Unit-Disk Range Searching and Applications

Haitao Wang
Department of Computer Science, Utah State University, Logan, UT, USA

Abstract

Given a set P of n points in the plane, we consider the problem of computing the number of points of P in a query unit disk (i.e., all query disks have the same radius). We show that the main techniques for simplex range searching can be adapted to this problem. For example, by adapting Matoušek's results, we can build a data structure of $O(n)$ space so that each query can be answered in $O(\sqrt{n})$ time; alternatively, we can build a data structure of $O(n^2/\log^2 n)$ space with $O(\log n)$ query time. Our techniques lead to improvements for several other classical problems in computational geometry.

1. Given a set of n unit disks and a set of n points in the plane, the batched unit-disk range counting problem is to compute for each disk the number of points in it. Previous work [Katz and Sharir, 1997] solved the problem in $O(n^{4/3}\log n)$ time. We give a new algorithm of $O(n^{4/3})$ time, which is optimal as it matches an $\Omega(n^{4/3})$-time lower bound. For small χ, where χ is the number of pairs of unit disks that intersect, we further improve the algorithm to $O(n^{2/3}\chi^{1/3}+n^{1+\delta})$ time, for any $\delta>0$.
2. The above result immediately leads to an $O(n^{4/3})$ time optimal algorithm for counting the intersecting pairs of circles for a set of n unit circles in the plane. The previous best algorithms solve the problem in $O(n^{4/3}\log n)$ deterministic time [Katz and Sharir, 1997] or in $O(n^{4/3}\log^{2/3} n)$ expected time by a randomized algorithm [Agarwal, Pellegrini, and Sharir, 1993].
3. Given a set P of n points in the plane and an integer k, the distance selection problem is to find the k-th smallest distance among all pairwise distances of P. The problem can be solved in $O(n^{4/3}\log^2 n)$ deterministic time [Katz and Sharir, 1997] or in $O(n\log n+n^{2/3}k^{1/3}\log^{5/3} n)$ expected time by a randomized algorithm [Chan, 2001]. Our new randomized algorithm runs in $O(n\log n+n^{2/3}k^{1/3}\log n)$ expected time.
4. Given a set P of n points in the plane, the discrete 2-center problem is to compute two smallest congruent disks whose centers are in P and whose union covers P. An $O(n^{4/3}\log^5 n)$-time algorithm was known [Agarwal, Sharir, and Welzl, 1998]. Our techniques yield a deterministic algorithm of $O(n^{4/3}\log^{10/3} n\cdot(\log\log n)^{O(1)})$ time and a randomized algorithm of $O(n^{4/3}\log^3 n\cdot(\log\log n)^{1/3})$ expected time.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases Unit disks, disk range searching, batched range searching, distance selection, discrete 2-center, arrangements, cuttings

Digital Object Identifier 10.4230/LIPIcs.SWAT.2022.32

Related Version *Full Version*: https://arxiv.org/abs/2204.08992

Funding This research was supported in part by NSF under Grant CCF-2005323.

1 Introduction

We consider unit-disk range counting queries. Given a set P of n points in the plane, the problem is to build a data structure so that the number of points of P in D can be computed efficiently for any query unit disk D (i.e., all query disks have the same known radius).

Our problem is a special case of the general disk range searching problem in which each query disk may have an arbitrary radius. Although we are not aware of any previous work particulary for our special case, the general problem has been studied before [4, 5, 19, 29, 32]. First of all, it is well-known that the lifting method can reduce the disk range searching

in the d-dimensional space to half-space range searching in $(d+1)$-dimensional space; see, e.g., [20, 32]. For example, using Matoušek's results in 3D [27], with $O(n)$ space, each disk query in the plane can be answered in $O(n^{2/3})$ time. Using the randomized results for general semialgebraic range searching [5, 29], one can build a data structure of $O(n)$ space in $O(n^{1+\delta})$ expected time that can answer each disk query in $O(\sqrt{n}\log^{O(1)} n)$ time, where (and throughout the paper) δ denotes any small positive constant. For deterministic results, Agarwal and Matoušek's techniques [4] can build a data structure of $O(n)$ space in $O(n\log n)$ time, and each query can be answered in $O(n^{1/2+\delta})$ time.

A related problem is to report all points of P in a query disk. If all query disks are unit disks, the problem is known as *fixed-radius neighbor problem* in the literature [9, 14, 17, 18]. Chazelle and Edelsbrunner [18] gave an optimal solution (in terms of space and query time): they constructed a data structure of $O(n)$ space that can answer each query in $O(\log n + k)$ time, where k is the output size; their data structure can be constructed in $O(n^2)$ time. By a standard lifting transformation that reduces the problem to the halfspace range reporting queries in 3D, Chan and Tsakalidis [12] constructed a data structure of $O(n)$ space in $O(n\log n)$ time that can answer each query in $O(\log n + k)$ time; the result also applies to the general case where the query disks may have arbitrary radii. Refer to [1, 2, 28] for excellent surveys on range searching.

In this paper, we focus on unit-disk counting queries. By taking advantage of the property that all query disks have the same known radius, we manage to adapt the techniques for simplex range searching to our problem. We show that literally all main results for simplex range searching can be adapted to our problem with asymptotically the same performance. For example, by adapting Matoušek's result in [26], we build a data structure of $O(n)$ space in $O(n\log n)$ time and each query can be answered in $O(\sqrt{n}\log^{O(1)} n)$ time. By adapting Matoušek's result in [27], we build a data structure of $O(n)$ space in $O(n^{1+\delta})$ time and each query can be answered in $O(\sqrt{n})$ time. By adapting Chan's randomized result in [11], we build a data structure of $O(n)$ space in $O(n\log n)$ expected time and each query can be answered in $O(\sqrt{n})$ time with high probability.

In addition, we obtain the following trade-off: After $O(nr)$ space and $O(nr(n/r)^{\delta})$ time preprocessing, each query can be answered in $O(\sqrt{n/r})$ time, for any $1 \le r \le n/\log^2 n$. In particularly, setting $r = n/\log^2 n$, we can achieve $O(\log n)$ query time, using $O(n^2/\log^2 n)$ space and $O(n^2/\log^{2-\delta} n)$ preprocessing time. To the best of our knowledge, the only previous work we are aware of with $O(\log n)$ time queries for the disk range searching problem is a result in [24],[1] which can answer each general disk query in $O(\log n)$ time with $O(n^2\log n)$ space and preprocessing time.

Probably more interestingly to some extent, our techniques can be used to derive improved algorithms for several classical problems, as follows. Our results are the first progress since the previous best algorithms for these problems were proposed over two decades ago.

Batched unit-disk range counting. Let P be a set of n points and $\mathcal{D}$ be a set of m (possibly overlapping) congruent disks in the plane. The problem is to compute for all disks $D \in \mathcal{D}$ the number of points of P in D. The algorithm of Katz and Sharir [24] solves the problem in $O((m^{2/3}n^{2/3} + m + n)\log n)$ time. By using our techniques for unit-disk range searching and adapting a recent result of Chan and Zheng [13], we obtain a new algorithm of $O(n^{2/3}m^{2/3} + m\log n + n\log m)$ time. We further improve the algorithm so that the complexities are sensitive to χ, the number of pairs of disks of $\mathcal{D}$ that intersect. The runtime of the algorithm is $O(n^{2/3}\chi^{1/3} + m^{1+\delta} + n\log n)$.

[1] See Theorem 3.1 [24]. The authors noted in their paper that the result was due to Pankaj K. Agarwal.

On the negative side, Erickson [21] proved a lower bound of $\Omega(n^{2/3}m^{2/3}+m\log n+n\log m)$ time for the problem in a so-called *partition algorithm model*, even if each disk is a half-plane (note that a half-plane can be considered as a special unit disk of infinite radius). Therefore, our algorithm is optimal under Erickson's model.

Counting intersections of congruent circles. As discussed in [24], the following problem can be immediately solved using batched unit-disk range counting: Given a set of n congruent circles of radius r in the plane, compute the number of intersecting pairs. To do so, define P as the set of the centers of circles and define $\mathcal{D}$ as the set of congruent disks centered at points of P with radius $2r$. Then apply the batched unit-disk range counting algorithm on P and $\mathcal{D}$. The algorithm runs in $O(n^{4/3})$ time, matching an $\Omega(n^{4/3})$-time lower bound [21]. To the best of our knowledge, the previous best results for this problem are a deterministic algorithm of $O(n^{4/3}\log n)$ time [24] and a randomized algorithm of $O(n^{4/3}\log^{2/3} n)$ expected time [3]. Agarwal, Pellegrini, and Sharir [6] also studied the problem for circles of different radii and gave an $O(n^{3/2+\delta})$ time deterministic algorithm.

Distance selection. Let P be a set of n points in the plane. Given an integer k in the range $[1, n(n-1)/2]$, the problem is to find the k-th smallest distance among all pairwise distances of P; let λ^* denote the k-th smallest distance. Given a value λ, the decision problem is to decide whether $\lambda \geq \lambda^*$. We refer to the original problem as the optimization problem.

Chazelle [15] gave the first subquadratic algorithm of $O(n^{9/5}\log^{4/5} n)$ time. Agarwal, Aronov, Sharir, and Suri [3] presented randomized algorithms that solve the decision and optimization problems in $O(n^{4/3}\log^{2/3} n)$ and $O(n^{4/3}\log^{8/3} n)$ expected time, respectively. Goodrich [22] later gave a deterministic algorithm of $O(n^{4/3}\log^{8/3} n)$ time for the optimization problem. Katz and Sharir [24] proposed a deterministic algorithm of $O(n^{4/3}\log n)$ time for the decision problem and used it to solve the optimization problem in $O(n^{4/3}\log^2 n)$ deterministic time. Using the decision algorithm of [3], Chan's randomized technique [10] solved the optimization problem in $O(n\log n + n^{2/3}k^{1/3}\log^{5/3} n)$ expected time.

Our algorithm for the batched unit-disk range counting problem can be used to solve the decision problem in $O(n^{4/3})$ time. Combining it with the randomized technique of Chan [10], the optimization problem can now be solved in $O(n\log n + n^{2/3}k^{1/3}\log n)$ expected time.

Discrete 2-center. Let P be a set of n points in the plane. The discrete 2-center problem is to find two smallest congruent disks whose centers are in P and whose union covers P. Agarwal, Sharir, and Welzl [8] gave an $O(n^{4/3}\log^5 n)$-time algorithm. Using our techniques for unit-disk range searching, we reduce the time of their algorithm to $O(n^{4/3}\log^{10/3} n(\log\log n)^{O(1)})$ deterministic time or to $O(n^{4/3}\log^3 n(\log\log n)^{1/3})$ expected time by a randomized algorithm.

In the following, we present our algorithms for unit-disk range searching in Section 2. The other problems are discussed in Section 3. Section 4 concludes the paper. Due to the space limit, many proofs are omitted but can be found in the full paper.

2 Unit-disk range searching

In this section, we present our algorithms for the unit-disk range searching problem. Our goal is to show that the main techniques for simplex range searching can be used to solve our problem. In particular, we show that, after overcoming many difficulties, the techniques of Matoušek in [26] and [27] as well as the results of Chan [11] can be adapted to our problem with asymptotically the same performance.

We assume that the radius of unit disks is 1. In the rest of this section, unless otherwise stated, a disk refers to a unit disk. We begin with an overview of our approach.

An overview. We roughly (but not precisely) discuss the main idea. We first implicitly build a grid G of side length $1/\sqrt{2}$ such that any query disk D only intersects $O(1)$ cells of G. This means that it suffices to build a data structure for the subset $P(C')$ of the points of P in each individual cell C' of G with respect to query disks whose centers are in another cell C that is close to C'. A helpful property for processing $P(C')$ with respect to C is that for any two disks with centers in C, their boundary portions in C' cross each other at most once. More importantly, we can define a duality relationship between points in C and disk arcs in C' (and vice versa): a point p in C is dual to the arc of the boundary of D_p in C', where D_p is the disk centered at p. This duality helps to obtain a *Test Set Lemma* that is crucial to the algorithms in [11, 26, 27]. With these properties and some additional observations, we show that the algorithm for computing cuttings for hyperplanes [16] can be adapted to the disk arcs in C'. With the cutting algorithms and the Test Set Lemma, we show that the techniques in [11, 26, 27] can be adapted to unit-disk range searching for the points of $P(C')$ with respect to the query disks centered in C.

The rest of this section is organized as follows. In Section 2.1, we reduce the problem to problems with respect to pairs of cells (C, C'). Section 2.2 introduces some basic concepts and observations that are fundamental to our approach. We adapt the cutting algorithm of Chazelle [16] to our problem in Section 2.3. Section 2.4 proves the Test Set Lemma. In the subsequent subsections, we adapt the algorithms of [11, 26, 27], whose query times are all $\Omega(\sqrt{n})$ with $O(n)$ space. Section 2.8 presents the trade-offs between the preprocessing and the query time. Section 2.9 finally summarizes all results.

2.1 Reducing the problem to pairs of grid cells

For each point p in the plane, we use $x(p)$ and $y(p)$ to denote its x- and y-coordinates, respectively, and we use D_p to denote the disk centered at p. For any region A in the plane, we use $P(A)$ to denote the subset of points of P in A, i.e., $P(A) = P \cap A$.

We will compute a set $\mathcal{C}$ of $O(n)$ pairwise-disjoint square cells in the plane with the following properties. (1) Each cell has side length $1/\sqrt{2}$. (2) Every two cells are separated by an axis-parallel line. (3) For a disk D_p with center p, if p is not in any cell of $\mathcal{C}$, then $D_p \cap P = \emptyset$. (4) Each cell C of $\mathcal{C}$ is associated with a subset $N(C)$ of $O(1)$ cells of $\mathcal{C}$, such that for any disk D with center in C, every point of $P \cap D$ is in one of the cells of $N(C)$. (5) Each cell C' of $\mathcal{C}$ is in $N(C)$ for a constant number of cells $C \in \mathcal{C}$. The following is a key lemma for reducing the problem to pairs of square cells. The proof is in the full paper.

▶ **Lemma 1.**

1. *The set $\mathcal{C}$ with the above properties, along with the subsets $P(C)$ and $N(C)$ for all cells $C \in \mathcal{C}$, can be computed in $O(n \log n)$ time and $O(n)$ space.*
2. *With $O(n \log n)$ time and $O(n)$ space preprocessing, given any disk with center p, we can determine whether p is in a cell C of $\mathcal{C}$, and if yes, return the set $N(C)$ in $O(\log n)$ time.*

With Lemma 1 in hand, to solve the unit disk range searching problem, for each cell $C \in \mathcal{C}$ and each cell $C' \in N(C)$, we will preprocess the points of $P(C')$ with respect to the query disks whose centers are in C. Suppose the preprocessing time (resp. space) for each such pair (C, C') is $f(m) = \Omega(m)$, where $m = |P(C')|$. Then, by the property (5) of $\mathcal{C}$, the total preprocessing time (resp., space) for all such pairs (C, C') is $f(n)$. In the following, we will describe our preprocessing algorithm for (C, C'). Since $N(C) \subset \mathcal{C}$ and the points of P in each cell of $\mathcal{C}$ are already known by Lemma 1, $P(C')$ is available to us. To simplify the

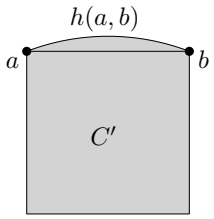

Figure 1 Illustrating $\overline{C'}$, which is the grey region.

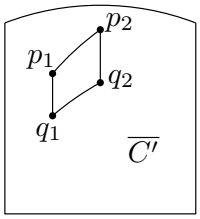

Figure 2 Illustrating an upper arc pseudo-trapezoid in $\overline{C'}$.

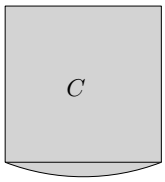

Figure 3 Illustrating $\overline{C}$, which is the grey region.

notation, we assume that all points of P are in C', i.e., $P(C') = P$. Note that if $C = C'$, then the problem is trivial because any disk centered in C' covers the entire cell. We thus assume $C \neq C'$. Due to the property (2) of $\mathcal{C}$, without loss of generality, in the following we assume that C and C' are separated by a horizontal line such that C is below the line.

2.2 Basic concepts and observations

We use $\overline{ab}$ to denote the line segment connecting two points a and b,. For any compact region A in the plane, let ∂A denote the boundary of A, e.g., if A is a disk, then ∂A is a circle.

Consider a disk D whose center is in C. As the side length of C' is $1/\sqrt{2}$, $\partial D \cap C'$ may contain up to two arcs of the circle ∂D. For this reason, we enlarge C' to a region $\overline{C'}$ so that $\partial D \cap \overline{C'}$ contains at most one arc. The region $\overline{C'}$ is defined as follows (e.g., see Fig. 1).

Let a and b be the two vertices of C' on its top edge. Let D_{ab} be the disk whose center is below $\overline{ab}$ and whose boundary contains both a and b. Let $h(a,b)$ be the arc of ∂D_{ab} above $\overline{ab}$ and connecting a and b. Define $\overline{C'}$ to be the region bounded by $h(a,b)$, and the three edges of C' other than $\overline{ab}$. As the side length of C' is $1/\sqrt{2}$, for any disk D whose center is in C, $\partial D \cap \overline{C'}$ is either $\emptyset$ or a single arc of ∂D (which is on the upper half-circle of ∂D). Let e_b denote the bottom edge of C'.

Consider a disk D. An arc h on the upper half-circle of ∂D (i.e., the half-circle above the horizontal line through its center) is called an *upper disk arc* (or *upper arc* for short); *lower arcs* are defined symmetrically. Note that an upper arc is x-monotone, i.e., each vertical line intersects it at a single point if not empty. If h is an arc of a disk D, then we say that D is the *underlying disk* of h and the center of D is also called the *center* of h. An arc h in $\overline{C'}$ is called a *spanning arc* if both endpoints of h are on $\partial\overline{C'}$. As we are mainly dealing with upper arcs of $\overline{C'}$ whose centers are in C, in the following unless otherwise stated, an upper arc always refers to one whose center is in C. The following is an easy but crucial observation that makes it possible to adapt many techniques for dealing with lines in the plane to spanning upper arcs of $\overline{C'}$.

▶ **Observation 2.** *Suppose h is an upper arc in $\overline{C'}$, and e is a vertical line segment or an upper arc in $\overline{C'}$. Then, h and e can intersect each other at most once.*

Proof. If e is a vertical segment, since h is x-monotone, h and e can intersect each other at most once. If e is an upper arc, since both e and h are upper arcs of disks whose centers are in C and they are both in $\overline{C'}$, they can intersect each other at most once. ◀

Pseudo-trapezoids. Let $h(p_1,p_2)$ be an upper arc with p_1 and p_2 as its left and right endpoints, respectively. Define $h(q_1,q_2)$ similarly, such that $x(p_1) = x(q_1)$ and $x(p_2) = x(q_2)$. Assume that $h(p_1,p_2)$ and $h(q_1,q_2)$ do not cross each other and $h(p_1,p_2)$ is above $h(q_1,q_2)$. The region σ bounded by the two arcs and the two vertical lines $\overline{p_1q_1}$ and $\overline{p_2q_2}$ is called an *upper-arc pseudo-trapezoid* (e.g., see Fig. 2). We call $\overline{p_1q_1}$ and $\overline{p_2q_2}$ the two *vertical sides* of

σ, and call $h(q_1, q_1)$ and $h(p_1, p_2)$ the *top arc* and *bottom arc* of σ, respectively. The region σ is also considered as an upper-arc pseudo-trapezoid if the bottom arc $h(q_1, q_2)$ is replaced by a line segment $\overline{q_1q_2}$ on e_b (for simplicity, we still refer to $\overline{q_1q_2}$ as the bottom-arc of σ). In this way, $\overline{C'}$ itself is an upper-arc pseudo-trapezoid. Note that for any pseudo-trapezoid σ in $\overline{C'}$ and a disk D centered in C, $\partial D \cap \sigma$ is either empty or an upper arc.

The counterparts of C (with respect to C'). The above definitions in C' (with respect to C) have counterparts in C (with respect to C') with similar properties. First, we define $\overline{C}$ in a symmetric way as $\overline{C'}$, i.e., a lower arc connecting the two bottom vertices of C' is used to bound $\partial\overline{C}$; e.g., see Fig. 3. Also, we define *lower-arc pseudo-trapezoids* and *spanning lower arcs* similarly, and unless otherwise stated, a lower arc in $\overline{C}$ refer to one whose center is in C'. In the following, unless otherwise stated, properties, algorithms, and observations for the concepts of C' with respect to C also hold for their counterparts of C with respect to C'.

Duality. We define a duality relationship between upper arcs in $\overline{C'}$ and points in C. For an upper arc h in $\overline{C'}$, we consider its center as its *dual point* in C. For a point $q \in C$, we consider the upper arc $\partial D_q \cap \overline{C'}$ as its *dual arc* in $\overline{C'}$ if it is not empty. Similarly, we define duality relationship between lower arcs in $\overline{C}$ and points in C'. Note that if the boundary of a disk centered at a point $p \in P$ does not intersect $\overline{C}$, then the point p can be ignored from P in our preprocessing because among all disks centered in C one disk contains p if and only if all other disks contain p. Henceforth, without loss of generality, we assume that ∂D_p intersects $\overline{C}$ for all points $p \in P$, implying that every point of P is dual to a lower arc in $\overline{C}$. Note that our duality is similar in spirit to the duality introduced by Agarwal and Sharir [7] between points and pseudo-lines.

2.3 Computing hierarchical cuttings for disk arcs

Let H be a set of n spanning upper arcs in $\overline{C'}$. For a compact region A of $\overline{C'}$, we use H_A to denote the set of arcs of H that intersect the relative interior of A. By adapting its definition for hyperplanes, e.g., [16, 27], a *cutting* for H is a collection Ξ of closed cells (each of which is an upper-arc pseudo-trapezoid) with disjoint interiors, which together cover the entire $\overline{C'}$. The *size* of Ξ is the number of cells in Ξ. For a parameter $1 \le r \le n$, a $(1/r)$-*cutting* for H is a cutting Ξ satisfying $|H_\sigma| \le n/r$ for every cell $\sigma \in \Xi$.

We will adapt the algorithm of Chazelle [16] to computing a $(1/r)$-cutting of size $O(r^2)$ for H. It is actually a sequence of *hierarchical cuttings*. Specifically, we say that a cutting Ξ' *c-refines* a cutting Ξ if every cell of Ξ' is contained in a single cell of Ξ and every cell of Ξ contains at most c cells of Ξ'. Let $\Xi_0, \Xi_1, \ldots, \Xi_k$ be a sequence of cuttings such that Ξ_0 consists of the single cell $\overline{C'}$ (recall that $\overline{C'}$ itself is an upper arc pseudo-trapezoid), and every Ξ_i is a $(1/\rho^i)$-cutting of size $O(\rho^{2i})$ which c-refines Ξ_{i-1}, for two constants ρ and c. In order to make Ξ_k a $(1/r)$-cutting, we set $k = \lceil \log_\rho r \rceil$. The above sequence of cuttings is called a *hierarchical* $(1/r)$-*cutting* of H. If a cell $\sigma \in \Xi_{j-1}$ contains a cell $\sigma' \in \Xi_j$, we say that σ is the *parent* of σ' and σ' is a *child* of σ. Hence, one could view Ξ as a tree structure with Ξ_0 as the root.

We have the following theorem.

▶ **Theorem 3** (The Cutting Theorem). *Let χ denote the number of intersections of the arcs of H. For any $r \le n$, a hierarchical $(1/r)$-cutting of size $O(r^2)$ for H (together with the sets H_σ for every cell σ of Ξ_i for all $0 \le i \le k$) can be computed in $O(nr)$ time; more specifically, the size of the cutting is bounded by $O(r^{1+\delta} + \chi \cdot r^2/n^2)$ and the running time of the algorithm is bounded by $O(nr^\delta + \chi \cdot r/n)$.*

▶ Remark. For a set of lower arcs in $\overline{C}$, we can define cuttings similarly with lower-arc pseudo-trapezoids as cells; the same result as Theorem 3 also holds for computing lower-arc cuttings. Also note that the algorithm is optimal if the subsets H_σ's need to be computed.

To prove Theorem 3, we adapt Chazelle's algorithm for computing cuttings for hyperplanes [16]. It was stated in [7] that Chazelle's algorithm can be extended to compute such a cutting of size $O(r^{1+\delta} + \chi \cdot r^2/n^2)$ in $O(n^{1+\delta} + \chi \cdot r/n)$ time. However, no details were provided in [7]. For completeness and also for helping the reader to better understand our cutting, we present the algorithm details in the full paper, where we actually give a more general algorithm that also works for other curves in the plane (e.g., circles or circular arcs of different radii, pseudo-lines, line segments, etc.). Note that our result reduces the factor $n^{1+\delta}$ in the above time complexity of [7] to nr^δ.

The weighted case. To adapt the simplex range searching algorithms in [11, 26, 27], we will need to compute cuttings for a weighted set H of spanning upper arcs in $\overline{C'}$, where each arc $h \in H$ has a nonnegative weight $w(h)$. The hierarchical $(1/r)$-cutting can be naturally generalized to the weighted case (i.e., the interior of each pseudo-trapezoid in a $(1/r)$-cutting can be intersected by upper arcs of H of total weight at most $w(H)/r$, where $w(H)$ is the total weight of all arcs of H). By a method in [25], any algorithm computing a hierarchical $(1/r)$-cutting for a set of hyperplanes can be converted to the weighted case with only a constant factor overhead. We can use the same technique to extend any algorithm computing a hierarchical $(1/r)$-cutting for a set of upper arcs to the weighted case.

2.4 Test Set Lemma

A critical component in all simplex range searching algorithms in [11, 26, 27] is a Test Set Lemma. Using the duality, we obtain a similar result for our problem in the following lemma, whose proof is in the full paper. For any pseudo-trapezoid σ in $\overline{C'}$, we say that an upper arc h *crosses* σ if h intersects the interior of σ.

▶ **Lemma 4** (Test Set Lemma). *For any parameter $r \le n$, there exists a set Q of at most r spanning upper arcs in $\overline{C'}$, such that for any collection Π of interior-disjoint upper-arc pseudo-trapezoids in $\overline{C'}$ satisfying that each pseudo-trapezoid contains at least $n/(c \cdot r)$ points of P for some constant $c > 0$, the following holds: if κ is the maximum number of pseudo-trapezoids of Π crossed by any upper arc of Q, then the maximum number of pseudo-trapezoids of Π crossed by any upper arc in $\overline{C'}$ is at most $O(\kappa + \sqrt{r})$.*

With our Cutting Theorem and the Test Set Lemma, we proceed to adapt the simplex range searching algorithms in [11, 26, 27] to our problem in the following subsections.

2.5 A data structure based on pseudo-trapezoidal partitions

We first extend the simplicial partition for hyperplanes in [26] to our problem, which we rename *pseudo-trapezoidal partition*. A *pseudo-trapezoidal partition* for P is a collection $\Pi = \{(P_1, \sigma_1), \ldots, (P_m, \sigma_m)\}$, where the P_i's are pairwise disjoint subsets forming a partition of P, and each σ_i is a relatively open upper-arc pseudo-trapezoid in $\overline{C'}$ containing all points of P_i. The pseudo-trapezoidal partition we will compute has the following additional property: $\max_{1 \le i \le m} |P_i| < 2 \cdot \min_{1 \le i \le m} |P_i|$, i.e., all subsets have roughly the same size. Note that the trapezoids σ_i's may overlap. The subsets P_i's are called *classes* of Π.

For any upper arc h in $\overline{C'}$, we define its *crossing number* with respect to Π as the number of pseudo-trapezoids of Π crossed by h. The crossing number of Π is defined as the maximum crossing numbers of all upper arcs h in $\overline{C'}$. The following Theorem 5 corresponds

to Theorem 3.1 [26]. Its proof, which is in the full paper, is similar to Theorem 3.1 in [26], with our Test Set Lemma and our Cutting Theorem. Note that similar result as the theorem is already known for pseudo-lines with respect to points [7].

▶ **Theorem 5** (Partition Theorem)**.** *Let s be an integer $2 \leq s < n$ and $r = n/s$. There exists a pseudo-trapezoidal partition Π for P, whose classes P_i satisfy $s \leq |P_i| < 2s$, and whose crossing number is $O(\sqrt{r})$.*

Lemma 6, which corresponds to Theorem 4.7(i) [26], computes a pseudo-trapezoidal partition and will be used in the algorithm for Theorem 7. The proof is in the full paper.

▶ **Lemma 6.** *For any fixed $\delta > 0$, if $s \geq n^{\delta}$, then a pseudo-trapezoidal partition as in the Partition Theorem (whose classes $|P_i|$ satisfy $s \leq |P_i| < 2s$ and whose crossing number is $O(\sqrt{r})$) can be constructed in $O(n \log r)$ time, where $r = n/s$.*

Using Lemma 6, we can obtain the following theorem, whose proof is in the full paper.

▶ **Theorem 7.** *A data structure of $O(n)$ space can be built in $O(n \log n)$ time, so that given a disk D centered in C, the number of points of P in D can be computed in $O(\sqrt{n}(\log n)^{O(1)})$ time.*

▶ Remark. It is easy to modify the algorithm to answer the *outside-disk queries*: compute the number of points of P *outside* any query disk, with asymptotically the same complexities. This is also the case for other data structures given later, e.g., Theorems 8, 9, 10.

2.6 A data structure based on hierarchical cuttings

Using our Cutting Theorem and the Test Set Lemma, we can adapt the techniques of Matoušek [27] to our problem. We have the following theorem, whose proof is in the full paper.

▶ **Theorem 8.** *We can build an $O(n)$ space data structure for P in $O(n^{1+\delta})$ time for any small constant $\delta > 0$, such that given any disk D whose center is in C, the number of points of P in D can be computed in $O(\sqrt{n})$ time.*

2.7 A randomized result

We have the following theorem by adapting the randomized result of Chan [11].

▶ **Theorem 9.** *We can build an $O(n)$ space data structure for P in $O(n \log n)$ expected time by a randomized algorithm, such that given any disk D whose center is in C, the number of points of P in D can be computed in $O(\sqrt{n})$ time with high probability.*

The data structure is a partition tree, denoted by T, obtained by recursively subdividing $\overline{C'}$ into cells each of which is an upper-arc pseudo-trapezoid. Each node v of T corresponds to a cell, denoted by σ_v. If v is the root, then σ_v is $\overline{C'}$. If v is not a leaf, then v has $O(1)$ children whose cells form a disjoint partition of σ_v. Define $P_v = P \cap \sigma_v$. The set P_v is not explicitly stored at v unless v is a leaf, in which case $|P_v| = O(1)$. The cardinality $|P_v|$ is stored at v. The height of T is $O(\log n)$. If κ is the maximum number of pseudo-trapezoids of T that are crossed by any upper arc in $\overline{C'}$, then $\kappa = O(\sqrt{n})$ holds with high probability. The partition tree T can be built by a randomized algorithm of $O(n \log n)$ expected time. The space of T is $O(n)$. More details for Theorem 9 are in the full paper.

2.8 Trade-offs

Using cuttings and the results of Theorems 8 and 9, we can obtain the following trade-offs between preprocessing and query time by standard techniques [2, 27]. The proof is in the full paper.

▶ **Theorem 10.**
1. *We can build an $O(nr)$ space data structure for P in $O(nr(n/r)^\delta)$ time, such that given any query disk D whose center is in C, the number of points of P in D can be computed in $O(\sqrt{n/r})$ time, for any $1 \le r \le n/\log^2 n$.*
2. *We can build an $O(nr)$ space data structure for P in $O(nr\log(n/r))$ expected time, such that given any query disk D whose center is in C, the number of points of P in D can be computed in $O(\sqrt{n/r})$ time with high probability, for any $1 \le r \le n/\log^2 n$.*

In particular, for the large space case, i.e., $r = n/\log^2 n$, we can obtain the following corollary by Theorem 10(1) (a randomized result with slightly better preprocessing time can also be obtained by Theorem 10(2)).

▶ **Corollary 11.** *We can build an $O(n^2/\log^2 n)$ space data structure for P in $O(n^2/\log^{2-\delta} n)$ time, such that given any query disk D whose center is in C, the number of points of P in D can be computed in $O(\log n)$ time.*

2.9 Wrapping things up

All above results on P are for a pair of cells (C, C') such that all points of P are in C' and centers of query disks are in C. Combining the above results with Lemma 1, we can obtain our results for the general case where points of P and query disk centers can be anywhere in the plane. The proof of Corollary 12 summarizes the overall algorithm.

▶ **Corollary 12.** *We have the following results for the unit-disk range counting problem.*
1. *An $O(n)$ space data structure can be built in $O(n\log n)$ time, with $O(\sqrt{n}(\log n)^{O(1)})$ query time.*
2. *An $O(n)$ space data structure can be built in $O(n^{1+\delta})$ time for any small constant $\delta > 0$, with $O(\sqrt{n})$ query time.*
3. *An $O(n)$ space data structure can be built in $O(n\log n)$ expected time by a randomized algorithm, with $O(\sqrt{n})$ query time with high probability.*
4. *An $O(n^2/\log^2 n)$ space data structure can be built in $O(n^2/\log^{2-\delta} n)$ time, with $O(\log n)$ query time.*
5. *An $O(nr)$ space data structure can be built in $O(nr(n/r)^\delta)$ time, with $O(\sqrt{n/r})$ query time, for any $1 \le r \le n/\log^2 n$.*
6. *An $O(nr)$ space data structure can be built in $O(nr\log(n/r))$ expected time by a randomized algorithm, with $O(\sqrt{n/r})$ query time with high probability, for any $1 \le r \le n/\log^2 n$.*

Proof. In the preprocessing, we compute the information and data structure in Lemma 1, which takes $O(n\log n)$ time and $O(n)$ space. For each pair of cells (C, C') with $C \in \mathcal{C}$ and $C' \in N(C)$, we construct the data structure on $P(C')$, i.e., $P \cap C'$, with respect to query disks centered in C, e.g., those in Theorems 7, 8, 9, and 10. As discussed before, due to property (5) of $\mathcal{C}$, the total preprocessing time and space is the same as those in the above theorems. Given a query disk D with center q, by Lemma 1(2), we determine whether q is in a cell C of $\mathcal{C}$ in $O(\log n)$ time. If no, then $D \cap P = \emptyset$ and thus we simply return 0. Otherwise, the data structure returns $N(C)$. Then, for each $C' \in N(C)$, we use the data structure constructed for (C, C') to compute $|P(C') \cap D|$. We return $|P \cap D| = \sum_{C' \in N(C)} |P(C') \cap D|$. As $|N(C)| = O(1)$, the total query time is as stated in the above theorems. ◀

▶ Remark. As in [11, 26, 27], all above results can be extended to the weighted case (or the more general semigroup model) where each point of P has a weight.

3 Applications

In this section, we show that our techniques for the disk range searching problem can be used to solve several other problems. More specifically, our techniques yield improved results for three classical problems: batched range counting, distance selection, and discrete 2-center.

3.1 Batched unit-disk range counting

Let P be a set of n points and $\mathcal{D}$ be a set of m (possibly overlapping) unit disks in the plane. The *batched unit-disk range counting* problem (also referred to as *offline range searching* in the literature) is to compute for each disk $D \in \mathcal{D}$ the number of points of P in D.

Let Q denote the set of centers of the disks of $\mathcal{D}$. For each point $q \in Q$, we use D_q to denote the unit disk centered at q.

We first apply Lemma 1 on P. For each point $q \in Q$, by Lemma 1(2), we first determine whether q is in a cell C of $\mathcal{C}$. If no, then D_q does not contain any point of P and thus it can be ignored for the problem; without loss of generality, we assume that this case does not happen to any disk of $\mathcal{D}$. Otherwise, let C be the cell of $\mathcal{C}$ that contains q. By Lemma 1(2), we further find the set $N(C)$ of C. In this way, in $O((n+m)\log n)$ time, we can compute $Q(C)$ for each cell C of $\mathcal{C}$, where $Q(C)$ is the subset of points of Q in C. Define $\mathcal{D}(C)$ as the set of disks of $\mathcal{D}$ whose centers are in $Q(C)$. Let $P(C) = P \cap C$.

In what follows, we will consider the problem for $P(C')$ and $\mathcal{D}(C)$ for each pair (C, C') of cells with $C \in \mathcal{C}$ and $C' \in N(C)$. Combining the results for all such pairs leads to the result for P and $\mathcal{D}$ (the details on this will be discussed later). To simplify the notation, we assume that $P(C') = P$ and $\mathcal{D}(C) = \mathcal{D}$ (thus $Q(C) = Q$). Hence, our goal is to compute $|P \cap D|$ for all disks $D \in \mathcal{D}$.

If $C = C'$, then all points of P are in D for each disk $D \in \mathcal{D}$ and thus the problem is trivial. Below we assume $C \neq C'$. Without loss of generality, we assume that C' and C are separated by a horizontal line and C' is above the line. We assume that each point of P defines a lower arc in $\overline{C}$ since otherwise the point can be ignored. We also assume that the boundary of each disk of $\mathcal{D}$ intersects $\overline{C}'$, i.e., each point q of Q is dual to an upper arc h_q in $\overline{C'}$, since otherwise the disk can be ignored. Observe that a point p is in D_q if and only if p is below the upper arc h_q (we say that p is below h_q if p is below the upper half boundary of D_q), for any $p \in P$ and $q \in Q$. Hence, the problem is equivalent to computing the number of points of P below each upper arc of H, where $H = \{h_q \mid q \in Q\}$.

Given a set of n points and a set of m lines in the plane, Chan and Zheng [13] recently gave an $O(m^{2/3}n^{2/3} + n\log m + m\log n)$ time algorithm to compute the number of points below each line (alternatively, compute the number of points inside the lower half-plane bounded by each line). We can easily adapt their algorithm to solve our problem. Indeed, the main techniques of Chan and Zheng's algorithm we need to adapt to our problem are the hierarchical cuttings and duality. Using our Cutting Theorem and our definition of duality, we can apply the same technique and solve our problem in $O(m^{2/3}n^{2/3} + n\log m + m\log n)$ time, with $n = |P|$ and $m = |H| = |\mathcal{D}|$. Thus we have the following theorem; the proof is in the full paper.

▶ **Theorem 13.** *We can compute, for all disks $D \in \mathcal{D}$, the number of points of P in D in $O(m^{2/3}n^{2/3} + n\log m + m\log n)$, with $n = |P|$ and $m = |\mathcal{D}|$.*

Let χ denote the number of intersections of the arcs of H, and thus $\chi = O(m^2)$. Using our Cutting Theorem and Theorem 13, we further improve the algorithm for small χ.

▶ **Theorem 14.** *We can compute, for all disks $D \in \mathcal{D}$, the number of points of P in D in $O(n^{2/3}\chi^{1/3} + m^{1+\delta} + n \log n)$ time, with $n = |P|$ and $m = |\mathcal{D}|$.*

Proof. We start with computing a hierarchical $(1/r)$-cutting $\Xi_0, \ldots, \Xi_k$ for H, where $r = \min\{m/8, (m^2/\chi)^{1/(1-\delta)}\}$ and δ refers to the parameter in the Cutting Theorem. By our Cutting Theorem, the size of the cutting, denoted by K, is bounded by $O(r^\delta + \chi \cdot r^2/m^2)$ and the time for computing the cutting is $O(mr^\delta + \chi \cdot r/m)$. Since the parameter r depends on χ, which is not available to us, we can overcome the issue by using the standard trick of doubling. More specifically, initially we set χ to a constant. Then we run the algorithm until it exceeds the running time specified based on the guessed value of χ. Next, we double the value χ and run the algorithm again. We repeat this process until when the algorithm finishes before it reaches the specified running time for a certain value of χ. In this way, we run the cutting construction algorithm at most $O(\log \chi)$ time. Therefore, the total time for constructing the desired cutting is $O((mr^\delta + \chi \cdot r/m) \log \chi)$.

Next, we reduce the problem into $O(K)$ subproblems and then solve each subproblem by Theorem 13, which will lead to the theorem.

For each point $p \in P$, we find the cell σ of Ξ_i that contains p and we store p in a *canonical subset* $P(\sigma)$ of P (which is initially $\emptyset$), for all $0 \le i \le k$, i.e., $P(\sigma) = P \cap \sigma$; in fact, we only need to store the cardinality of $P(\sigma)$. For ease of exposition, we assume that no point of P lies on the boundary of any cell of Ξ_i for any i.

For each disk $D \in \mathcal{D}$, our goal is to compute the number of points of P in D, denoted by n_D. We process D as follows. We initialize $n_D = 0$. Let h be the upper arc of H defined by D, i.e., $h = \partial D \cap \overline{C'}$. Starting from $\Xi_0 = \overline{C'}$. Suppose σ is a cell of Ξ_i crossed by h (initially, $i = 0$ and σ is $\overline{C'}$) and $i < k$. For each child cell σ' of σ in Ξ_{i+1}, if σ' is contained in D, then we increase n_D by $|P(\sigma')|$ because all points of $P(\sigma')$ are contained in D. Otherwise, if h crosses σ', then we proceed on σ'. In this way, the points of $P \cap D$ not counted in n_D are those contained in cells $\sigma \in \Xi_k$ that are crossed by h. To count those points, we perform further processing as follows.

For each cell σ in Ξ_k, if $|P_\sigma| > n/K$, then we arbitrarily partition $P(\sigma)$ into subsets of size between $n/(2K)$ and n/K, called *standard subsets* of $P(\sigma)$. As Ξ_k has $O(K)$ cells and $|P| = n$, the number of standard subsets of all cells of Ξ_k is $O(K)$. Denote by $\mathcal{D}_\sigma$ the subset of disks of $\mathcal{D}$ whose boundaries cross σ. Our problem is to compute for all disks $D \in \mathcal{D}_\sigma$ the number of points of $P(\sigma)$ contained in D, for all cells $\sigma \in \Xi_k$. To this end, for each cell σ of Ξ_k, for each standard subset $P'(\sigma)$ of $P(\sigma)$, we solve the batched unit-disk range counting problem on the point set $P'(\sigma)$ and the disk set $\mathcal{D}_\sigma$ by Theorem 13. Note that $|\mathcal{D}_\sigma| \le m/r$. As Ξ_k has $O(K)$ cells, we obtain $O(K)$ subproblems of size $(n/K, m/r)$ each. As discussed above, solving these subproblems also solves our original problem. It remains to analyze the time complexity of the algorithm, which can be found in the full paper. ◀

The general problem. The above results are for the case where points of P are in the square cell C' while centers of $\mathcal{D}$ are all in C. For solving the general problem where both P and $\mathcal{D}$ can be anywhere in the plane, as discussed before, we reduce the problem to the above case by Lemma 1. The properties of the set $\mathcal{C}$ guarantee that the complexities for the general problem are asymptotically the same as those in Theorem 13. To see this, we consider all pairs (C, C') with $C \in \mathcal{C}$ and $C' \in N(C)$. For the i-th pair (C, C'), let $n_i = |P(C')|$ and $m_i = |\mathcal{D}(C)|$. Then, solving the problem for the i-th pair (C, C') takes $O(n_i^{2/3} m_i^{2/3} + m_i \log n_i + n_i \log m_i)$

time by Theorem 13. Due to the properties (4) and (5) of $\mathcal{C}$, $\sum_i n_i = O(n)$ and $\sum_i m_i = O(m)$. Therefore, by Hölder's Inequality, $\sum_i n_i^{2/3} m_i^{2/3} \leq n^{1/3} \cdot \sum_i n_i^{1/3} m_i^{2/3} \leq n^{2/3} m^{2/3}$, and thus the total time for solving the problem for all pairs of cells is $O(n^{2/3}m^{2/3} + m \log n + n \log m)$. Similarly, the complexity of Theorem 14 also holds for the general problem, with χ as the number of pairs of disks of $\mathcal{D}$ that intersect.

Computing incidences between points and circles. It is easy to modify the algorithm to solve the following problem: Given n points and m unit circles in the plane, computing (either counting or reporting) the incidences between points and unit circles. The runtime is $O(n^{2/3}m^{2/3} + m \log n + n \log m)$ or $O(n^{2/3}\chi^{1/3} + m^{1+\delta} + n \log n)$, where χ is the number of intersecting pairs of the unit circles. Although the details were not given, Agarwal and Sharir [7] already mentioned that an $n^{2/3}m^{2/3}2^{O(\log^*(n+m))} + O((m+n)\log(m+n))$ time algorithm can be obtained by adapting Matoušek's technique [27]. (The same problem for circles of arbitrary radii is considered in [7]. Refer to [30] for many other incidence problems.) Our result further leads to an $O(n^{4/3})$-time algorithm for the *unit-distance detection* problem: Given n points in the plane, is there a pair of points at unit distance? Erickson [21] gave a lower bound of $\Omega(n^{4/3})$ time for the problem in his partition algorithm model.

3.2 The distance selection problem

Given a set P of n points in the plane and an integer k in the range $[0, n(n-1)/2]$, the distance selection problem is to compute the k-th smallest distance among the distances of all pairs of points of P. Let λ^* denote the k-th smallest distance to be computed. Given a value λ, the *decision problem* is to decide whether $\lambda \geq \lambda^*$. Using our batched range counting algorithm, we can easily obtain the following lemma.

▶ **Lemma 15.** *Given a value λ, whether $\lambda \geq \lambda^*$ can be decided in $O(n^{4/3})$ time.*

Proof. We can use our algorithm for the batched unit-disk range counting problem. Indeed, let $\mathcal{D}$ be the set of congruent disks centered at the points of P with radius λ. By Theorem 13, we can compute in $O(n^{4/3})$ time the cardinality $|\Pi|$, where Π is the set of all disk-point incidences (D, p), where $D \in \mathcal{D}$, $p \in P$, and D contains p. Observe that for each pair of points (p_i, p_j) of P whose distance is at most λ, it introduces two pairs in Π. Also, each point p_i introduces one pair in Π because p_i is contained in the disk of $\mathcal{D}$ centered at p_i. Hence, the number of pairs of points of P whose distances are at most λ is equal to $(|\Pi| - n)/2$. Clearly, $\lambda \geq \lambda^*$ if and only if $(|\Pi| - n)/2 \geq k$. ◀

Plugging Lemma 15 into a randomized algorithm of Chan [10] (i.e., Theorem 5 [10]), λ^* can be computed in $O(n \log n + n^{2/3}k^{1/3} \log n)$ expected time.

3.3 The discrete 2-center problem

Let P be a set of n points in the plane. The discrete 2-center problem is to find two smallest congruent disks whose centers are in P and whose union covers P. Let λ^* be the radius of the disks in an optimal solution. Given λ, the *decision problem* is to decide whether $\lambda \geq \lambda^*$.

Agarwal, Sharir, and Welzl [8] gave an $O(n^{4/3} \log^5 n)$ time algorithm by solving the decision problem first. A *key subproblem* in their decision algorithm [8] is: Preprocess P to compute a collection $\mathcal{P}$ of *canonical subsets* of P, $\{P_1, P_2, \ldots\}$, so that given a query point p in the plane, the set P_p of points of P *outside* the unit disk centered at p can be represented as the union of a sub-collection $\mathcal{P}_p$ of canonical subsets and $\mathcal{P}_p$ can be found efficiently (it suffices to give the "names" of the canonical subsets of $\mathcal{P}_p$). Note that here the radius of unit disks is λ.

Roughly speaking, suppose we can solve the above key subproblem with preprocessing time T such that $\sum_{P_i \in \mathcal{P}} |P_i| = M$ and $|\mathcal{P}_p|$ for any query point p is bounded by $O(\tau)$ (and $|\mathcal{P}_p|$ can be found in $O(\tau)$ time); then the algorithm of [8] can solve the decision problem in $O(T + M \log n + \tau \cdot n \log^3 n)$ time. The optimal radius λ^* can thus be found by binary search on all pairwise distances of P (in each iteration, find the k-th smallest distance using a distance selection algorithm); the total time is $O((T_1 + T_2) \log n)$, where T_1 is the time of the distance selection algorithm and T_2 is the time of the decision algorithm.

Note that the logarithmic factor of $M \log n$ in the above running time of the decision algorithm of [8] is due to that for each canonical subset $P_i \in \mathcal{P}$, we need to compute the common intersection of all unit disks centered at the points of P_i, which takes $O(|P_i| \log n)$ time [23]. However, if all points of P_i are sorted (e.g., by x-coordinate or y-coordinate), then the common intersection can be computed in $O(|P_i|)$ time [31]. Therefore, if we can guarantee that all canonical subsets are sorted, then the runtime of the decision algorithm of [8] can be bounded by $O(T + M + \tau \cdot n \log^3 n)$.

Using our techniques for unit-disk range searching, we present new solutions to the above key subproblem. We show that after $T = O(n^{4/3} \log^2 n (\log \log n)^{1/3})$ expected time preprocessing by a randomized algorithm, we can compute $M = O(n^{4/3} \log^2 n/(\log \log n)^{2/3})$ sorted canonical subsets of P so that $\tau = O(n^{1/3}(\log \log n)^{1/3}/\log n)$ holds with high probability. Consequently, the decision problem can be solved in $O(n^{4/3} \log^2 n (\log \log n)^{1/3})$ expected time, and thus λ^* can be computed in $O(n^{4/3} \log^3 n (\log \log n)^{1/3})$ expected time if we use the $O(n^{4/3} \log^2 n)$ time distance selection algorithm in [24]. We also have another slightly slower deterministic result. After $T = O(n^{4/3} \log^{7/3} n (\log \log n)^{1/3})$ time preprocessing algorithm, we can compute $M = O(n^{4/3} \log^{7/3} n/(\log \log n)^{2/3})$ sorted canonical subsets of P so that $\tau = O(n^{1/3}(\log \log n)^{O(1)}/\log^{2/3} n)$. Consequently, the decision problem can be solved in $O(n^{4/3} \log^{7/3} n (\log \log n)^{O(1)})$ time, and thus λ^* can be computed in $O(n^{4/3} \log^{10/3} n (\log \log n)^{O(1)})$ time.

▶ Remark. It is straightforward to modify our algorithms to achieve the same results for the following *inside-disk* problem: represent the subset of points of P *inside* D as a collection of pairwise-disjoint canonical sets for any query disk D.

In what follows, we present our solutions to the above subproblem. We apply Lemma 1 on the set P to compute the set $\mathcal{C}$ of square cells. As before, we first reduce the problem to the same problem with respect to pairs of cells (C, C') of $\mathcal{C}$, by using Lemma 1 as well as the following lemma (whose proof is in the full paper, by modifying of the algorithm for Lemma 1); then we will solve the problem using our techniques for disk range searching.

▶ **Lemma 16.** *We can compute in $O(n \log n)$ time a collection of $O(n)$ sorted canonical subsets of P whose total size is $O(n \log n)$, such that for any cell C of $\mathcal{C}$, there are $O(\log n)$ pairwise-disjoint canonical subsets whose union consists of the points of P that are not in the cells of $N(C)$, and we can find those canonical subsets in $O(\log n)$ time.*

Let D_p be the unit disk centered at a point p in the plane. If p is not in any cell of $\mathcal{C}$, then $D_p \cap P = \emptyset$ and thus we can return the entire set P as a canonical subset. Henceforth, we only consider the case where p is in a cell C of $\mathcal{C}$. According to Lemma 16, it suffices to find canonical subsets to cover all points of $P \cap C'$ not in D_p for all cells $C' \in N(C)$. As $|N(C)| = O(1)$, it suffices to consider one such cell $C' \in N(C)$. Hence, as before, the problem reduces to a pair of square cells (C, C') of $\mathcal{C}$ with $C' \in N(C)$. If $C' = C$, then we know that all points of $P \cap C'$ are in D_p. Hence, we assume that $C' \neq C$. Without loss of generality, we assume that C' and C are separated by a horizontal line and C is below the line. The problem is to process all points of $P \cap C'$, such that given any query disk D_p whose center p is in C, we can find a collection of disjoint canonical subsets whose union is the set of points of $P \cap C'$ not in D_p. To simplify the notation, we assume that all n points of P are in C'.

Our data structure combines some techniques for the disk range searching problem. As remarked before, all our results on disk range searching with respect to (C, C') can be applied to find the number of points of P outside any query disk D whose center is in C (indeed, the disk D defines a spanning upper arc h in $\overline{C'}$, and points in D lie on one side of h while points outside D lie on the other side of h). Hence, our main idea is to examine our disk range searching data structures and define canonical subsets of P in these data structures. For each query disk D, we apply the query algorithm on D, which will produce a collection of canonical subsects. The crux is to carefully design the disk range searching data structure (e.g., by setting parameters to some appropriate values) so that the following are as small as possible (tradeoffs are needed): the preprocessing time, the total size of all canonical subsets of the data structure, which is M, and the total number of canonical subsets for each query disk D, which is τ. In the following, whenever we say "apply our query algorithm on D", we mean "finding points outside D". We will present two results, a randomized result based on Chan's partition trees [11] and a slightly slower deterministic result.

3.3.1 The randomized result

Our data structure has three levels. We will present them from the lowest level to the highest one. We start with the lowest level in the following lemma, which relies on the partition tree T built in Theorem 9. The proof is in the full paper. For any disk D, we use $P \setminus D$ to refer to the subset of the points of P not in D.

▶ **Lemma 17.** *We can compute in $O(n \log n)$ expected time a data structure with $O(n)$ sorted canonical subsets of P whose total size is $O(n \log n)$, so that for any disk D whose center is in C, we can find in $O(\kappa)$ time $O(\kappa)$ pairwise-disjoint canonical sets whose union is $P \setminus D$, where $\kappa = O(\sqrt{n})$ holds with high probability.*

In the next lemma we add the second level to the data structure of Lemma 17. The proof is in the full paper.

▶ **Lemma 18.** *We can compute in $O(n^2 \log\log n / \log^2 n)$ expected time a data structure with $O(n^2/\log^2 n)$ sorted canonical subsets of P whose total size is $O(n^2 \log\log n/\log^2 n)$, so that for any disk D whose center is in C, we can find in $O(\kappa)$ time $O(\kappa)$ pairwise-disjoint canonical sets whose union is $P \setminus D$, where $\kappa = O(\log n)$ holds with high probability.*

We finally add the top-level data structure in Lemma 19, whose proof is in the full paper.

▶ **Lemma 19.** *For any $r < n/\log^{\omega(1)} n$, we can compute in $O(n \log n + nr \log\log r/\log^2 r)$ expected time a data structure with $O(nr/\log^2 r)$ sorted canonical subsets of P whose total size is $O(n \log(n/r) + nr \log\log r/\log^2 r)$, so that for any disk D whose center is in C, we can find in $O(\kappa)$ time $O(\kappa)$ pairwise-disjoint canonical sets whose union is $P \setminus D$, where $\kappa = O(\sqrt{n/r} \log r)$ holds with high probability.*

By setting $r = n^{1/3} \log^4 n/(\log\log n)^{2/3}$ in Lemma 19, we can obtain the following result.

▶ **Corollary 20.** *We can compute in $O(n^{4/3} \log^2 n (\log\log n)^{1/3})$ expected time by a randomized algorithm a data structure with $O(n^{4/3} \log^2 n/(\log\log n)^{2/3})$ sorted canonical subsets of P whose total size is $O(n^{4/3} \log^2 n (\log\log n)^{1/3})$, so that for any disk D whose center is in C, we can find in $O(\kappa)$ time $O(\kappa)$ pairwise-disjoint canonical sets whose union is $P \setminus D$, where $\kappa = O(n^{1/3} (\log\log n)^{1/3}/\log n)$ holds with high probability.*

As discussed before, plugging our above result in the algorithm of [8], we can solve the decision version of the discrete 2-center problem in $O(n^{4/3}\log^2 n(\log\log n)^{1/3})$ expected time. Using the decision algorithm and the $O(n^{4/3}\log^2 n)$-time distance selection algorithm in [24], the discrete 2-center problem can be solved in $O(n^{4/3}\log^3 n(\log\log n)^{1/3})$ expected time.

▶ **Theorem 21.** *Given a set P of n points in the plane, the discrete 2-center problem can be solved in $O(n^{4/3}\log^3 n(\log\log n)^{1/3})$ expected time by a randomized algorithm.*

3.3.2 The deterministic result

The deterministic result also has three levels, which correspond to Lemmas 17, 18, and 19, respectively, but instead uses deterministic techniques. More specifically, we use the partition tree of Theorem 7 to obtain the lowest level data structure; the second level follows the same algorithm as Lemma 18 with the deterministic lowest level structure; the top level structure makes use of a partial half-space decomposition scheme of [27]. The next three lemmas present the three data structures, respectively, with their proofs in the full paper.

▶ **Lemma 22.** *We can compute in $O(n\log n)$ time a data structure with $O(n)$ sorted canonical subsets of P whose total size is $O(n\log\log n)$, so that for any disk D whose center is in C, we can find in $O(\sqrt{n}(\log n)^{O(1)})$ time $O(\sqrt{n}(\log n)^{O(1)})$ pairwise-disjoint canonical sets whose union is $P\setminus D$.*

▶ **Lemma 23.** *We can compute in $O(n^2\cdot\log\log n/\log^2 n)$ time a data structure with $O(n^2/\log^2 n)$ sorted canonical subsets of P whose total size is $O(n^2\log\log\log n/\log^2 n)$, so that for any disk D whose center is in C, we can find in $O(\log n(\log\log n)^{O(1)})$ time $O(\log n(\log\log n)^{O(1)})$ pairwise-disjoint canonical sets whose union is $P\setminus D$.*

▶ **Lemma 24.** *For any $r\le n$, we can compute in $O(n\sqrt{r}+n\log n+r^2+(n^2/r)\cdot\log r\cdot\log\log(n/r)/\log^2(n/r))$ time a data structure with $O(r\log r+(n^2/r)\log r/\log^2(n/r))$ sorted canonical subsets of P whose total size is $O(n\log^2 r+(n^2/r)\log r\log\log\log(n/r)/\log^2(n/r))$, so that for any disk D whose center is in C, we can find in $O(\sqrt{r}\log(n/r)(\log\log(n/r))^{O(1)})$ time $O(\sqrt{r}\log(n/r)(\log\log(n/r))^{O(1)})$ pairwise-disjoint canonical sets whose union is $P\setminus D$.*

By setting $r=n^{2/3}(\log\log n)^{2/3}/\log^{10/3} n$ in the preceding lemma, we obtain the following result.

▶ **Corollary 25.** *We can compute in $O(n^{4/3}\log^{7/3} n(\log\log n)^{1/3})$ time a data structure with a total of $O(n^{4/3}\log^{7/3} n/(\log\log n)^{2/3})$ sorted canonical subsets of P whose total size is upper-bounded by $O(n^{4/3}\log^{7/3} n\log\log\log n/(\log\log n)^{2/3})$, so that for any disk D whose center is in C, we can find in $O(n^{1/3}(\log\log n)^{O(1)}/\log^{2/3} n)$ time $O(n^{1/3}(\log\log n)^{O(1)}/\log^{2/3} n)$ pairwise-disjoint canonical sets whose union is $P\setminus D$.*

According to our discussion before, plugging our above result in the algorithm of [8], we can solve the decision version of the discrete 2-center problem in $O(n^{4/3}\log^{7/3} n(\log\log n)^{O(1)})$ time. Using the decision algorithm and the $O(n^{4/3}\log^2 n)$-time distance selection algorithm in [24], the discrete 2-center problem can be solved in $O(n^{4/3}\log^{10/3} n(\log\log n)^{O(1)})$ time.

▶ **Theorem 26.** *Given a set P of n points in the plane, the discrete 2-center problem can be solved in $O(n^{4/3}\log^{10/3} n(\log\log n)^{O(1)})$ time.*

4 Concluding remarks

Our techniques are likely to find other applications. Generally speaking, our techniques may be useful for solving problems involving a set of congruent disks in the plane. Our paper demonstrates that well-studied techniques for arrangements of lines may be adapted to solving problems involving arrangements of congruent disks. The general idea is to first reduce the problem to the same problem with respect to a pair of square cells using an algorithm like Lemma 1. Then, to tackle the problem on a pair of square cells (C, C'), we need to deal with an arrangement of spanning upper arcs in $\overline{C'}$ such that the centers of the underlying disks of these arcs are all in C. The properties of spanning upper arcs (e.g., Observation 2), along with the duality between the upper arcs in $\overline{C'}$ and the points in C, make an upper-arc arrangement "resemble" a line arrangement so that many algorithms and techniques on line arrangements may be easily adapted to the upper-arc arrangements.

References

1 Pankaj K. Agarwal. Range searching. In *Handbook of Discrete and Computational Geometry,* Csaba D. Tóth, Joseph O'Rourke, and Jacob E. Goodman (eds.), pages 1057–1092. CRC Press, 3rd edition, 2017.

2 Pankaj K. Agarwal. Simplex range searching and its variants: a review. In *A Journey Through Discrete Mathematics*, pages 1–30. Springer, 2017.

3 Pankaj K. Agarwal, Boris Aronov, Micha Sharir, and Subhash Suri. Selecting distances in the plane. *Algorithmica*, 9:495–514, 1993.

4 Pankaj K. Agarwal and Jiří Matoušek. On range searching with semialgebraic sets. *Discrete and Computational Geometry*, 11:393–418, 1994.

5 Pankaj K. Agarwal, Jiří Matoušek, and Micha Sharir. On range searching with semialgebraic sets. II. *SIAM Journal on Computing*, 42:2039–2062, 2013.

6 Pankaj K. Agarwal, Marco Pellegrini, and Micha Sharir. Counting circular arc intersections. *SIAM Journal on Computing*, 22:778–793, 1993.

7 Pankaj K. Agarwal and Micha Sharir. Pseudoline arrangements: Duality, algorithms, and applications. *SIAM Journal on Computing*, 34:526–552, 2005.

8 Pankaj K. Agarwal, Micha Sharir, and Emo Welzl. The discrete 2-center problem. *Discrete and Computational Geometry*, 20:287–305, 1998.

9 Jon L. Bentley and Hermann A. Maurer. A note on Euclidean near neighbor searching in the plane. *Information Processing Letters*, 8:133–136, 1979.

10 Timothy M. Chan. On enumerating and selecting distances. *International Journal of Computational Geometry and Application*, 11:291–304, 2001.

11 Timothy M. Chan. Optimal partition trees. *Discrete and Computational Geometry*, 47:661–690, 2012.

12 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete and Computational Geometry*, 56:866–881, 2016.

13 Timothy M. Chan and Da Wei Zheng. Hopcroft's problem, log-star shaving, 2D fractional cascading, and decision trees. In *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 190–210, 2022.

14 Bernard Chazelle. An improved algorithm for the fixed-radius neighbor problem. *Information Processing Letters*, 16:193–198, 1983.

15 Bernard Chazelle. New techniques for computing order statistics in Euclidean space. In *Proceedings of the 1st Annual Symposium on Computational Geometry (SoCG)*, pages 125–134, 1985.

16 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9(2):145–158, 1993.

17 Bernard Chazelle, Richard Cole, Franco P. Preparata, and Chee-Keng Yap. New upper bounds for neighbor searching. *Information and Control*, 68:105–124, 1986.

18 Bernard Chazelle and Herbert Edelsbrunner. Optimal solutions for a class of point retrieval problems. *Journal of Symbolic Computation*, 1:47–56, 1985.

19 Bernard Chazelle and Emo Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete and Computational Geometry*, 4(5):467–489, 1989.

20 Herbert Edelsbrunner. *Algorithmis in Combinatorial Geometry*. Heidelberg, 1987.

21 Jeff Erickson. New lower bounds for Hopcroft's problem. *Discrete and Computational Geometry*, 16:389–418, 1996.

22 Michael T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proceedings of the 9th Annual Symposium on Computational Geometry (SoCG)*, pages 73–82, 1993.

23 John Hershberger and Subhash Suri. Finding tailored partitions. *Journal of Algorithms*, 3:431–463, 1991.

24 Matthew J. Katz and Micha Sharir. An expander-based approach to geometric optimization. *SIAM Journal on Computing*, 26(5):1384–1408, 1997.

25 Jiří Matoušek. Cutting hyperplane arrangement. *Discrete and Computational Geometry*, 6:385–406, 1991.

26 Jiří Matoušek. Efficient partition trees. *Discrete and Computational Geometry*, 8(3):315–334, 1992.

27 Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(1):157–182, 1993.

28 Jiří Matoušek. Geometric range searching. *ACM Computing Survey*, 26:421–461, 1994.

29 Jiří Matoušek and Zuzana Patáková. Multilevel polynomial partitions and simplified range searching. *Discrete and Computational Geometry*, 54:22–41, 2015.

30 Micha Sharir. Computational geometry column 65. *SIGACT News*, 48:68–85, 2017.

31 Haitao Wang. On the planar two-center problem and circular hulls. In *Proceedings of the 36th International Symposium on Computational Geometry (SoCG)*, pages 68:1–68:14, 2020.

32 Frances F. Yao. A 3-space partition and its applications. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC)*, pages 258–263, 1983.