




Constrained Two-Line Center Problems

Taehoon Ahn  

Graduate School of Artificial Intelligence,
Pohang University of Science and Technology, Republic of Korea

Sang Won Bae  

Division of Artificial Intelligence and Computer Science,
Kyonggi University, Suwon, Republic of Korea

Abstract

Given a set P of n points in the plane, the two-line center problem asks to find two lines that minimize the maximum distance from each point in P to its closer one of the two resulting lines. The currently best algorithm for the problem takes $O(n^2 \log^2 n)$ time by Jaromczyk and Kowaluk in 1995. In this paper, we present faster algorithms for three variants of the two-line center problem in which the orientations of the resulting lines are constrained. Specifically, our algorithms solve the problem in $O(n \log n)$ time when the orientations of both lines are fixed; in $O(n \log^3 n)$ time when the orientation of one line is fixed; and in $O(n^2 \alpha(n) \log n)$ time when the angle between the two lines is fixed, where $\alpha(n)$ denotes the inverse Ackermann function.

2012 ACM Subject Classification Theory of computation \rightarrow Computational geometry

Keywords and phrases two-line center problem, geometric location problem, geometric optimization

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.5

Related Version *Full Version:* <https://arxiv.org/abs/2409.13304> [7]

Funding *Taehoon Ahn:* Supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2019-0-01906).

Sang Won Bae: Supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00251168).

1 Introduction

Given a set P of n points in the plane \mathbb{R}^2 , the *two-line center problem* asks to find two lines that minimize the maximum distance from each point in P to its closer one of the two resulting lines. In 1991, Agarwal and Sharir [4] presented the first subcubic $O(n^2 \log^5 n)$ -time algorithm for the two-line center problem, in which they solved the decision version in $O(n^2 \log^3 n)$ time using their machinery [3] to maintain the width of a point set under a prescribed sequence of changes and then to apply the parametric search technique. (See also its full version [5].) In 1995, Jaromczyk and Kowaluk [24] presented an $O(n^2 \log^2 n)$ -time algorithm and also discussed an $O(n^2 \log n)$ -time decision algorithm. Glozman et al. [21, 22] exhibited how any D -time decision algorithm for the two-line center problem can be converted to an optimization algorithm of $O(n^2 \log n + D \log n)$ time using sorted matrices. Later, Katz and Sharir [26] introduced an expander-based approach and showed how to solve the problem in $O(n^2 \log^3 n + D \log n)$ time. There was no significant progress since then and $O(n^2 \log^2 n)$ still remains the best known upper bound [22, 24].

This paper addresses constrained variants of the two-line center problem, and aims to provide efficient algorithms for the constrained problems, particularly faster than $O(n^2 \log^2 n)$ time, and to provide new observations and algorithmic techniques for any future breakthrough on the problem. The currently fastest algorithm by Jaromczyk and Kowaluk [24] indeed considers several constrained problems, tackled by different methods. Though not having explicitly mentioned in [24], their approach yields an $O(n \log^2 n)$ -time algorithm when a fixed



© Taehoon Ahn and Sang Won Bae;

licensed under Creative Commons License CC-BY 4.0

35th International Symposium on Algorithms and Computation (ISAAC 2024).

Editors: Julián Mestre and Anthony Wirth; Article No. 5; pp. 5:1–5:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

point in P should be the farthest to the resulting lines, after an $O(n^2)$ -time preprocessing. Recently, Bae [10] presented an $O(n^2)$ -time algorithm for the two-*parallel*-line center problem, in which the two resulting lines are supposed to be parallel.

In this paper, we solve three variants of the two-line center problem, constrained about the orientations of the resulting two lines. Following summarizes our results and approaches:

- (1) (*Two fixed orientations*) Given two orientations θ and ϕ , we present an $O(n \log n)$ -time algorithm that solves the two-line center problem in which the two resulting lines are constrained to have orientations θ and ϕ . If the input points P are given as a sorted list in one of the specified orientations, then the running time can be reduced to $O(n)$.
- (2) (*One fixed orientation*) Given an orientation ϕ , we present an $O(n \log^3 n)$ -time algorithm that solves the two-line center problem in which one of the resulting lines is constrained to have orientation ϕ . We first devise an $O(n \log^2 n)$ -time decision algorithm for this constrained problem using the data structure by Agarwal and Sharir [3]. In spite of having such an efficient decision algorithm, it is not immediate to achieve a sub-quadratic time optimization algorithm by applying known techniques; as introduced above, all known techniques for the two-line center problem require at least quadratic-time additional overhead [5, 22, 26]. To overcome this difficulty, we use our decision algorithm as a subroutine to find an interval narrow enough to reduce the possible number of candidate configurations to $O(n)$ and apply the dynamic width structure by Chan [13].
- (3) (*Fixed angle of intersection*) Given a real β , we present an $O(n^2 \alpha(n) \log n)$ -time algorithm that solves the two-line center problem in which the two resulting lines are constrained to make angle β , where $\alpha(n)$ denotes the inverse Ackermann function. As in the second problem, we start by presenting a decision algorithm and apply the known technique [22] to obtain a favorably narrow interval that contains the optimal width value. We then consider a sweeping process in which we rotate a strip of variable width within the interval, and prove that it suffices to find an optimal solution by simulating the process.

To our best knowledge, the three constrained problems have not been considered in the literature. Note that the two-parallel-line center problem studied in [10] is a more constrained variant of our problems: In the first problem (of two fixed orientations), the special case of $\theta = \phi$ can be solved in $O(n)$ time, and the third problem (of fixed angle) for $\beta = 0$ indeed asks to find a two-parallel-line center, which can be solved in $O(n^2)$ time [10].

Due to space limit, most proofs are omitted, but can be found in the full version [7].

Related work

The two-line center problem is a special case of the k -line center problem for $k \geq 1$. For $k = 1$, known as the *width* problem, one can solve the problem in $O(n \log n)$ time [29], or in $O(n)$ time if the convex hull of P is given [31]. In three dimensions, the width of n points in \mathbb{R}^3 can be computed in $O(n^{3/2+\epsilon})$ expected time by Agarwal and Sharir [6]. In higher dimensions $d \geq 4$, Chan [12] showed how to compute the width in $O(n^{\lceil d/2 \rceil})$ time. In the plane \mathbb{R}^2 , the k -line center problem is known to be NP-hard when k is part of the input [28], while efficient approximation algorithms are known [1, 2]. Agarwal et al. [2] presented an efficient approximation algorithm. Exact algorithms for $k \leq 2$ are presented as aforementioned, while any nontrivial exact algorithm for $k \geq 3$ is, however, unknown. An efficient $(1 + \epsilon)$ -approximation algorithm for $k = 2$ is presented by Agarwal et al. [1]. Very recently, several constrained variants of the k -line center problem and its generalization in high dimensions have been considered. Das et al. [16] presented an approximation algorithm for the k -line center problem where the resulting lines are constrained to be axis-parallel. Chung et al. [15] considered a variant of the parallel k -line center problem. Ahn et al. [8] presented first algorithms for the problem of finding two parallel *slabs* in \mathbb{R}^d for $d \geq 3$.

Not being restricted to the line center problems, there have been an enormous amount of results on constrained variants of those problems of finding optimal locations of one or more geometric shapes enclosing input objects. Such results on constrained problems usually provided more efficient solutions than those for the original (unconstrained) problems or played important roles as stepping stones to later breakthroughs. Constrained two-square problems [25] and the problem of covering points by two disjoint rectangles [27] are such examples.

Some preliminaries

A *strip* σ is the closed region between two parallel lines and its *width* is the distance between the two lines. A pair of two strips will be called a *two-strip* and the width of a two-strip mean the larger width of its two members. Note that the two-line center problem is equivalent to the problem of finding a two-strip of minimum width that enclose given points. The *orientation* of a line is a real value $\theta \in [0, \pi)$ such that θ is the angle swept from a horizontal line in counterclockwise direction to the line. Similarly, a strip is said to have an orientation θ when its bounding lines are in orientation θ . For any set P of points and orientation $\theta \in [0, \pi)$, we denote by $\sigma_\theta(P)$ the minimum-width strip in orientation θ that encloses P . We denote $\text{width}_\theta(P) := \text{width}(\sigma_\theta(P))$. The *width* of point set P , denoted by $\text{width}(P)$, is the smallest width of a strip that encloses P .

2 Two fixed orientations

In this section, we consider the first constrained problem where the orientations of two line centers should given values θ and ϕ . We assume that $\phi = 0$ without loss of generality.

We start by sorting the n points in P in the nondecreasing order of y -coordinates and let $p_1, \dots, p_n \in P$ be in this order. For $0 \leq i \leq n$, let $P_i := \{p_1, \dots, p_i\}$ and $\bar{P}_i := P \setminus P_i = \{p_{i+1}, \dots, p_n\}$. It is straightforward in $O(n)$ time to incrementally construct the strips $\sigma_\theta(P_i)$ and $\sigma_\theta(\bar{P}_i)$ in orientation θ for all $0 \leq i \leq n$. We then observe the following.

► **Lemma 1.** *Given P as a sorted list as above, P can be processed in $O(n)$ -time so that $\sigma_\theta(P_i \cup \bar{P}_j)$ can be answered in $O(1)$ time for any query pair (i, j) of indices.*

Consider any minimum-width two-strip (σ_1, σ_2) enclosing P such that its orientations are 0 and θ , respectively. Observe that σ_1 includes a contiguous sequence p_{i+1}, \dots, p_j of points in P for some indices $0 \leq i \leq j \leq n$, while σ_2 covers the points in $P_i \cup \bar{P}_j$. Hence, the problem can be solved by searching for $O(n^2)$ possible bipartitions of P , namely, $(P_i \cup \bar{P}_j, P \setminus (P_i \cup \bar{P}_j))$ for $0 \leq i \leq j \leq n$, and evaluating the widths of the two strips enclosing each part of desired bipartitions.

Let $w_1(i, j) := \text{width}_0(\{p_{i+1}, \dots, p_j\})$ be the width of the smallest horizontal strip enclosing $j - i$ points $p_{i+1}, \dots, p_j \in P$, that is, the difference of the y -coordinates of p_{i+1} and p_j . Let $w_2(i, j) := \text{width}_\theta(P_i \cup \bar{P}_j)$ be the width of the smallest strip in orientation θ enclosing $P_i \cup \bar{P}_j = \{p_1, \dots, p_i, p_{j+1}, \dots, p_n\}$. Define $w(i, j) := \max\{w_1(i, j), w_2(i, j)\}$. Our task is to minimize $w(i, j)$ over all $0 \leq i \leq j \leq n$. This can be done by evaluating $w_1(i, j)$ and $w_2(i, j)$ for at most $4n$ pairs (i, j) of indices due to the monotonicity of w_1 and w_2 . More precisely, observe that

$$w_1(i, j) \leq w_1(i, j+1) \quad \text{and} \quad w_1(i, j) \leq w_1(i-1, j),$$

while

$$w_2(i, j) \geq w_2(i, j+1) \quad \text{and} \quad w_2(i, j) \geq w_2(i-1, j)$$

by definition. Hence, our algorithm initially sets $i = j = 0$ and repeatedly increases j by one until it holds that $w_1(0, j) \leq w_2(0, j)$ and $w_1(0, j + 1) \geq w_2(0, j + 1)$. Then for each $i = 1, \dots, n$ in this order, it repeatedly increases j by one until it holds that $w_1(i, j) \leq w_2(i, j)$ and $w_1(i, j + 1) \geq w_2(i, j + 1)$ for the current i . This way, our algorithm probes at most $4n$ pairs (i, j) . For a given pair (i, j) , in $O(1)$ time we can evaluate $w_1(i, j)$ by definition and $w_2(i, j)$ by Lemma 1. We thus conclude the following.

► **Theorem 2.** *Given a set P of n points and two orientations $\theta, \phi \in [0, \pi)$, the two-line center problem where the resulting lines have orientations θ and ϕ can be computed in $O(n \log n)$ time, or in $O(n)$ time, provided P is sorted in orientation either θ or ϕ .*

3 One fixed orientation

In this section, we solve the second constrained problem: given a fixed orientation ϕ , find two strips of minimum width whose union encloses P such that one of the two strips is in orientation ϕ . Throughout this section, a pair of two such strips (σ_1, σ_2) , where σ_1 is in orientation ϕ , will be simply called a *constrained two-strip*, and assume that $\phi = 0$.

To find a constrained two-strip of minimum width enclosing P , one could make use of a data structure for the dynamic width maintenance [13, 17]. Observe that there are $O(n^2)$ possible bipartitions of P induced by a constrained two-strip (σ_1, σ_2) since there are $O(n^2)$ distinct subsets of P that can be enclosed by a horizontal strip σ_1 . This approach, however, does not seem to avoid a quadratic running time, since the point set we would maintain undergoes $\Theta(n^2)$ updates. Another common approach is to apply known techniques, such as the parametric search [5], the expander-based method [26], or the one based on a sorted matrix [22]. These techniques also require at least quadratic time overhead.

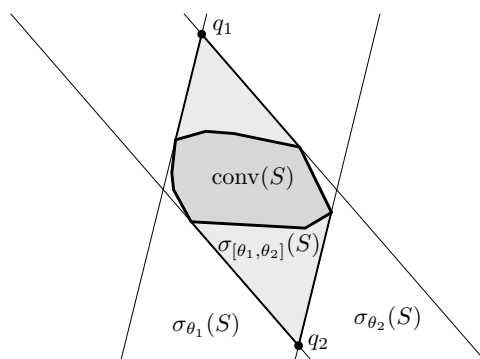
Despite this difficulty, we present a near-linear $O(n \log^3 n)$ -time algorithm based on our $O(n \log^2 n)$ -time decision algorithm and Chan's structure of dynamic width maintenance [13]. Note that the decision problem can be solved in $O(n \log^3 n)$ time by a direct application of the machinery of Agarwal and Sharir [3]. In the following, we show how to shave another logarithmic factor, while still using the data structure of Agarwal and Sharir.

3.1 Data structures for dynamic width decision and maintenance

Agarwal and Sharir [3] showed that in $O(n \log^3 n)$ time the *offline dynamic width decision problem* can be solved: Given a parameter $\omega > 0$ and a sequence of n insert/delete operations on a set S of points, initially consisting of at most n points, determine whether there is any moment such that $\text{width}(S) \leq \omega$ during the n updates on S . Their algorithm builds a segment tree based on the life-spans of the points, that is, the time intervals in which each point is a member of S , and traverse it with a secondary data structure \mathcal{D} that maintains necessary information about the width of the current S using linear space.

The data structure \mathcal{D} consists of two balanced binary search trees that store the edges of the convex hull $\text{conv}(S)$, ordered by their orientations, and maintains a certain collection of invariants, which suffice to decide in $O(1)$ time whether or not $\text{width}(S) \leq \omega$ for the current set S . Agarwal and Sharir showed that how to update \mathcal{D} per insertion of a point into S , and also how to undo the latest insertion, recovering the structure \mathcal{D} to the status before the latest insertion. Summarizing, we have:

► **Lemma 3** (Agarwal and Sharir [3]). *Suppose the data structure \mathcal{D} with a parameter ω has been built on a set S of n points. Then, we can decide whether or not $\text{width}(S) \leq \omega$ in $O(1)$ time, and \mathcal{D} can be maintained in $O(\log^2 n)$ worst-case time for the following updates: inserting a point to S and undoing the latest insertion.*



■ **Figure 1** Illustration of $\sigma_{[\theta_1, \theta_2]}(S) = \text{conv}(S \cup \{q_1, q_2\})$ (shaded in light gray) when $\theta_1 < \theta_2$.

Chan [13] presented how to exactly maintain $\text{width}(S)$ over fully online updates on S . Its amortized time per update is $O(\sqrt{n} \log^3 n)$, based on the following data structure.

► **Lemma 4** (Chan [13]). *There is a data structure \mathcal{W} for a set S of n points that supports deletions of points from S and queries of the following kind: given a query point set Q , report $\text{width}(S \cup Q)$. The total preprocessing and deletion time is $O(n \log^3 n)$ and the query time is $O(|Q| \log^3(n + |Q|))$. The space required for maintaining the structure \mathcal{W} is $O(n \log n)$.*

Though Chan did not discuss the space requirement for his method, it is not difficult to see it as stated above from construction [13]. Note that if the online updates are deletions only, then this results in an $O(n \log^3 n)$ -time algorithm that maintains the exact width.

3.2 Orientation-constrained width

Let S be a set of points, and let $\theta_1 \leq \theta_2$ be two orientations. Define the $[\theta_1, \theta_2]$ -constrained width of S to be

$$\text{width}_{[\theta_1, \theta_2]}(S) := \min_{\theta \in [\theta_1, \theta_2]} \text{width}_{\theta}(S).$$

Note that $\text{width}(S) = \text{width}_{[0, \pi]}(S)$ and $\text{width}_{[\theta, \theta]}(S) = \text{width}_{\theta}(S) = \text{width}(\sigma_{\theta}(S))$. Also, define

$$\sigma_{[\theta_1, \theta_2]}(S) := \bigcap_{\theta \in [\theta_1, \theta_2]} \sigma_{\theta}(S).$$

Note that $\sigma_{[\theta_1, \theta_2]}(S)$ is the convex hull of S and two more points from the boundary of the intersection of two strips $\sigma_{\theta_1}(S)$ and $\sigma_{\theta_2}(S)$. See Figure 1 for an illustration.

► **Lemma 5.** *For any finite set S of points, it holds that $\text{width}_{[\theta_1, \theta_2]}(S) = \text{width}(\sigma_{[\theta_1, \theta_2]}(S))$.*

As will be seen later, we are also interested in *orientation-constrained width decision queries*. More precisely, we are given a query interval $[\theta_1, \theta_2] \subseteq [0, \pi)$ of orientations and want to decide whether there exists $\theta \in [\theta_1, \theta_2]$ such that the width of $\sigma_{\theta}(S)$ is at most ω or, equivalently, whether $\text{width}_{[\theta_1, \theta_2]}(S) \leq \omega$. It turns out that the structure \mathcal{D} of Lemma 3 by Agarwal and Sharir is helpful for this type of queries as well, with the aid of Lemma 5.

► **Lemma 6.** *Provided the data structure \mathcal{D} on a point set S of n points with parameter ω is available, an orientation-constrained width decision query on S for width ω can be answered in $O(\log^2 n)$ worst-case time using $O(\log n)$ additional space.*

5:6 **Constrained Two-Line Center Problems**

Now, consider two sets S_1 and S_2 of points in the plane that can be separated by a line, that is, $\text{conv}(S_1) \cap \text{conv}(S_2) = \emptyset$. Then, there are exactly two outer common tangent lines ℓ_1 and ℓ_2 . Let $\theta_1 \leq \theta_2$ be the orientations of ℓ_1 and ℓ_2 . We say that S_1 dominates S_2 if $\sigma_{[\theta_1, \theta_2]}(S_2) \subseteq \sigma_{[\theta_1, \theta_2]}(S_1)$. By construction, note that either S_1 or S_2 dominates the other. We also mean by the distance between two convex, compact sets A and B , denoted by $d(A, B)$, the minimum length of translation vectors τ such that A and $B + \tau$ have a common point.

► **Lemma 7.** *With the above notations, suppose that S_1 dominates S_2 . Then, it holds that:*

- (i) $\text{width}_{[\theta_1, \theta_2]}(S_1 \cup S_2) = \text{width}_{[\theta_1, \theta_2]}(S_1)$.
- (ii) *If $\text{width}(S_1 \cup S_2) < d(\text{conv}(S_1), \text{conv}(S_2))$, then $\text{width}(S_1 \cup S_2) = \text{width}_{[\theta_1, \theta_2]}(S_1)$.*

Proof. Since S_1 dominates S_2 , we have

$$S_2 \subseteq \sigma_{[\theta_1, \theta_2]}(S_2) \subseteq \sigma_{[\theta_1, \theta_2]}(S_1).$$

Lemma 5 implies that

$$\sigma_{[\theta_1, \theta_2]}(S_1 \cup S_2) = \sigma_{[\theta_1, \theta_2]}(S_1),$$

so the first statement (i) follows. See Figure 2(a).

Suppose $\text{width}(S_1 \cup S_2) < d(\text{conv}(S_1), \text{conv}(S_2))$. Let $\sigma^* = \sigma_{\theta^*}(S_1 \cup S_2)$ be a minimum-width strip enclosing $S_1 \cup S_2$ whose orientation is θ^* . We claim that $\theta^* \in [\theta_1, \theta_2]$. If this claim is true, then, by definition, we have

$$\sigma_{[\theta_1, \theta_2]}(S_1) = \sigma_{[\theta_1, \theta_2]}(S_1 \cup S_2) \subseteq \sigma_{\theta^*}(S_1 \cup S_2) = \sigma^*,$$

so

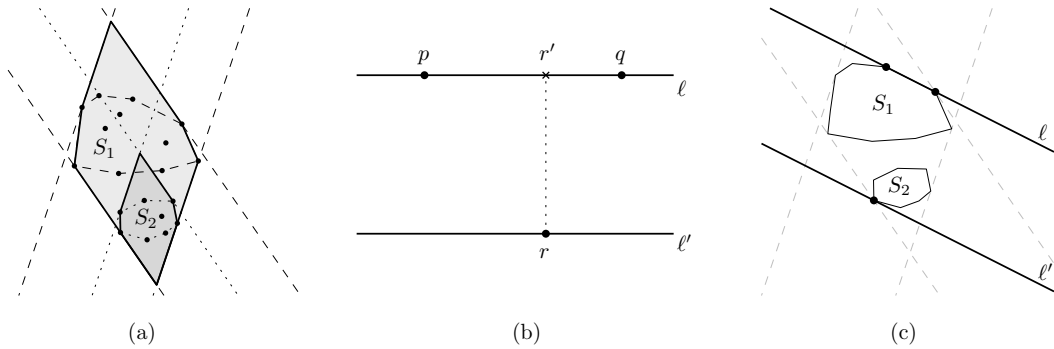
$$\text{width}(\sigma_{[\theta_1, \theta_2]}(S_1)) \leq \text{width}(S_1 \cup S_2),$$

on one hand. On the other hand, since $S_1 \cup S_2$ is a subset of $\sigma_{[\theta_1, \theta_2]}(S_1 \cup S_2) = \sigma_{[\theta_1, \theta_2]}(S_1)$, we also have

$$\text{width}(\sigma_{[\theta_1, \theta_2]}(S_1)) \geq \text{width}(S_1 \cup S_2),$$

and the second statement (ii) is thus proved.

Hence, we are done by proving the claim that $\theta^* \in [\theta_1, \theta_2]$. As σ^* is minimal among those enclosing $S_1 \cup S_2$, its boundary contains three points p, q, r from $S_1 \cup S_2$ such that p and q lie on a common bounding line ℓ of σ^* , r lies on the other bounding line ℓ' , and the perpendicular foot r' of r to ℓ lies in between p and q . See Figure 2(b) for an illustration.



■ **Figure 2** Illustrations to the proof of Lemma 7.

We first exclude the possibility that both p and q belong to a common set, S_1 or S_2 , and r to the other. Suppose for a contradiction that, say, $p, q \in S_1$ and $r \in S_2$. By our assumption that $d(\text{conv}(S_1), \text{conv}(S_2)) > \text{width}(S_1 \cup S_2)$, the distance d from r to its perpendicular foot r' is strictly larger than $\text{width}(S_1 \cup S_2)$, while, however, the distance d is also the width of σ^* , a contradiction to the assumption that the width of σ^* determines $\text{width}(S_1 \cup S_2)$.

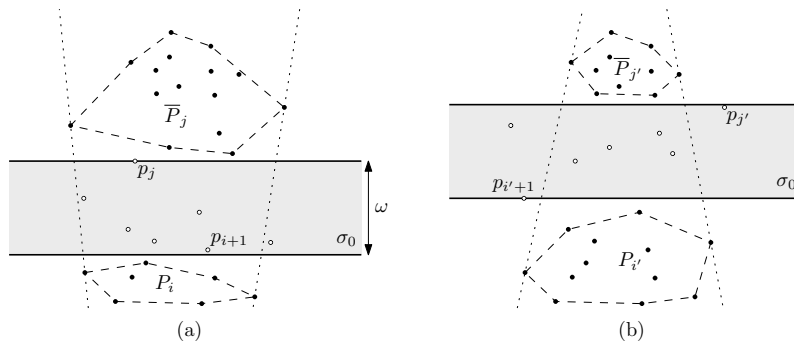
Now, suppose that $\theta^* \notin [\theta_1, \theta_2]$. Then, observe that both bounding lines ℓ and ℓ' of σ^* cannot intersect a common set, S_1 or S_2 ; that is, $\ell \cap S_i \neq \emptyset$ if and only if $\ell' \cap S_i = \emptyset$, for $i = 1, 2$. (See Figure 2(c).) This implies that $p, q \in \ell$ belong to one common set, S_1 or S_2 , and $r \in \ell'$ belongs to the other set, which is forbidden by the above argument. Thus, we have $\theta \in [\theta_1, \theta_2]$, and the claim is true. ◀

3.3 Decision algorithm

We describe our decision algorithm for a given parameter $\omega > 0$. The points $p_1, \dots, p_n \in P$ are assumed to be sorted in the y -coordinates, and we let $P_i = \{p_1, \dots, p_i\}$ and $\bar{P}_i = P \setminus P_i$. Consider a horizontal strip σ_0 of width ω and sweep the plane by translating σ_0 upwards from below. At any moment of this sweeping process, the points $P \setminus \sigma_0$ outside of σ_0 is partitioned into P_i and \bar{P}_j for some $0 \leq i \leq j \leq n$ such that all points of P_i lies below σ_0 and all points of \bar{P}_j lies above σ_0 . Note that the indices i and j representing such a separation by σ_0 do not decrease during the process, and it always holds that $d(\text{conv}(P_i), \text{conv}(\bar{P}_j)) > \omega$. Also, by this monotonicity of i and j , observe that \bar{P}_j dominates P_i from the beginning until some moment and P_i dominates \bar{P}_j from that moment to the end. See Figure 3.

We maintain convex hulls, $\text{conv}(P_i)$ and $\text{conv}(\bar{P}_j)$, and the data structure \mathcal{D} with parameter ω on set P_i . Maintaining the convex hulls $\text{conv}(P_i)$ and $\text{conv}(\bar{P}_j)$ can be done in $O(n \log n)$ total time [11, 23]; in our case, updates on P_i and \bar{P}_j are offline. Initially, we have $i = j = 0$, so $\text{conv}(P_i) = \emptyset$, $\text{conv}(\bar{P}_j) = \text{conv}(P)$, and \mathcal{D} is initialized for an empty set P_0 .

In the main loop of our decision algorithm, as σ_0 moves upwards, we do nothing until P_i becomes dominating \bar{P}_j while we maintain the data structure \mathcal{D} by inserting relevant points. For each pair (i, j) such that P_i dominates \bar{P}_j , we decide whether $\text{width}(P_i \cup \bar{P}_j) \leq \omega$ or not. If it is the case, we stop the algorithm and report YES; otherwise, we proceed the algorithm. The decision is made as follows: We first compute the outer common tangents of $\text{conv}(P_i)$ and $\text{conv}(\bar{P}_j)$ in $O(\log^2 n)$ time using $\text{conv}(P_i)$ and $\text{conv}(\bar{P}_j)$. (See Figure 3.) Let $\theta_1 \leq \theta_2$ be the orientations of the two common tangents. We then perform an orientation-constrained width decision query on P_i with query interval $[\theta_1, \theta_2]$. This can be done in $O(\log^2 n)$ time by Lemma 6 using \mathcal{D} . If the answer to this query is positive, then we conclude that $\text{width}(P_i \cup \bar{P}_j) \leq \omega$; otherwise, we conclude that $\text{width}(P_i \cup \bar{P}_j) > \omega$. The correctness of this decision is guaranteed by the following lemma, which is a direct application of Lemma 7.



■ **Figure 3** Snapshots of the sweeping process: (a) \bar{P}_j dominates P_i and (b) $P_{i'}$ dominates $\bar{P}_{j'}$.

► **Lemma 8.** *For (i, j) such that P_i dominates \overline{P}_j and $d(\text{conv}(P_i), \text{conv}(\overline{P}_j)) > \omega$, we have $\text{width}(P_i \cup \overline{P}_j) \leq \omega$ if and only if $\text{width}_{[\theta_1, \theta_2]}(P_i) \leq \omega$.*

This way, we check all the pairs (i, j) such that P_i dominates \overline{P}_j during the sweeping process. The other pairs (i, j) such that \overline{P}_j dominates P_i can be handled in a symmetric way by moving the horizontal strip σ_0 downwards. Therefore, we conclude the following.

► **Theorem 9.** *Given a set P of n points, $\phi \in [0, \pi)$, and $\omega > 0$, we can decide in $O(n \log^2 n)$ time and $O(n)$ space whether there is a constrained two-strip of width ω .*

3.4 Optimization algorithm

Let w^* be our target optimal width, that is, the minimum width of a constrained two-strip enclosing P . As above, suppose that $p_1, \dots, p_n \in P$ are sorted in their y -coordinates. Let W_1 be the set of all differences of y -coordinates between two points in P . Since W_1 can be represented by a sorted matrix [19], we can find two consecutive values $w_0, w_1 \in W_1$ such that $w_0 < w^* \leq w_1$ in $O(n \log^3 n)$ time using our decision algorithm (Theorem 9) and an efficient selection algorithm for a sorted matrix [20].

At this stage, observe that, for any $w_0 < w < w_1$, the sequence of changes on the sets P_i and \overline{P}_j of points below and above σ_0 is the same as the horizontal strip σ_0 of width w moves upwards. Let X be the set of those pairs (i, j) of indices such that P_i and \overline{P}_j appear as the sets of points below and above σ_0 , respectively.

► **Lemma 10.** *It holds that*

$$w^* = \min\{w_1, \min_{(i,j) \in X} \{\text{width}(P_i \cup \overline{P}_j)\}\}.$$

Thus, we are done by computing the width of $P_i \cup \overline{P}_j$ for $(i, j) \in X$. Even better, to compute w^* and an optimal two-strip, it suffices to evaluate the exact value of $\text{width}(P_i \cup \overline{P}_j)$ only for those $(i, j) \in X$ such that $\text{width}(P_i \cup \overline{P}_j) < w_1$. Let W be the set of values of $\text{width}(P_i \cup \overline{P}_j)$ that are less than w_1 . Below, we show how to compute the set W .

For the purpose, we take any value w with $w_0 < w < w_1$ and simulate the translational sweeping process with the horizontal strip σ_0 of width w in a similar way as done for the decision algorithm. Here, we sweep the plane by moving σ_0 downwards from above. We initialize the data structure \mathcal{D} with parameter $\omega = w_1$ on point set $P_n = P$ by inserting n points p_1, \dots, p_n in this order, and maintain \mathcal{D} by deleting points in the reversed order to represent P_i , as σ_0 moves downwards. In addition, we initialize the structure \mathcal{W} of Lemma 4 for point set $P_n = P$, and maintain it to store P_i by deleting points in the same order. We also maintain the convex hulls $\text{conv}(P_i)$ and $\text{conv}(\overline{P}_j)$.

During the sweeping process, we only handle those $(i, j) \in X$ such that P_i dominates \overline{P}_j , so we stop the process as soon as \overline{P}_j dominates P_i . (Those $(i, j) \in X$ such that \overline{P}_j dominates P_i can be handled in a symmetric way by moving σ_0 upwards.) Consider such a pair (i, j) . Our goal is to compute $\text{width}(P_i \cup \overline{P}_j)$ only when it is less than w_1 . Note that $d(\text{conv}(P_i), \text{conv}(\overline{P}_j)) \geq w_1$. We compute the outer common tangents of $\text{conv}(P_i)$ and $\text{conv}(\overline{P}_j)$ and let $\theta_1 \leq \theta_2$ be their orientations. As in the decision algorithm, we test whether or not $\text{width}_{[\theta_1, \theta_2]}(P_i) \leq w_1$ by Lemma 6 using \mathcal{D} . Lemma 7 implies the following.

► **Lemma 11.** *Provided P_i dominates \overline{P}_j , $\text{width}(P_i \cup \overline{P}_j) \geq w_1$ if $\text{width}_{[\theta_1, \theta_2]}(P_i) > w_1$.*

If it turns out that $\text{width}_{[\theta_1, \theta_2]}(P_i) > w_1$, then we can discard the pair (i, j) and proceed the algorithm by Lemma 11. Otherwise, we compute the exact value of $\text{width}_{[\theta_1, \theta_2]}(P_i)$. If $\theta_1 = \theta_2$, this can be done in $O(\log n)$ time using convex hulls $\text{conv}(P_i)$ and $\text{conv}(\overline{P}_j)$; if $\theta_1 < \theta_2$,

by Lemma 5, we have $\text{width}_{[\theta_1, \theta_2]}(P_i) = \text{width}(\sigma_{[\theta_1, \theta_2]}(P_i))$ and $\sigma_{[\theta_1, \theta_2]}(P_i) = \text{conv}(P_i \cup Q)$ where $Q = \{q_1, q_2\}$ consists of two points as described above. (See also Figure 1.) Hence, in this case, we can compute $\text{width}_{[\theta_1, \theta_2]}(P_i)$ in $O(\log^3 n)$ time by Lemma 4 with a query set $Q = \{q_1, q_2\}$ to W . Again, Lemma 7 implies the following.

► **Lemma 12.** *$\text{width}_{[\theta_1, \theta_2]}(P_i) < w_1$ if and only if $\text{width}(P_i \cup \overline{P}_j) < w_1$. Moreover, if $\text{width}_{[\theta_1, \theta_2]}(P_i) < w_1$, then $\text{width}(P_i \cup \overline{P}_j) = \text{width}_{[\theta_1, \theta_2]}(P_i)$.*

By Lemma 12, we have $\text{width}(P_i \cup \overline{P}_j) = \text{width}_{[\theta_1, \theta_2]}(P_i)$ and it is a member of W if and only if the computed value of $\text{width}_{[\theta_1, \theta_2]}(P_i)$ is strictly smaller than w_1 .

This way, we can collect the values in W in $O(n \log^3 n)$ time. By Lemma 10, the minimum value in W is w^* , if W is nonempty; or $w^* = w_1$, if $W = \emptyset$. The corresponding two-strip of width w^* can be computed and stored during the execution of the algorithm. Hence, we finally conclude the following.

► **Theorem 13.** *Given a set P of n points in the plane and an orientation ϕ , a two-line center for P in which one of the two lines is constrained to be in orientation ϕ can be computed in $O(n \log^3 n)$ time and $O(n \log n)$ space.*

4 Fixed angle of intersection

In this section, we solve the third constrained two-line center problem in which, given a real value $0 \leq \beta \leq \pi/2$, the difference of the orientations of the two resulting lines is exactly β . Throughout this section, for convenience, a *constrained two-strip* denotes a pair of strips whose orientations differ by β . Let w^* be the minimum width of a constrained two-strip enclosing P . We start by describing optimal configurations.

► **Lemma 14.** *There exists a minimum-width constrained two-strip (σ_1, σ_2) enclosing P that falls into one of the following cases:*

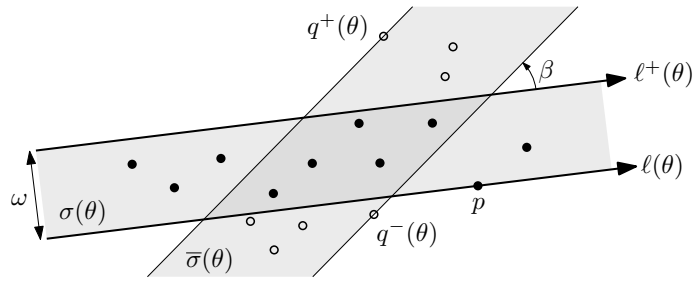
- (i) *Either $w^* = \text{width}(\sigma_1) > \text{width}(\sigma_2)$ or $w^* = \text{width}(\sigma_2) > \text{width}(\sigma_1)$, and one of the four bounding lines of σ_1 and σ_2 contains two points in P .*
- (ii) *It holds that $w^* = \text{width}(\sigma_1) = \text{width}(\sigma_2)$, and each of the four bounding lines of σ_1 and σ_2 contains a point in P .*

In the following, we present an algorithm that runs in $O(n^2 \alpha(n) \log n)$ time using $O(n^2)$ space. Our algorithm follows a similar flow as for the second problem, consisting of two phases: (1) find a favorably narrow interval $(w_0, w_1]$ that includes our target width w^* , and (2) proceed the search for w^* with the aid of $(w_0, w_1]$. We first present an efficient decision algorithm, and then describe each of the two phases.

4.1 Decision algorithm

Let $\omega > 0$ be a given parameter, and our goal is to decide whether or not $\omega \geq w^*$. Throughout this section, we regard θ as a *direction* from the range $[0, 2\pi)$, taken by modulo 2π , and assume that no three points in P are collinear.

We consider a rotational sweeping process with *fixed width* ω described as follows: Take any point $p \in P$ as a pivot. For direction $\theta \in [0, 2\pi)$, let $\ell(\theta)$ and $\ell^+(\theta)$ be two directed lines in θ such that $\ell(\theta)$ is through p and $\ell^+(\theta)$ is at distance ω to the left of $\ell(\theta)$. We simultaneously rotate both lines $\ell(\theta)$ and $\ell^+(\theta)$ counterclockwise by increasing θ , and consider the strip $\sigma(\theta)$ bounded by the two lines. Taking the second strip $\overline{\sigma}(\theta) := \sigma_{\theta+\beta}(P \setminus \sigma(\theta))$ as



■ **Figure 4** A snapshot at θ of the rotational sweeping process with fixed width ω and pivot p .

the minimum-width strip in orientation $\theta + \beta$ enclosing the rest of points in P , our goal is to decide if there exists $\theta \in [0, 2\pi)$ such that $\text{width}(\bar{\sigma}(\theta)) \leq \omega$ for some $p \in P$. See Figure 4, in which points in $P \cap \sigma(\theta)$ are depicted by dots and those in $P \setminus \sigma(\theta)$ by small circles.

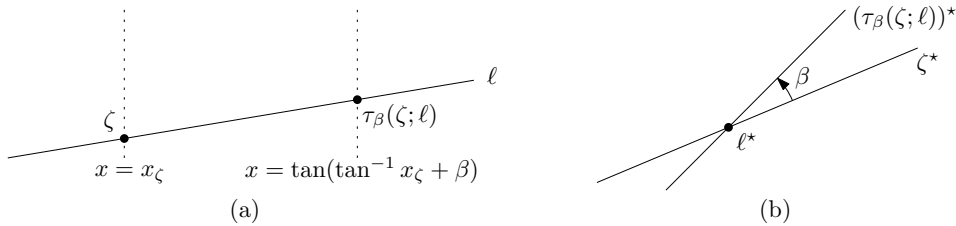
This sweeping process can be simulated by maintaining the dynamic convex hull $\text{conv}(P \setminus \sigma(\theta))$ and its two extreme points $q^-(\theta)$ and $q^+(\theta)$ that define $\bar{\sigma}(\theta)$. Since the number of updates on $P \setminus \sigma(\theta)$ is $O(n)$, it can be done in $O((n + E) \log n)$ time [11], where E denotes the number of changes of the two extreme points $q^-(\theta)$ and $q^+(\theta)$. As will be seen later, $E = O(n\alpha(n))$ and hence the decision can be made in $O(n^2\alpha(n) \log n)$ time.

In order to see why $E = O(n\alpha(n))$ and even to improve the running time, we find it more useful and convenient to discuss the problem in the dual setting. Consider the standard dual transformation that maps each point $r = (a, b) \in \mathbb{R}^2$ into a non-vertical line $r^* : \{y = ax - b\}$, and vice versa. Let $L := \{p^* \mid p \in P\}$ be the set of n lines dual to each point in P . For a fixed pivot $p = (a, b) \in P$, the trace of $l^+(\theta)$ in the dual environment draws a hyperbola $\{y = ax - b \pm \omega\sqrt{1 + x^2}\}$ [5]. We take the upper branch of the hyperbola, denoted by

$$h_p : \{y = ax - b + \omega\sqrt{1 + x^2}\},$$

and let $H := \{h_p \mid p \in P\}$. By this choice, we restrict ourselves to considering half the domain of directions, namely $[\pi/2, 3\pi/2)$; the other case can be handled symmetrically by considering the lower branches of the hyperbolas. Note that the dual of $l^+(\theta)$ for $\theta \in (\pi/2, 3\pi/2)$ is the intersection point between h_p and vertical line $\{x = \tan(\theta)\}$. Hence, the first strip $\sigma(\theta)$ appears as a vertical segment between p^* and h_p at $x = \tan(\theta)$. Similarly, the dual of the second strip $\bar{\sigma}(\theta)$ is a vertical segment at $x = \tan(\theta + \beta)$ that crosses all but those lines in L intersected by $(\sigma(\theta))^*$.

For better exposition, we consider an operation that rotates a given line around a given point by angle β . In the dual setting, such a rotation can be performed by the following mapping: for any non-vertical line ℓ and a point $\zeta \in \ell$, we define $\tau_\beta(\zeta; \ell)$ to be the intersection



■ **Figure 5** (a) Illustration for the mapping $\tau_\beta(\cdot; \ell)$ on line ℓ and (b) its dual representation.

point between ℓ and the vertical line $\{x = \tan(\tan^{-1}(x_\zeta) + \beta)\}$, where x_ζ is the x -coordinate of ζ . See Figure 5. For a line segment s on ℓ , let $\tau_\beta(s)$ be the segment of ℓ obtained by applying the β -shifting map τ_β to all points on s along ℓ , that is, $\tau_\beta(s) = \bigcup_{\zeta \in s} \tau_\beta(\zeta; \ell)$. Note that $\tau_\beta(s)$ may consist of two half-lines, if the x -coordinates of s are large enough.

Now, consider the pieces of lines in L below p^* or above h_p . Let S be the set of these segments and half-lines, and $T := \{\tau_\beta(s) \mid s \in S\}$ be the set of shifted segments. We then observe that the two lines bounding $\bar{\sigma}(\theta)$ correspond to the highest and lowest points of the intersection $T \cap \{x = \tan(\theta + \beta)\}$. Thus, $\bar{\sigma}(\theta)$ and its width over $\theta \in (\pi/2, 3\pi/2)$ are determined by the lower and upper envelopes, $\mathcal{L}(T)$ and $\mathcal{U}(T)$, of T . This implies that the number E of changes of the two extreme points $q^-(\theta)$ and $q^+(\theta)$ indeed counts the number of vertices in $\mathcal{L}(T)$ and $\mathcal{U}(T)$, so $E = O(n\alpha(n))$ [30] and the decision problem can be solved in $O(n^2\alpha(n) \log n)$ time. In the following, we improve it to $O(n^2\alpha(n))$ time.

We start with the directed line $\ell(\pi/2)$ through any $p \in P$, and rotate it around the pivot p by increasing θ until it hits another point $p' \in P$. Whenever $\ell(\theta)$ hits another point p' , we switch the pivot to p' and continue the rotation around the new pivot p' . Observe that this motion of $\ell(\theta)$ preserves the number k of points in P that lie on $\ell(\theta)$ or on its right side, except some moments when $\ell(\theta)$ contains two points. In the dual setting, the trace of $\ell(\theta)$ is well known as the k -level of the arrangement $\mathcal{A}(L)$ of lines in L . More precisely, for $k = 1, \dots, n$, the k -level of $\mathcal{A}(L)$, denoted by L_k , is the monotone chain consisting of all edges e of $\mathcal{A}(L)$ such that there are exactly $k - 1$ lines strictly below any point in the relative interior of e . Similarly, the trace of line $\ell^+(\theta)$ at distance ω from $\ell(\theta)$ is the k -level of the arrangement $\mathcal{A}(H)$ of n hyperbolas in H , denoted by H_k .

Let L_k^- be the region strictly below L_k and H_k^+ be that strictly above H_k . For each $r \in P$ and $1 \leq k \leq n$, define

$$S_{k,r}^+ := r^* \cap H_k^+, \quad S_{k,r}^- := r^* \cap L_k^-, \quad T_{k,r}^+ := \tau_\beta(S_{k,r}^+), \quad \text{and} \quad T_{k,r}^- := \tau_\beta(S_{k,r}^-),$$

and let T_k^+ and T_k^- be the collections of segments and half-lines in $T_{k,r}^+$ and $T_{k,r}^-$, respectively, over all $r \in P$. Our goal is then to compute the envelopes $\mathcal{L}(T_k^+ \cup T_k^-)$ and $\mathcal{U}(T_k^+ \cup T_k^-)$ for all k . As discussed above, these envelopes explicitly describe the changes of the extreme points defining the second strip $\bar{\sigma}(\theta)$ whose orientation is $\theta + \beta$, so we can decide whether $\text{width}(\bar{\sigma}(\theta)) \leq \omega$ for some θ in time linear to the total complexity of the envelopes.

Let $\mathcal{L}_k^+ := \mathcal{L}(T_k^+)$, $\mathcal{U}_k^+ := \mathcal{U}(T_k^+)$, $\mathcal{L}_k^- := \mathcal{L}(T_k^-)$, and $\mathcal{U}_k^- := \mathcal{U}(T_k^-)$. We compute these four families of envelopes separately for all k . Then, $\mathcal{L}(T_k^+ \cup T_k^-)$ and $\mathcal{U}(T_k^+ \cup T_k^-)$ can be obtained by merging two envelopes. For our purpose, the following observation is essential.

► **Lemma 15.** *For any $1 \leq k \leq n - 1$ and $r \in P$,*

$$T_{k+1,r}^+ \subset T_{k,r}^+ \quad \text{and} \quad T_{k,r}^- \subset T_{k+1,r}^-.$$

Therefore, every point on \mathcal{L}_k^+ is below or on \mathcal{L}_{k+1}^+ ; every point on \mathcal{U}_k^+ is above or on \mathcal{U}_{k+1}^+ ; every point on \mathcal{L}_k^- is above or on \mathcal{L}_{k+1}^- ; every point on \mathcal{U}_k^- is below or on \mathcal{U}_{k+1}^- .

This naturally suggests us computing each family of envelopes in an incremental way. In the following, we describe how to compute $\mathcal{L}_n^+, \mathcal{L}_{n-1}^+, \dots, \mathcal{L}_1^+$ in this order. The other three families can also be handled symmetrically and analogously.

Initially, we compute $\mathcal{A}(L)$ and $\mathcal{A}(L \cup H)$, and for each vertex v of $\mathcal{A}(L \cup H)$, we collect its images under τ_β into a set Ξ . More precisely, we use a dictionary structure for Ξ indexed by pairs in $L \times (L \cup H)$, such as an array-based $n \times 2n$ matrix. For each pair (ℓ, ℓ') with $\ell, \ell' \in L$ such that $v = \ell \cap \ell'$ is a vertex of $\mathcal{A}(L \cup H)$, we store $\tau_\beta(v; \ell)$; for (ℓ, h) with $\ell \in L$ and $h \in H$, we store $\tau_\beta(v; \ell)$ for each $v \in \ell \cap h$. We also associate each point $\xi \in \Xi$ with the

edge e of $\mathcal{A}(L)$ such that $\xi \in e$, so that given a pair in $L \times (L \cup H)$ we can locate in $O(1)$ time on which edges of $\mathcal{A}(L)$ its relevant points $\xi \in \Xi$ lie. Note that at most two points are stored at each entry of Ξ . The initialization can be done in $O(n^2)$ time using $O(n^2)$ space.

Let $T_{n+1}^+ = \mathcal{L}_{n+1}^+ = \emptyset$, and suppose \mathcal{L}_{k+1}^+ has been correctly computed. Let T^* be the set of segments obtained from $T_{k,r}^+ \setminus T_{k+1,r}^+$ for all $r \in P$. We then have $\mathcal{L}_k^+ = \mathcal{L}(\mathcal{L}_{k+1}^+ \cup T^*)$ by Lemma 15, so can be computed by merging \mathcal{L}_{k+1}^+ and $\mathcal{L}(T^*)$. Computing $\mathcal{L}(T^*)$ is done in three steps: specify T^* , compute the arrangement $\mathcal{A}(T^*)$, and extract $\mathcal{L}(T^*)$ from $\mathcal{A}(T^*)$.

To specify T^* , we walk along H_k and H_{k+1} in $\mathcal{A}(L \cup H)$ and find out all intersections $H_k \cap r^*$ and $H_{k+1} \cap r^*$ for each $r \in P$. We are then able to extract all segments of r^* that lie in between H_k and H_{k+1} . For each such segment s , $\tau_\beta(s)$ is a member of T^* . This can be done in $O(m_k + m_{k+1} + n)$ time, where m_i denotes the number of vertices of $\mathcal{A}(L \cup H)$ along H_i . Note that the number of segments in T^* is at most $m_k + m_{k+1}$, since the endpoints of their preimages under the β -shifting map τ_β are all from the vertices along H_k and H_{k+1} .

Note that every $t \in T^*$ is a segment of a line in L , so $\mathcal{A}(T^*)$ is a clipped portion of the entire arrangement $\mathcal{A}(L)$. In addition, the endpoints of t are members of Ξ , so we can find out their exact locations in $\mathcal{A}(L)$ in $O(1)$ time per each. Thus, we can construct $\mathcal{A}(T^*)$ by tracing segments $t \in T^*$ in $\mathcal{A}(L)$ in $O(|T^*| + v_k) = O(m_k + m_{k+1} + v_k)$ time, where v_k denotes the number of vertices of $\mathcal{A}(L)$ we encounter. Note that $\mathcal{A}(T^*)$ consists of exactly $m_k + m_{k+1} + v_k$ vertices and at most $m_k + m_{k+1} + 2v_k$ edges.

As $\mathcal{A}(T^*)$ forms a plane graph, possibly being disconnected, it turns out that its lower envelope $\mathcal{L}(T^*)$ can be obtained in time linear to its complexity. Here, we make use of the following two algorithmic tools: First, a linear-time algorithm for computing the vertical decomposition (or the trapezoidation) of a simple polygon [14,18] can be applied for computing the lower envelope of a connected plane graph.

► **Lemma 16.** *Given a connected plane graph G , consisting of m line segments, its lower envelope $\mathcal{L}(G)$ can be computed in $O(m)$ time.*

Second, Asano et al. [9] presented a linear-time algorithm for computing the lower envelope of disjoint line segments, provided their endpoints are sorted. It is not difficult to see that their algorithm also works even if we replace “segments” by “monotone chains.”

► **Lemma 17** (Asano et al. [9]). *Let C_1, C_2, \dots, C_l be mutually disjoint monotone chains with m line segments in total. If a sorted list of their endpoints is given, then their lower envelope $\mathcal{L}(\bigcup_i C_i)$ can be computed in $O(m)$ time.*

Back to our problem, we apply Lemma 16 to each connected component of $\mathcal{A}(T^*)$, resulting in disjoint monotone chains C_1, C_2, \dots , whose lower envelope is $\mathcal{L}(T^*)$. Recall that we already have the sorted list of endpoints of T^* , since those endpoints have been obtained by walking along H_k and H_{k+1} in $\mathcal{A}(L \cup H)$ and applying the β -shifting map τ_β , and the map τ_β preserves the order along H_k and H_{k+1} . Hence, we can extract a sorted list of endpoints of the chains C_i in additional $O(m_k + m_{k+1})$ time, which allows us to apply Lemma 17 to obtain $\mathcal{L}(T^*)$. The total time for this third step is proportional to the number of edges in $\mathcal{A}(T^*)$, so $O(m_k + m_{k+1} + v_k)$ time.

Finally, to compute \mathcal{L}_k^+ , we linearly scan \mathcal{L}_{k+1}^+ and $\mathcal{L}(T^*)$, simultaneously. This takes time linear to the total complexity of \mathcal{L}_{k+1}^+ and $\mathcal{L}(T^*)$. Note that \mathcal{L}_{k+1}^+ consists of $O(|T_{k+1}^+| \alpha(|T_{k+1}^+|)) = O(m_{k+1} \alpha(m_{k+1}))$ edges since it is the lower envelope of line segments [30]. So, the total time we spend to incrementally construct \mathcal{L}_k^+ from \mathcal{L}_{k+1}^+ for each $1 \leq k \leq n$ is bounded by $O(n + m_k + m_{k+1} \alpha(m_{k+1}) + v_k)$.

By iterating k from n down to 1, we conclude our decision algorithm.

► **Theorem 18.** *Given a set P of n points, an angle β , and a parameter ω , we can decide whether or not $\omega \geq w^*$ in $O(n^2\alpha(n))$ time and $O(n^2)$ space.*

4.2 First phase of the optimization algorithm

From now on, we describe our optimization algorithm. Its first phase is done as follows.

Let W_2 be the set of all pairwise distances among points in P . We first obtain two consecutive values $w'_0 < w'_1 \in W_2$ such that $w^* \in (w'_0, w'_1]$. This is easily done in $O(n^2\alpha(n) \log n)$ time by sorting W_2 and performing a binary search on W_2 using our decision algorithm presented in Theorem 18. Next, let W_3 be the set of $n \binom{n}{2}$ values obtained as follows: for any pair $p, q \in P$ with $p \neq q$, collect the distances from each $r \in P$ to the line through p and q . We then find two consecutive values $w''_0 < w''_1 \in W_3$ such that $w^* \in (w''_0, w''_1]$. This can be done in $O(n^2\alpha(n) \log n)$ time by the technique of Gluzman et al. [22], again using our decision algorithm. Note that the two-strip of width w''_1 is the best solution of case (i) of Lemma 14. We then choose $w_0 := \max\{w'_0, w''_0\}$ and $w_1 := \min\{w'_1, w''_1\}$, and obtain:

► **Lemma 19.** *In $O(n^2\alpha(n) \log n)$ time, we can find two values $w_0 \leq w_1$ such that $w_0 < w^* \leq w_1$ and no member in $W_2 \cup W_3$ lies in (w_0, w_1) .*

4.3 Second phase of the optimization algorithm

For each $p \in P$, let w_p^* be the minimum possible width of constrained two-strips (σ_1, σ_2) such that p lies on the boundary of σ_1 . It is obvious that $w^* = \min_{p \in P} w_p^*$. The second phase of our algorithm computes the exact value of w_p^* , if $w_p^* < w_1$; or reports $w_p^* \geq w_1$, otherwise. Note that, if $w_p^* < w_1$, then the corresponding optimal two-strip falls in case (ii) described in Lemma 14 by Lemma 19. In the following, let $p \in P$ be fixed and called the *pivot*.

Updates in the sweeping process with fixed width

Before describing the algorithm, we discuss essential ingredients of its correctness, based on Lemma 19. Let $w \in (w_0, w_1)$ be any value. We consider the sweeping process with fixed width w and fixed pivot p , as described at the beginning of Section 4.1. (See Figure 4.) Recall that the first strip $\sigma(\theta)$ is determined by two directed lines $\ell(\theta)$ and $\ell^+(\theta)$ such that $\ell(\theta)$ goes through p and $\ell^+(\theta)$ is at distance w to the left of $\ell(\theta)$, and the second strip $\bar{\sigma}(\theta)$ in orientation $\theta + \beta$ encloses the rest of points in $P \setminus \sigma(\theta)$. Let $P(\theta) := P \cap \sigma(\theta)$. Then, during this sweeping process as θ increases, $P(\theta)$ undergoes a sequence of *updates*. We identify each update by a pair of its involved point $r \in P$ and its *type* determined by one of the four combinations of the following:

- An update is *right* if it happens when $\ell(\theta)$ hits r ; or *left* when $\ell^+(\theta)$ hits r
- An update is *leaving* if r is being deleted from $P(\theta)$; or *approaching*, otherwise.

Thus, two updates are the same if their involved points and their types are equal.

Let Υ_w be the set of those updates occurred on $P(\theta)$ during the sweeping process with fixed width w over $\theta \in [0, 2\pi)$. Observe that there are two possibilities for each $r \in P \setminus \{p\}$: By Lemma 19, the distance between r and the pivot p is either at most w_0 or at least w_1 . Thus, if r falls in the former case, there are exactly two updates for r in Υ_w whose types are right leaving and right approaching; in the latter case, there are exactly four updates for r in Υ_w with each of the four possible types. This implies that the set Υ_w is invariant under the choice of $w \in (w_0, w_1)$, so we write $\Upsilon = \Upsilon_w$ for any $w \in (w_0, w_1)$.

Fix an arbitrary right leaving update $v_0 \in \Upsilon$ in which $r_0 \in P \setminus \{p\}$ is involved, and assume that both p and r_0 lie along $\ell(0)$ in this order, that is, p and r_0 lie on the horizontal line $\ell(0)$ and r_0 is to the right of p ; this can be easily achieved by a proper rotation of the

axes. For $v \in \Upsilon$ and $w \in (w_0, w_1)$, let $\phi_v(w) \in [0, 2\pi)$ be the direction at which v occurs during the sweeping process with fixed width w . From the above discussion, we know that ϕ_v is a well-defined function from (w_0, w_1) to $[0, 2\pi)$. Lemma 19 implies the following.

► **Lemma 20.** *There is no $w \in (w_0, w_1)$ such that $\phi_v(w) = \phi_{v'}(w)$ for any two distinct $v, v' \in \Upsilon$. Moreover, for each $v \in \Upsilon$, $\phi_v(w)$ is either constant if v is right, continuously increasing if v is left leaving, or continuously decreasing if v is left approaching.*

For $w \in (w_0, w_1)$, we consider a total order \prec_w on Υ such that $v \prec_w v'$ if and only if $\phi_v(w) < \phi_{v'}(w)$. Note that its totality is guaranteed by Lemma 20 and v_0 is the least element in Υ under \prec_w . Lemma 20 further implies that the ordering \prec_w on Υ remains the same over all $w \in (w_0, w_1)$: assuming any swap between \prec_w and $\prec_{w'}$ for $w_0 < w < w' < w_1$, one can face with some $w'' \in (w, w')$ and $v, v' \in \Upsilon$ such that $\phi_v(w'') = \phi_{v'}(w'')$, due to the continuity of functions ϕ_v and $\phi_{v'}$, so a contradiction.

Hence, we have a universal total ordering \prec on Υ such that $\prec = \prec_w$ for any $w \in (w_0, w_1)$. Let $v_0, v_1, \dots, v_{m-1} \in \Upsilon$ be the updates in Υ listed in this order \prec , where $m := |\Upsilon|$. For each $0 \leq i \leq m-1$, let $I_i := \{\phi_{v_i}(w) \mid w_0 < w < w_1\}$. Lemma 20 implies that I_i consists of a single element if v_i is a right update; otherwise, I_i forms an open interval if v_i is a left update. The following summarizes more implications of Lemma 20 about the intervals I_i . Two intervals I and I' are said to be *properly nested* if one includes the other, say $I' \subset I$, in such a way that both endpoints of I' lie in the relative interior of I .

► **Lemma 21.** *Two intervals I_i and I_j are never properly nested. If I_i and I_j overlap, then either both of v_i and v_j are left leaving or both are left approaching.*

For $0 \leq i \leq m-1$, let P_i be the resulting set after executing the first $i+1$ updates v_0, \dots, v_i on the subset of points in P lying on or to the left of $\ell(0)$ whose distances to $\ell(0)$ are at most w_0 . Note that $P_0 = (\sigma_0 \cap P) \setminus \{r_0\}$ where σ_0 denotes the horizontal strip of width w_0 such that $\ell(0)$ bounds σ_0 from below. Let $Q_i := P \setminus P_i$, and define

$$\omega_i(\theta) := \text{width}_\theta(P_i) \quad \text{and} \quad \bar{\omega}_i(\theta) := \text{width}_{\theta+\beta}(Q_i)$$

for $\theta \in [0, 2\pi)$. Let $r_i \in P \setminus \{p\}$ be the point involved in v_i .

► **Lemma 22.** *For any left leaving update $v_i \in \Upsilon$, $\omega_{i-1}(\theta) = \text{width}_\theta(\{p, r_i\})$ over $\theta \in I_i$, and is an increasing function over I_i whose infimum and supremum are w_0 and w_1 , respectively.*

Description of algorithm

The second phase of our algorithm simulates a similar sweeping process as before, but with the first strip $\sigma(\theta)$ having *variable* width: Let $\omega: [0, 2\pi) \rightarrow \mathbb{R}$ be a function, which will be specified later. We redefine $\ell^+(\theta)$ to be the line at distance $\omega(\theta)$ to the left of $\ell(\theta)$, and thus $\sigma(\theta)$ to have width $\omega(\theta)$. The second strip $\bar{\sigma}(\theta)$ in orientation $\theta + \beta$ is determined as before to tightly enclose the rest of points in $P \setminus \sigma(\theta)$. Let $\bar{\omega}(\theta) := \text{width}(\bar{\sigma}(\theta))$. This way, the process is completely determined by the width function $\omega(\theta)$.

Our width function $\omega(\theta)$ will be fully determined by when to execute each update $v_i \in \Upsilon$. For $0 \leq i \leq m-1$, let ϕ_i be the direction at which the i -th update $v_i \in \Upsilon$ is executed in our algorithm. We choose the ϕ_i 's by the following rules:

- If v_i is a right update, ϕ_i is the only direction in I_i , that is, $I_i = \{\phi_i\}$.
- If v_i is a left approaching update, ϕ_i is chosen to be the larger endpoint of I_i .
- If v_i is a left leaving update, ϕ_i is chosen to be the smallest direction θ such that $\omega_{i-1}(\theta) = \bar{\omega}_{i-1}(\theta)$ over $\theta \in I_i$, if exists; otherwise, ϕ_i is the larger endpoint of I_i .

Note that $\phi_0 = 0$ and let $\phi_m := 2\pi$. It is obvious that either $\phi_i \in I_i$ or ϕ_i is the larger endpoint of I_i . Less obvious is that the resulting ϕ_i 's indeed obey the ordering \prec of Υ .

► **Lemma 23.** *It holds that $0 = \phi_0 < \phi_1 \leq \phi_2 \leq \dots \leq \phi_{m-1} \leq \phi_m = 2\pi$.*

The function $\omega(\theta)$ is then set up as follows: $\omega(0) := w_0$ and $\omega(\theta) := \max\{w_0, \omega_i(\theta)\}$ for $\theta \in (\phi_i, \phi_{i+1}]$ and $0 \leq i \leq m-1$. We then obtain a conditional correctness of our algorithm.

► **Lemma 24.** *Suppose $w_p^* < w_1$, and let θ^* and $\theta^* + \beta$ be the directions of the bounding lines of a corresponding two-strip of width w_p^* such that the pivot p lies on the right bounding line of direction θ^* . If $\theta^* \notin I_i$ for all left approaching updates $v_i \in \Upsilon$, then there is a left leaving update $v_j \in \Upsilon$ such that $\theta^* = \phi_j \in I_j$ and $w_p^* = \omega(\theta^*) = \omega_{j-1}(\theta^*) = \bar{\omega}_{j-1}(\theta^*) = \bar{\omega}(\theta^*)$.*

Thus, we can compute w_p^* and its corresponding two-strip by checking each ϕ_i such that $\phi_i \in I_i$ and $v_i \in \Upsilon$ is a left leaving update, provided the condition of Lemma 24 is satisfied. The other case, where w_p^* is *not* determined by left leaving updates, can be handled by a *reversed* sweeping process that rotates $\sigma(\theta)$ clockwise by decreasing θ from 2π to 0; note that in this reversed process each approaching update becomes a leaving update, and vice versa.

Now, the detailed implementation is presented. Simulating the sweeping process with function $\omega(\theta)$ can be done by maintaining a dynamic set Q , representing $P \setminus \sigma(\theta)$, and its convex hull $\text{conv}(Q)$. First, we compute the updates $v_0, v_1, \dots, v_{m-1} \in \Upsilon$ together with their intervals I_i , and also precompute ϕ_i for all right updates and left approaching updates $v_i \in \Upsilon$. Initially, $Q = Q_0$ and $Q = Q_i$ while we are in $\theta \in (\phi_i, \phi_{i+1}]$ for each $0 \leq i \leq m$. We also maintain the two extreme points of Q that determine $\bar{\sigma}(\theta)$: this can be done by two types of queries on $\text{conv}(Q)$, finding two tangents of $\text{conv}(Q)$ in a given direction and finding the next extreme point of $\text{conv}(Q)$ neighboring the current one. Each of these convex hull queries can be answered in $O(\log n)$ amortized time [11].

While we rotate $\sigma(\theta)$ as increasing θ , we execute updates $v_i \in \Upsilon$ at $\theta = \phi_i$ if ϕ_i has already been computed. Recall that only the execution times ϕ_i of left leaving update v_i are not precomputed, so they are evaluated during the sweeping process: Suppose the current direction θ lies in I_i for a left leaving update v_i and the first $j \leq i$ updates v_0, \dots, v_{j-1} have already been executed, that is, $Q = Q_{j-1}$ currently at θ and $\bar{\omega}(\theta) = \bar{\omega}_{j-1}(\theta)$. At this moment θ , note that $\theta \in I_j \cap I_i$ and v_j is also a left leaving update by Lemma 21. Hence, Lemma 22 implies that $\omega(\theta) = \omega_{j-1}(\theta) = \text{width}_\theta(\{p, r_j\})$. We then solve the equation $\omega_{j-1}(\phi) = \bar{\omega}_{j-1}(\phi)$. Since the two functions ω_{j-1} and $\bar{\omega}_{j-1}$ are sinusoidal over a range in which the two extreme points of Q_{j-1} do not change [10], this can be done in time proportional to the number of such changes while $Q = Q_{j-1}$. As soon as we find a solution $\phi \in I_j$ such that $\omega_{j-1}(\phi) = \bar{\omega}_{j-1}(\phi)$, we know that $\phi_j = \phi$ by our rules; otherwise, ϕ_j is chosen to be the larger endpoint of I_j .

Since $m = O(n)$, the overall time we spend is bounded by $O(n \log n + E \log n)$, where E denotes the number of changes of the extreme points of Q that define the second strip $\bar{\sigma}(\theta)$. In the dual setting, as done for the decision algorithm, those changes correspond to the vertices of the lower and upper envelopes of $O(n)$ line segments, so we have $E = O(n\alpha(n))$ [30]. By iterating pivots $p \in P$, the second phase of the algorithm can be implemented in $O(n^2\alpha(n) \log n)$ total time.

Therefore, we conclude the following result.

► **Theorem 25.** *Given a set P of n points and a parameter $\beta \in [0, \pi/2]$, the two-line center problem with a constraint that the resulting two lines should make an angle of β can be solved in $O(n^2\alpha(n) \log n)$ time using $O(n^2)$ space.*

References

- 1 Pankaj K. Agarwal, Cecilia M. Procopiuc, and Kasturi R. Varadarajan. A $(1+\varepsilon)$ -approximation algorithm for 2-line-center. *Computational Geometry: Theory and Applications*, 26:119–128, 2003. doi:10.1016/S0925-7721(03)00017-8.
- 2 Pankaj K. Agarwal, Cecilia M. Procopiuc, and Kasturi R. Varadarajan. Approximation algorithms for a k -line center. *Algorithmica*, 42(3):221–230, 2005. doi:10.1007/s00453-005-1166-x.
- 3 Pankaj K. Agarwal and Micha Sharir. Off-line dynamic maintenance of the width of a planar point set. *Computational Geometry: Theory and Applications*, 1:65–78, 1991. doi:10.1016/0925-7721(91)90001-U.
- 4 Pankaj K. Agarwal and Micha Sharir. Planar geometric location problems and maintaining the width of a planar set. In *Proceedings of the 2nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1991)*, pages 449–458. SIAM, 1991. URL: <http://dl.acm.org/citation.cfm?id=127787.127865>.
- 5 Pankaj K. Agarwal and Micha Sharir. Planar geometric location problems. *Algorithmica*, 11(2):185–195, 1994. doi:10.1007/BF01182774.
- 6 Pankaj K. Agarwal and Micha Sharir. Efficient randomized algorithms for some geometric optimization problems. *Discrete & Computational Geometry*, 16(4):317–337, 1996. doi:10.1007/BF02712871.
- 7 Taehoon Ahn and Sang Won Bae. Constrained two-line center problems, 2024. arXiv:2409.13304.
- 8 Taehoon Ahn, Chaeyoon Chung, Hee-Kap Ahn, Sang Won Bae, Otfried Cheong, and Sang Duk Yoon. Minimum-width double-slabs and widest empty slabs in high dimensions. In J.A. Soto and A. Wiese, editors, *Proceedings of the 16th Latin American Theoretical Informatics (LATIN 2024), Part I*, volume 14578 of *LNCS*, pages 303–317, 2024. doi:10.1007/978-3-031-55598-5_20.
- 9 Takao Asano, Tetsuo Asano, Leonidas Guibas, John Hershberger, and Hiroshi Imai. Visibility of disjoint polygons. *Algorithmica*, 1:49–63, 1986. doi:10.1007/BF01840436.
- 10 Sang Won Bae. Minimum-width double-strip and parallelogram annulus. *Theoretical Computer Science*, 833:133–146, 2020. doi:10.1016/j.tcs.2020.05.045.
- 11 Gerth Stølting Brodal and Riko Jacob. Dynamic planar convex hull. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 617–626, 2002. doi:10.1109/SFCS.2002.1181985.
- 12 T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. *International Journal of Computational Geometry and Applications*, 12(1-2):67–85, 2002. doi:10.1142/S0218195902000748.
- 13 Timothy M. Chan. A fully dynamic algorithm for planar width. *Discrete & Computational Geometry*, 30:17–24, 2003. doi:10.1007/s00454-003-2923-8.
- 14 Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(3):485–524, 1991. doi:10.1007/BF02574703.
- 15 Chayoon Chung, Taehoon Ahn, Sang Won Bae, and Hee-Kap Ahn. Parallel line centers with guaranteed separation. In *Proceedings of the 35th Canadian Conference on Computational Geometry (CCCG 2023)*, pages 153–160, 2023.
- 16 Arun Kumar Das, Sandip Das, and Joydeep Mukherjee. Approximation algorithms for orthogonal line centers. *Discrete Applied Mathematics*, 338:69–76, 2023. doi:10.1016/j.dam.2023.05.014.
- 17 David Eppstein. Incremental and decremental maintenance of planar width. *Journal of Algorithms*, 37:570–577, 2000. doi:10.1006/jagm.2000.1107.
- 18 Alain Fournier and Delfin Y. Montuno. Triangulating simple polygons and equivalent problems. *ACM Transactions on Graphics*, 3:153–174, 1984. doi:10.1145/357337.357341.

- 19 Greg N. Frederickson and Donald B. Johnson. The complexity of selection and ranking in $X+Y$ and matrices with sorted columns. *Journal of Computer and System Science*, 24:197–208, 1982. doi:10.1016/0022-0000(82)90048-4.
- 20 Greg N. Frederickson and Donald B. Johnson. Generalized selection and ranking: Sorted matrices. *SIAM Journal on Computing*, 13(1):14–30, 1984. doi:10.1137/0213002.
- 21 Alex Glozman, Klara Kedem, and Gregory Shpitalnik. On some geometric selection and optimization problems via sorted matrices. In *Proceedings of the 4th International Workshop on Algorithms and Data Structures (WADS 1995)*, volume 955 of *LNCS*, pages 26–37, 1995. doi:10.1007/3-540-60220-8_48.
- 22 Alex Glozman, Klara Kedem, and Gregory Shpitalnik. On some geometric selection and optimization problems via sorted matrices. *Computational Geometry: Theory and Applications*, 11(1):17–28, 1998. doi:10.1016/S0925-7721(98)00017-0.
- 23 John Hershberger and Subhash Suri. Off-line maintenance of planar configurations. *Journal of Algorithms*, 21:453–475, 1991. doi:10.1006/jagm.1996.0054.
- 24 Jerzy Jaromczyk and Mirosław Kowaluk. The two-line center problem from a polar view: A new algorithm and data structure. In *Proceedings of the 4th International Workshop on Algorithms and Data Structures (WADS 1995)*, volume 955 of *LNCS*, pages 13–25, 1995. doi:10.1007/3-540-60220-8_47.
- 25 Matthew J. Katz, Klara Kedem, and Michael Segal. Constrained square-center problems. In *Proceedings of the 6th Scandinavian Workshop on Algorithm Theory (SWAT 1998)*, volume 1432, pages 95–106. Springer, 1998. doi:10.1007/BFb0054358.
- 26 Matthew J. Katz and Micha Sharir. An expander-based approach to geometric optimization. *SIAM Journal on Computing*, 26(5):1384–1408, 1997. doi:10.1137/S0097539794268649.
- 27 Sang-Sub Kim, Sang Won Bae, and Hee-Kap Ahn. Covering a point set by two disjoint rectangles. *International Journal of Computational Geometry & Applications*, 21(3):313–330, 2011. doi:10.1142/S0218195911003676.
- 28 Nimrod Megiddo and Arie Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, 1982. doi:10.1016/0167-6377(82)90039-6.
- 29 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer Verlag, 1985. doi:10.1007/978-1-4612-1098-6.
- 30 Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.
- 31 Godfried T. Toussaint. Solving geometric problems with the rotating calipers. In *Proceedings of the 2nd Mediterranean Electrotechnical Conference (IEEE MELECON 1983)*, 1983.