symmetry

Article

Maximum Coverage by *k* Lines

Chaeyoon Chung [1], **Antoine Vigneron** [2] **and Hee-Kap Ahn** [3,*]

[1] Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang 37673, Republic of Korea; chaeyoon17@postech.ac.kr
[2] Department of Computer Science and Engineering, Ulsan National Institute of Science and Technology, Ulsan 44919, Republic of Korea; antoine@unist.ac.kr
[3] Graduate School of Artificial Intelligence, Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang 37673, Republic of Korea
* Correspondence: heekap@postech.ac.kr

Abstract: Given a set of n disks in the plane, we study the problem of finding k lines that together intersect the maximum number of input disks. We consider three variants of this problem with the following constraints on the solution: (1) no constraint on the lines, (2) the k lines should be parallel and (3) the k lines should pass through a common point. For $k = 2$, we give $O(n^3 \log n)$-time algorithms for all three cases. For any fixed $k \geq 3$, we give an $O(n^{3k/2})$-time algorithm for (1). For variants (2) and (3), the running times of our algorithms vary from $O(n^4)$ to $O(n^6)$.

Keywords: computational geometry; geometric optimization; maximum coverage; partial hitting sets; shape fitting; exact algorithm

Citation: Chung, C.; Vigneron, A.; Ahn, H.-K. Maximum Coverage by *k* Lines. *Symmetry* **2024**, *16*, 206. https://doi.org/10.3390/sym16020206

Academic Editor: Alexander Shelupanov

Received: 16 January 2024
Revised: 2 February 2024
Accepted: 7 February 2024
Published: 9 February 2024

1. Introduction

Given a set of n disks in the plane, we study the problem of finding k lines that together intersect the maximum number of input disks. In other words, we want to maximize the number of disks that intersect at least one output line. We study two other variants of this problem, where the k output lines should be parallel, and where the k lines should pass through a common point. The problems that we consider in this paper are the following. (See Figure 1).

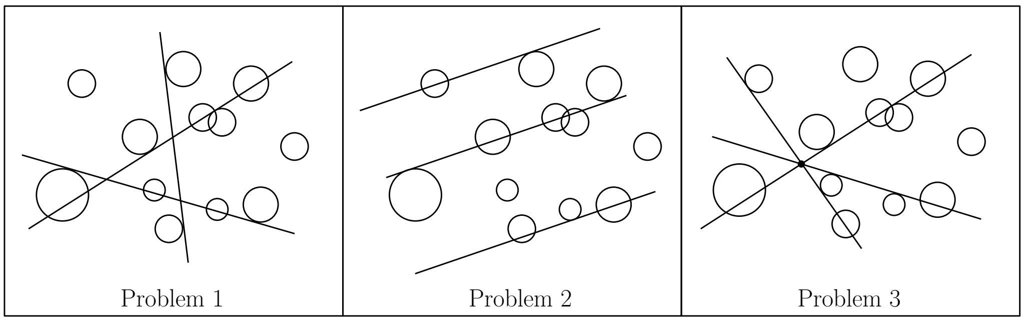

Figure 1. Examples of optimal solutions for Problems 1, 2 and 3 when $k = 3$.

Problem 1 (Maximum Coverage by k Lines)**.** *Given a set of n disks in the plane and a positive integer k, find k lines that together intersect the maximum number of input disks.*

Problem 2 (Maximum Coverage by k Parallel Lines)**.** *Given a set of n disks in the plane and a positive integer k, find k parallel lines that together intersect the maximum number of input disks.*

Problem 3 (Maximum Coverage by k Lines through a Point)**.** *Given a set of n disks in the plane and a positive integer k, find k lines that together intersect the maximum number of input disks and pass through a common point.*

In this paper, we give algorithms for the three above problems. We will assume that k is a constant. A summary of our results is given in Table 1. In particular, for $k = 1$, Problem 1, 2 and 3 coincide, and we give an $O(n^2)$-time algorithm. Given a set of points in the plane, the problem of finding a line that covers the maximum number of points is 3SUM-hard [1], so an $O(n^\alpha)$-time algorithm for any $\alpha < 2$ is currently out of reach for Problem 1, 2 and 3, even when $k = 1$.

Table 1. The running times of our algorithms for Problem 1, 2 and 3 for a constant k. Space usage is $O(n)$ if not explicitly stated.

	Problem 1	**Problem 2**	**Problem 3**
$k = 1$		$O(n^2)$ time, $O(n^2)$ space	
$k = 2$	$O(n^3 \log n)$ time	$O(n^3 \log n)$ time	$O(n^3 \log n)$ time
		$O(n^3)$ time, $O(n \log n)$ space	
$k = 3$	$O(n^{3k/2})$ time	$O(n^4)$ time	$O(n^5 \log n)$ time
			$O(n^5)$ time, $O(n^2)$ space
$k \geq 4$	$O(n^{3k/2})$ time	$O(n^4)$ time	$O(n^6)$ time

Related Work

The one-dimensional version of our problem, where we want to find a set of k points that intersect the maximum number of input intervals, is known as the *partial interval hitting set* problem [2]. Jansen et al. [3] and Chrobak et al. [4] gave an $O(kn^2)$-time algorithm.

Dumitrescu and Jiang [5] showed that the problem of hitting the maximum number of input points with k lines is APX-hard. In this paper, we consider the more general problem of hitting the maximum number of disks in the plane using k lines. To the best of our knowledge, the problem of finding k lines that together intersect the maximum number of input disks (or other geometric figures) has not been studied yet.

In the most general, combinatorial setting, we are given a collection of subsets of a larger set, and we want to find k such subsets whose union is the largest. This problem is known as the *maximum coverage problem*, and the simple greedy algorithm provides a $1 - 1/e$ approximation of the optimum [6]. Another maximum coverage problem was studied by Jin et al. They considered the problem of covering a maximum number of input points by k copies of the same rectangle or disk [7]. They provide linear-time approximation schemes, and give references to related problems.

The combinatorics of intersections between lines and balls have also been studied. In particular, Ha et al. showed that, for n disjoint spheres, if $n \geq 7$, then at most three different permutations of the spheres can appear along a line that crosses them all [8].

One motivation for studying maximum coverage by lines is to find line patterns in a two-dimensional dataset. Input disks may represent data points, each given with an imprecision that is equal to their radii. Other computational geometry problems have been studied under this imprecision model, such as the Hausdorff distance [9], the Delaunay triangulation [10] and the discrete Fréchet distance [11].

Our problem of finding line patterns is related to edge detection, which is a fundamental problem in image processing and computer vision, where the goal is to extract edges, lines or junctions from images [12]. Therefore, our algorithms could find applications in 3D reconstruction or image registration, by identifying the main lines or edges in the image. Our algorithms for lines through a point, or parallel lines, could help find symmetries in such an image.

Our algorithms could also be used for a facility location problem wherein one wants to place line facilities that are close to a set of demand regions; for instance, we may want to place k satellites at different orbits, that together cover the largest possible number of population centers. Line facility location has been studied by Cheung and Daescu, in the context of weighted regions [13].

2. Preliminaries and Notations

2.1. Partial Interval Hitting Set

In Sections 4 and 5, we use a reduction to the *partial interval hitting set* problem [2]. Given a set of n closed intervals $I_j = [s_j, t_j]$ for $j = 1, \dots, n$ on the real line, and a positive integer γ, the partial interval hitting set problem, shortly PIHS, is to find a set H of γ points that together hit the maximum number of intervals. We say that a point q hits an interval I if $q \in I$.

Chrobak et al. ([4] Section 3.1) gave a dynamic-programming algorithm for this problem which runs in $O(\gamma n^2)$ time. Their algorithm uses $O(n^2)$ space for pre-computing $O(n^2)$ values. We show in Section 6 that we can reduce the space usage to $O(\gamma n)$ while keeping the same time bound. In addition, when $\gamma = 2$, we show that we can maintain the solution for intervals with moving endpoints in $O(n)$ time per event using $O(n^2)$ space.

2.2. Notations

We denote by $\mathcal{D}$ a set of n closed disks in the plane. We will consider that a point is a disk of radius 0. We will assume that $\bigcap \mathcal{D} = \emptyset$, as otherwise, a trivial solution to our problem is to take a line that intersects $\bigcap \mathcal{D}$.

The *orientation* θ of a line is the angle that it forms with horizontal. More precisely, it is the angle swept from a horizontal line to ℓ in counterclockwise direction. Therefore, we have $\theta \in [0, \pi)$. Given an orientation $\theta \neq \pi/2$, and a disk $D \in \mathcal{D}$, the lowest line with orientation θ that is not below any point of D is called the *upper tangent* to D with orientation θ. Similarly, the highest line with orientation θ that is not above any point of D is called the *lower tangent* to D with orientation θ. When $\theta = \pi/2$, the upper tangent to D is the leftmost vertical line that is not to the left to any point of D, and the lower tangent to D is the rightmost vertical line that is not to the right to any point of D. A line is said to be *tangent* to D if it is either the lower or the upper tangent to D for some orientation. A line that is tangent to two different disks in $\mathcal{D}$ is called a *common tangent* of $\mathcal{D}$.

Given a directed line $\vec{\ell}$ in the plane, the *orientation* of $\vec{\ell}$ denotes the angle swept from the x-axis to $\vec{\ell}$ in the counterclockwise direction. The orientation of a directed line is in $[0, 2\pi)$. Given a disk D and an angle $\theta \in [0, 2\pi)$, the directed line tangent to D with orientation θ is the directed line with orientation θ that intersects the boundary of D and that has no point of D to its right.

For a disk D and a line ℓ tangent to it, we refer to the process of rotating ℓ around D while keeping it tangent to D as *rotating ℓ tangentially around D*.

We say that two disks D_1 and D_2 *cross* if $D_1 \cap D_2 \neq \emptyset$, and neither $D_1 \subset D_2$ nor $D_2 \subset D_1$. In this case, D_1 and D_2 have two common tangents. Two disks in $\mathcal{D}$ have at most four common tangents, which happens when D_1 and D_2 are disjoint and have a positive radius. Two disks are said to be *tangent* if their boundaries intersect at exactly one point. We allow disks to be points, and hence have radius 0.

3. Maximum Coverage by k Lines

Given a set $\mathcal{D}$ of n disks in the plane and a positive integer k, we would like to find k lines that together intersect the maximum number of input disks. In this section, we give efficient algorithms for this problem by reducing it to the problem of computing the depth of a collection of boxes, which is related to Klee's problem [14].

3.1. A First Algorithm

We begin with the observation below.

Lemma 1. *Let ℓ be a line tangent to a disk $D \in \mathcal{D}$. Suppose that there is no common tangent between D and any other disk in $\mathcal{D}$. Then, there is a common tangent ℓ' between two disks in $\mathcal{D}$ such that any disk intersected by ℓ is intersected by ℓ'.*

Proof. As there is no common tangent between D and any other disk in $\mathcal{D}$, each disk in $\mathcal{D}$ is either contained in D or contains D. There must be more than one disk contained in D, as otherwise, D or the disk that it contains, if it exists, is a subset of $\bigcap \mathcal{D}$, which we assumed to be empty (see Section 2.2), a contradiction.

There must be a pair of disks D_1 and D_2 that are both contained in D, and either cross or are disjoint. Otherwise, the j disks contained in D would form a chain of inclusions $D_{i_1} \subset D_{i_2} \subset \cdots \subset D_{i_j}$, and we would have $D_{i_1} \subset \bigcap \mathcal{D}$. This is a contradiction, as we assumed that $\bigcap \mathcal{D} = \emptyset$.

We take ℓ' to be a common tangent to D_1 and D_2. Then, ℓ' intersects all the disks that contain D_1, and since ℓ only intersects these disks, the set of disks intersected by ℓ' is a superset of those intersected by ℓ. □

It follows that we can choose the k lines of our solution to be common tangents between disks in $\mathcal{D}$:

Lemma 2. *There exists an optimal set of k lines, each of which is a common tangent of $\mathcal{D}$.*

Proof. Let ℓ_1 be a line in an optimal set of k lines. We translate ℓ_1 in a direction orthogonal to itself until it first becomes tangent to some disk $D_1 \in \mathcal{D}$. Let ℓ_2 denote this translated line.

Suppose that there is a common tangent between D_1 and some other disk in $\mathcal{D}$. We rotate ℓ_2 tangentially around D_1 until it first becomes a line ℓ_3 tangent to some other disk D_2. Then, the set of disks intersected by ℓ_3 contains the set of disks intersected by ℓ_1. Therefore, we can replace ℓ_1 with ℓ_3 in our optimal solution.

On the other hand, if there is no common tangent between D_1 and any other disk in $\mathcal{D}$, then by Lemma 2, we can replace ℓ_1 in our solution with a line ℓ' that is a common tangent. We repeat this process until all the lines in our solution are common tangents. □

Therefore, the maximum intersection problem can be solved as follows. We first compute all the common tangents of $\mathcal{D}$. Among these $O(n^2)$ lines, we choose k of them and count the number of disks that are intersected by the chosen lines. Then, we return the k lines that intersect the largest number of disks. It takes $O(n^{2k+1})$ time in total. In the two sections below, we present more efficient algorithms.

3.2. Maximum Coverage by One Line

In this section, we assume that $k = 1$. We give an $O(n^2)$-time algorithm to solve the maximum coverage problem, which improves on the $O(n^3)$-time algorithm described above. By Lemma 2, we only need to find a common tangent of $\mathcal{D}$ that intersects the maximum number of disks in $\mathcal{D}$. Using the duality transformation, we can find an optimal line and the set of disks intersected by this line in $O(n^2)$ time as follows:

Theorem 1. *Given a set $\mathcal{D}$ of n disks in the plane, we can find a line that intersects the maximum number of disks in $O(n^2)$ time.*

Proof. We use a standard point-line duality transformation ([15] Chapter 8). A line $\ell : y = ax - b$ in the primal plane corresponds to a point $\ell^* = (a, b)$ in the dual plane. For a disk D, let $\mathcal{U}_D$ be the set of all upper tangents to D. Then, the set of points $\mathcal{U}_D^* = \{\ell^* : \ell \in \mathcal{U}_D\}$ forms a curve in the dual plane. Analogously, we define $\mathcal{V}_D$ to be the set of all lower tangents to D and consider the curve $\mathcal{V}_D^*$ formed by the points $\{\ell^* : \ell \in \mathcal{V}_D\}$. We compute these two dual curves for all disks in $\mathcal{D}$ and compute the arrangement of them. As these curves are algebraic curves of constant degree, their arrangement $\mathcal{A}$ has a $O(n^2)$ size, and we can compute $\mathcal{A}$ in $O(n^2)$ time [16].

Let $D \in \mathcal{D}$. We rotate an upper tangent ℓ of D tangentially around D and update the number of disks intersected by ℓ each time it becomes tangent to another disk, or ceases to be tangent to it. In the dual plane, this simply means that we follow the dual curve $\mathcal{U}_D^*$ in $\mathcal{A}$, and add or remove the disk corresponding to any dual curve that we cross. Therefore, it can be achieved in $O(1)$ time per vertex of $\mathcal{A}$ that is along $\mathcal{U}_D^*$, and an additional $O(n)$ time to find the number of disks that are intersected when we start rotating ℓ.

We perform the same traversal for each disk in $\mathcal{D}$, and we perform it for lower tangents as well. In total, it takes time $O(n^2)$ as $\mathcal{A}$ has a size $O(n^2)$. During this traversal, we record the best line that was found so far, and we return it after all the traversals are completed. □

3.3. Maximum Coverage by $k \geq 2$ Lines

By Lemma 2, there is a subset $\bar{\mathcal{D}} = \{D_1, \dots, D_k\} \subseteq \mathcal{D}$ of size k and an optimal solution $\ell_1, \dots, \ell_k$ such that each line ℓ_i is tangent to D_i. We show that, if we know $\bar{\mathcal{D}}$, we are able to find an optimal solution by reducing the problem to *the depth problem* [17]: Given a set of n boxes in $\mathbb{R}^d$, find a point $p \in \mathbb{R}^d$ that maximizes the number of boxes containing p. It is known that the depth problem can be solved in $O(n \log n)$ time when $d = 2$ [18], and in $O(n^{d/2})$ time when $d > 2$ using linear space [17].

Lemma 3. *Given a subset $\bar{\mathcal{D}} = \{D_1, \dots, D_k\} \subset \mathcal{D}$ of size k, we can find k lines $\ell_1, \dots, \ell_k$ such that each line ℓ_i is tangent to D_i and the number of intersected disks in $\mathcal{D}$ is maximized in $O(n \log n)$ time when $k = 2$ and in $O(n^{k/2})$ time when $k \geq 3$.*

Proof. We first show that finding k such tangent lines can be reduced to the depth problem. For a disk $D \in \bar{\mathcal{D}}$, let $\vec{\ell}(\theta)$ be the directed line tangent to D with orientation θ. Let D' be another disk in $\mathcal{D} \setminus \bar{\mathcal{D}}$. While increasing θ from 0 to 2π, the tangent $\vec{\ell}(\theta)$ intersects D' in at most two intervals of angles in $[0, 2\pi]$. We say that these intervals are *induced by* (D, D').

For each disk $D' \in \mathcal{D} \setminus \bar{\mathcal{D}}$, we compute the intervals induced by (D, D') for every $D \in \bar{\mathcal{D}}$. Then, there are at most $2k$ such intervals corresponding to D'.

Let $i \in \{1, \dots, l\}$ and let S_i, S_i' be the intervals induced by (D_i, D'), where S_i' is possibly empty. We associate to these intervals S_i and S_i' the slab consisting of the points $(\theta_1, \dots, \theta_k) \in [0, 2\pi]^k$ such that $\min S_i \leq \theta_i \leq \max S_i$, and if $S_i' \neq \emptyset$, the slab consisting of the points that satisfy $\min S_i' \leq \theta_i \leq \max S_i'$. These slabs are orthogonal to the θ_i-axis. As $k = O(1)$, the union of these slabs for all disks in $\bar{\mathcal{D}}$ has constant complexity. More precisely, the union of these slabs is the union of $O(1)$ interior-disjoint boxes. (See Figure 2). We replace the union of the slabs by these boxes.

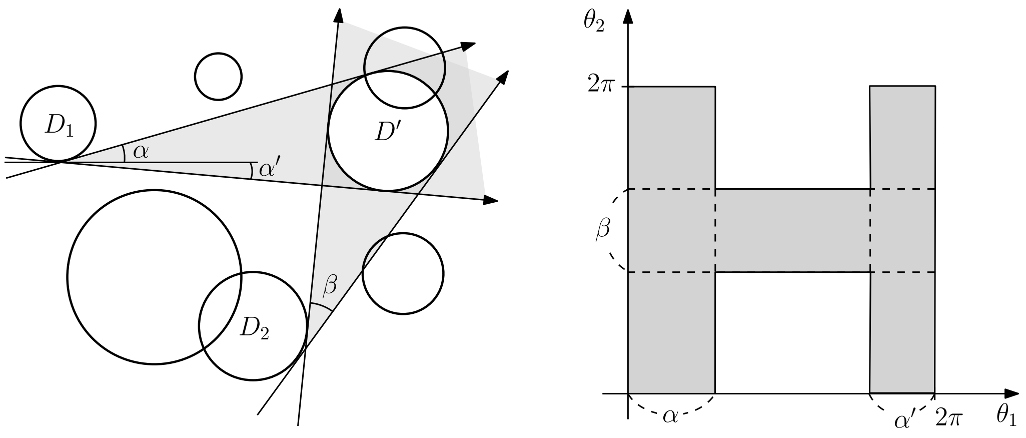

Figure 2. (**Left**) a set of seven disks in the plane, where and $k = 2$, and $\bar{\mathcal{D}} = \{D_1, D_2\}$. (**Right**) The union of slabs induced by (D_1, D') and (D_2, D').

We repeat this for every disk $D' \in \mathcal{D} \setminus \bar{\mathcal{D}}$ and obtain $O(n)$ boxes. Then, for any sequence of angles $(\theta_1, \dots, \theta_k)$ of the directed tangent lines, the number of boxes containing

$(\theta_1, \ldots, \theta_k)$ is the number of disks in $\mathcal{D} \setminus \bar{D}$ intersected by the k tangent lines. Therefore, finding an optimum solution for a fixed $\bar{D}$ reduces to the depth problem for $O(n)$ boxes.

One issue here is that we should count only once the boundary between two boxes corresponding to the same disk D', but in degenerate cases where the boundaries of two boxes overlap, we need to count twice their common boundary. The maximum depth algorithm by Chan [17] can handle this situation. $\square$

For every subset of k distinct disks of $\mathcal{D}$, we find a set of k lines such that each line is tangent to a different disk in this subset, and that together intersect the maximum number of disks in $\mathcal{D}$. Among those $O(n^k)$ sets of lines, we choose one with the largest number of intersected disks, which is an optimal solution. It follows that:

Theorem 2. *Given a set of n disks in the plane and a positive integer k, we can find k lines that together intersect the maximum number of disks in $\mathcal{D}$ in $O(n^3 \log n)$ time for $k = 2$ and in $O(n^{3k/2})$ time for $k \geq 3$, using $O(n)$ space.*

4. Maximum Coverage by k Parallel Lines

Given a set $\mathcal{D}$ of n disks in the plane and an integer $k \geq 2$, we would like to find k parallel lines that together intersect the maximum number of disks in $\mathcal{D}$. We first show that this problem can be solved by reducing it to the partial interval hitting set problem. Then, we show how to improve the running time for small k.

We begin with an observation about optimal sets of k parallel lines. Given an optimal set $\{\ell_1, \ldots, \ell_k\}$ of k parallel lines, it is always possible to translate a line of this optimal solution so that the line becomes tangent to an input disk, while keeping the same set of intersected disks. Therefore, we may assume that each line ℓ_i is tangent to a disk D_i, for $i = 1, \ldots, k$.

For each $i \in \{1, \ldots, k\}$, we may assume that there exists a common tangent between D_i and some other disk in $\mathcal{D}$. Otherwise, for each $D \in \mathcal{D} \setminus \{D_i\}$, we must have $D \subset D_i$ or $D_i \subset D$. Therefore, without loss of generality, the disks $D_1, \ldots, D_k$ form a chain of inclusion $D_1 \subset \cdots \subset D_k$. In addition, all the other disks are either contained in D_1 or contain D_k. No disk in $\mathcal{D} \setminus \{D_1, \ldots, D_k\}$ can be contained in D_1, as otherwise we could improve the solution by translating ℓ_1 until it crosses this disk. It follows that $D_1 = \bigcap \mathcal{D}$, contradicting our assumption that $\bigcap \mathcal{D} = \emptyset$.

We then rotate ℓ_i tangentially around D_i simultaneously for all $i = 1, \ldots, k$ until one of the k lines becomes tangent to another disk or ceases to be tangent to it. Let $\ell'_1, \ldots, \ell'_k$ denote the k lines after the rotation. As the set of disks intersected by ℓ'_i contains the set of disks intersected by ℓ_i for all $i = 1, \ldots, k$, we can replace our optimal set of parallel lines with $\{\ell'_1, \ldots, \ell'_k\}$. Thus we have the following observation.

Observation 1. *There exists an optimal set of k parallel lines such that every line is tangent to an input disk and at least one of them is a common tangent of $\mathcal{D}$.*

Once we know the direction of an optimal set of lines, we project the disks in $\mathcal{D}$ onto a line orthogonal to this direction. We obtain n closed intervals on this line. The problem of finding k lines with this orientation that intersect the maximum number of disks is equivalent to the problem of finding k points that together intersect the maximum number of intervals, which is the PIHS problem. In Section 6, Lemma 6, we show that it can be solved in $O(n^2)$ time using $O(n)$ space.

By Observation 1, we have $O(n^2)$ possible orientations. So we can solve the maximum intersection problem by k parallel lines in $O(n^4)$ time as k is a constant. When $k = 2$, we give two different algorithms which improve the running time and we provide a space-time trade-off.

Improvements for $k = 2$

Let D_1 and D_2 be two disks in $\mathcal{D}$. Let ℓ_1 and ℓ_2 be two parallel lines such that ℓ_i is tangent to D_i for $i = 1, 2$. For $i = 1, 2$, we maintain an array of $O(n)$ Booleans that records for each disk in $\mathcal{D}$ whether it is intersected by ℓ_i, and we keep a counter for the number of disks in $\mathcal{D}$ which are intersected by ℓ_1 or ℓ_2. Now we simultaneously rotate the two lines tangentially around D_1 and D_2, respectively.

Whenever ℓ_1 becomes tangent to a disk other than D_1, an element of the boolean list for ℓ_1 needs to be updated and the number of intersected disks may change. The same holds for ℓ_2 and D_2. We call this an *event*. As any two disks have at most 4 common tangents, there are $O(n)$ events in total. We precompute them, and sort them according to the orientation of the common tangents that they correspond to in $O(n \log n)$ time.

For each event, the number of disks in $\mathcal{D}$ intersected by ℓ_1 or ℓ_2 increases or decreases by at most one, and we can update the boolean lists and compute the number of intersected disks in $O(1)$ time. There are $O(n)$ events in total, so it takes $O(n)$ time.

We repeat this for every pair of disks in $\mathcal{D}$ and find an event where the number of intersected disks is maximized. By Observation 1, the two parallel lines ℓ_1 and ℓ_2 corresponding to this event is an optimal solution. Thus, we obtain the following result.

Lemma 4. *Given a set $\mathcal{D}$ of n disks in the plane, we can find two parallel lines that together intersect the maximum number of disks in $\mathcal{D}$ in $O(n^3 \log n)$ time using $O(n)$ space.*

We can improve the running time to $O(n^3)$ time using $O(n \log n)$ space as follows.

1. Let $T = \lfloor \log n \rfloor$. Partition the disks into $\lceil n/T \rceil$ disjoint groups, each consisting of T disks, except possibly the last group containing less than T disks. Let $\mathcal{G} = \{G_1, \ldots, G_{\lceil n/T \rceil}\}$ be the set of groups.
2. For every subset of $\mathcal{S} \subseteq \mathcal{G}$ of size at most two,
    (a) Let $\mathcal{D}_\mathcal{S} = \bigcup \mathcal{S}$. For each disk $D \in \mathcal{D}_\mathcal{S}$, compute the sorted list of the other disks in $\mathcal{D}$ intersected by the tangent line rotating around D in $O(n \log n)$ time. It takes $O(Tn \log n)$ time for all disks in $\mathcal{D}_\mathcal{S}$.
    (b) For a fixed pair $D_1, D_2 \in \mathcal{D}_\mathcal{S}$, we can find the maximum number of disks intersected by ℓ_1, ℓ_2 in $O(n)$ time as the ordering of the events has been precomputed. So over all the pairs $D_1, D_2 \in \mathcal{D}_\mathcal{S}$, it takes $O(nT^2)$ time to find optimal lines ℓ_1, ℓ_2.

We consider $O((n/T)^2)$ subsets of $\mathcal{G}$, and we spend $O(Tn \log n) + O(nT^2)$ time for each subset. Thus we spend $O((n^3 \log n)/T + n^3)$ time in total, which is $O(n^3)$ time.

Theorem 3. *Given a set of n disks in the plane and a positive integer k, we can find k parallel lines that together intersect the maximum number of disks in $\mathcal{D}$ in $O(n^4)$ time using $O(n)$ space. For $k = 2$, we can find such two parallel lines in $O(n^3 \log n)$ time using $O(n)$ space and in $O(n^3)$ time using $O(n \log n)$ space.*

5. Maximum Coverage by k Lines through a Point

Concurrent lines are lines that meet at a common point, called their *concurrency point*. Given a set $\mathcal{D}$ of n disks in the plane and a positive integer k, we want to find k concurrent lines that together intersect the maximum number of disks in $\mathcal{D}$. We can solve this problem in $O(n^2)$ time by Theorem 1 for $k = 1$.

For $k = 2$, a simple modification of the algorithm from Theorem 2 gives a solution in $O(n^3 \log n)$ time. The idea is the following. We use the plane-sweep algorithm by Asano and Imai to compute the depth of our set of rectangles [18]. We need to rule out the solutions consisting of two parallel lines, which correspond to points along the diagonal $x = y$ in our arrangement of rectangles. So at each event $x = x_i$ of the sweep, we add a negative weight -2 to the point (x_i, x_i), which guarantees that it will not be returned as an optimal solution.

Therefore, we consider the case where $k \geq 3$, and we make the assumption that no two intersecting lines together intersect all the disks in $\mathcal{D}$, as otherwise we can simply solve the problem using our algorithm for $k = 2$.

Lemma 5. *Suppose that no two intersecting lines together intersect all the disks in $\mathcal{D}$. Then there exists an optimal set of k lines for $k \geq 3$ such that every line is tangent to an input disk and at least two of them are common tangents of $\mathcal{D}$.*

Proof. Let $\mathcal{L}^* = \{\ell_1^*, \ldots, \ell_k^*\}$ be an optimal solution to our problem. Let p^* be their concurrency point. When we rotate a line ℓ_i^* about p^*, as ℓ_i^* does not intersect all the disks in $\mathcal{D}$, it must at some point become tangent to a disk in $\mathcal{D}$. So we rotate each line ℓ_i^* clockwise until it first becomes tangent to some disk in $\mathcal{D}$, obtaining a line ℓ_i. Then $\mathcal{L} = \{\ell_1, \ldots, \ell_k\}$ is still an optimal solution, and each line ℓ_i is tangent to a disk $D_i \in \mathcal{D}$.

We move the intersection point $p = \bigcap \mathcal{L}$ along ℓ_1, while rotating each ℓ_i, $i \geq 2$ tangentially around D_i. By our assumption, no line parallel to ℓ_1 intersects all the disks in $\mathcal{D}$. So ℓ_i must become tangent to some disk in $\mathcal{D} \setminus \{D_i\}$ at some point for every $i \geq 2$. Let p' be the point corresponding to the first such event, when p moves in either direction along ℓ_1. Then we obtain a new optimal solution $\mathcal{L}' = \{\ell_1', \ldots, \ell_k'\}$ such that ℓ_j' is a common tangent of $\mathcal{D}$ for some j with $2 \leq j \leq k$. If ℓ_1 is a common tangent of $\mathcal{D}$, we are done. Otherwise, we apply the same argument as above to ℓ_j', obtaining a second common tangent. $\square$

By Lemma 5, there exists an optimal set of k lines whose concurrency point is the intersection of two common tangents of $\mathcal{D}$. For every pair of common tangents (ℓ, ℓ'), we consider the point of intersection as a candidate for being the concurrency point of an optimal solution. Thus we have $O(n^4)$ candidates for being the concurrency point of an optimal solution.

Once we pick a concurrency candidate p, we can reduce our problem to the partial interval hitting set problem. Let ℓ and ℓ' be the two common tangents that intersect at p. Without loss of generality, we assume that ℓ is parallel to the x-axis. Let $\bar{\mathcal{D}}$ be the set of disks in $\mathcal{D}$ which are intersected by neither ℓ nor ℓ'. For every disk D in $\bar{\mathcal{D}}$, let $I_D \in [0, \pi)$ be the interval of angles θ such that the line through p with orientation θ intersects D. Note that I_D is a closed interval in $[0, \pi)$. Now we solve PIHS for these intervals I_D for $D \in \bar{\mathcal{D}}$, and $\gamma = k - 2$. Let $\Gamma = \{\theta_1, \ldots, \theta_{k-2}\}$ denote an optimal solution for the problem. Let $\mathcal{L}_p$ denote the set of $k - 2$ lines of orientation $\theta_1, \ldots, \theta_{k-2}$ passing through p. Since $\mathcal{L}_p$ is a set of $k - 2$ lines that pass through p and that together intersect the maximum number of disks in $\bar{\mathcal{D}}$, $\mathcal{L}_p \cup \{\ell, \ell'\}$ is an optimal set of k lines if ℓ and ℓ' are contained in an optimal solution. Therefore, we can compute an optimal set of k concurrent lines by repeating this process for every concurrency candidate.

For $k = 3$, we solve PIHS for $\gamma = 1$, so we need to find a point that hits the maximum number of intervals, which can be done in $O(n \log n)$ time by a simple scan after sorting the endpoints of the intervals. For $k \geq 4$, Chrobak et al. ([4] Section 3.1) show that PIHS can be solved in $O(n^2)$ time, and we show that it can be done using only $O(n)$ space in Section 6. Thus we can solve this problem in $O(n^5 \log n)$ time and $O(n)$ space for $k = 3$, and in $O(n^6)$ time and $O(n)$ space for $k \geq 4$.

We now show how the running time can be reduced to $O(n^5)$ for $k = 3$, using $O(n^2)$ space. Our approach is based on a result presented in Section 6 that shows how to update a solution to PIHS for a set of intervals with moving endpoints where an event occurs when two endpoints collide or separate. We show in Lemma 7 that the solution can be updated in $O(n)$ time per event.

Let ℓ denote a common tangent of $\mathcal{D}$, and assume without loss of generality that ℓ is parallel to the x-axis. We show that we can find two lines ℓ_1 and ℓ_2 such that their intersection point lies on ℓ, and the number of disks in $\mathcal{D}$ intersected by $\{\ell, \ell_1, \ell_2\}$ is maximized, in $O(n^3)$ time. As the set of disks intersected by ℓ does not change for a fixed line ℓ, we do not take this set of disks into consideration.

Let $\bar{\mathcal{D}}_\ell$ denote the set of disks in $\mathcal{D}$ that do not intersect ℓ. For a point p on ℓ, let $\mathcal{I}_p$ be the collection of intervals which are defined for $\bar{\mathcal{D}}_\ell$ around p in the same manner as we explained in the reduction step above. We solve PIHS for $\mathcal{I}_p$ and $\gamma = 2$, and let $\{\theta_1, \theta_2\}$ be the solution. Let ℓ_{θ_1} and ℓ_{θ_2} denote the two lines through p with orientations θ_1 and θ_2, respectively. These two lines together intersect the maximum number of disks in $\bar{\mathcal{D}}_\ell$ while passing through p.

Now suppose that we move p along ℓ. The endpoints of the intervals in $\mathcal{I}_p$ will move along the real line. A solution to PIHS changes only if an event occurs, as otherwise the ordering of endpoints of the interval remains the same. The key point is that an event occurs only if p is on a concurrency candidate for ℓ. The reason is that, for a point $q \in \ell$, there exists two intervals in $\mathcal{I}_q$ such that two endpoints from distinct intervals lie on the same point if and only if there exists a line which passes through q and is tangent to two disks in $\mathcal{D}$.

There are $O(n^2)$ concurrency candidates along ℓ in total, and they can be precomputed and sorted along ℓ in $O(n^2 \log n)$ time. From the pairs of disks which determine each concurrency candidate on ℓ, the corresponding event can also be found directly. We handle all the events in order and store the best two lines, $\ell_{\theta_1^*}$ and $\ell^*_{\theta_2}$, which together intersect the maximum number of disks in $\bar{\mathcal{D}}_\ell$. In Section 6, Lemma 7, we show how to handle each event in $O(n)$ time. Thus, we can find $\ell_{\theta_1^*}$ and $\ell^*_{\theta_2}$ in $O(n^3)$ time in total.

We repeat this for every common tangent of $\mathcal{D}$. By Lemma 5, there always exists an optimal set of k lines such that one of the lines is a common tangent to two disks in $\mathcal{D}$. Thus, the best set of k lines found during the repetition is an optimal solution. Hence, we have just proved the following theorem.

Theorem 4. *Given a set of n disks in the plane and a positive integer k, we can find k lines that pass through a common point and that together intersect the maximum number of disks in $O(n^6)$ time using $O(n)$ space. When $k = 3$, we can solve this problem in $O(n^5 \log n)$ time using $O(n)$ space, and $O(n^5)$ time using $O(n^2)$ space.*

6. On the Partial Interval Hitting Set Problem

As we mentioned above, given n closed intervals $I_j = [s_j, t_j]$, $j = 1, \ldots, n$ on the real line, and a positive integer γ, the partial hitting set problem is to find a set H of γ points on the real line that together hit the maximum number of intervals. We say that a point q hits an interval I if $q \in I$ [2]. It is easy to see that, after shifting the solution points to the right until they each meet a right endpoint, that we may assume that the points in the solution are the right endpoints of γ input intervals.

Chrobak et al. ([4] Section 3.1) gave a dynamic-programming algorithm for this problem that runs in $O(\gamma n^2)$ time. We give a sketch of their algorithm. First, it sorts the intervals using their right endpoints and relabels them so that $t_1 \leq t_2 \leq \ldots \leq t_n$. Let $T_{h,b}$ be the maximum number of input intervals that can be hit by a subset $H \subseteq \{t_1, t_2, \ldots, t_b\}$ such that $|H| \leq h$ and $t_b \in H$, where $h \in \{1, 2, \ldots \gamma\}$ and $b \in \{1, 2, \ldots, n\}$. Let $w_{a,b}$ be the number of intervals I_i such that $t_a < s_i \leq t_b \leq t_i$, namely the intervals that are hit by t_b but not by t_a. We first set $T_{1,b}$ to the number of intervals that contain t_b. Similarly, we set $T_{h,1}$ to the number of intervals that contain t_1. Then, for every $h = 2, 3, \ldots, \gamma$ and for every $b = 2, 3, \ldots, n$, we can compute $T_{h,b}$ using the recurrence relation $T_{h,b} = \max_{a<b}\{T_{h-1,a} + w_{a,b}\}$.

The output value is $\max_b T_{\gamma,b}$. Chrobak et al. use $O(n^2)$ space as their algorithm precomputes all values $w_{a,b}$ in time $O(n^2)$. We show how the space usage can be reduced to $O(\gamma n)$ without increasing the time bound.

Lemma 6. *Given n closed intervals on the real line, and a positive integer γ, we can find a set of γ points on the line that together hit the maximum number of intervals in $O(\gamma n^2)$ time using $O(\gamma n)$ space.*

Proof. We first compute the sorted list of the interval endpoints in increasing order. We let $T_{h,1}$ be the number of intervals that contain t_1 for $h = 1, \ldots, \gamma$. These values can be computed in $O(n)$ time. For a fixed b with $1 < b \leq n$, $T_{h,b}$ can be computed for all $h = 1, \ldots, \gamma$ in $O(\gamma n)$ time once $T_{h,i}$ and $w_{i,b}$ are computed for all $h = 1, \ldots, \gamma$ and all $i = 1, \ldots, b-1$.

We show that $w_{i,b+1}$ for all $i = 1, \ldots, b$ can be computed in (amortized) $O(n)$ time once $w_{i,b}$ is computed for all $i = 1, \ldots, b-1$. For any fixed b with $1 \leq b < n$, let S_b be the number of intervals I_i such that $t_b < s_i \leq t_{b+1}$, and let E_b be the number of intervals I_i such that $t_i = t_b$. Let E_b^a be the number of intervals I_i such that $t_a < s_i < t_i = t_b$. For any integers a, b with $1 \leq a < b \leq n$, $w_{a,b+1} = w_{a,b} + S_b - E_b^a$. Thus, S_b and E_b^i for all $i = 1, \ldots, b-1$ must be computed in advance when we compute $w_{i,b+1}$ from $w_{i,b}$ for all $i = 1, \ldots, b-1$. By scanning the sorted list of interval endpoints, we can compute S_b in $O(S_b)$ time and E_b^i for all $i = 1, \ldots, b-1$ in $O(n \cdot E_b)$ time. So we can compute $w_{i,b+1}$ for all $i = 1, \ldots, b-1$ in $O(S_b + n \cdot E_b)$ time. Additionally, we set $w_{b,b+1} = S_b$.

For every $b = 2 \ldots, n$, we compute $w_{a,b}$ for all $a < b$ and then compute $T_{h,b}$ for all h. It takes $O(S_{b-1} + n \cdot E_{b-1} + \gamma n)$ time for every $b = 2, \ldots, n$. As $\sum_b S_b = n$ and $\sum_b E_b = n$, it takes $O(\gamma n^2)$ time in total using $O(\gamma n)$ space. □

We now consider the case where the interval endpoints move along the real line, so each endpoint p is given as a function $p(t)$ of the time $t \in \mathbb{R}$. For two intervals I_i and I_j ($i \neq j$), let p and q denote two endpoints such that $p \in \{s_i, t_i\}$ and $q \in \{s_j, t_j\}$. Let $t_0 \in \mathbb{R}$. If $p(t_0^-) \neq q(t_0^-)$, that is, if for any $t < t_0$ that is close enough to t_0 we have $p(t) \neq q(t)$, and then $p(t_0) = q(t_0)$, we say that p and q *collide*. On the other hand, if $p(t_0) = q(t_0)$ and $p(t_0^+) \neq q(t_0^+)$, that is, if for any $t > t_0$ that is close enough to t_0 we have $p(t) \neq q(t)$, then we say that p and q *separate*. We say that an *event* occurs when two endpoints collide or separate. An event at time t_0 is one of the following types:

(LL) Two points p and q, which are both left endpoints of intervals, collide or separate. (See Figure 3a).
(LR) Two points p and q, where p is the left endpoint of an interval and q is the right endpoint of another interval, satisfy $p(t_0^-) < q(t_0^-)$ and collide at t_0, or p and q separate at t_0 and $p(t_0^+) < q(t_0^+)$. (See Figure 3b).
(RL) Two points p and q, where p is the right endpoint of an interval and q is the left endpoint of another interval, satisfy $p(t_0^-) < q(t_0^-)$ and collide at t_0, or p and q separate at t_0 and $p(t_0^+) < q(t_0^+)$. (See Figure 3c).
(RR) Two points p and q, which are both right endpoints of intervals, collide or separate. (See Figure 3d).

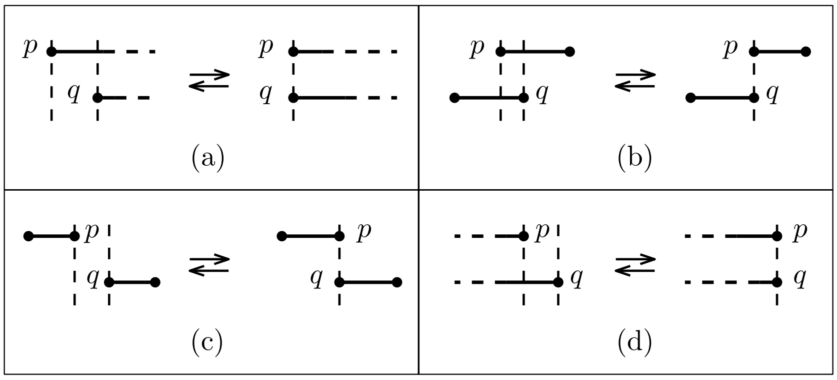

Figure 3. Types of events where two interval endpoints p and q collide or separate. (**a**) Type LL (**b**) Type LR (**c**) Type RL (**d**) Type RR

Lemma 7. *At each event, we can update an optimal solution to* PIHS *for $\gamma = 2$ in $O(n)$ time after $O(n^2)$-time preprocessing using $O(n^2)$ space.*

Proof. Let $\mathcal{W}$ denote the table which stores all $w_{a,b}$, and let $\mathcal{T}$ denote the table which stores all $T_{h,b}$. For $\gamma = 2$, we show that an event where two interval endpoints p and q collide or separate can be handled in $O(n)$ time so that all elements of $\mathcal{W}$ and $\mathcal{T}$ store correct values reflecting the situation right after the event. We first assume that no three intervals endpoints coincide at any time. At the end of this proof, we explain how to handle these degenerate cases.

For an event of type LL or LR, the set of intervals which are hit by a right endpoint does not change at all. Thus nothing in $\mathcal{W}$ and $\mathcal{T}$ needs to change.

For an event of type RL, let I_i and I_j be the two intervals involved in the event such that $t_i = p$ and $s_j = q$. Then at this event, the set of intervals that are hit by t_i will change, and the set of intervals hit by any other point does not change. The number of intervals hit by t_i increases by 1 for the case where p and q collide, and this number decreases by 1 for the other case where p and q separate. It follows that among all $T_{1,b}$ for $b = 1, \dots, n$, $T_{1,i}$ is the only element which changes. The update of $T_{1,i}$ can be done in $O(1)$ time. Among all $T_{2,b}$ for $b = 1, \dots, n$, however, more than one element can change. We show that we can compute the changes in $O(n)$ time in total. For both cases where the points collide or separate, $w_{a,b}$ changes only if $a = i$ or $b = i$ in the following way. The value $w_{a,i}$ increases or decreases by 1 for all $a = 1, \dots, i-1$. The value $w_{i,b}$ increases or decreases by 1 if $t_b \leq t_j$ for $b = i+1, \dots, n$. Thus we can update $\mathcal{W}$ in $O(n)$ time in total.

The observation that $w_{a,b}$ changes only if $a = i$ or $b = i$ implies that $T_{2,b}$ changes only if $T_{1,i} + w_{i,b}$ becomes larger than its original value for $b = i+1, \dots, n$. This can be checked in $O(1)$ time for every $b = i+1, \dots, n$ once $T_{1,i}$ and $w_{i,b}$ were computed. Using additional $O(n)$ time, we update $T_{2,i}$ using the recurrence relation. Thus we can update $\mathcal{T}$ in $O(n)$ time in total. Then we can report a new solution by computing $\max_b T_{2,b}$.

For an event of type RR, let I_i and I_j be the two intervals involved in the event such that $t_i = p$, $t_j = q$, and $p < q$ holds right after the event if p and q separate at this event. If $i > j$, which happens only for the case where p and q separate, we just switch the labels of two intervals so that $i < j$ holds. Then I_j is the only interval such that the set of intervals which are hit by its right endpoint changes by this event. The size of the set increases by 1 for the case where p and q collide, and the size of the set decreases by 1 for the case where p and q separate. It follows that among all $T_{1,b}$ for $b = 1, \dots, n$, $T_{1,j}$ is the only element which changes. The update of $T_{1,j}$ can be done in $O(1)$ time. Among all $T_{2,b}$ for $b = 1, \dots, n$, however, more than one element can change. We show that we can compute the changes in $O(n)$ time in total. For both cases where points collide or separate, $w_{a,b}$ changes only if $b = j$. More precisely, the value $w_{a,j}$ increases or decreases by 1 if $t_a < s_i$. Thus we can update $\mathcal{W}$ in $O(n)$ time in total.

The observation that $T_{1,j}$ is the only element which changes among all $T_{1,b}$ for $b = 1, \dots, n$ implies that $T_{2,b}$ changes only if $T_{1,j} + w_{j,b}$ becomes larger than its original value for $b = j+1, \dots, n$. This can be checked in $O(1)$ time for every $b = j+1, \dots, n$ once $T_{1,j}$ and $w_{j,b}$ were computed. Using additional $O(n)$ time, we update $T_{2,j}$ using the recurrence relation. Thus we can update $\mathcal{T}$ in $O(n)$ time in total. Then we can report a new solution by computing $\max_b T_{2,b}$.

We now explain how to handle degenerate cases where more than 2 endpoints collide or separate at the same time t. So we have several events of the type LL, LR, RL or RR occurring at time t. In this case, we first separately handle in an arbitrary order each of the events where two points collide, in the way that is described above. Then we obtain the solution at time t with maximum coverage. After this, we handle in an arbitrary order all the events where two points separate at time t. □

7. Discussion and Conclusions

We addressed the problem of finding k lines that together intersect the maximum number of input disks. We considered two other variants, where the k output lines should be parallel, and where the k lines should pass through a common point. We presented the first algorithms for these problems.

For $k = 1$, the three problems coincide, and we give an $O(n^2)$ time algorithm by applying a geometric dualization. As this problem is 3SUM-hard even for covering points, an $O(n^\alpha)$-time algorithm for any $\alpha < 2$ is currently out of reach.

For the problem of finding $k \geq 2$ lines that together intersect the maximum number of input disks, we first show that there exists an optimal set of k lines, each of which is tangent to two input disks. Using this observation, we show that the problem can be reduced to the problem of computing the depth of a set of boxes. The running time of our algorithm is $O(n^3 \log n)$ when $k = 2$ and is $O(n^{3k/2})$ time when $k \geq 3$.

For the problem of finding k parallel lines that together intersect the maximum number of input disks for $k \geq 2$, it can be reduced to the Partial Interval Hitting Set problem once the direction of the output lines is fixed. We first give $O(n^4)$-time algorithm by observing that there exists an optimal set of k parallel lines such that every line is tangent to an input disk and at least one of them is tangent to two input disks in $\mathcal{D}$. Then we reduce the time complexity when $k = 2$, at the expense of increasing the space usage by a logarithmic factor.

Our results are the first nontrivial results of these three problems, which are maximum coverage problems in geometric settings. One natural question is to extend our results to other geometric settings, for instance to covering balls by lines in $\mathbb{R}^3$. Another possible direction for further work is to consider approximation algorithms. The maximum coverage problem for arbitrary sets is known to be NP-hard, and the straightforward greedy algorithm gives a $(1 - 1/e)$-approximation of the optimum. Can we find better approximation algorithms in geometric settings?

Author Contributions: Conceptualization, methodology and writing: C.C., A.V. and H.-K.A. All authors have read and agreed to the published version of the manuscript.

Funding: H.-K.Ahn was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (RS-2023-00219980). C.Chung 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)). A. Vigneron was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2022R1F1A107586911).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gajentaan, A.; Overmars, M. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.* **2012**, *45*, 140–152. [CrossRef]
2. Damaschke, P. Refined algorithms for hitting many intervals. *Inf. Process. Lett.* **2017**, *118*, 117–122. [CrossRef]
3. Jansen, K.; Scheffler, P.; Woeginger, G. The disjoint cliques problem. *RAIRO Rech. OpÉrationnelle* **1997**, *31*, 45–66. [CrossRef]
4. Chrobak, M.; Golin, M.; Lam, T.W.; Nogneng, D. Scheduling with gaps: New models and algorithms. *J. Sched.* **2021**, *24*, 381–403. [CrossRef]
5. Dumitrescu, A.; Jiang, M. On the approximability of covering points by lines and related problems. *Comput. Geom.* **2015**, *48*, 703–717. [CrossRef]
6. Hochbaum, D.S.; Pathria, A. Analysis of the greedy approach in problems of maximum k-coverage. *Nav. Res. Logist.* **1998**, *45*, 615–627. [CrossRef]
7. Jin, K.; Li, J.; Wang, H.; Zhang, B.; Zhang, N. Near-linear time approximation schemes for geometric maximum coverage. *Theor. Comput. Sci.* **2018**, *725*, 64–78. [CrossRef]
8. Ha, J.; Cheong, O.; Goaoc, X.; Yang, J. Geometric permutations of non-overlapping unit balls revisited. *Comput. Geom.* **2016**, *53*, 36–50. [CrossRef]
9. Knauer, C.; Löffler, M.; Scherfenberg, M.; Wolle, T. The directed Hausdorff distance between imprecise point sets. *Theor. Comput. Sci.* **2011**, *412*, 4173–4186. [CrossRef]
10. Löffler, M.; Snoeyink, J. Delaunay triangulation of imprecise points in linear time after preprocessing. *Comput. Geom. Theory Appl.* **2010**, *43*, 234–242. [CrossRef]

11. Ahn, H.; Knauer, C.; Scherfenberg, M.; Schlipf, L.; Vigneron, A. Computing the Discrete Fréchet distance with imprecise input. *Int. J. Comput. Geom. Appl.* **2012**, *22*, 27–44. [CrossRef]
12. Ziou, D.; Tabbone, S. Edge detection techniques—An overview. *Pattern Recognit. Image Anal. C/C Raspoznavaniye Obraz. I Anal. Izobr.* **1998**, *8*, 537–559.
13. Cheung, Y.K.; Daescu, O. Line segment facility location in weighted subdivisions. In *Algorithmic Aspects in Information and Management*; Goldberg, A.V., Zhou, Y., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 100–113.
14. Klee, V. Can the Measure of $\bigcup_1^n [a_i, b_i]$ be Computed in Less Than $O(n \log n)$ Steps? *Am. Math. Mon.* **1977**, *84*, 284.
15. Berg, M.d.; Cheong, O.; Kreveld, M.v.; Overmars, M. *Computational Geometry: Algorithms and Applications*, 3rd ed.; Springer: Berlin/Heidelberg, Germany, 2008.
16. Amato, N.; Goodrich, M.; Ramos, E. Computing the arrangement of curve segments: Divide-and-conquer algorithms via sampling. In Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, USA, 9–11 January 2000; pp. 705–706.
17. Chan, T.M. Klee's Measure Problem Made Easy. In Proceedings of the 54th Annual Symposium on Foundations of Computer Science, Berkeley, CA, USA, 26–29 October 2013; pp. 410–419.
18. Imai, H.; Asano, T. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *J. Algorithms* **1983**, *4*, 310–323. [CrossRef]