An Optimal Algorithm for Shortest Paths in Unweighted Disk Graphs

Bruce W. Brewer
Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA

Haitao Wang
Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA

Abstract

Given in the plane a set S of n points and a set of disks centered at these points, the disk graph $G(S)$ induced by these disks has vertex set S and an edge between two vertices if their disks intersect. Note that the disks may have different radii. We consider the problem of computing shortest paths from a source point $s \in S$ to all vertices in $G(S)$ where the length of a path in $G(S)$ is defined as the number of edges in the path. The previously best algorithm solves the problem in $O(n \log^2 n)$ time. A lower bound of $\Omega(n \log n)$ is also known for this problem under the algebraic decision tree model. In this paper, we present an $O(n \log n)$ time algorithm, which matches the lower bound and thus is optimal. Another virtue of our algorithm is that it is quite simple.

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

Keywords and phrases disk graphs, weighted Voronoi diagrams, shortest paths

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.31

1 Introduction

Let S be a set of n points in the plane. Each point of S is associated with a disk centered at the point. Note that the disks may have different radii. The *disk graph* $G(S)$ induced by these disks has vertex set S and an edge between two vertices if their disks intersect. We consider the single-source-shortest-path (SSSP) problem of computing shortest paths from a source point $s \in S$ to all vertices in $G(S)$ where the length of a path in $G(S)$ is defined as the number of edges in the path.

The problem was studied previously. A straightforward solution is to first construct $G(S)$ explicitly and then run a breadth-first-search algorithm, which takes $\Omega(n^2)$ time since $G(S)$ may have $\Omega(n^2)$ edges in the worst case. Kaplan, Katz, Saban, and Sharir [15] proposed the first subquadratic time algorithm, and their algorithm runs in $O(n \log^4 n)$ time. Later Klost [18] gave an $O(n \log^2 n)$ time improved solution. On the other hand, it is known that the problem has an $\Omega(n \log n)$ time lower bound under the algebraic decision tree model even if all disks have the same radius [6].

In this paper, we present a new algorithm whose runtime is $O(n \log n)$, which matches the $\Omega(n \log n)$ lower bound and thus is optimal. Another virtue of the algorithm is that it is quite simple. Indeed, the most complex step of our algorithm is constructing a static additively-weighted Voronoi diagram, which can be easily done using Fortune's sweeping algorithm [14]. Aside from that, the algorithm's description in the paper is self-contained.

Concurrently with our work, de Berg and Cabello have also achieved an $O(n \log n)$ time algorithm using a different approach [4].

Related work. Disk graphs are among the most well-studied topics in computational geometry due to their applications and geometric properties, e.g., [1, 3, 7, 8, 10, 13, 20, 24]. The SSSP problem we consider is actually an *unweighted case* because the length of each edge of

$G(S)$ is defined to be one. In the *weighted* case, the length of each edge is defined to be the Euclidean distance between the two disk centers corresponding to the two vertices of the edge. Kaplan, Katz, Saban, and Sharir [15] also gave an algorithm for the weighted case, and the algorithm runs in $O(n \log^6 n)$ time. An, Oh, and Xue [2] very recently derived an improved algorithm of $O(n \log^4 n)$ time.

The SSSP problem on *unit-disk graphs* (in which all disks have the same radius) has also been extensively studied. Cabello and Jejčič [6] first solved the unweighted case in $O(n \log n)$ time, matching the $\Omega(n \log n)$ lower bound [6]. If the points are presorted by x- and y-coordinates, Chan and Skrepetos [9] showed that it can be solved in $O(n)$ additional time. The weighted case was first solved by Roditty and Segal [22] in $O(n^{4/3+\epsilon})$ time for any $\epsilon > 0$. Later, Cabello and Jejčič [6] solved it in $O(n^{1+\epsilon})$ time. The algorithm by Cabello and Jejčič [6] is bottlenecked by a dynamic bicromatic closest pair data structure, which has since seen improvements by Kaplan, Mulzer, Roditty, Seiferth, and Sharir [16] and also by Liu [19]. Recently, Wang and Xue [25] gave a new algorithm of $O(n \log^2 n)$ time. This has been slightly reduced to $O(n \log^2 n / \log \log n)$ by Brewer and Wang [5]. Furthermore, Brewer and Wang [5] proved that the problem can be solved using $O(n \log n)$ comparisons under the algebraic decision tree model, matching an $\Omega(n \log n)$ lower bound in the same model [6].

Kaplan, Katz, Saban, and Sharir [15] also studied a "reversed version" of the shortest path problem in disk graphs. Chan and Huang [8] very recently improved the algorithm of [15]. Both algorithms used the previous SSSP algorithm for disk graphs as a decision procedure. With our new SSSP algorithm, these algorithms can be slightly improved and also simplified.

In the following, after introducing some notation and concepts in Section 2, we present our algorithm in Section 3.

2 Preliminaries

We follow the notation already defined in Section 1, e.g., n, S, s, $G(S)$.

We define S_i for $i = 0, 1, 2, \ldots$ to be the set of vertices of $G(S)$ whose distance from s in $G(S)$ is equal to i. Note that $S_0 = \{s\}$. For convenience, we define $S_{\leq i} = \cup_{j \leq i} S_j$.

For any two points p, q in the plane, let $|pq|$ denote the (Euclidean) distance between p and q.

To differentiate from other points in the plane, we refer to points of S as *sites*. For any site $v \in S$, let D_v denote the input disk centered at v and r_v the radius of D_v. Notice that two sites $u, v \in S$ have an edge (u, v) in $G(S)$ if and only if $|uv| - r_u - r_v \leq 0$. We consider r_v as the *weight* of v, and for any point $p \in \mathbb{R}^2$, we define the *(additively) weighted distance* from p to v as $d_v(p) = |vp| - r_v$. Note that if p is outside the disk D_v, then $d_v(p)$ is the smallest distance from p to any point of D_v.

For any subset $S' \subseteq S$ and a point $p \in \mathbb{R}^2$, we define a *nearest neighbor* of p to be a site in S' whose weighted distance to p is minimal. Formally, we define $\beta_p(S') = \arg\min_{v \in S'} d_v(p)$ to be a nearest neighbor of p in S'. In cases where p has more than one nearest neighbor in S', we let $\beta_p(S')$ refer to an arbitrary one. We also define $d_{S'}(p) = \min_{v \in S'} d_v(p)$.

With respect to the weights r_v of the sites v of S, we will make heavy use of the *(additively) weighted Voronoi diagram* of S, which is a partition of the plane into Voronoi regions, edges, and vertices [14, 23]. The Voronoi region $\mathrm{Vor}(u)$ of a site $u \in S$ is usually defined as the set of points nearer to u than to any other site in S. For convenience, we will let $\mathrm{Vor}(u)$ also include its boundary, defining $\mathrm{Vor}(u)$ as the set of points at least as near to u than to any other site in S. Formally, $\mathrm{Vor}(u) = \{p \in \mathbb{R}^2 : \forall\, v \in S \setminus \{u\},\ d_u(p) \leq d_v(p)\}$. This way, the

plane is covered by the Voronoi regions. The Voronoi edge between two sites $u, v \in S$ is defined by $\mathcal{E}(u,v) = \{p \in \mathbb{R} : \forall\, w \in S \setminus \{u,v\}, d_u(p) = d_v(p) < d_w(p)\}$. Note that $\mathcal{E}(u,v)$ may have multiple connected components, each of which is a straight line segment or a hyperbolic arc [14, 23]. Voronoi vertices are the points that are equidistant to at least three sites in S such that all other sites in S are farther than these. We use $\mathcal{VD}(S)$ to denote the weighted Voronoi diagram of S.

Let $\mathcal{DT}(S)$ denote the dual graph of $\mathcal{VD}(S)$. Specifically, S is the vertex set of $\mathcal{DT}(S)$, and the edge set of $\mathcal{DT}(S)$ consists of all pairs of sites $u, v \in S$ whose Voronoi regions share a Voronoi edge, i.e., $\mathcal{E}(u,v) \neq \emptyset$. Note that if the sites of S all have the same weight (i.e., all disks have the same radius), then $\mathcal{DT}(S)$ is the Delaunay triangulation of S. We define the neighborhood $N_{\mathcal{DT}}(u)$ of a site $u \in S$ to be the set of sites that share an edge with u in $\mathcal{DT}(S)$. By extension, we define the neighborhood $N_{\mathcal{DT}}(S')$ of a set $S' \subseteq S$ by $N_{\mathcal{DT}}(S') = \bigcup_{u \in S'} N_{\mathcal{DT}}(u)$.

For ease of exposition, we make the general position assumption that no three sites of S lie on the same line. The assumption can be lifted without affecting the results, e.g., by standard perturbation techniques [12, 26].

3 The algorithm

In this section, we present our algorithm to compute the sets S_i for $i = 0, 1, 2, \ldots$. It is easy to extend our algorithm to also compute predecessor information for the shortest paths.

We follow the same algorithmic scheme of Cabello and Jejčič [6] for unit-disk graphs and manage to extend their algorithm to our general disk graph case. In what follows, we first describe the algorithm, then discuss the implementation and time analysis, and finally prove its correctness.

3.1 Algorithm description

We begin with constructing the weighted Voronoi diagram $\mathcal{VD}(S)$. Then the algorithm runs in iterations, and each i-th iteration will compute the set S_i. Initially, we have $S_0 = \{s\}$. In the i-th iteration for $i \geq 1$, we compute S_i as follows, assuming that $S_{\leq i-1}$ has already been computed. Let $Q = N_{\mathcal{DT}}(S_{i-1}) \setminus S_{\leq i-1}$, i.e., the neighbors of S_{i-1} in $\mathcal{DT}(S)$ that are not in S_{i-1}. We remove a site $v \in Q$ from Q and determine whether $d_{S_{i-1}}(v) \leq r_v$. If it is, then we add v to S_i and update $Q = Q \cup (N_{\mathcal{DT}}(v) \setminus S_{\leq i})$, i.e., we add the neighbors of v in $\mathcal{DT}(S)$ that are not in $Q \cup S_{\leq i}$ to Q. We repeat this until $Q = \emptyset$, at which point the i-th iteration is complete. After the i-th iteration, we check if $S_i = \emptyset$. If so, then we are done, and the algorithm stops. Otherwise, we continue to the $i+1$-th iteration. Algorithm 1 gives the pseudocode.

3.2 Algorithm implementation and time analysis

We now discuss how to implement the algorithm efficiently.

First of all, computing $\mathcal{VD}(S)$ and thus $\mathcal{DT}(S)$ can be done in $O(n \log n)$ time [14]. Note that the combinatorial sizes of both $\mathcal{VD}(S)$ and $\mathcal{DT}(S)$ are $O(n)$ [14, 23].

Next, notice that the number of vertices added to Q on Line 7 in iteration i is at most the sum of the degrees of the vertices of S_{i-1} in the graph $\mathcal{DT}(S)$. Because the sets S_i are disjoint and $\mathcal{DT}(S)$ has $O(n)$ edges, this means that the number of vertices added to Q by Line 7 throughout the whole algorithm is $O(n)$. By the same reasoning, we conclude that

Algorithm 1 SSSP algorithm.

Input: A set of point sites S, radius r_v for each site $v \in S$, source site $s \in S$.
Output: The sets S_i for $i = 0, 1, \ldots$

```
1  i ← 0;
2  S_0 ← {s};
3  Build VD(S) and DT(S);
4  while S_i ≠ ∅ do
5  |  i ← i + 1;
6  |  S_i ← ∅;
7  |  Q ← N_DT(S_{i−1}) \ S_{≤i−1} ;
8  |  while Q ≠ ∅ do
9  |  |  v ← pop(Q);
10 |  |  if d_{S_{i−1}}(v) ≤ r_v then
11 |  |  |  S_i ← S_i ∪ {v};
12 |  |  |  Q ← Q ∪ (N_DT(v) \ S_{≤i}) ;
13 return S_i for i = 0, 1, ...;
```

the number of vertices added to Q by Line 12 throughout the whole algorithm is also $O(n)$. We can also know that the while loop on Line 8 has at most $O(n)$ iterations throughout the whole algorithm because a vertex is removed from Q in each iteration.

It remains to determine the time complexity for the operation of checking whether $d_{S_{i-1}}(v) \leq r_v$ on Line 10. To this end, it suffices to find $\beta_v(S_{i-1})$, i.e., the nearest neighbor of v in S_{i-1}. To find $\beta_v(S_{i-1})$, in the beginning of the i-th iteration, we compute the weighted Voronoi diagram for S_{i-1}, denoted by $\mathcal{VD}(S_{i-1})$, which takes $O(|S_{i-1}| \log |S_{i-1}|)$ time [14], and then construct a point location data structure for $\mathcal{VD}(S_{i-1})$ in $O(|S_{i-1}|)$ time [11,17]. After that, $\beta_v(S_{i-1})$ can be found in $O(\log n)$ time by a point location query with v on $\mathcal{VD}(S_{i-1})$. As such, the total time of Line 10 in the whole algorithm is $O(n \log n)$. As $\sum_i |S_{i-1}| = n$, the overall time for constructing $\mathcal{VD}(S_{i-1})$ in the whole algorithm is $O(n \log n)$.

We thus conclude that the time complexity of the whole algorithm is $O(n \log n)$.

3.3 Algorithm correctness

We start with proving the following two observations that our algorithm's correctness relies on.

▶ **Observation 1.** *For any two sites $u, v \in S$, they are connected by an edge in $G(S)$ if and only if $d_v(u) \leq r_u$ or $d_u(v) \leq r_v$.*

Proof. Recall that u and v have an edge if and only if $|uv| - r_u - r_v \leq 0$. Notice that $|uv| - r_u - r_v \leq 0$ holds if and only if $d_v(u) \leq r_u$ or $d_u(v) \leq r_v$. ◀

▶ **Observation 2.** *For any site $v \in S \setminus S_{\leq i-1}$, v is in S_i if and only if $d_{S_{i-1}}(v) \leq r_v$.*

Proof. By definition, S_{i-1} has a site u with $d_u(v) = d_{S_{i-1}}(v)$. If $d_{S_{i-1}}(v) \leq r_v$, then $d_u(v) \leq r_v$. By Observation 1, $G(S)$ has an edge connecting u and v. As $u \in S_{i-1}$ and $v \in S \setminus S_{\leq i-1}$, we obtain that $v \in S_i$.

If $v \in S_i$, then there is an edge in $G(S)$ connecting v to a site $u \in S_{i-1}$. By Observation 1, $d_u(v) \leq r_v$. It then follows that $d_{S_{i-1}}(v) \leq d_u(v) \leq r_v$. ◀

We now prove that the algorithm is correct. Let S'_i be the set S_i computed by our algorithm and let S_i be the "correct" set of vertices at distance i from s in $G(S)$. Then, our goal is to prove that $S'_i = S_i$ for all i. We do so by induction on i. Our base case is that $S'_0 = S_0$, which is obviously true because they are both $\{s\}$. Our inductive hypothesis is that $S'_j = S_j$ for all $j < i$. To prove that $S'_i = S_i$, we will argue both $S'_i \subseteq S_i$ and $S'_i \supseteq S_i$.

First of all, since $S'_j = S_j$ for all $j < i$, our algorithm guarantees that every site of S'_i is at distance i from s in $G(S)$ and thus $S'_i \subseteq S_i$. To see this, a site v is added to S'_i in Line 11 because $d_{S_{i-1}}(v) \leq r_v$. By Observation 2, $v \in S_i$.

We next prove that $S'_i \supseteq S_i$. Consider a site $v \in S_i$. Our goal is to show that v is also in S'_i. We will prove in Lemma 3 that there exists a vertex $u \in S_{i-1}$ and a u-v path π in $\mathcal{DT}(S)$ such that all internal vertices of π are in S_i (note that *internal vertices* refer to vertices of π excluding u and v). Let the vertices of π be $u = w_0, w_1, \ldots, w_k = v$. We argue that all w_j with $1 \leq j \leq k$ are added to S'_i. To this end, because $w_j \in S_i$ for all $1 \leq j \leq k$, it is sufficient to argue that w_j is added to Q because when w_j is popped from Q, it will pass the check on Line 10 by Observation 2 and then be added to S'_i. Indeed, w_1 is added to Q on Line 7 because $w_1 \in N_{\mathcal{DT}}(u)$. Since $w_2 \in N_{\mathcal{DT}}(w_1)$, w_2 will be added to Q on Line 12 no later than right after w_1 is added to S'_i on Line 11. Following the same argument, $w_3, w_4, \ldots, w_k = v$ will all be added to Q and thus added to S'_i as well. This proves that $v \in S'_i$. Therefore, $S'_i \supseteq S_i$.

In summary, we have proved $S'_i = S_i$ and thus the correctness of the algorithm.

▶ **Lemma 3.** *For any $i > 0$ and any site $v \in S_i$, there exist a site $u \in S_{i-1}$ and a u-v path in $\mathcal{DT}(S)$ such that all internal vertices of the path are in S_i.*

Proof. We extend the proof of [6, Lemma 1] for the unit-disk graph case to our general disk graph problem.

Let $u = \beta_v(S_{i-1})$, i.e., u is a nearest neighbor of v in S_{i-1}. Consider the line segment $\overline{uv}$ and the sites whose Voronoi regions in $\mathcal{VD}(S)$ intersect $\overline{uv}$. Let $w_0, w_1, \ldots, w_k$ be these sites in the order that their Voronoi regions intersect $\overline{uv}$ with $w_0 = u$ and $w_k = v$. Note that repetitions are possible, so it could be that $w_j = w_{j'}$ for $j \neq j'$. See Figure 1 for an example. For ease of discussion, we assume that $\overline{uv}$ does not contain any Voronoi vertices of $\mathcal{VD}(S)$ (otherwise, we could replace v by a point arbitrarily close to v and then follow the same argument). This implies that, for every $0 \leq j \leq k-1$, $\overline{uv}$ crosses the Voronoi edge between w_j and w_{j+1}. This further implies that w_j and w_{j+1} are adjacent in $\mathcal{DT}(S)$, so $\pi = w_0, w_1, \ldots, w_k$ is a u-v path in $\mathcal{DT}(S)$.

Next, we need to show that all internal vertices w_j $(1 \leq j \leq k-1)$ of π are in S_i. We will first prove that there is an edge (u, w_j) in $G(S)$ connecting u and w_j, meaning that $w_j \in S_{i-2} \cup S_{i-1} \cup S_i$ since $u \in S_{i-1}$. Then, we will show that $w_j \notin S_{i-2} \cup S_{i-1}$, so $w_j \in S_i$ must hold. This will complete the proof of the lemma.

Proving $G(S)$ has an edge (u, w_j). We first argue that $G(S)$ has an edge (u, v). Indeed, since $v \in S_i$, there is a vertex $u' \in S_{i-1}$ connecting to v by an edge in $G(S)$. By Observation 1, $d_{u'}(v) \leq r_v$. Since $u = \beta_v(S_{i-1})$, we have $d_u(v) \leq d_{u'}(v) \leq r_v$. By Observation 1, $G(S)$ has an edge (u, v).

Consider any w_j with $1 \leq j \leq k-1$. We now argue that $G(S)$ has an edge (u, w_j). Recall that $\overline{uv}$ intersects $\mathrm{Vor}(w_j)$. Let p be a point of $\overline{uv} \cap \mathrm{Vor}(w_j)$; see Figure 2. We can derive the following:

$$d_{w_j}(u) = |w_j u| - r_{w_j} \overset{(1)}{\leq} |w_j p| + |pu| - r_{w_j} \overset{(2)}{\leq} |vp| + |pu| - r_v \overset{(3)}{=} |vu| - r_v = d_v(u) \overset{(4)}{\leq} r_u,$$

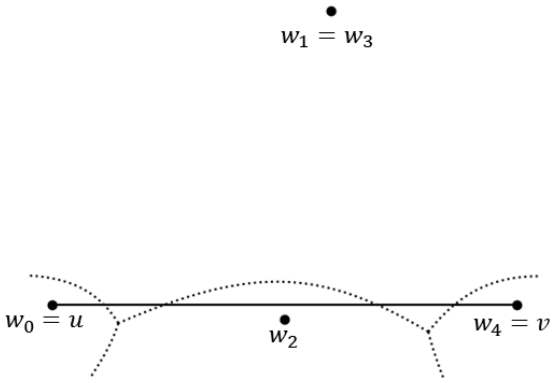

Figure 1 The segment $\overline{uv}$ passes through $\mathrm{Vor}(u)$, $\mathrm{Vor}(w_1)$, $\mathrm{Vor}(w_2)$, $\mathrm{Vor}(w_3)$, and $\mathrm{Vor}(v)$, where $w_1 = w_3$.

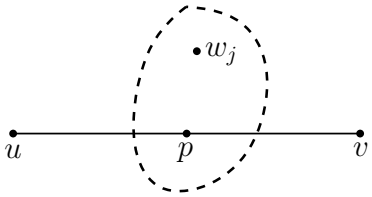

Figure 2 The region bounded by the dashed curve is $\mathrm{Vor}(w_j)$.

where (1) is due to the triangle inequality, (2) is due to $p \in \mathrm{Vor}(w_j)$ (therefore, $|w_jp| - r_{w_j} = d_{w_j}(p) \leq d_v(p) = |vp| - r_v$), (3) is due to $p \in \overline{uv}$, and (4) is due to Observation 1 and the fact that $G(S)$ has an edge (u, v). Consequently, by Observation 1, $G(S)$ has an edge (u, w_j).

Proving $w_j \notin S_{i-2} \cup S_{i-1}$. To prove that $w_j \notin S_{i-2} \cup S_{i-1}$, $1 \leq j \leq k-1$, as before, we pick a point $p \in \overline{uv} \cap \mathrm{Vor}(w_j)$ and derive the following:

$$d_{w_j}(v) \overset{(1)}{<} |w_jp| + |pv| - r_{w_j} \overset{(2)}{\leq} |up| + |pv| - r_u \overset{(3)}{=} |uv| - r_u = d_u(v) \overset{(4)}{\leq} r_v,$$

where (1) is due to the triangle inequality and our general position assumption (because $p \in \overline{uv}$, equality occurs only if u, v, and w_j are collinear, violating our general position assumption), (2) is due to $p \in \mathrm{Vor}(w_j)$ (therefore, $|w_jp| - r_{w_j} = d_{w_j}(p) \leq d_u(p) = |up| - r_u$), (3) is due to $p \in \overline{uv}$, and (4) is due to Observation 1 and the fact that $G(S)$ has an edge (u, v), as proven above.

Notice that $w_j \notin S_{i-1}$ since otherwise $d_{w_j}(v) < d_u(v)$ proved above would contradict with $u = \beta_v(S_{i-1})$. Also, since $G(S)$ has an edge (w_j, v) and $v \in S_i$, w_j cannot be in S_{i-2} (since otherwise v would be in $S_{\leq i-1}$, contradicting with that $v \in S_i$). This proves that $w_j \notin S_{i-2} \cup S_{i-1}$. ◀

The following theorem summarizes our result.

▶ **Theorem 4.** *Given a set of n disks in the plane and a source disk, shortest paths from the source to all other vertices in the disk graph induced by all disks can be computed in $O(n \log n)$ time.*

Extension to the L_∞ (resp., L_1) metric. In the L_∞ metric, each disk becomes a square centered at a site of S. For this case, an $O(n \log n)$-time algorithm was given by Klost [18] to solve the SSSP problem, which uses the involved techniques in [3]. We can solve this problem

in $O(n \log n)$ time in a much simpler way by making the following changes to Algorithm 1. First, use the L_∞ metric to measure the distance $|pq|$ of any two points p, q in the plane. Second, use additively-weighted Voronoi diagrams in the L_∞ metric instead. Note that constructing an additively-weighted Voronoi diagram for a set of n points in the L_∞ metric is equivalent to constructing a Voronoi diagram for a set of n squares in the L_∞ metric. This latter problem can be solved in $O(n \log n)$ time, e.g., by the algorithm of Papadopoulou and Lee [21]. As such, Algorithm 1 can be modified to solve the L_∞ metric case in $O(n \log n)$ time. The algorithm is much simpler than the one in [18]. The L_1 case can be treated analogously.

References

1. Shinwoo An, Kyungjin Cho, and Eunjin Oh. Faster algorithms for cycle hitting problems on disk graphs. In *Proceedings of the 18th Algorithms and Data Structures Symposium (WADS)*, pages 29–42, 2023. doi:10.1007/978-3-031-38906-1_3.
2. Shinwoo An, Eunjin Oh, and Jie Xue. Single-source shortest path problem in weighted disk graphs. In *Proceedings of the 41st International Symposium on Computational Geometry (SoCG)*, pages 7:1–7:15, 2025. doi:10.4230/LIPIcs.SoCG.2025.7.
3. Alexander Baumann, Haim Kaplan, Katharina Klost, Kristin Knorr, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Dynamic connectivity in disk graphs. *Discrete and Computational Geometry*, 71:214–277, 2024. doi:10.1007/s00454-023-00621-x.
4. Mark de Berg and Sergio Cabello. An $O(n \log n)$ algorithm for single-source shortest paths in disk graphs. In *Proceedings of the 33rd Annual European Symposium on Algorithms (ESA)*, page to appear, 2025. doi:10.48550/arXiv.2506.07571.
5. Bruce W. Brewer and Haitao Wang. An improved algorithm for shortest paths in weighted unit-disk graphs. In *Proceedings of the 36th Canadian Conference on Computational Geometry (CCCG)*, pages 57–64, 2024. arXiv:2407.03176.
6. Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry: Theory and Applications*, 48:360–367, 2015. doi:10.1016/j.comgeo.2014.12.003.
7. Sergio Cabello and Wolfgang Mulzer. Minimum cuts in geometric intersection graphs. *Computational Geometry: Theory and Applications*, 94(101720), 2021. doi:10.1016/j.comgeo.2020.101720.
8. Timothy M. Chan and Zhengcheng Huang. Faster algorithms for reverse shortest path in unit-disk graphs and related geometric optimization problems: Improving the shrink-and-bifurcate technique. In *Proceedings of the 41st International Symposium on Computational Geometry (SoCG)*, pages 32:1–32:14, 2025. doi:10.4230/LIPIcs.SoCG.2025.32.
9. Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC)*, pages 24:1–24:13, 2016. doi:10.4230/LIPIcs.ISAAC.2016.24.
10. Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990. doi:10.1016/0012-365X(90)90358-O.
11. Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986. doi:10.1137/0215023.
12. Herbert Edelsbrunner and Ernst P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9:66–104, 1990. doi:10.1145/77635.77639.
13. Jared Espenant, J. Mark Keil, and Debajyoti Mondal. Finding a maximum clique in a disk graph. In *Proceedings of the 39th International Symposium on Computational Geometry (SoCG)*, pages 30:1–30:17, 2023. doi:10.4230/LIPIcs.SoCG.2022.49.

14 Steven Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987. doi:10.1007/BF01840357.

15 Haim Kaplan, Matthew J. Katz, Rachel Saban, and Micha Sharir. The unweighted and weighted reverse shortest path problem for disk graphs. In *Proceedings of the 31st Annual European Symposium on Algorithms (ESA)*, pages 67:1–67:14, 2023. doi:10.4230/LIPIcs.ESA.2023.67.

16 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discrete and Computational Geometry*, 64:838–904, 2020. doi:10.1007/s00454-020-00243-7.

17 David G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12:28–35, 1983. doi:10.1137/0212002.

18 Katharina Klost. An algorithmic framework for the single source shortest path problem with applications to disk graphs. *Computational Geometry: Theory and Applications*, 111:101979, 2023. doi:10.1016/j.comgeo.2022.101979.

19 Chih-Hung Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. *SIAM Journal on Computing*, 51:723–765, 2022. doi:10.1137/20M1388371.

20 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. Subexponential parameterized algorithms on disk graphs (extended abstract). In *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2031, 2022. doi:10.1137/1.9781611977073.80.

21 Evanthia Papadopoulou and D.T. Lee. The L_∞-Voronoi diagram of segments and VLSI applications. *International Journal of Computational Geometry and Application*, 11(101720):503–528, 2001. doi:10.1142/S0218195901000626.

22 Liam Roditty and Michael Segal. On bounded leg shortest paths problems. *Algorithmica*, 59:583–600, 2011. doi:10.1007/s00453-009-9322-3.

23 Micha Sharir. Intersection and closest-pair problems for a set of planar discs. *SIAM Journal on Computing*, 14:448–468, 1985. doi:10.1137/0214034.

24 Anastasiia Tkachenko and Haitao Wang. Dominating set, independent set, discrete k-center, dispersion, and related problems for planar points in convex position. In *Proceedings of the 42nd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 73:1–73:20, 2025. doi:10.4230/LIPIcs.STACS.2025.73.

25 Haitao Wang and Jie Xue. Near-optimal algorithms for shortest paths in weighted unit-disk graphs. *Discrete and Computational Geometry*, 64:1141–1166, 2020. doi:10.1007/s00454-020-00219-7.

26 C.-K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *Journal of Computer and System Sciences*, 40:2–18, 1990. doi:10.1016/0022-0000(90)90016-E.