# Smallest $k$-Enclosing Rectangle Revisited

**Timothy M. Chan**
Dept. of Computer Science, University of Illinois at Urbana-Champaign, USA
tmc@illinois.edu

**Sariel Har-Peled**
Dept. of Computer Science, University of Illinois at Urbana-Champaign, USA
sariel@illinois.edu

## Abstract

Given a set of $n$ points in the plane, and a parameter $k$, we consider the problem of computing the minimum (perimeter or area) axis-aligned rectangle enclosing $k$ points. We present the first near quadratic time algorithm for this problem, improving over the previous near-$O(n^{5/2})$-time algorithm by Kaplan et al. [23]. We provide an almost matching conditional lower bound, under the assumption that $(\min, +)$-convolution cannot be solved in truly subquadratic time. Furthermore, we present a new reduction (for either perimeter or area) that can make the time bound sensitive to $k$, giving near $O(nk)$ time. We also present a near linear time $(1+\varepsilon)$-approximation algorithm to the minimum area of the optimal rectangle containing $k$ points. In addition, we study related problems including the 3-sided, arbitrarily oriented, weighted, and subset sum versions of the problem.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Geometric optimization, outliers, approximation algorithms, conditional lower bounds

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.23

**Related Version** A full version is available at https://arxiv.org/abs/1903.06785.

**Funding** *Timothy M. Chan*: Supported in part by NSF AF award CCF-1814026.
*Sariel Har-Peled*: Supported in part by NSF AF award CCF-1421231.

## 1 Introduction

Given a set $P$ of $n$ points in the plane, and a parameter $k$, consider the problem of computing the smallest area/perimeter axis-aligned rectangle that contains $k$ points of $P$. (Unless stated otherwise, rectangles are axis-aligned by default.) This problem and its variants have a long history. Eppstein and Erickson [18] studied an exhaustive number of variants of this problem for various shapes.

For the minimum perimeter variant, the first work on this problem seems to be Aggarwal et al. [1], who showed a brute force algorithm with running time $O(n^3)$. Recently, Kaplan et al. [23] gave an algorithm with running time $O(n^{5/2} \log^2 n)$ that works for both minimum perimeter and area.

Several works derived algorithms with running time sensitive to $k$, the number of points in the shape. Aggarwal et al. [1] showed an algorithm for the minimum perimeter with running time $O(k^2 n \log n)$. This was improved to $O(n \log n + k^2 n)$ by Eppstein and Erickson [18] or alternatively by Datta et al. [16]. Kaplan et al.'s algorithm [23] for the $k$-insensitive case, coupled with these previous techniques [18, 16], results in an $O(n \log n + nk^{3/2} \log^2 k)$ running time, which is currently the state of the art.

Known techniques [18, 16] reduce the problem to solving $O(n/k)$ instances of size $O(k)$. These reductions work only for the perimeter case, not the area case – in particular, there are incorrect attributions in the literature to results on the minimum area rectangle – see the introduction of de Berg et al. [17] for details. De Berg et al. described an algorithm with running time $O(n \log^2 n + nk^2 \log n)$ for minimum area. Both de Berg et al. [17] and Kaplan et al. [23] left as an open question whether there is a reduction from the minimum-area problem to about $\widetilde{O}(n/k)$ instances of size $O(k)$, where $\widetilde{O}$ hides[1] polynomial factors in $\log n$ and $1/\varepsilon$. Such a reduction would readily imply an improved algorithm.

## Our results

We revisit the above problems and provide significantly improved algorithms:

1. Exact smallest $k$-enclosing rectangle. In Section 2.1 we describe an algorithm for the minimum $k$-enclosing rectangle (either area or perimeter) with running time $O(n^2 \log n)$ (see Theorem 2). It is based on a new divide-and-conquer approach, which is arguably simpler than Kaplan et al.'s algorithm. Known reductions mentioned above then lead to an $O(n \log n + nk \log k)$-time algorithm for computing the minimum perimeter rectangle.
2. $k$-sensitive running time for smallest area. In Section 2.2 we describe a reduction of the minimum-area problem to $O(\frac{n}{k} \log \frac{n}{k})$ instances of size $O(k)$ (see Theorem 8). Our reduction uses *shallow cutting* for 3-sided rectangular ranges [22] and is conceptually simple.
   Plugging this the aforementioned new $O(n^2 \log n)$-time algorithm leads to $O(nk \log \frac{n}{k} \log k)$ time algorithm for computing the minimum area $k$-enclosing rectangle (see Corollary 9). Thus, our new result strictly improves upon both Kaplan et al.'s and de Berg et al.'s results for all $k$, from constant to $\Theta(n)$.

The smallest enclosing rectangle problem is amenable to sampling. Kaplan et al. used samples in an approximation algorithm, with running time $\widetilde{O}(n/k)$, that computes a rectangle containing at least $(1-\varepsilon)k$ points of a prescribed perimeter, where $k$ is the maximum number of points in any such rectangle. Similarly, using relative approximations [21], de Berg et al. [17] showed an algorithm that computes, in $\widetilde{O}(n)$ time, a rectangle containing $\geq (1-\varepsilon)k$ points, where $k$ is the maximum number of points in any rectangle of a prescribed area. The "dual" problem, of approximating the minimum area rectangle containing $k$ points seems harder, since sampling does not directly apply to it.

1. Approximating the area of the smallest $k$-enclosing rectangle. In Section 2.3, we present an approximation algorithm that computes, in $O(n \log n)$ expected time, a rectangle containing $k$ points of area $\leq (1+\varepsilon)\alpha^*$, for a constant $\varepsilon \in (0,1)$, where $\alpha^*$ is the smallest-area of such a rectangle (see Theorem 13).

We next present a flotilla of related results:

1. 3-sided smallest $k$-enclosing rectangle. In Section 3.1 we (slightly) speed up the exact algorithm for the 3-sided rectangles case (i.e., rectangles that must have their bottom edge on the $x$-axis). The running time is $O\big(n^2/2^{\Omega(\sqrt{\log n})}\big)$, and is obtained using known results on the *(min,+)-convolution* problem [7, 28] (see Theorem 16).
2. Arbitrarily oriented smallest $k$-enclosing rectangle. In Section 3.2 we briefly consider the variant where the rectangle may not be axis-aligned. We show that this problem can be solved in $O(n^3 \log n + n^3 k/2^{\Omega(\sqrt{\log k})})$ time, slightly improving a previous result of $O(n^3 k)$ [15] when $k$ is not too small.

[1] We reserve the right, in the future, to use the $\widetilde{O}$ to hide any other things we do not like.

3. Minimum-weight $k$-enclosing rectangle. In Section 3.3 we show how to extend our $O(n^2 \log n)$-time algorithm to the related problem of finding a minimum-weight rectangle that contains $k$ points, for $n$ given weighted points in the plane (see Theorem 17).
4. Subset sum for $k$-enclosing rectangle. In Section 3.4, we study the problem of finding a rectangle that contains $k$ points and has a prescribed weight $W$ (or as close as one can get to it). The running time of the new algorithm is $O(n^{5/2} \log n)$ (see Theorem 19).
5. Conditional lower bound. In Section 3.5, we prove that our near quadratic algorithm for exact minimum (perimeter or area) $k$-enclosing rectangle is near optimal up to an arbitrarily small polynomial factor, under a "popular" conjecture that the (min,+)-convolution problem cannot be solved in truly subquadratic time [14].

## 2 Smallest $k$-enclosing rectangle

### 2.1 An exact near-quadratic algorithm

Our $O(n^2 \log n)$-time algorithm for minimum $k$-enclosing rectangles is based on divide-and-conquer. It has some similarity with an $O(n^2)$-time divide-and-conquer algorithm by Barbay et al. [5] for a different problem (finding the minimum-weight rectangle for $n$ weighted points in the plane, without any $k$-enclosing constraint), but the new algorithm requires more ingenuity.

We start with a semi-dynamic data structure for a 1D subproblem:

▶ **Lemma 1.** *Given a set $P$ of $n$ points in 1D with $q$* marked *points, and an integer $k$, we can maintain an $O(q^2)$-space data structure, with $O(n \log n + nq)$ preprocessing time, that supports the following operations:*

- *report the shortest interval containing $k$ points of $P$ in $O(q)$ time;*
- *delete a marked point in $O(q)$ time;*
- *unmark a marked point in $O(q)$ time.*

**Proof.** Sort the points $P$, and let $p_1, \ldots, p_n$ he resulting order. Consider the (implicit) matrix $M = P - P$. Formally, the entry $M_{ij}$ is $p_j - p_i$ (we are interested only in the top right part of this matrix) – such an entry can be computed in $O(1)$ time directly from the sorted point set. The optimal quantity of interest is the minimum on the $k$th diagonal; that is, $\alpha(M) = \min_i M_{i,i+k-1}$. When a marked point get deleted, this corresponds to deleting a row and a column of $M$ – the quantity of interest remains the minimum along the $k$th diagonal. Such a deletion, as far as a specific entry of the top right of the matrix is concerned, either (i) removes it, (ii) keeps it in its place, (iii) shift it one diagonal down as its moves left, or (iv) keep it on the same diagonal as it shifts both up and left (see Figure 1).

In particular, any sequence of at most $q$ deletions of elements can shift an entry in the matrix at most $q$ diagonals down. This implies that we need to keep track only of the $k, \ldots, k+q$ diagonals of this matrix. To do better, observe that if we track the elements of an original diagonal of interest, the deletions can fragment the diagonal into at most $O(q)$ groups, where each group still appear as contiguous run of the original diagonal.

To this end, let a **fragment** of a diagonal be either (i) a singleton entry that appears in a row or column of a marked point, or (ii) a maximum contiguous portion of the diagonal which does not touch any singleton entries from (i). It is easy to verify that the $k$th diagonal of the matrix at any given point in time is made out of a sequence of at most $3k$ fragments, where each fragment is an original fragment of one of the diagonals in the range $k, \ldots, k+q$.

As such, instead of storing all the elements of a fragment, we only maintain the minimum entry of the fragment (together with the information of what pairs of points it corresponds to). After this compression, a diagonal of interest can be represented as a linked list of $O(q)$

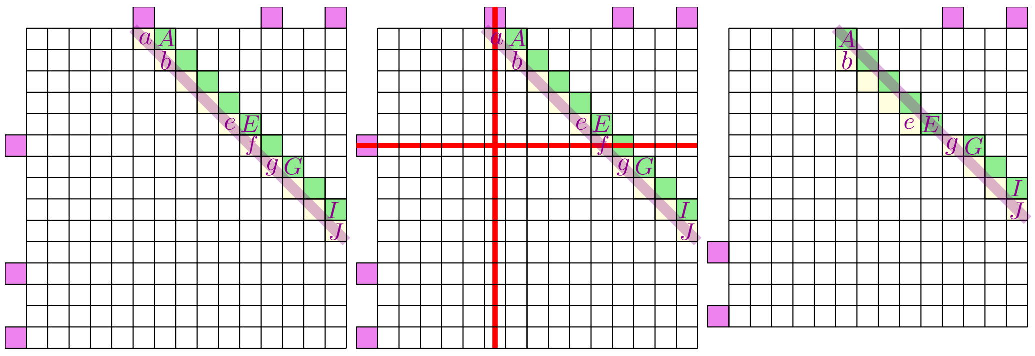


**Figure 1** Behold the matrix!

fragment summaries. In the preprocessing stage, the algorithm computes this representation for the $k$ to $k+q$ diagonals (using this representation). This requires $O(q^2)$ space, and $O(nq)$ time.

A deletion of a marked point then corresponds to taking a contiguous block of linked fragments at the $i$th list and moving it to list $i-1$, doing this surgery for $i = k, \ldots, k+q$. The blocks being moved start and end in singleton entries that correspond to the deleted point. We also need to remove these two singleton elements, and merge the two adjacent fragment summaries that are no longer separated by a singleton. This surgery for all the $q+1$ lists of interest can be done in $O(q)$ time (we omit the tedious but straightforward details).

A query corresponds to scanning the $k$th diagonal and reporting the minimum value stored along it. An unmarking operation corresponds to merging two fragment summaries and the singleton separating them into a single fragment summary, and doing this for all the $q+1$ lists. Both operations clearly can be done in $O(q)$ time. ◀

▶ **Theorem 2.** *Given a set $P$ of $n$ points in the plane and an integer $k$, one can compute, in $O(n^2 \log n)$ time, the smallest-area/perimeter axis-aligned rectangle enclosing $k$ points.*

**Proof.** We do divide-and-conquer by $y$-coordinates. Given a set $P$ of $n$ points in the plane, and horizontal slabs $\sigma$ and $\tau$, each containing $q$ points of $P$, we describe a recursive algorithm to find a smallest $k$-enclosing axis-aligned rectangle containing $P$, under the restriction that the top edge is inside $\sigma$ and the bottom edge is inside $\tau$. It is assumed that either $\sigma$ is completely above $\tau$, or $\sigma = \tau$. It is also assumed that all points above $\sigma$ or below $\tau$ have already been deleted from $P$. There can still be a large number of points in $P - (\sigma \cup \tau)$ (recursion will lower $q$ but not necessarily $n$). We will not explicitly store the points in $P - (\sigma \cup \tau)$, but rather "summarize" the points in an $O(q^2)$-space structure. Namely, we assume that the $x$-coordinates of $P$ are maintained in the 1D data structure $\mathcal{S}$ of Lemma 1, where the marked points are the $O(q)$ points in $P \cap (\sigma \cup \tau)$.

The algorithm proceeds as follows:

1. If $q = 1$, then report the answer by querying $\mathcal{S}$ in $O(1)$ time. Else:
2. Divide $\sigma$ into two horizontal subslabs $\sigma_1$ and $\sigma_2$, each containing $q/2$ points of $P$. Likewise divide $\tau$ into $\tau_1$ and $\tau_2$.
3. For each $i, j \in \{1, 2\}$, recursively solve the problem for the slabs $\sigma_i$ and $\tau_j$;[2] to prepare for the recursive call, make a copy of $\mathcal{S}$, delete the (marked) points in $P \cap (\sigma \cup \tau)$ above

[2] If $\sigma = \tau$, one of the four recursive calls is unnecessary.

$\sigma_i$ or below $\tau_j$, and unmark the remaining points in $P \cap (\sigma \cup \tau) - (\sigma_i \cup \tau_j)$, as shown in Figure 2. The time needed for these $O(q)$ deletions and unmarkings, and for copying $\mathcal{S}$, is $O(q^2)$ (we emphasize that this bound is independent of $n$).

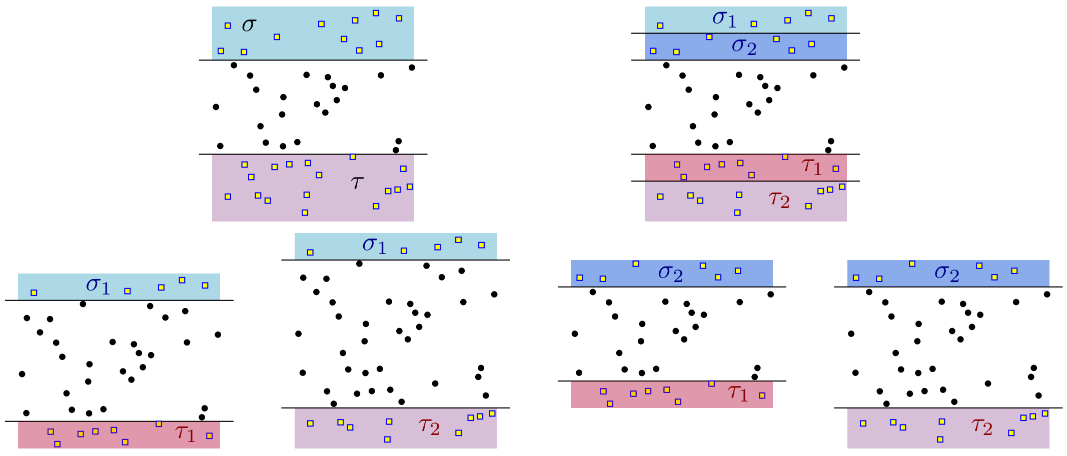


**Figure 2** Divide et impera for the problem at hand.

The running time satisfies the recurrence

$$T(n, q) = 4\,T(n, q/2) + O(q^2),$$

with $T(n, 1) = O(1)$, which gives $T(n, q) = O(q^2 \log q)$. Initially, $\sigma = \tau$ is the entire plane, with $q = n$; the data structure $\mathcal{S}$ can be preprocessed in $O(n^2)$ time. Thus, the total running time is $O(n^2 \log n)$. ◀

One can readily get an algorithm with $k$-sensitive running time for the perimeter case, by reducing the problem into $O(n/k)$ instance of size $O(k)$. This reduction is well known [18, 16] in this case – approximate the smallest enclosing disk containing $k$ points in $O(n \log n)$ time, partition the plane into a grid with side length proportional to the radius of this disk, and then solve the problem for each cluster (i.e., $3 \times 3$ group of grid cells) that contains at least $k$ points of $P$, using our above algorithm. We thus get the following.

▶ **Corollary 3.** *Given a set $P$ of $n$ points in the plane and an integer $k$, one can compute, in $O(n \log n + nk \log k)$ time, the smallest-perimeter axis-aligned rectangle enclosing $k$ points of $P$.*

The $O(n \log n)$ term can be eliminated in the word RAM model, using a randomized linear-time algorithm for approximate smallest $k$-enclosing disk [20] (which requires integer division and hashing).

A similar reduction for the minimum-area case is more challenging, and was left as an open problem in previous work [23]. The difficulty arises because the optimal-area rectangle may be long and thin, with side length potentially much bigger than the radius of the minimum $k$-enclosing disk. Nonetheless, we show that such a reduction is possible (with an extra logarithmic factor) in the next subsection.

## 2.2 $k$-sensitive running time for smallest area

Our starting point is a shallow cutting lemma for 3-sided ranges [22]. (It can be viewed as an orthogonal variant of Matoušek's shallow cutting lemma for halfspaces [24].)

▶ **Lemma 4** ([22]). *Given a set $P$ of $n$ points in the plane, lying above a horizontal line $\ell$, and a parameter $k$, one can compute a family $\mathcal{F}$ of at most $2\lceil n/k \rceil$ subsets of $P$, each of size at most $6k$. The collection of sets can be computed in $O(n)$ time if the $x$-coordinates have been pre-sorted. For any axis-aligned rectangle $R$ with its bottom edge lying on $\ell$, that contains less than $k$ points of $P$, we have $P \cap R \subseteq PA$ for some $PA \in \mathcal{F}$.*

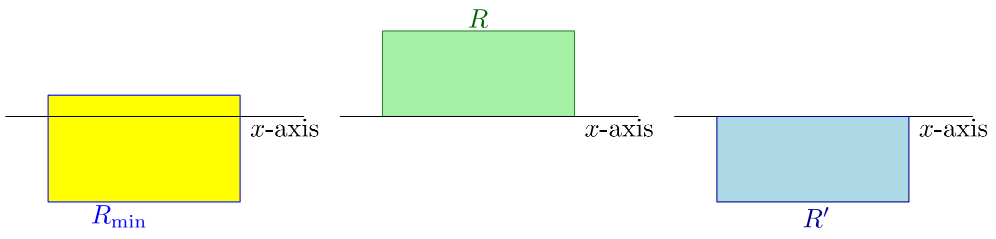
▶ **Definition 5.** *Let $\mathcal{R}$ be the set of all axis-aligned rectangles in the plane. A* ***scoring function*** *is a function $f : \mathcal{R} \to \mathbb{R}$, with the following properties:*
1. *Translation invariant: $\forall p \in \mathbb{R}^2$, we have $f(p + R) = f(R)$.*
2. *Monotonicity: $\forall R, R' \in \mathcal{R}$, such that $R \subseteq R'$, we have that $f(R) \leq f(R')$.*

Functions that satisfy the above definition include area, perimeter, diameter, and enclosing radius of a rectangle.

For a set $U \subseteq \mathbb{R}^d$, let $\updownarrow U = \{(x, -y) \mid (x, y) \in U\}$ be the **reflection** of $U$ through the $x$-axis. Similarly, let $|U|_y = \{(x, |y|) \mid (x, y) \in U\}$ be the **folding** of $U$ through the $x$-axis.

▶ **Lemma 6.** *Given a set $P$ of $n$ points in the plane, a parameter $k$, and a scoring function $f$, let $R_{\min}$ be the minimum score axis-aligned rectangle that contains $k$ points of $P$, and intersects the $x$-axis. Then, one can compute a family $\mathcal{F}$ of $O(n/k)$ subsets of $P$, such that (i) each set of $\mathcal{F}$ is of size $\leq 12k$, and (ii) $R_{\min} \cap P \subseteq PA$, for some $PA \in \mathcal{F}$.*

**Proof.** Let $P' = |P|_y$ be the "folding" of $P$ over the $x$-axis, and let $\mathcal{F}'$ be the cover of $P'$ by sets of size $\leq 12k$, as computed by Lemma 4 for rectangles containing at most $2k$ points. Let $\mathcal{F}$ be the corresponding family of sets for $P$. We claim that $\mathcal{F}$ has the desired property.



Let $R = |R_{\min}|_y$ be the folding of $R_{\min}$, and let $R' = \updownarrow R$. Observe that $f(R) = f(R') \leq f(R_{\min})$ because of the translation invariance of $f$, and monotonicity of $f$.

If $|R \cap P| > k$ then one can shrink it so that it contains only $k$ points of $P$, but this would imply that $R_{\min}$ is not the minimum, a contradiction. The case that $|R' \cap P| > k$ leads to a similar contradiction. We conclude that $R \cup R'$ contains at most $2k$ points of $P$. Implying that $R = |R_{\min}|_y$ contains at most $2k$ points of $P'$. As such, there is a set $PA' \in \mathcal{F}'$ that contains $R \cap P'$. Now, let $PA$ be the corresponding set in $\mathcal{F}$ to $PA'$. Since $R_{\min} \subseteq R \cup R'$, it follows that $R_{\min} \cap P \subseteq (R \cup R') \cap P \subseteq PA$, as desired. ◀

▶ **Lemma 7.** *Given a set $P$ of $n$ points in the plane, a parameter $k$, and a scoring function $f$, let $R_{\min}$ be the minimum score rectangle that contains $k$ points of $P$. One can compute, in $O(n \log n)$ time, a family $\mathcal{F}$ of $O(\frac{n}{k} \log \frac{n}{k})$ subsets of $P$, such that (i) each subset of $\mathcal{F}$ is of size $\leq 12k$, and (ii) $R_{\min} \cap P \subseteq PA$, for some $PA \in \mathcal{F}$.*

**Proof.** Find a horizontal line $\ell$ that splits $P$ evenly, and compute the family of Lemma 6. Now recurse on the points above $\ell$ and the points below $\ell$. The recursion bottoms out when the number of points is $\leq 12k$. The correctness is by now standard – as soon as a recursive call picks a line that stabs the optimal rectangle, the family generated for this line contains the desired set. The $x$-coordinates need to be pre-sorted just once at the beginning. ◀

▶ **Theorem 8.** *Let $P$ be a set of $n$ points in the plane, $k$ be a parameter, $f$ be a scoring function for rectangles, and let* **`alg`** *be an algorithm the computes, in $T_{\mathtt{alg}}(m)$ time, the axis-aligned rectangle containing $k$ points in a set of $m$ points that minimizes $f$. Then one can compute the rectangle containing $k$ points of $P$ that minimizes $f$, in time $O\big(n \log n + T_{\mathtt{alg}}(12k)\frac{n}{k}\log\frac{n}{k}\big)$.*

**Proof.** Compute the family of sets $\mathcal{F}$ using Lemma 7, and then apply `alg` to each set in this family. ◀

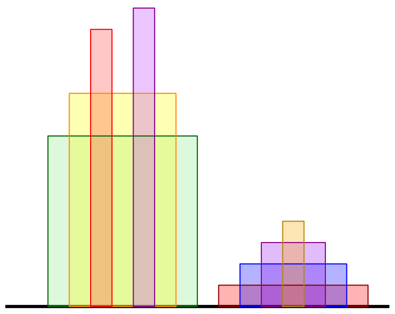
Combining Theorem 2 with Theorem 8 gives the following.

▶ **Corollary 9.** *Given a set $P$ of $n$ points in the plane and an integer $k$, one can compute, in $O(nk \log \frac{n}{k} \log k)$ time, the smallest-area axis-aligned rectangle enclosing $k$ points of $P$.*

For the case when $t = n - k$ is very small (i.e., finding the smallest enclosing axis-aligned rectangle with $t$ *outliers*), there is an easy reduction (for both perimeter and area) yielding $O(n+T_{\mathtt{alg}}(4t))$ time [26, 2], by keeping the $t$ leftmost/rightmost/topmost/bottommost points. Immediately from Theorem 2, we get $O(n + t^2 \log t)$ running time.

## 2.3 An approximation algorithm for smallest area

In this subsection, we give an efficient approximation algorithm for the smallest-area $k$-enclosing rectangle problem. The smallest perimeter case is straightforward to approximate, by grid rounding, but the area case is tougher, again because the optimal rectangle may be long and thin.



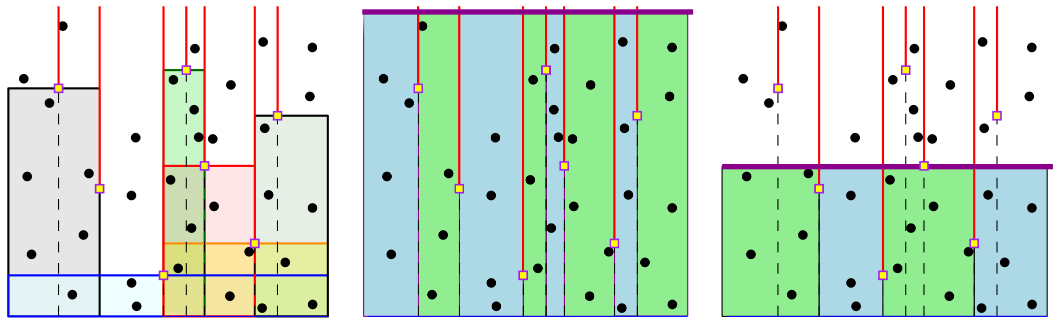
▶ **Definition 10.** *A* ***laminar*** *family of 3-sided rectangles is a collection $\mathcal{R}$ of axis-aligned rectangles with the bottom edges lying on the $x$-axis, such that for every pair of rectangles $[a, b] \times [0, c]$ and $[a', b'] \times [0, c']$ in $\mathcal{R}$, one of the following is true:*

- *$[a, b] \cap [a', b'] = \emptyset$, or*
- *$[a, b] \subseteq [a', b']$ and $c > c'$, or*
- *$[a', b'] \subseteq [a, b]$ and $c' > c$.*

Standard range trees can answer orthogonal range counting queries (counting the number of points inside rectangular ranges) in logarithmic time per query (this has been improved to $O(\sqrt{\log n})$ in the offline setting by Chan and Pătraşcu [12]). The following lemma shows how to achieve constant time per query in the offline laminar special case, which will be useful later in our approximation algorithm.

▶ **Lemma 11.** *Let $P$ be a set of $n$ points, and let $\mathcal{R}$ be a laminar family of $O(n)$ 3-sided rectangles in the plane. Suppose that we are given a* ***designated point*** *on the top edge of each rectangle in $\mathcal{R}$, and the $x$- and $y$-coordinates of all the designated points and all the points of $P$ have been pre-sorted. Then we can count, for each rectangle $R \in \mathcal{R}$, the number of points of $P$ inside the rectangle, in $O(n)$ total time.*

**Proof.** We describe a sweep algorithm, with a horizontal sweep line $\ell$ moving downward. Let $X$ denote the set of $x$-coordinates of all points of $P$ and all designated points of the rectangles of $\mathcal{R}$. Consider the union of $R \cap \ell$ over all $R \in \mathcal{R}$; it can be expressed as a union of disjoint intervals. Let $\mathcal{I}_\ell$ be the $x$-projection of these disjoint intervals. Store the following collection $\Gamma_\ell$ of disjoint sets in a *union-find* data structure [27]: for each interval $I \in \mathcal{I}_\ell$, define the set $X \cap I$, and for each $a \in X$ not covered by $\mathcal{I}_\ell$, define the singleton set $\{a\}$. Create a linked list $L_\ell$ containing these sets in $\Gamma_\ell$ ordered by $x$. For each set in $\Gamma_\ell$, we store a count of the number of points of $P$ below $\ell$ with $x$-coordinates inside the set.



Suppose that the sweep line $\ell$ hits the top edge of a rectangle $R$ with $x$-projection $[a, b]$. By definition of a laminar family, any interval in $\mathcal{I}_\ell$ that intersects $[a, b]$ must be contained in $[a, b]$. We find the set in $\Gamma_\ell$ that contains the $x$-coordinate of the designated point of $R$. From this set, we walk through the list $L_\ell$ in both directions to find all sets contained in $[a, b]$, and replace these sets with their union in $\Gamma_\ell$ and $L_\ell$. The count for the new set is the sum of the counts of the old sets; this also gives the output count for the rectangle $R$.

Next, suppose that the sweep line $\ell$ hits a point $p \in P$. We find the set in $\Gamma_\ell$ that contains the $x$-coordinate of $p$, and decrement its count. (For example, if the set is a singleton, its count changes from 1 to 0.)

The entire sweep performs $O(n)$ union and find operations. Gabow and Tarjan [19] gave a linear-time union-find algorithm for the special case where the "union tree" is known in advance; their algorithm is applicable, since the union tree here is just a path of the elements ordered by $x$. ◀

We first solve the approximate decision problem:

▶ **Lemma 12.** *Given a set $P$ of $n$ points in the plane, a value $\alpha$, and parameters $k$ and $\varepsilon \in (0, 1)$, one can either compute a $k$-enclosing axis-aligned rectangle $R'$ such that* $\mathrm{area}(R') \le \big(1 + O(\varepsilon)\big)\alpha$*, or conclude that the smallest-area $k$-enclosing axis-aligned rectangle has area greater than $\alpha$. The running time of the algorithm is $O\big(\varepsilon^{-3} \log \varepsilon^{-1} \cdot n \log n\big)$.*

**Proof.** It is sufficient to solve the problem for the case where the rectangle must intersect a horizontal line $\ell$ in $O((1/\varepsilon)^3 \log(1/\varepsilon) \cdot n)$ time, assuming that the $x$- and $y$-coordinates of the given points $P$ have been pre-sorted. Then standard divide-and-conquer by $y$-coordinates gives an $O((1/\varepsilon)^3 \log(1/\varepsilon) \cdot n \log n)$-time algorithm for the general problem. Pre-sorting needs to be done only once at the beginning.

Without loss of generality, assume that $\ell$ is the $x$-axis. Suppose there exists a rectangle $R^*$ intersecting $\ell$ that contains at least $k$ points and has area at most $\alpha$. By symmetry, we may assume that $R^*$ has greater area above $\ell$ than below, and that the top edge passes through some input point $p = (p_x, p_y) \in P$. Then the height $h^*$ of $R^*$ is between $p_y$ and $2p_y$, and the width $w^*$ is at most $\alpha/p_y$.

Without loss of generality, assume that all $x$-coordinates are in $[0, 1/3]$. Define a one-dimensional **quadtree interval** (also known as a *dyadic interval*) to be an interval of the form $[\frac{m}{2^i}, \frac{m+1}{2^i}]$. It is known that every interval of length $w < 1/3$ is contained in a quadtree interval of length $O(w)$ after shifting the interval by one of two possible values $s \in \{0, 1/3\}$ (this is a special case of a shifting lemma for $d$-dimensional quadtrees [6, 8]). Thus, suppose that $[p_x - \alpha/p_y, p_x + \alpha/p_y]$ is contained in the interval $[\frac{m}{2^i} + s, \frac{m+1}{2^i} + s]$, where the length $\frac{1}{2^i}$ is equal to the smallest power of 2 greater than $c\alpha/p_y$, for some constant $c$. (Note that $i$ is a nondecreasing function of $p_y$.) Without loss of generality, assume that $1/\varepsilon = 2^E$ for an integer $E$. Define a family of $O(1/\varepsilon^3)$ **canonical** rectangles of the form

$$[\tfrac{m}{2^i} + \tfrac{j}{2^{i+E}} + s,\ \tfrac{m}{2^i} + \tfrac{j'}{2^{i+E}} + s] \ \times\ [-j''\varepsilon p_y, p_y]$$

over all possible indices $j, j', j'' \in \{0, \ldots, 1/\varepsilon\}$ such that $p_x \in [\frac{m}{2^i} + \frac{j}{2^{i+E}} + s,\ \frac{m}{2^i} + \frac{j'}{2^{i+E}} + s]$.

By rounding, $R^*$ is contained in a canonical rectangle $R'$ with height at most $h^* + O(\varepsilon)p_y \leq (1 + O(\varepsilon))h^*$ and width at most

$$w^* + O(\varepsilon)\alpha/p_y \leq \big(1 + O(\varepsilon)\big)\alpha/h^*,$$

and thus area at most $(1 + O(\varepsilon))\alpha$. So, it suffices to count the number of points inside each canonical rectangle and return the smallest area among those rectangles containing at least $k$ points.

To speed up range counting, observe that for canonical rectangles with the same $j, j', j''$, the same $s \in \{0, 1/3\}$, and the same value for $(i \bmod E)$, the portion of the rectangles above (resp. below) $\ell$ forms a laminar family. This is because: (i) in the $x$-projections, if a pair of intervals intersects, one interval must be contained in the other; (ii) as the height of the 3-sided rectangle increases, $p_y$ increases, and so $i$ can only increase (or stay the same), and so the width of the rectangle can only decrease (or stay the same). Thus, we can apply Lemma 11 to compute the counts of the points inside each rectangle, for all canonical rectangles with a fixed $j, j', j'', s$ and $(i \bmod E)$, in $O(n)$ time (for each canonical rectangle, we can use a point $(p_x, p_y) \in P$ on the top edge, and a corresponding point $(p_x, -j''\varepsilon p_y)$ on the bottom edge, as the designated points). The number of choices for $j, j', j'', s$ and $(i \bmod E)$ is $O((1/\varepsilon)^3 \log(1/\varepsilon))$. ◀

▶ **Theorem 13.** *Given a set $P$ of $n$ points in the plane, and parameters $k$ and $\varepsilon \in (0, 1)$, one can compute a $k$-enclosing rectangle $R'$ such that* $\mathrm{area}(R') \leq (1 + \varepsilon)\mathrm{opt}(P, k)$*, where* $\mathrm{opt}(P, k)$ *is the area of the smallest axis-aligned rectangle containing $k$ points of $P$. The expected running time of the algorithm is $O\big((1/\varepsilon)^3 \log(1/\varepsilon) \cdot n \log n\big)$.*

**Proof.** We can use known techniques for reducing optimization problems to decision problems. We give a self-contained description of one approach based on Chan's randomized technique [9].

Let $b$ be a sufficiently large constant. Divide the plane into $b$ columns (vertical slabs) each containing $n/b$ points. Similarly divide the plane into $b$ rows (horizontal slabs) each containing $n/b$ points. These steps take linear time by invoking a selection algorithm $O(b)$ times. For each quadruple $\tau = (c, c', r, r')$ where $c$ and $c'$ are columns (with $c$ left of $c'$ or $c = c'$) and $r$ and $r'$ are rows (with $r$ below $r'$ or $r = r'$), consider the subproblem of finding the smallest-area rectangle containing $k$ points of $P$, subject to the extra constraints that the left edge of the rectangle lies in $c$, the right edge lies in $c'$, the bottom edge lies in $r$, and the top edge lies in $r'$. To solve this subproblem, it suffices to consider the at most $4n/b$ points in $P \cap (c \cup c' \cup r \cup r')$. To ensure that the extra constraints are satisfied, we add $4n/b$ copies of the four intersection points formed by the right boundary of $c$, the left boundary of $c'$, the top boundary of $r$, and the bottom boundary of $r'$; and we add $16n/b$ to $k$. (Straightforward modifications can be made in the special case when $c = c'$ or $r = r'$.) Let $P_\tau$ be the resulting point set of size at most $20n/b$ points, and $k_\tau$ be the resulting value of $k$. We thus have

$$\mathrm{opt}(P, k) = \min_\tau \mathrm{opt}(P_\tau, k_\tau).$$

To compute an approximation to the minimum, we consider the at most $b^4$ quadruples in *random order* $\tau_1, \tau_2, \ldots$ and keep track of an approximate minimum $\alpha$ with the invariant that $\alpha \le \min\{\mathrm{opt}(P_{\tau_1}, k_{\tau_1}), \ldots, \mathrm{opt}\big(P_{\tau_{i-1}}, k_{\tau_{i-1}}\big)\} < (1+\varepsilon)\alpha$ after the $(i-1)$th iteration. Let $\varepsilon'$ be such that $(1+\varepsilon')^2 = 1+\varepsilon$; note that $\varepsilon' = \Theta(\varepsilon)$. At the $i$th iteration, we run the approximate decision procedure for $P_{\tau_i}$ twice, at values $\alpha$ and $\alpha/(1+\varepsilon')$, which allows us to conclude one of the following:

- $\mathrm{opt}(P_{\tau_i}, k_{\tau_i}) \ge \alpha$. In this case, we can continue to the next iteration and the invariant is maintained.
- $\alpha/(1+\varepsilon') \le \mathrm{opt}(P_{\tau_i}, k_{\tau_i}) < (1+\varepsilon')\alpha$. In this case, we reset $\alpha$ to $\alpha/(1+\varepsilon')$ and the invariant is maintained.
- $\mathrm{opt}(P_{\tau_i}, k_{\tau_i}) < \alpha$. In this case, we recursively compute an approximation $\alpha_i$ to the quantity $\mathrm{opt}(P_{\tau_i}, k_{\tau_i})$, satisfying $\alpha_i \le \mathrm{opt}(P_{\tau_i}, k_{\tau_i}) < (1+\varepsilon)\alpha_i$. We reset $\alpha$ to $\alpha_i$ and the invariant is maintained.

We land in the third case only if $\mathrm{opt}(P_{\tau_i}, k_{\tau_i})$ is the smallest among the $i$ values $\mathrm{opt}(P_{\tau_1}, k_{\tau_1})$, $\ldots$, $\mathrm{opt}(P_{\tau_i}, k_{\tau_i})$, which happens with probability at most $1/i$. Thus, the expected number of recursive calls is bounded by the $(b^4)$th Harmonic number $\sum_{i=1}^{b^4} 1/i < \ln(b^4) + 1$. The expected running time satisfies the recurrence

$$T(n) \;\le\; (4\ln b + 1)\, T(20n/b) + O\big((1/\varepsilon)^3 \log(1/\varepsilon) \cdot n \log n\big),$$

which gives $T(n) = O\big((1/\varepsilon)^3 \log(1/\varepsilon) \cdot n \log n\big)$ when $b = 1000$, for example. ◀

## 3 Extensions

### 3.1 3-sided smallest $k$-enclosing rectangle

In this subsection, we give a slightly faster algorithm for the 3-sided variant of the problem, finding the smallest-area/perimeter rectangle enclosing $k$ points, under the restriction that the bottom edge lies on the $x$-axis. The improvement uses the latest result on the (min,+)-convolution problem, and is interesting in view of a reduction in Section 3.5 in the reverse direction, establishing essentially an equivalence of the 3-sided problem to (min,+)-convolution.

▶ Problem 14. (min,+)-Convolution. Given real numbers $a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}$, compute $c_\ell = \min_{i=0}^{\ell}(a_i + b_{\ell-i})$ for all $\ell = 0, \ldots, 2n-2$.

Let $T_{\mathrm{convol}}(n)$ be the time complexity of the (min,+)-convolution problem. As observed by Bremner et al. [7], the problem can be reduced to (min,+)-matrix multiplication, and using the current best result by Williams [28] (derandomized by Chan and Williams [13]), $T_{\mathrm{convol}}(n) = O(n^2/2^{\Omega(\sqrt{\log n})})$. We use (min,+)-convolution to speed up the preprocessing time of the 1D data structure from Section 2.1.

▶ **Lemma 15.** *The preprocessing time in Lemma 1 can be reduced to* $O((n/q)T_{\mathrm{convol}}(q) + q^3)$.

**Proof.** Divide the $n \times n$ matrix $M$ vertically into $n/q$ submatrices $M_1, \ldots, M_{n/q}$ each of dimension $n \times q$. For each submatrix $M_i$, we consider the portions of the diagonals $k, \ldots, k+q$ that are within $M_i$ – each such portion will be called a **chunk**. We precompute the minimum of the entries in each chunk. For a fixed $i$, this is equivalent to computing $\min_{qi<j\le q(i+1)}(p_{j+\ell-1} - p_j)$ for all $\ell \in \{k, \ldots, k+q\}$. Notice that after some massaging of the sequence (negating, reversing, and padding), this computation can be reduced to (min,+)-convolution over $O(q)$ elements, and can thus be done in $O(T_{\mathrm{convol}}(q))$ time. The total time over all $i$ is $O((n/q)T_{\mathrm{convol}}(q))$.

Recall that in the preprocessing algorithm in Lemma 1, we need to compute the minimum of each fragment in the $k, \ldots, k+q$ diagonals. Each fragment can be decomposed into some number of disjoint chunks plus $O(q)$ extra elements. Over all $O(q)$ diagonals, there are $O(q^2)$ fragments and $O(n/q \cdot q) = O(n)$ chunks in total. Thus, we can compute the minima of all fragments in $O(q^2 \cdot q + n/q \cdot q) = O(q^3 + n)$ time, after the above precomputation of the minima of all chunks. ◀

▶ **Theorem 16.** *Given a set $P$ of $n$ points in the plane and integer $k$, one can compute, in $O(n^2/2^{\Omega(\sqrt{\log n})})$ time, the smallest-area/perimeter axis-aligned rectangle enclosing $k$ points of $P$, under the restriction that the bottom edge lies on the $x$-axis.*

**Proof.** Divide the plane into $n/q$ horizontal slabs each containing $q$ points, for some parameter $q$ to be set later.

Take such a slab $\sigma$. We solve the subproblem of finding a smallest $k$-enclosing axis-aligned rectangle under the restriction that the top edge is in $\sigma$ and the bottom edge is on the $x$-axis. To this end, we first delete all points above $\sigma$ or below the $x$-axis. We build the 1D data structure $\mathcal{S}$ in the lemma for the $x$-coordinates of the surviving points, where the marked points are the $q$ points in $\sigma$. The preprocessing time is $O((n/q)T_{\mathrm{convol}}(q) + q^3)$. Then for each point $p \in \sigma$, we can compute a smallest $k$-enclosing axis-aligned rectangle where the top edge has $p$'s $y$-coordinate and bottom edge is on the $x$-axis, by making a copy of $\mathcal{S}$, deleting all points in $\sigma$ above $p$, and querying $\mathcal{S}$. The time needed for the $O(q)$ deletions, and for copying $\mathcal{S}$, is $O(q^2)$. The total time over all $p \in \sigma$ is $O(q^3)$.

We return the minimum (by area or perimeter) of all the rectangles found. The overall running time over all $n/q$ slabs $\sigma$ is

$$O((n/q) \cdot ((n/q)T_{\mathrm{convol}}(q) + q^3)).$$

With $T_{\mathrm{convol}}(q) = O(q^2/2^{\Omega(\sqrt{\log q})})$, we can set $q = n^{1/3}$, for example, and obtain the final time bound $O(n^2/2^{\Omega(\sqrt{\log q})})$. ◀

For $k$-sensitive bounds, we can apply the shallow cutting technique from Section 2.2 (which is easier for 3-sided rectangles) and obtain an $O(n \log n + nk/2^{\Omega(\sqrt{\log k})})$ time bound.

## 3.2 Arbitrarily oriented smallest $k$-enclosing rectangle

We briefly consider the problem of computing a smallest-area/perimeter arbitrarily oriented rectangle (not necessarily axis-aligned) enclosing $k$ points. The optimal rectangle is defined by 5 points, with one edge containing 2 points $p_1^*$ and $p_2^*$. Given a fixed choice of $p_1^*$ and $p_2^*$, we can use a rotation and translation to make $p_1^*p_2^*$ lie on the $x$-axis and thereby obtain a 3-sided axis-aligned rectangle problem, which can be solved in $O(n \log n + nk/2^{\Omega(\sqrt{\log k})})$ time. Exhaustively trying all pairs $p_1^*p_2^*$ then gives $O(n^3 \log n + n^2k/2^{\Omega(\sqrt{\log k})})$ total time.

## 3.3 Minimum-weight $k$-enclosing rectangle

Our $O(n^2 \log n)$-time algorithm can be adapted to solve the following related problem. (Without the $k$ constraint, the problem has an $O(n^2)$-time algorithm [5].)

▶ **Theorem 17.** *Given a set $P$ of $n$ points in the plane each with a real weight, and an integer $k$, one can compute, in $O(n^2 \log n)$ time, the axis-aligned rectangle enclosing $k$ points minimizing the total weight of the points inside.*

**Proof.** We follow the same approach as in Section 2.1, with the following differences in the data structure of Lemma 1. For every fragment we maintain the minimum weight solution. Using prefix sums, the entry $M_{i,j}$ in the matrix contains the total weight of the elements from $i$ to $j$. As before, we break the $q+1$ diagonals of entry into fragments, where each fragment summary maintains the minimum weight encountered.

A deletion of a marked point $p$ of weight $w$ would result is an insertion of a fixup entry, of value $-w$ into a linked list of a diagonal where $p$ appeared as a singleton (when crossing a column of $p$), and a fixup entry of value $+w$ when encountering the row column of $p$. The real value of a fragment is the value stored in the fragment plus the total sum of the fixups appearing before it in the linked list of its diagonal. As such, during query the real value can be computed in $O(q)$ time overall, as this list is being scanned. When we merge two adjacent fragments separated by a singleton, we should increase the later fragment by the fixup value at the singleton before taking the minimum. Clearly, all the operations can be implemented in $O(q)$ time.

Now, we can use the divide-and-conquer algorithm in the proof of Theorem 2 with no change. ◀

As an application, we can solve the following problem: given $n$ points in the plane each colored red or blue, and an integer $k$, find an axis-aligned rectangle enclosing exactly $k$ points minimizing the number of red points inside. This is a special case of the problem in the above theorem, where the red points have weight 1 and blue points have weight 0, and can thus be solved in $O(n^2 \log n)$ time.

Similarly, we can solve for other variants of the red/blue problem, for example, finding a $k$-enclosing rectangle maximizing (or minimizing) the number of red points, or finding a $k$-enclosing rectangle with exactly a given number $k_r$ of red points. (For the latter, the following observation allows us to reduce the 1D subproblem to querying for the maximum and minimum: given a set $P$ of red/blue points in 1D and a value $k$, let $K_r$ denote the set of all possible values $k_r$ for which there exists an interval containing $k$ points of $P$ and exactly $k_r$ red points; then $K_r$ forms a contiguous range of integers, and thus contains all numbers between $\min(K_r)$ and $\max(K_r)$.)

## 3.4 Subset sum for $k$-enclosing rectangle

A more challenging variant of the weighted problem is to find a rectangle enclosing exactly $k$ points with total weight exactly $W$ (similar to subset sum), or more generally, find an axis-aligned rectangle enclosing exactly $k$ points with total weight closest to $W$.

We use a different approach, using a 1D data structure that is static but can "plan for" a small number of deletions.

▶ **Lemma 18.** *Given a set $P$ of $n$ points in 1D and integers $k$ and $q$, we can build a static data structure, with $O(nq \log n)$ preprocessing time, that supports the following type of queries in $O(q \log n)$ time: for any subset $D \subset P$ of at most $q$ points and any weight $W$, find an interval containing $k$ points of $P - D$ with weight closest to $W$.*

**Proof.** See full version [10]. ◀

▶ **Theorem 19.** *Given $n$ points in the plane each with a real weight, and given a real number $W$ and an integer $k$, one can compute, in $O(n^{5/2} \log n)$ time, an axis-aligned rectangle enclosing exactly $k$ points with total weight closest to $W$.*

**Proof.** See full version [10]. ◀

We can further improve the running time for small $k$:

▶ **Theorem 20.** *Given $n$ points in the plane each with a real weight, and given a real number $W$ and an integer $k$, one can compute, in $O(n^2 \sqrt{k} \log k)$ time, an axis-aligned rectangle enclosing exactly $k$ points with total weight closest to $W$.*

**Proof.** See full version [10]. ◀

As an application, we can solve the following problem: given $n$ colored points in the plane with $d$ different colors, and integers $k_1, \dots, k_d$, with $k_1 + \cdots + k_d = k$, find an axis-aligned rectangle enclosing exactly $k_i$ points of the $i$th color. The problem was proposed by Barba et al. [4], who gave an $O(n^2 k)$-time algorithm. (It may be viewed as a geometric variant of the jumbled or histogram indexing problem for strings [11].) It is a special case of the problem from Theorem 19: we can give points with color $i$ a weight of $M^i$ for a sufficiently large $M$, e.g., $M = n+1$, and set the target to $W = \sum_{i=1}^{d} k_i M^i$. Since weights require $O(d \log n)$ bits, each addition has $O(d)$ cost, and so the running time becomes $O(dn^2 \sqrt{k} \log k)$. The weights can be reduced to $O(\log n)$ bits by randomized hashing (for example, by randomly selecting $M$ from $\{0, \dots, p-1\}$ and working with numbers modulo $p$ for an $O(\log n)$-bit prime $p$), since there are only polynomially (i.e., $O(n^4)$) many combinatorially different rectangles. This way, the running time can be reduced to $O(n^2 \sqrt{k} \log k)$ – this improves Barba et al.'s result.

## 3.5 Conditional lower bounds

We can prove that the smallest-perimeter $k$-enclosing axis-aligned rectangle problem do not have truly subquadratic (*i.e.*, $O(n^{2-\delta})$) algorithms, under the conjecture that (min,+)-convolution does not have a truly subquadratic algorithm. Our proof holds for the 3-sided version of the problem, which complements nicely with our upper bound in Section 3.1 using (min,+)-convolution.

We describe a reduction from the following decision problem, which Cygan et al. [14] showed does not have a truly subquadratic algorithm under the (min,+)-convolution conjecture.

▶ Problem 21. (min,+)-Convolution Decision. Given real numbers $a_0, \dots, a_{n-1}, b_0, \dots, b_{n-1}$, and $c_0, \dots, c_{n-1}$, decide whether

$$\forall \ell : \; c_\ell \le \min_{i+j=\ell} (a_i + b_j).$$

▶ **Theorem 22.** *If there is a $T(n)$-time algorithm for computing the smallest-perimeter/area axis-aligned rectangle enclosing $k$ points for a given set of $n$ points in the plane and a given number $k$ (with or without the constraint that the bottom edge lies on the $x$-axis), then there is an $O(T(O(n))$-time algorithm for Problem 21.*

**Proof.** See full version [10]. ◀

A similar reduction holds for the minimum-weight $k$-enclosing rectangle problem from Theorem 17:

▶ **Theorem 23.** *If there is a $T(n)$-time algorithm for computing the minimum-weight axis-aligned rectangle enclosing $k$ points for a given set of $n$ weighted points in the plane and number $k$ (with or without the constraint that the bottom edge lies on the $x$-axis), then there is an $O(T(O(n))$-time algorithm for Problem 21.*

**Proof.** See full version [10]. ◀

A near-quadratic conditional lower bound for the minimum-weight rectangle problem without the $k$ constraint was given by Backurs et al. [3] (under a different "popular" conjecture about the complexity of maximum-weight clique).

We can similarly prove that the subset-sum variant of the $k$-enclosing rectangle problem from Theorem 19 (or its 3-sided variant) does not have truly subquadratic algorithms, under the conjecture that the *convolution-3SUM* problem (given real numbers $a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}, c_0, \ldots, c_{n-1}$, decide whether $c_\ell = a_i + b_{\ell-i}$ for some $i$ and $\ell$) does not have a truly subquadratic algorithm (which is known to be true under the conjecture that 3SUM for integers does not have a truly subquadratic algorithm [25]).

## References

1 Alok Aggarwal, Hiroshi Imai, Naoki Katoh, and Subhash Suri. Finding $k$ points with minimum diameter and related problems. *J. Algorithms*, 12(1):38–56, 1991. `doi:10.1016/0196-6774(91)90022-Q`.

2 Rossen Atanassov, Prosenjit Bose, Mathieu Couture, Anil Maheshwari, Pat Morin, Michel Paquette, Michiel H. M. Smid, and Stefanie Wuhrer. Algorithms for optimal outlier removal. *J. Discrete Algorithms*, 7(2):239–248, 2009. `doi:10.1016/j.jda.2008.12.002`.

3 Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. In *Proc. 43rd Int. Colloq. Automata Lang. Prog.* (ICALP), pages 81:1–81:13, 2016.

4 Luis Barba, Stephane Durocher, Robert Fraser, Ferran Hurtado, Saeed Mehrabi, Debajyoti Mondal, Jason Morrison, Matthew Skala, and Mohammad Abdul Wahid. On $k$-enclosing objects in a coloured point set. In *Proc. 25th Canad. Conf. Comput. Geom.* (CCCG), 2013. URL: `http://cccg.ca/proceedings/2013/papers/paper_35.pdf`.

5 Jérémy Barbay, Timothy M. Chan, Gonzalo Navarro, and Pablo Pérez-Lantero. Maximum-weight planar boxes in $O(n^2)$ time (and better). *Inform. Process. Lett.*, 114(8):437–445, 2014. `doi:10.1016/j.ipl.2014.03.007`.

6 Marshall W. Bern. Approximate closest-point queries in high dimensions. *Inf. Process. Lett.*, 45(2):95–99, 1993. `doi:10.1016/0020-0190(93)90222-U`.

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

8 Timothy M. Chan. Approximate nearest neighbor queries revisited. *Discrete & Computational Geometry*, 20(3):359–373, 1998. `doi:10.1007/PL00009390`.

9 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22(4):547–567, 1999. `doi:10.1007/PL00009478`.

10 Timothy M. Chan and Sariel Har-Peled. Smallest $k$-enclosing rectangle revisited. *CoRR*, abs/1903.06785, 2019. `arXiv:1903.06785`.

11 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proc. 47th Annu. ACM Sympos. Theory Comput.* (STOC), pages 31–40, 2015. `doi:10.1145/2746539.2746568`.

12 Timothy M. Chan and Mihai Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proc. 21st ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 161–173, 2010. `doi:10.1137/1.9781611973075.15`.

**13** Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov–Smolensky. In *Proc. 27th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 1246–1255, 2016. `doi:10.1137/1.9781611974331.ch87`.

**14** Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to $(\min, +)$-convolution. In *Proc. 44th Int. Colloq. Automata Lang. Prog.* (ICALP), volume 80 of *LIPIcs*, pages 22:1–22:15, 2017. `doi:10.4230/LIPIcs.ICALP.2017.22`.

**15** Sandip Das, Partha P. Goswami, and Subhas C. Nandy. Smallest $k$-point enclosing rectangle and square of arbitrary orientation. *Inform. Process. Lett.*, 94(6):259–266, 2005. `doi:10.1016/j.ipl.2005.02.013`.

**16** Amitava Datta, Hans-Peter Lenhof, Christian Schwarz, and Michiel H. M. Smid. Static and dynamic algorithms for $k$-point clustering problems. *J. Algorithms*, 19(3):474–503, 1995. `doi:10.1006/jagm.1995.1048`.

**17** Mark de Berg, Sergio Cabello, Otfried Cheong, David Eppstein, and Christian Knauer. Covering many points with a small-area box. *CoRR*, abs/1612.02149, 2016. `arXiv:1612.02149`.

**18** David Eppstein and Jeff Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discrete Comput. Geom.*, 11:321–350, 1994. URL: `http://jeffe.cs.illinois.edu/pubs/small.html`.

**19** Harold N. Gabow and Robert Endre Tarjan. A linear-time algorithm for a special case of disjoint set union. *J. Comput. Sys. Sci.*, 30(2):209–221, 1985. `doi:10.1016/0022-0000(85)90014-5`.

**20** Sariel Har-Peled and Soham Mazumdar. Fast algorithms for computing the smallest $k$-enclosing disc. *Algorithmica*, 41(3):147–157, 2005. URL: `https://sarielhp.org/p/03/min_disk/`.

**21** Sariel Har-Peled and Micha Sharir. Relative $(p, \varepsilon)$-approximations in geometry. *Discrete Comput. Geom.*, 45(3):462–496, 2011. `doi:10.1007/s00454-010-9248-1`.

**22** Allan Grønlund Jørgensen and Kasper Green Larsen. Range selection and median: Tight cell probe lower bounds and adaptive data structures. In *Proc. 22nd ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 805–813, 2011. `doi:10.1137/1.9781611973082.63`.

**23** Haim Kaplan, Sasanka Roy, and Micha Sharir. Finding axis-parallel rectangles of fixed perimeter or area containing the largest number of points. In *Proc. 26th Annu. Euro. Sympos. Alg.* (ESA), volume 87 of *LIPIcs*, pages 52:1–52:13, 2017. `doi:10.4230/LIPIcs.ESA.2017.52`.

**24** Jiří Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2:169–186, 1992.

**25** Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proc. 42nd Annu. ACM Sympos. Theory Comput.* (STOC), pages 603–610, 2010. `doi:10.1145/1806689.1806772`.

**26** Michael Segal and Klara Kedem. Enclosing $k$ points in the smallest axis parallel rectangle. *Inform. Process. Lett.*, 65(2):95–99, 1998. `doi:10.1016/S0020-0190(97)00212-3`.

**27** Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. Assoc. Comput. Mach.*, 22(2):215–225, 1975. `doi:10.1145/321879.321884`.

**28** Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proc. 46th Annu. ACM Sympos. Theory Comput.* (STOC), pages 664–673, 2014. `doi:10.1145/2591796.2591811`.