Parallel Line Centers with Guaranteed Separation[*]

Chaeyoon Chung[†] Taehoon Ahn[*] Sang Won Bae[‡] Hee-Kap Ahn[§]

Abstract

Given a set P of n points in the plane and an integer $k \geq 1$, the k-line-center problem asks k slabs whose union encloses P that minimizes the maximum width of the k slabs. In this paper, we introduce a new variant of the k-line-center problem for $k \geq 2$, in which the resulting k lines are parallel and a prescribed separation between two line centers is guaranteed. More precisely, we define a measure of separation, namely the gap-ratio of k parallel slabs, to be the minimum distance between any two slabs, divided by the width of the smallest slab enclosing the k slabs. We present the first and efficient algorithms for the following problems: (1) Given a real $0 < \rho \leq 1$, compute k parallel slabs of minimum width that cover P with gap-ratio at least ρ. (2) Compute k parallel slabs that covers P with maximum possible gap-ratio. Our algorithms run in $O(\rho^{-k} \cdot kn \log n)$ and $O(\rho_{\max}^{-k} \cdot kn \log n)$ time, respectively, where $\rho_{\max}$ denotes the maximum possible gap-ratio of any k parallel slabs that cover P. Both algorithms use $O(n)$ space.

1 Introduction

In many practical situations of geometric facility location, one would like to locate two or more facilities of a certain shape that serve a given set P of input customers (e.g., points) in the plane, in such a way that the interference between two facilities and/or between two customers served by different facilities is minimized. Hence, most preferred in this case are mutually disjoint facilities with separation guaranteed or maximized between the covering regions of any two facilities. In this paper, we consider such a variant of the *k-line-center problem* in the plane for $k \geq 2$ with separation guaranteed or maximized between any two line centers and between their covering regions. By this goal regarding separation, it is obvious that the resulting k line centers are parallel and mutually disjoint. A more precise description of our problem is given below.

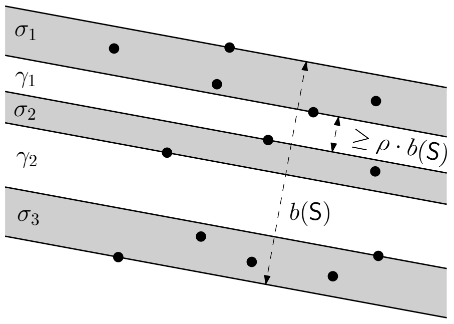

Figure 1: An example of a 3-slab cover S of 13 points in P. Here, we have $w(\mathsf{S}) = w(\sigma_1) = w(\sigma_3)$ and $g(\mathsf{S}) = w(\gamma_1) \geq \rho \cdot b(\mathsf{S})$. Hence, the k-slab S is a minimum-width k-slab cover of P whose gap-ratio is at least ρ.

As an input point is assigned to its closest line center under the Euclidean metric, the covering region of a line center forms a *slab*. So the problem is equivalent to finding k parallel and mutually disjoint slabs $\sigma_1, \sigma_2, \ldots, \sigma_k$ such that their union encloses P and $\sigma_i \cap P \neq \emptyset$ for each $i = 1, \ldots, k$. We call a sequence S of such k parallel slabs a *k-slab cover* of P, or just a *k-slab* regardless of the relation to P. For each slab σ, its *width* $w(\sigma)$ is the orthogonal distance between its two bounding lines. Consider a k-slab $\mathsf{S} = (\sigma_1, \ldots, \sigma_k)$. (See Figure 1.) The *width* of S, denoted by $w(\mathsf{S})$, is defined to be $\max\{w(\sigma_1), \ldots, w(\sigma_k)\}$, and the *breadth* of S, denoted by $b(\mathsf{S})$, is the orthogonal distance between the two outermost bounding lines of S. The region between two consecutive slabs σ_i and σ_{i+1} in S is called a *gap*, denoted by γ_i. Each gap γ_i also forms a slab, so $w(\gamma_i)$ denotes its width. The *gap-width* of S is defined to be the minimum of $w(\gamma_i)$ over $i = 1, \ldots, k-1$, denoted by $g(\mathsf{S})$. Our measure of separation in the problem is then defined to be the ratio of the gap-width over the breadth, namely, the *gap-ratio* of S is $\rho(\mathsf{S}) := g(\mathsf{S})/b(\mathsf{S})$.

The goal of the problem is thus to find an optimal k-slab cover of P regarding two objective criteria: width and gap-ratio. More precisely, we consider the following

[*]C.Chung, T.Ahn, and H.-K.Ahn were supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2017-0-00905, Software Star Lab (Optimal Data Structure and Algorithmic Applications in Dynamic Geometric Environment)) and (No. 2019-0-01906, Artificial Intelligence Graduate School Program(POSTECH)). S.W.Bae was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2018R1D1A1B07042755 and No. RS-2023-00251168).

[†]Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Republic of Korea, {chaeyoon17, sloth}@postech.ac.kr

[‡]Division of Artificial Intelligence and Computer Science, Kyonggi University, Suwon, Republic of Korea, swbae@kgu.ac.kr

[§]Department of Computer Science and Engineering, Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Pohang, Republic of Korea, heekap@postech.ac.kr

problems for given $k \geq 2$ and a set P of n points:

1. For a given real $0 < \rho \leq 1$, find a minimum-width k-slab cover of P whose gap-ratio is at least ρ.
2. Find a maximum-gap-ratio k-slab cover of P.

Related work. The *k-line-center* problem is equivalent to finding k slabs, being not necessarily parallel, of min-max width whose union encloses the input points. The 1-line-center problem can be solved in $O(n \log n)$ time by computing the center line of a minimum-width slab of the points [9, 11]. For the 2-line-center problem, Agarwal and Sharir [3] gave an $O(n^2 \log^5 n)$-time algorithm, and Jaromczyk and Kowaluk [12] improved it to $O(n^2 \log^2 n)$ time. Agarwal et al. [1] gave an $(1+\epsilon)$-approximation algorithm for the problem that runs in $O(n(\log n + \epsilon^{-2}\log(1/\epsilon)) + \epsilon^{-7/2}\log(1/\epsilon))$ time. When k is part of input, the k-line-center problem is NP-complete because it is known to be NP-complete to decide whether n points can be covered by k lines [13]. Agarwal et al. [2] gave an $(1+\epsilon)$-approximation algorithm that runs in $O(n \log n)$ time with the constant factor depending on k and ϵ.

If the line centers are constrained to be parallel in the k-line center problem, the problem is equivalent to finding a minimum-width k-slab cover in our sense. In this case, Bae [5] presented an $O(n^2)$-time algorithm for the case of $k = 2$. No non-trivial algorithm is known for each $k > 3$, to the best of our knowledge.

Our results and approach. We present efficient algorithms for the above two problems. Our algorithms run in $O(\rho^{-k} \cdot kn \log n)$ and $O(\rho_{\max}^{-k} \cdot kn \log n)$ time, respectively, where $\rho_{\max}$ denotes the maximum possible gap-ratio of any k parallel slabs that cover P. Both algorithms use $O(n)$ space.

We first consider Problem 1 of computing a minimum-width k-slab cover of P whose gap-ratio is at least a given parameter $\rho \in (0, 1]$. To tackle the problem, we introduce a concept of a *separator* of a k-slab, defined to be a sequence of $k-1$ points each of which lies in its distinct gap. We describe efficient algorithms to compute an optimal k-slab cover that respects a fixed separator in Section 3. Then, in Section 4, we show how to solve Problem 1 by testing $O(1/\rho^k)$ candidate separators. In Section 5, we show that the algorithms in Sections 3 and 4 are helpful enough to achieve an efficient algorithm for Problem 2.

2 Preliminaries

The line segment connecting two points p and q is denote by pq and its length by $|pq|$. The *orientation* of a line ℓ in the plane is the angle swept from a vertical line in counterclockwise direction to ℓ. Hence, the orientation θ of each line falls in the range $\theta \in [0, \pi)$. The *orientation* of a linear object, including line segments, slabs, and k-slabs, is that of a line parallel to it.

For any two points p, q and orientation $\theta \in [0, \pi)$, define $d_\theta(p, q)$ to be the orthogonal distance between two lines in orientation θ through p and q. It can be written $d_\theta(p, q) = |pq| \cdot |\sin(\theta - \theta_{pq})|$, where θ_{pq} is the orientation of pq. Thus, $d_\theta(p, q)$ for fixed p and q is a sinusoidal function in θ. A function is called *sinusoidal* if it is of the form $a \sin(\theta + b)$ for some constants $a, b \in \mathbb{R}$.

Consider any non-vertical k-slab $\mathsf{S} = (\sigma_1, \ldots, \sigma_k)$ such that σ_i lies above σ_{i+1} and its i-th gap γ_i lies between σ_i and σ_{i+1} for $i = 1, \ldots, k-1$. We call S a (k, ρ)-*slab* if its gap-ratio $\rho(\mathsf{S}) \geq \rho$ for a constant $0 < \rho \leq 1$. Let $\mathsf{R} = (r_1, \ldots, r_{k-1})$ be a sequence of $k-1$ points on a common line in the order along the line. If it holds $r_i \in \gamma_i$ for each $i = 1, \ldots, k-1$, we call R a *separator* of S and say that the k-slab S respects the separator R.

3 (k, ρ)-Slabs Respecting a Given Separator

In this section, we present two algorithms that compute a minimum-width (k, ρ)-slab cover S^* of P respecting a given separator R. Since the k slabs in a k-slab cover of P are mutually disjoint and each of them encloses at least one point of P, we can assume that $0 < \rho \leq 1/(k-1)$. There is no k-slab cover of P for $\rho > 1/(k-1)$.

Let $\mathsf{R} = (r_1, \ldots, r_{k-1})$ be a given separator. Without loss of generality, we assume that the points $r_1, \ldots, r_{k-1}$ in R lie on a common vertical line in this order along it downwards. For each orientation $\theta \in [0, \pi)$, let $\ell_i(\theta)$ be the line in orientation θ through r_i for $i = 1, \ldots, k-1$. Then, these $k-1$ lines partition P into k disjoint subsets $P_1(\theta), \ldots, P_k(\theta)$ such that $P_1(\theta)$ consists of all points in P lying on or above $\ell_1(\theta)$, $P_i(\theta)$ for $1 < i \leq k-1$ consists all points in P lying on or above $\ell_i(\theta)$ and below $\ell_{i-1}(\theta)$, and $P_k(\theta)$ consists of the rest that lies below $\ell_{k-1}(\theta)$.

For $i = 1, \ldots, k$, let $\sigma_i(\theta)$ be the minimum-width slab in orientation θ that encloses $P_i(\theta)$. Let $\mathsf{S}(\theta) := (\sigma_1(\theta), \ldots, \sigma_k(\theta))$ be the k-slab cover of P consisting of these k parallel slabs. Let $\gamma_1(\theta), \ldots, \gamma_{k-1}(\theta)$ be the gaps of $\mathsf{S}(\theta)$. Note that $\mathsf{S}(\theta)$ respects the given separator R by definition. Thus, our goal is to find an optimal orientation $\theta^* \in (0, \pi)$ that minimizes the width of $\mathsf{S}(\theta)$ for all θ such that the gap-ratio of $\mathsf{S}(\theta)$ is at least the given threshold ρ.

By an abuse of notations, we let $w_i(\theta) = w(\sigma_i(\theta))$ for $i = 1, \ldots, k$ and $g_i(\theta) = w(\gamma_i(\theta))$ for $i = 1, \ldots, k-1$. Also, we let $w(\theta) = w(\mathsf{S}(\theta))$, $g(\theta) = g(\mathsf{S}(\theta))$, $b(\theta) = b(\mathsf{S}(\theta))$, and $\rho(\theta) = \rho(\mathsf{S}(\theta))$. By definition, note that $w(\theta) = \max w_i(\theta)$, $g(\theta) = \min g_i(\theta)$, and $\rho(\theta) = g(\theta)/b(\theta)$.

For each $\theta \in (0, \pi)$ and $1 \leq i \leq k$, we denote by $q_i^+(\theta)$ and $q_i^-(\theta)$ the two points of $P_i(\theta)$ such that every point of $P_i(\theta)$ lies in between the two lines ℓ^+ and ℓ^- in orientation θ passing through $q_i^+(\theta)$ and $q_i^-(\theta)$, respectively, and ℓ^+ lies above ℓ^-. We call $q_i^+(\theta)$ and $q_i^-(\theta)$ the *extreme* points of $P_i(\theta)$. Observe that the i-th slab $\sigma_i(\theta)$ is determined by the two lines in orientation θ through $q_i^+(\theta)$ and $q_i^-(\theta)$, and the i-th gap $\gamma_i(\theta)$ by the two lines in orientation θ through $q_i^-(\theta)$ and $q_{i+1}^+(\theta)$. This implies that their widths $w_i(\theta)$ and $g_i(\theta)$ are *sinusoidal* functions over a subdomain in which $q_i^+(\theta)$, $q_i^-(\theta)$, and $q_{i+1}^+(\theta)$ remain the same [5].

Note that if $P_i(\theta)$ consists of a single point p, we have $q_i^+(\theta) = q_i^-(\theta) = p$ and $w_i(\theta) = 0$.

If $P_i(\theta) = \emptyset$, $q_i^+(\theta)$ and $q_i^-(\theta)$ are not defined. Since the i-th slab σ_i contains no point of P, the k-slab $\mathsf{S}(\theta)$ is not defined. Thus, in this case, we set $w_i(\theta) = 0$ and $g_{i-1}(\theta) = g_i(\theta) = 0$ so that $g(\theta) = \rho(\theta) = 0$, and thus our algorithms filter out such orientations $\theta \in [0, \pi)$ in searching for an optimal (k, ρ)-slab cover of P.

3.1 First algorithm using $O(kn)$ space

In the following lemma, we show that each of these width functions is indeed piecewise sinusoidal and can be specified in an efficient way. This can be done by applying a geometric dualization [8, Chapter 8], the Zone Theorem [7] in the arrangement of lines, and efficient algorithms to compute the zone of a line in the arrangement [4, 15].

Lemma 1 *Each of the functions $w_1, \ldots, w_k$, $g_1, \ldots, g_{k-1}$, and b is piecewise sinusoidal with $O(n)$ breakpoints over the domain $[0, \pi)$, and an explicit description of each can be computed in $O(n \log n)$ time.*

The proof of Lemma 1 can be found in the full version.

By Lemma 1, we can also specify the functions w and g. Recall that w is the upper envelope of the w_i's and g is the lower envelope of the g_i's. Hence, we observe that w and g are piecewise sinusoidal with $O(kn)$ breakpoints, and can be computed explicitly in $O(kn \log k)$ time by a merge-sort-like recursive merging on k (or $k-1$) functions of $O(n)$ complexity. In conclusion we have the following algorithm.

Lemma 2 *Given P, $k \geq 2$, $0 < \rho \leq 1$, and a separator R as defined above, a minimum-width (k, ρ)-slab cover of P respecting R can be computed in $O(kn \log n)$ time and $O(kn)$ space, if exists.*

Proof. Here, we describe our algorithm. First, we compute the full descriptions of functions w, g, and b. For functions w and g, we apply Lemma 1 to obtain the full descriptions of $w_1, \ldots, w_k$ and $g_1, \ldots, g_{k-1}$ in $O(kn \log n)$ time. Then, we recursively compute the upper envelope of two upper envelopes: one for $w_1, \ldots, w_{\lfloor k/2 \rfloor}$ and the other for $w_{\lfloor k/2 \rfloor + 1}, \ldots, w_k$. This can be done in time linear to the complexity of the two upper envelopes, which is $O(kn)$ time, because any two sinusoidal curves cross $O(1)$ times in any subdomain of $(0, \pi)$. Since the recursion depth is bounded by $O(\log k)$, we can compute the full description of w in $O(kn \log k)$ time. Computing g can be done analogously. Note that functions w and g are piecewise sinusoidal with $O(kn)$ breakpoints, and function b is piecewise sinusoidal with $O(n)$ breakpoints.

Next, we specify the intervals of ρ-valid orientations. We call an orientation $\theta \in (0, \pi)$ *ρ-valid* if the gap-ratio $\rho(\theta)$ of $\mathsf{S}(\theta)$ is at least ρ. We can specify the intervals of ρ-valid orientations by solving equation $g(\theta) = \rho \cdot b(\theta)$. As both functions g and b are piecewise sinusoidal, we can find all the zeros of the equation in $O(kn)$ time, and these zeros are endpoints of the ρ-valid intervals. Note that the number of ρ-valid intervals is also bounded by $O(kn)$ since any two sinusoidal curves cross $O(1)$ times in any subdomain of $(0, \pi)$. At this stage, if there is no ρ-valid orientation, we report that there is no (k, ρ)-slab cover of P respecting R.

Finally, for each ρ-valid interval I, we minimize the width $w(\theta)$ of $\mathsf{S}(\theta)$ over $\theta \in I$. Since function w is piecewise sinusoidal with $O(kn)$ breakpoints and there are $O(kn)$ such intervals, we can find an optimal orientation θ^* that minimizes w over all ρ-valid orientations in total $O(kn)$ time.

Hence, the total running time is bounded by $O(kn \log n)$. It is not difficult to see that the space spent during the execution of the algorithm is $O(kn)$. $\square$

3.2 Reducing the space usage

In the following, we show how to reduce the space complexity of Lemma 2 to $O(n)$. This improved algorithm runs in an angular sweeping fashion by handling events and updating necessary invariants related to $\mathsf{S}(\theta)$ as θ increases from 0 to π.

Data structures, variables, and invariants. At any moment $\theta \in [0, \pi)$ during the execution of our algorithm, we maintain the following:

- A fully dynamic structure CH_i for each $i = 1, \ldots, k$. CH_i stores the convex hull of $P_i(\theta)$ and supports an extreme point query in a given direction, a tangent line query through a given point, a neighbor query, and an insertion/deletion in amortized $O(\log n)$ time using $O(n)$ space. Such a data structure is known by Brodal and Jacob [6].
- $2k$ lists $\mathsf{W}_1, \ldots, \mathsf{W}_k$, $\mathsf{G}_1, \ldots, \mathsf{G}_{k-1}$, and B. The list W_i stores sinusoidal functions with domain such that its tail stores the current sinusoidal form of function w_i and its predecessors store some previous sinusoidal pieces of w_i in the order. Similarly, G_i stores sinusoidal pieces of g_i, and B stores those of b. These $2k$ lists will be maintained so that the total number of elements does not exceed $5n$.

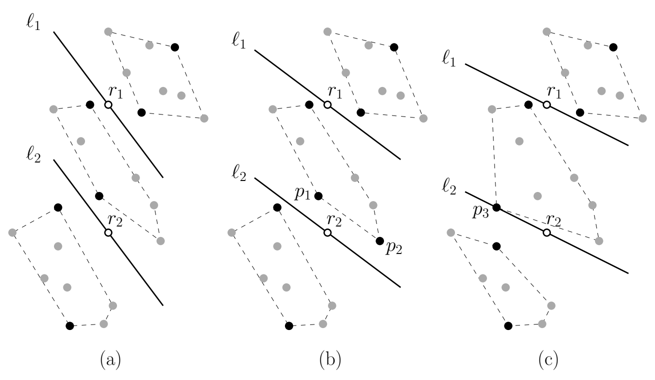

Figure 2: The convex hulls of $P_1(\theta)$, $P_2(\theta)$, and $P_3(\theta)$, respectively, are depicted by dashed lines, and the extreme points for each are shown as black dots. (b) A slab event occurs when a side p_1p_2 of the convex hull of $P_2(\theta)$ is parallel to the rotating lines. (c) A cross event occurs when ℓ_2 touches a point p_3.

- The two extreme points $q_i^+ = q_i^+(\theta)$ and $q_i^- = q_i^-(\theta)$ of $P_i(\theta)$ for each $i = 1, \ldots, k$. If $P_i(\theta) = \emptyset$, these two variables are set to `nil`.

Events. Our *events* correspond to changes of the extreme points q_i^+ and q_i^- for $i = 1, \ldots, k$. We distinguish two types of events:

- A *slab event* occurs when two or more points of P are contained in a boundary line of slab $\sigma_i(\theta)$ of $\mathsf{S}(\theta)$ for some i. Each slab event is associated with a tuple (ϕ, p^+, p^-, i), where ϕ is the orientation when the event occurs and (p^+, p^-) is the new pair of extreme points of $P_i(\theta)$ right after $\theta = \phi$.
- A *cross event* occurs when a point in P lies on $\ell_i(\theta)$ for some i. Each cross event is associated with a tuple (ϕ, p, i), where ϕ is the orientation at which the event occurs and the line $\ell_i(\theta)$ hits p at $\theta = \phi$.

Initialization. For the initialization, our algorithm sets $\theta = 0$ and computes all the above data structures and variables accordingly. Here, we extend the function $d_\theta(p, q)$ in such a way that $d_\theta(p, q) = 0$ when either p or q is `nil`. Recall that the points $r_1, \ldots, r_{k-1}$ in the given separator R lie on a vertical line, so all the lines $\ell_1(0), \ldots, \ell_{k-1}(0)$ coincide with the same vertical line. Hence, $P_1(0)$ consists of those in P lying on the right side and $P_k(0)$ the rest of those in P, while $P_2(0) = \cdots = P_{k-1}(0) = \emptyset$. For each $i = 1, \ldots, k$, we insert the points in $P_i(0)$ into CH_i after initializing CH_i to be empty, and specify the pair of extreme points (q_i^+, q_i^-) of $P_i(0)$. We initialize the lists $\mathsf{W}_1, \ldots, \mathsf{W}_k$, $\mathsf{G}_1, \ldots, \mathsf{G}_{k-1}$, and B as empty lists. For each $i = 1, \ldots, k$, we append a node with $(d_\theta(q_i^+, q_i^-), [0, \pi))$ to W_i. For each $i = 1, \ldots, k-1$, and append a node with $(d_\theta(q_i^-, q_{i+1}^+), [0, \pi))$ to G_i. And, append a node with $(d_\theta(q_1^+, q_k^-), [0, \pi))$ to B.

Additionally, we need a priority queue Q, called the *event queue*, which will store at most $O(k)$ events which will occur, prioritized by their occurring orientations. Since only $P_1(0)$ and $P_k(0)$ can be non-empty, we create next slab events and cross events for $P_1(0)$ and $P_k(0)$ by tangent line queries and neighbor queries to CH_1 and CH_k, and insert them into Q.

Main loop. In the main loop of our algorithm, until the event queue Q becomes empty, we extract the next event from Q after the current orientation θ, handle the event, and evaluate partial width functions to compute an optimal solution. The last evaluation task is performed by procedure Evaluate, which will be described later.

Let e be the event extracted from Q, and ϕ be the orientation of e. We handle e according to its type:

- Suppose e is a slab event associated with (ϕ, p^+, p^-, i). Then the pair of extreme points (q_i^+, q_i^-) of $P_i(\theta)$ has to change to (p^+, p^-) after ϕ. So we set q_i^+ to p^+ and q_i^- to p^-.

  Next, we update W_i. Let $(f, [\phi_0, \pi))$ be the tail of W_i. We modify it to $(f, [\phi_0, \phi))$ and append a new node $(d_\theta(p^+, p^-), [\phi, \pi))$ to the tail of W_i. Similarly, we update G_{i-1} and G_i if $1 < i < k$: Modify the right endpoint of the domain of the function at the tail of G_{i-1} (and also of G_i) to ϕ, and append a new node $(d_\theta(q_{i-1}^-, p^+), [\phi, \pi))$ to G_{i-1} and another $(d_\theta(p^-, q_{i+1}^+), [\phi, \pi))$ to G_i. If either $i = 1$ or $i = k$, we also modify B. If $i = 1$, we update G_1 as done above, modify the right endpoint of the domain of the function at the tail of B to ϕ, and append a new node $(d_\theta(p^+, q_k^-), [\phi, \pi))$ to B. If $i = k$, we update G_k as done above, modify the right endpoint of the domain of the function at the tail of B to ϕ, and append a new node $(d_\theta(q_1^+, p^-), [\phi, \pi))$ to B.

  Finally, we create the next possible slab event e' for $P_i(\theta)$ by finding the next extreme pair (q^+, q^-) of $P_i(\theta)$ as θ increases from ϕ. This can be done by two neighbor queries to the convex hull CH_i. To verify that this pair (q^+, q^-) indeed make the corresponding slab event occur at $\theta = \phi'$, we check the orientations of the tangents from r_i and r_{i-1} to CH_i by tangent line queries. Only if ϕ' is not bigger than these orientations of the tangents, the slab event for (q^+, q^-) occurs at ϕ'. In this case, we insert the slab event at ϕ' associated with (q^+, q^-, i).
- Suppose e is a cross event associated with (ϕ, p, i). In this case, p is about to cross line $\ell_i(\phi)$. We first update CH_i and CH_{i+1} by an insertion and a deletion of p. We then update q_i^- and q_{i+1}^+ correctly by extreme point queries to CH_i and CH_{i+1}.

Next, we update the function lists W_i, W_{i+1}, and G_i. For each of these list, if $(f, [\phi_0, \pi))$ is its tail, we modify it to $(f, [\phi_0, \phi))$. Also, we append $(d_\theta(q_i^+, q_i^-), [\phi, \pi))$ to W_i, $(d_\theta(q_{i+1}^+, q_{i+1}^-), [\phi, \pi))$ to W_{i+1}, and $(d_\theta(q_i^-, q_{i+1}^+), [\phi, \pi))$ to G_i.

Finally, we create the next cross event for $\ell_i(\theta)$ by tangent line queries to CH_i and CH_{i+1} and insert them into Q. Also, we create next slab events for $P_i(\theta)$ and for $P_{i+1}(\theta)$ with the updated extreme points, and insert each into Q after verification as described in handling a slab event.

After handling event e as above, we set θ to ϕ. If the event queue Q becomes empty, we set θ to π. Finally in the main loop, we call procedure Evaluate only if the number of handled events is divisible by n or the event queue Q becomes empty.

Procedure Evaluate. For each $\mathsf{L} \in \{\mathsf{W}_1, \ldots, \mathsf{W}_k, \mathsf{G}_1, \ldots, \mathsf{G}_{k-1}, \mathsf{B}\}$, let $(f, [\phi_0, \pi))$ be the tail of L. If $\phi_0 < \theta$, we modify the tail to $(f, [\phi_0, \theta))$ and append a new node $(f, [\theta, \pi))$ to the tail of L.

Observe that by construction the left endpoint of the domain of the function stored at the head of L is equal to some $\theta_0 < \theta$ for every list L, and the union of the domains of all nodes except the tail is exactly $[\theta_0, \theta)$. Therefore, for each $i = 1, \ldots, k$, the partial functions stored at the nodes of W_i, except its tail, form the function w_i restricted to domain $[\theta_0, \theta)$; for each $i = 1, \ldots, k-1$, those of G_i, except its tail, form the function g_i restricted to $[\theta_0, \theta)$; those of B, except its tail, form the function b restricted to $[\theta_0, \theta)$.

We compute the upper envelope w of w_i's restricted in domain $[\theta_0, \theta)$ and also the lower envelope g of g_i's restricted in domain $[\theta_0, \theta)$ as done in Lemma 2. Then, we consider the ρ-valid intervals in $[\theta_0, \theta)$ and maximize w in each ρ-valid interval. This way, we compute the minimum possible width of (k, ρ)-slab covers of P in the sub-domain $[\theta_0, \theta)$.

Finally, we delete all the nodes except the tail from each of the lists $\mathsf{W}_1, \ldots, \mathsf{W}_k$, $\mathsf{G}_1, \ldots, \mathsf{G}_{k-1}, \mathsf{B}$.

Analysis. The initialization step can be done in $O(n \log n)$ time. The main loop, except the call of procedure Evaluate takes $O(\log n)$ time per event. The number of handled events is bounded by $O(kn)$ by Lemma 1 since each event corresponds to one breakpoint of at most three of the functions $w_1, \ldots, w_k, g_1, \ldots, g_{k-1}, b$. Thus, the total time for handling events is bounded by $O(kn \log n)$. Since procedure Evaluate is called every n events and at most three nodes are appended to the lists of functions for each event, the total number of nodes stored in the lists does not exceed $2k + 3n \leq 5n$ at each call of Evaluate. Hence, the time spent for a call of Evaluate is bounded by $O(n \log k)$ and the total time spend for Evaluate is $O(kn \log k)$ since the number of calls is $O(k)$.

The structures $\mathsf{CH}_1, \ldots, \mathsf{CH}_k$ use $O(n)$ space in total [6]. The number of events stored in Q is bounded by $O(k)$ at any time. The total number of nodes stored in the function lists $\mathsf{W}_1, \ldots, \mathsf{W}_k$, $\mathsf{G}_1, \ldots, \mathsf{G}_{k-1}$, and B is bounded by $5n$ as analyzed above. Hence, the total space complexity is bounded by $O(n)$.

We finally conclude the following theorem.

Theorem 3 *Given a set P of n points, an integer $k \geq 2$, a real number $0 < \rho \leq 1$, and a separator R, a minimum-width (k, ρ)-slab cover of P respecting R can be computed in $O(kn \log n)$ time and $O(n)$ space, if exists. Otherwise, it is reported in the same time bound that there is no (k, ρ)-slab cover of P that respects R.*

4 Computing a Minimum-Width (k, ρ)-Slab Cover

In this section, we show how to compute a minimum-width (k, ρ)-slab cover of P in Problem 1. Recall that a real ρ is given, and it holds that $0 < \rho \leq 1/(k-1)$; otherwise, we just report that there is no (k, ρ)-slab cover of P.

The *directional width* of P in orientation $\theta \in [0, \pi)$ is the width of the smallest slab enclosing P in orientation θ, denoted by $d_\theta(P) := \max_{p,q \in P} d_\theta(p, q)$. A subset $K \subseteq P$ is called an *ϵ-coreset* for the directional width of P if $d_\theta(K) \geq (1 - \epsilon) d_\theta(P)$ for any orientation θ.

Our algorithm starts by computing a $(\rho/2)$-coreset $K \subseteq P$ for the directional width of P. Then, for each antipodal pair (p, q) of the convex hull of K, we generate candidate separators from the segment pq and apply the algorithm described in Theorem 3. In the following, we mainly focus on the correctness of our algorithm by proving that any (k, ρ)-slab cover of P indeed respects one of our generated separators.

4.1 Candidate separators from a coreset

Let $K \subseteq P$ be a $(\rho/2)$-coreset $K \subseteq P$ for the directional width of P.

Lemma 4 *For any (k, ρ)-slab cover $\mathsf{S} = (\sigma_1, \ldots, \sigma_k)$ of P, it holds that $K \cap \sigma_1 \neq \emptyset$ and $K \cap \sigma_k \neq \emptyset$.*

Proof. Suppose for a contradiction that there exists a (k, ρ)-slab cover $\mathsf{S}' = (\sigma'_1, \ldots, \sigma'_k)$ with gaps $\gamma'_1, \ldots, \gamma'_{k-1}$ such that $K \cap \sigma'_1 = \emptyset$ or $K \cap \sigma'_k = \emptyset$. Without loss of generality, we assume that $K \cap \sigma'_1 = \emptyset$. Let θ' denote the orientation of S'. We then have

$$d_{\theta'}(K) \geq (1 - \rho/2) b(\mathsf{S}')$$

on one hand, since K is a $(\rho/2)$-coreset of P and $b(\mathsf{S}') = d_{\theta'}(P)$. On the other hand, the assumption that $K \cap \sigma'_1 = \emptyset$ implies that

$$d_{\theta'}(K) \leq b(\mathsf{S}') - (w(\sigma'_1) + w(\gamma'_1)).$$

Plugging these two inequalities, we have

$$g(\mathsf{S}') \le w(\gamma_1') \le \rho/2 \cdot b(\mathsf{S}') - w(\sigma_1') < \rho \cdot b(\mathsf{S}'),$$

and thus $\rho(\mathsf{S}') = g(\mathsf{S}')/b(\mathsf{S}') < \rho$, a contradiction. $\square$

Furthermore, in Lemma 4, we can pick two points $p \in K \cap \sigma_1$ and $q \in K \cap \sigma_k$ such that (p, q) forms an antipodal pair of the convex hull of K.

Corollary 5 *For any (k, ρ)-slab cover $\mathsf{S} = (\sigma_1, \ldots, \sigma_k)$ of P, there is an antipodal pair (p, q) of K such that $p \in \sigma_1$ and $q \in \sigma_k$.*

For any two points p, q in the plane, let R_{pq} be the set of $\lceil 1/\rho \rceil$ points $p_j \in pq$ such that the distance from p to p_j is exactly $j \cdot |pq|/(\lceil 1/\rho \rceil + 1)$ for $1 \le j \le \lceil 1/\rho \rceil$.

Lemma 6 *For any (k, ρ)-slab cover $\mathsf{S} = (\sigma_1, \ldots, \sigma_k)$ of P, let p and q be any two points such that $p \in \sigma_1$ and $q \in \sigma_k$. Then, each gap of S contains at least one point in R_{pq}. Therefore, there is a separator $R = (r_1, \ldots, r_{k-1})$ of S such that $r_i \in R_{pq}$ for all $1 \le i \le k-1$.*

Proof. Let θ be the orientation of S and $\gamma_1, \ldots, \gamma_{k-1}$ be the gaps of S. Suppose for a contradiction that $\gamma_i \cap R_{pq} = \emptyset$ for some i with $1 \le i \le k-1$. Then, its width $w(\gamma_i)$ is small in the sense that

$$w(\gamma_i) \le \frac{d_\theta(p, q)}{\lceil 1/\rho \rceil + 1} < \rho \cdot d_\theta(p, q).$$

Since p and q are contained in the k-slab cover S, we also have $d_\theta(p, q) \le b(\mathsf{S})$, which implies that

$$g(\mathsf{S}) \le w(\gamma_i) < \rho \cdot b(\mathsf{S}),$$

a contradiction to the assumption that $\rho(\mathsf{S}) \ge \rho$. $\square$

4.2 Algorithm

Now, we are ready to describe our algorithm. Our algorithm first computes a $(\rho/2)$-coreset $K \subseteq P$ of size $O(1/\rho)$ for the directional width of P in $O(n)$ time [10, Chapter 20]. Then, it computes all antipodal pairs of the convex hull of K by using the rotating caliper method [14] after computing the convex hull of K. Finally, for each of these antipodal pair (p, q), it generates all possible $(k-1)$-combinations from the set R_{pq}, as candidate separators, and computes a minimum-width (k, ρ)-slab cover of P by applying Theorem 3.

The correctness of our algorithm is guaranteed by the above discussion through Lemmas 4 and 6. Since K is a subset of P and $|K| = O(1/\rho)$, the number of antipodal pairs is also bounded by $O(1/\rho)$. Hence, the number of generated candidate separators is bounded by $O(1/\rho^k)$. The following theorem summarizes the result.

Theorem 7 *Given a set P of n points in the plane, an integer $k \ge 2$, and a real $0 < \rho \le 1$, a minimum-width (k, ρ)-slab cover of P can be computed in $O(\rho^{-k} \cdot kn \log n)$ time and $O(n)$ space, if exists. Otherwise, it is reported in the same time bound that there is no (k, ρ)-slab cover of P.*

5 Computing a Maximum-Gap-Ratio k-Slab Cover

In this section, we present an algorithm computing a maximum-gap-ratio k-slab cover of P in Problem 2. Let $\rho_{\max}$ be the maximum possible real number such that there exists a $(k, \rho_{\max})$-slab cover of P.

Observe that procedure Evaluate in the algorithm of Theorem 3 can be easily modified for maximizing the gap-ratio, instead of minimizing the width. This is possible because procedure Evaluate indeed specifies all necessary functions $g(\theta)$ and $b(\theta)$ explicitly. With this modified version of procedure Evaluate, we have the following corollary of Theorem 7.

Corollary 8 *Given a set P of n points in the plane, an integer $k \ge 2$, and a real $0 < \rho \le 1$, a k-slab cover of P with maximum possible gap-ratio $\rho_{\max}$ can be computed in $O(\rho^{-k} kn \log n)$ time and $O(n)$ space, if $\rho \le \rho_{\max}$. Otherwise, it is reported in the same time bound that there is no (k, ρ)-slab cover of P.*

Thus, what remains is to find a value ρ with $0 < \rho < \rho_{\max}$, which is big enough. For any integer $i \ge 0$, let $\rho_i = 2^{-i/k}$. We then search for the integer t such that $\rho_{t-1} > \rho_{\max} \ge \rho_t$. This can be done by running the algorithm in Corollary 8 with $\rho = \rho_i$ for $i = 0, 1, \ldots$ in this order until it first outputs a k-slab cover of P. Then, the resulting k-slab cover of P indeed the one with maximum possible gap-ratio by Corollary 8.

In order to analyze the running time, observe that the time spent by the t applications of Corollary 8 is bounded by

$$\begin{aligned} &\sum_{i=0,\ldots,t} O(\rho_i^{-k} \cdot kn \log n) \\ &= \sum_{i=0,\ldots,t} O(2^i \cdot kn \log n) \\ &= O(2^t \cdot kn \log n) \\ &= O(\rho_t^{-k} \cdot kn \log n), \end{aligned}$$

since $\rho_i^{-k} = 2^i$. In addition, we have $\rho_t^{-k} < 2\rho_{\max}^{-k}$ since $\rho_{t-1} > \rho_{\max} \ge \rho_t$. This implies that the time complexity is bounded by $O(\rho_{\max}^{-k} \cdot kn \log n)$.

Theorem 9 *Given a set P of n points and an integer $k \ge 2$, a maximum-gap-ratio k-slab cover of P can be computed in $O(\rho_{\max}^{-k} \cdot kn \log n)$ time and $O(n)$ space, where $\rho_{\max}$ is the maximum possible gap-ratio of a k-slab cover of P.*

References

[1] P. K. Agarwal, C. M. Procopiuc, and K. R. Varadarajan. A $(1+\epsilon)$-approximation algorithm for 2-line-center. *Computational Geometry*, 26(2):119–128, 2003.

[2] P. K. Agarwal, C. M. Procopiuc, and K. R. Varadarajan. Approximation algorithms for a k-line center. *Algorithmica*, 42:221–230, 2005.

[3] P. K. Agarwal and M. Sharir. Planar geometric location problems. *Algorithmica*, 11:185–195, 1994.

[4] P. Alevizos, J.-D. Boissonnat, and F. Preparata. An optimal algorithm for the boundary of a cell in a union of rays. *Algorithmica*, 5:573–590, 1990.

[5] S. W. Bae. Minimum-width double-strip and parallelogram annulus. *Theoretical Computer Science*, 833:133–146, 2020.

[6] G. S. Brodal and R. Jacob. Dynamic planar convex hull. In *Symposium on Foundations of Computer Science*, pages 617–626. IEEE Computer Society, 2002.

[7] B. Chazelle, L. Guibas, and D.-T. Lee. The power of geometric duality. *BIT Numerical Mathematics*, 25(1):76–90, 1989.

[8] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Alogorithms and Applications.* Springer-Verlag, 2nd edition, 2000.

[9] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133, 1972.

[10] S. Har-Peled. *Geometric Approximation Algorithms.* American Mathematical Society, USA, 2011.

[11] M. E. Houle and G. T. Toussaint. Computing the width of a set. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):761–765, 1988.

[12] J. W. Jaromczyk and M. Kowaluk. The two-line center problem from a polar view: A new algorithm and data structure. In *Workshop on Algorithms and Data Structures*, pages 13–25. Springer, 1995.

[13] N. Megiddo and A. Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1:194–197, 1982.

[14] G. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE MELECON*, 1983.

[15] H. Wang. A simple algorithm for computing the zone of a line in an arrangement of lines. In *SIAM Symposium on Simplicity in Algorithms*, pages 79–86. SIAM, 2022.