Farthest-Point Voronoi Diagrams in the Presence of Rectangular Obstacles

Mincheol Kim¹ · Chanyang Seo² · Taehoon Ahn¹ · Hee-Kap Ahn³

Received: 18 April 2022 / Accepted: 29 December 2022 / Published online: 24 January 2023 © The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
We present an algorithm to compute the geodesic $L_1$ farthest-point Voronoi diagram of $m$ point sites in the presence of $n$ rectangular obstacles in the plane. It takes $O(nm + n \log n + m \log m)$ construction time using $O(nm)$ space. This is the first optimal algorithm for constructing the farthest-point Voronoi diagram in the presence of obstacles. We can construct a data structure in the same construction time and space that answers a farthest-neighbor query in $O(\log(n + m))$ time.

Keywords
Farthest-point Voronoi diagrams · Obstacles · Rectangles

This research was partly supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) Grant funded by the Korea government(MSIT) (No. 2017-0-00905, Software Star Lab (Optimal Data Structure and Algorithmic Applications in Dynamic Geometric Environment)) and (No. 2019-0-01906, Artificial Intelligence Graduate School Program(POSTECH)).

Hee-Kap Ahn
heekap@postech.ac.kr
Mincheol Kim
rucatia@postech.ac.kr
Chanyang Seo
chan8616@postech.ac.kr
Taehoon Ahn
sloth@postech.ac.kr

¹ Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea
² Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Pohang, Korea
³ Graduate School of Artificial Intelligence, Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea
1 Introduction

A Voronoi diagram of a set of sites is a subdivision of the space under consideration into subspaces by assigning points in the space to sites with respect to a certain proximity. Typical Voronoi assignment models are the nearest-point model and the farthest-point model where every point is assigned to its nearest site and its farthest site, respectively. There are results for computing Voronoi diagrams in the plane \([1, 14, 15, 26]\), under various metrics \([9, 17, 18, 24]\), or for various types of sites \([2, 8, 23]\).

For \(m\) point sites in the plane, the nearest-point and farthest-point Voronoi diagrams of the sites under the Euclidean metric can be constructed in \(O(m \log m)\) time \([15, 26]\). When the sites are contained in a simple polygon with no holes, the distance between any two points in the polygon, called the geodesic distance, is measured as the length of the shortest path contained in the polygon and connecting the points (called the geodesic path). There has been a fair amount of work computing the geodesic nearest-point and farthest-point Voronoi diagrams of \(m\) point sites in a simple \(n\)-gon \([3, 4, 21, 22]\) to achieve the lower bound \(\Omega(n + m \log m)\) \([3]\). Recently, optimal algorithms of \(O(n + m \log m)\) time were given for the geodesic nearest-point Voronoi diagram \([20]\) and for the geodesic farthest-point Voronoi diagram \([27]\).

The problem of computing Voronoi diagrams is more challenging in the presence of obstacles. Each obstacle plays as a hole and there can be two or more geodesic paths connecting two points that avoid those holes. The geodesic nearest-point Voronoi diagram of \(m\) point sites can be computed in \(O(m \log m + k \log k)\) time by applying the continuous Dijkstra paradigm \([16]\), where \(k\) is the number of total vertices of obstacles. However, no optimal algorithm is known for the farthest-point Voronoi diagram in the presence of obstacles in the plane, even when the obstacles are of elementary shapes such as axis-aligned line segments and rectangles. The best result of the geodesic farthest-point Voronoi diagram known so far takes \(O(mk \log^2 (m + k) \log k)\) time by Bae and Chwa \([5]\). They also showed that the total complexity of the geodesic farthest-point Voronoi diagram is \(\Theta(mk)\).

In the presence of \(n\) rectangular obstacles under \(L_1\) metric, there are a few results for farthest-neighbor queries. Ben-Moshe et al. \([7]\) presented a data structure with \(O(nm \log(n + m))\) construction time and \(O(nm)\) space for \(m\) point sites that supports farthest point queries in \(O(\log(n + m))\) time. They also showed that the \(L_1\) geodesic farthest-point Voronoi diagram has complexity \(\Theta(nm)\), but without presenting any algorithm for computing the diagram. Later Ben-Moshe et al. \([6]\) gave a tradeoff between the query time and the preprocessing/space such that a data structure of size \(O((n + m)^{1.5})\) can be constructed in \(O((n + m)^{1.5} \log^2(n + m))\) time to support farthest point queries in \(O((n + m)^{0.5} \log(n + m))\) time.

The geodesic center of a set of objects in a polygonal domain is the set of points in the domain that minimize the maximum geodesic distance from input objects. Thus, it can be obtained once the geodesic farthest-point Voronoi diagram of the objects is constructed. For \(m\) points in the presence of \(n\) axis-aligned rectangular obstacles in the plane, Choi et al. \([10]\) showed that the geodesic center of the points under the \(L_1\) metric consists of \(\Theta(nm)\) connected regions and gave an \(O(n^2m)\)-time algorithm to compute the geodesic center. Later, Ben-Moshe et al. \([7]\) gave an \(O(nm \log(n + m))\)-time algorithm for the problem.
Our Result In this paper, we present an algorithm that computes the geodesic $L_1$ farthest-point Voronoi diagram of $m$ points in the presence of $n$ rectangular obstacles in the plane in $O(nm + n \log n + m \log m)$ time using $O(nm)$ space. The running time and space complexity of our algorithm match the time and space lower bounds of the Voronoi diagram. Thus, it is the first optimal algorithm for computing the geodesic farthest-point Voronoi diagram in the presence of obstacles.

To do this, we construct a data structure for $L_1$ farthest-neighbor queries in $O(nm + n \log n + m \log m)$ time using $O(nm)$ space. This improves upon the results by Ben-Moshe et al. [7]. Moreover, the construction time and space are the best among the data structures supporting $O(\log (n + m))$ query time for $L_1$ farthest neighbors. Then we present an optimal algorithm to compute the explicit geodesic $L_1$ farthest-point Voronoi diagram in $O(nm + n \log n + m \log m)$ time using $O(nm)$ space, which matches the time and space lower bounds of the diagram.

As a byproduct, we compute the geodesic center under the $L_1$ metric in $O(nm + n \log n + m \log m)$ time. This result improves upon the algorithm by Ben-Moshe et al. [7].

Outline First, we construct four farthest-point maps, one for each of the four axis directions, either the $x$- or $y$-axis, and either positive or negative. In the course, we construct a data structure for $L_1$ farthest-neighbor queries in $O(nm + n \log n + m \log m)$ time using $O(nm)$ space. For each axis direction, we apply the plane sweep technique with a line orthogonal to the direction and moving along the direction. During the sweep, we maintain the status of the sweep line in a balanced binary search tree and its associated structures while handling events induced by the point sites and the sides of rectangles parallel to the sweep line. There are $m$ events induced by point sites and $O(n)$ events induced by rectangles. After sorting the events in $O(n \log n + m \log m)$ time, we show that we can handle all events induced by point sites in $O(nm)$ time. Additionally, we show that each event induced by a rectangle can be handled in $O(m + \log n)$ time. By the plane sweep, we construct a data structure consisting of $O(n + m)$ line segments parallel to the sweep line and $O(nm)$ points in $O(nm + n \log n + m \log m)$ time in total. Given a query, it uses axis-aligned ray shooting queries on the data structure to find the farthest site from the query. The four farthest-point maps are planar subdivisions, and they can be constructed during the plane sweep in the same time and space.

With the four farthest-point maps and the data structure for farthest-neighbor queries, we construct the geodesic $L_1$ farthest-point Voronoi diagram explicitly. First, we decompose the plane, excluding the holes, into rectangular faces using vertical line segments, each extending from a vertical side of a hole. Then, we partition each face in the decomposition into zones such that the farthest-point Voronoi diagram restricted to a zone coincides with the corresponding region of a farthest-point map. This partition is done by using the boundary between two farthest-point maps, which can be computed by traversing the cells in the two maps in which the boundary lies. Finally, we glue the corresponding regions along the boundaries of zones, and then glue all adjacent faces along their boundaries to obtain the geodesic $L_1$ farthest-point Voronoi diagram. We show that this can be done in $O(nm + n \log n + m \log m)$ time in total.
For the centers of \( m \) points in the presence of \( n \) axis-aligned rectangles in the plane, we can find them from the farthest-point Voronoi diagram in time linear to the complexity of the diagram.

2 Preliminaries

Let \( R \) be a set of \( n \) open disjoint rectangles and \( S \) be a set of \( m \) point sites lying in the free space \( F = \mathbb{R}^2 - \bigcup_{R \in R} R \). Throughout the paper, we use the \( L_1 \) metric. For ease of description, we omit \( L_1 \). We use \( x(p) \) and \( y(p) \) to denote the \( x \)-coordinate and \( y \)-coordinate of a point \( p \), respectively. For two points \( p \) and \( q \) in \( F \), we use \( pq \) to denote the line segment connecting them. Whenever we say a path connecting two points in \( F \), it is a path contained in \( F \). There can be two or more geodesic paths connecting two points \( p \) and \( q \) that avoid the holes. We use \( \pi(p, q) \) to denote a fixed geodesic path connecting \( p \) and \( q \), and use \( d(p, q) \) to denote the geodesic distance between \( p \) and \( q \), which is the length of \( \pi(p, q) \).

We use \( f(p) \) to denote the set of sites of \( S \) that are farthest from a point \( p \in F \) under the geodesic distance, that is, a site \( s \) is in \( f(p) \) if and only if \( d(s, p) \geq d(s', p) \) for all \( s' \in S \). If there is only one farthest site, we use \( f(p) \) to denote the site.

A horizontal line segment \( \ell \) can be represented by the two \( x \)-coordinates \( x_1(\ell) \) and \( x_2(\ell) \) of its endpoints (\( x_1(\ell) < x_2(\ell) \)) and the \( y \)-coordinate \( y(\ell) \) of them. For an axis-aligned rectangle \( R \), let \( x_1(R) \) and \( x_2(R) \) denote the \( x \)-coordinates of the left and right sides of \( R \).

A path is \( x \)-monotone if and only if the intersection of the path with any line perpendicular to the \( x \)-axis is connected. Likewise, a path is \( y \)-monotone if and only if the intersection of the path with any line perpendicular to the \( y \)-axis is connected. A path is \( xy \)-monotone if and only if the path is \( x \)-monotone and \( y \)-monotone. Observe that if a path connecting two points is \( xy \)-monotone, it is a geodesic path connecting the points.

2.1 Eight Monotone Paths from a Point

Choi and Yap [11] gave a way of partitioning the plane with rectangular holes into eight regions using eight \( xy \)-monotone paths from a point. We use their method to partition \( F \) as follows. Consider a horizontal ray emanating from a point \( s = p_1 \in F \) going rightwards. The ray stops when it hits a rectangle \( R \in R \) at a point \( p'_1 \). Let \( p_2 \) be the top-left corner of \( R \). We repeat this process by taking a horizontal ray from \( p_2 \) going rightwards until it hits a rectangle, and so on. The last horizontal ray goes to infinity. Then we obtain an \( xy \)-monotone path \( \pi_{ru}(s) = p_1 p'_1 p_2 p'_2 \ldots \) from \( s \) that alternates going rightwards and going upwards. See Fig. 1a.

By choosing two directions, one going either rightwards or leftwards horizontally, and one going either upwards or downwards vertically, and ordering the chosen directions, we define eight rectilinear \( xy \)-monotone paths from \( s \) with directions: rightwards-upwards (ru), upwards-rightwards (ur), upwards-leftwards (ul), leftwards-upwards (lu), leftwards-downwards (ld), downwards-leftwards (dl),
The eight paths partition $F$ into eight regions $\text{reg}_1,...,\text{reg}_8$. Region $\text{reg}_3$ consists of two subregions separated by a rectangle $R$. Every geodesic path from $s$ to $p$ is $y^+$-monotone and $p$ is $y^+$-reachable from $s$. Every geodesic path from $s$ to $q$ is $y^-$-monotone and $q$ is $y^-$-reachable from $s$.

Fig. 1 Gray rectangles are holes. a The eight paths partition $F$ into eight regions $\text{reg}_1,...,\text{reg}_8$. Region $\text{reg}_3$ consists of two subregions separated by a rectangle $R$. b Every geodesic path from $s$ to $p$ is $y^+$-monotone and $p$ is $y^+$-reachable from $s$. Every geodesic path from $s$ to $q$ is $y^-$-monotone and $q$ is $y^-$-reachable from $s$ (Color figure online)

Based on Lemma 1, we define a few more terms. For any point $p$ in $\text{reg}_2 \cup \text{reg}_3 \cup \text{reg}_4$ (and the boundaries of the regions), we say $p$ is $y^+$-reachable from $s$, and every geodesic path from $s$ to $p$ is $y^+$-monotone. Any point $q \in \text{reg}_6 \cup \text{reg}_7 \cup \text{reg}_8$ (and the boundaries of the regions) is $y^-$-reachable from $s$, and every geodesic path from $s$ to $q$ is $y^-$-monotone. See Fig. 1b. Similarly, any point $p \in \text{reg}_1 \cup \text{reg}_2 \cup \text{reg}_8$ (and the boundaries of the regions) is $x^+$-reachable from $s$, and every geodesic path from $s$ to $p$ is $x^+$-monotone. Any point $q \in \text{reg}_4 \cup \text{reg}_5 \cup \text{reg}_6$ (and the boundaries of the regions) is $x^-$-reachable from $s$, and every geodesic path from $s$ to $q$ is $x^-$-monotone.
3 Farthest-Point Maps

Based on Lemma 1 and the four directions of monotone paths in the previous section, we define four farthest-point maps. A farthest-point map $M_{y^+} = M_y(S)$ of $S$ in $F$ corresponding to the positive $y$-direction is a planar subdivision of $F$ into cells. For a point $p \in F$, a site $s \in S$ is a farthest site of $p$ in $M_{y^+}$ if $d(p, s) > d(p, s')$ for every site $s' \in S$ from which $p$ is $y^+$-reachable. If $p$ is $y^+$-reachable from no site in $S$, $p$ has no farthest site in $M_{y^+}$. Thus, a cell of $M_{y^+}$ is defined on $F \setminus C_\emptyset$, where $C_\emptyset$ denotes the set of points of $F$ that are $y^+$-reachable from no site in $S$. A site $s$ corresponds to one or more cells in $M_{y^+}$ with the property that a point $p \in F \setminus C_\emptyset$ lies in a cell of $s$ if and only if $d(p, s) > d(p, s')$ for every $s' \in S \setminus \{s\}$ from which $p$ is $y^+$-reachable.

We define $M_{y^-}$, $M_{x^+}$ and $M_{x^-}$ analogously with respect to their corresponding directions. Since the four maps have the same structural and combinatorial properties with respect to their corresponding directions, we describe only $M_{y^+}$ in the following. Let $B$ be an axis-aligned rectangular box such that $S$, $R$, and all vertices of the four farthest-point maps are contained in the interior of $B$. We focus on $F \cap B$ only, and often abuse $F$ to denote $F \cap B$.

In the following, we analyze the edges of $M_{y^+}$ using the bisectors of pairs of sites. Let $F(s, s')$ denote a set of points of $F$ that are $y^+$-reachable from two sites $s$ and $s'$. To be specific, $F(s, s')$ is an intersection of two regions, one lying above $\pi_{1u}(s)$ and $\pi_{1u}(s')$ and the other lying above $\pi_{1d}(s)$ and $\pi_{1d}(s')$. Thus, the boundary of $F(s, s')$ coincides with the upper envelope of $\pi_{1u}(s)$, $\pi_{ru}(s)$, $\pi_{lu}(s')$ and $\pi_{ru}(s')$. We use $F(s, s)$ to denote the set of points that are $y^+$-reachable from a site $s$.

For any two distinct sites $s, s' \in S$, their bisector consists of all points $x \in F$ satisfying $d(x, s) = d(x, s')$. Observe that the bisector may contain a two-dimensional region. We use $b(s, s')$ to denote the line segments and the boundary of the two-dimensional region in the bisector of $s$ and $s'$.

**Lemma 2** For any two sites $s$ and $s'$, $b(s, s') \cap F(s, s')$ consists of axis-aligned segments.

**Proof** Consider two sites $s$ and $s'$ in the plane with no holes. Then $b(s, s')$ contained in $B$ is a polygonal chain consisting of two parallel and axis-aligned segments, and one segment of slope 1 or $-1$ lying in between them. The segment of slope 1 or $-1$ is contained in region $[\min\{x(s), x(s')\}, \max\{x(s), x(s')\}] \times [\min\{y(s), y(s')\}, \max\{y(s), y(s')\}]$.

Now consider the bisector $b(s, s')$ of two sites $s$ and $s'$ in the freespace $F$. Due to the rectangle holes of $F$, the bisector may consist of two or more pieces. It, however, still consists of axis-aligned segments, and segments of slope $\pm 1$ under the $L_1$ metric [19].

We now focus on $b(s, s')$ restricted to $F(s, s')$ and show that no segment of slope $\pm 1$ appears in $b(s, s') \cap F(s, s')$. Assume to the contrary that $b(s, s') \cap F(s, s')$ has a segment $uv$ of slope 1 or $-1$. Let $q$ be any point on $uv$. Since $q \in F(s, s')$, $q$ is $y^+$-reachable from both $s$ and $s'$, and $\pi(s, q)$ and $\pi(s', q)$ are $y$-monotone. Let $q_1$ be the farthest point from $q$ on $\pi(s, q)$ such that $\pi(q_1, q)$ is $xy$-monotone. Likewise, let $q_2$ be the farthest point from $q$ on $\pi(s', q)$ such that $\pi(q_2, q)$ is $xy$-monotone. See Fig. 2. Clearly, $d(s, q_1) + d(q_1, q) = d(s', q_2) + d(q_2, q)$. Since $uv$ has slope 1 or $-1$, $\min\{x(q_1), x(q_2)\} \leq x(q) \leq \max\{x(q_1), x(q_2)\}$ and $\min\{y(q_1), y(q_2)\} \leq y(q) \leq \max\{y(q_1), y(q_2)\}$.
max\{y(q_1), y(q_2)\}. Then \(q\) is \(y^+\)-reachable from one of \(q_1\) and \(q_2\), and \(y^-\)-reachable from the other, implying that \(\pi(s, q)\) or \(\pi(s', q)\) is not \(y\)-monotone, a contradiction. Thus, no segment of slope 1 appears in \(b(s, s') \cap F(s, s')\). Analogously, we can show that \(b(s, s') \cap F(s, s')\) has no segment with slope \(-1\).

Let \(f_\delta(p)\) denote the set of farthest sites from a point \(p \in S\) among the sites from which \(p\) is \(\delta\)-reachable for \(\delta \in \{y^+, y^-, x^+, x^-\}\). For each horizontal segment of \(\pi_{lu}(s) \cup \pi_{ru}(s)\), we call the portion \(h\) of the segment such that \(f_{y^+}(p) = \{s\}\) for any point \(p \in h\), a \(b\)-edge. Observe that no point \(p'\) with \(x_1(h) \leq x(p') \leq x_2(h)\) and \(y(p') = y(h) - \varepsilon\) for any \(\varepsilon > 0\) is \(y^+\)-reachable from \(s\). Therefore, \(f_{y^+}(p) \neq f_{y^+}(p')\), and thus an \(b\)-edge is also an edge of \(M_{y^+}\). Since every edge of \(M_{y^+}\) is part of a bisector of two sites in \(S\) or a \(b\)-edge, it is either horizontal or vertical. See Fig. 3a.

**Corollary 1** Every edge of \(M_{y^+}\) is an axis-aligned line segment.

For sites contained in a simple polygon, Aronov et al. [4] gave a lemma, called Ordering Lemma, that the order of sites along the boundary of their convex hull is the same as the order of their Voronoi cells along the boundary of a simple polygon. We give a lemma on the order of sites in the presence of rectangular obstacles. We use it in analyzing the maps and Voronoi diagrams.
Lemma 3 Let \( pq \) be a horizontal segment contained in \( F \setminus C_y \) with \( x(p) < x(q) \). For any two sites \( f_p \in f_{x+}(p) \) and \( f_q \in f_{x+}(q) \), if \( f_p \notin f_{y+}(q) \) or \( f_q \notin f_{y+}(p) \), then \( x(f_p) > x(f_q) \).

Proof Ben-Moshe et al. [7] showed that \( x(f_p) \neq x(f_q) \).

Assume to contrary that \( x(f_p) < x(f_q) \). Consider two cases that there are (1) two geodesic paths, one from \( p \) to \( f_q \) and one from \( q \) to \( f_p \), intersecting each other at a point, say \( r \) (Fig. 4a), or (2) no two such geodesic paths intersecting each other (Fig. 4b).

For case (1), we have

\[
d(p, f_q) = d(p, r) + d(r, f_q) \quad \text{and} \quad d(q, f_p) = d(q, r) + d(r, f_p).
\]

We also observe that

\[
d(p, f_p) \leq d(p, r) + d(r, f_p) \quad \text{and} \quad d(q, f_q) \leq d(q, r) + d(r, f_q).
\]

Adding the two inequalities above, we obtain

\[
d(p, f_p) + d(q, f_q) \leq d(p, r) + d(r, f_p) + d(q, r) + d(r, f_q)
\]

\[
= d(p, f_q) + d(q, f_p).
\]

However, since \( f_p \notin f(q) \) or \( f_q \notin f(p) \), we have \( d(p, f_p) + d(q, f_q) > d(p, f_q) + d(q, f_p) \), a contradiction.

Now consider case (2) that there are no two geodesic paths, one from \( p \) to \( f_q \) and one from \( q \) to \( f_p \), intersecting each other (Fig. 4b). Since \( pq \) is horizontal with \( x(p) < x(q) \), \( x(f_p) < x(f_q) \) by assumption, every geodesic path from \( p \) to \( f_q \) and every geodesic path from \( q \) to \( f_p \) are \( y^+ \)-monotone, and no two geodesic paths intersect each other by the case, we have \( y(f_p) \neq y(f_q) \). Without loss of generality, assume \( y(f_p) < y(f_q) \). Then \( \pi_{tu}(f_q) \) intersects \( \pi(q, f_p) \) at a point, say \( r \) (Fig. 4c). Since \( x(f_p) < x(f_q) \leq x(r) \) and \( y(f_p) < y(f_q) \leq y(r) \), we have \( d(r, f_q) < d(r, f_p) \).
and thus
\[ d(q, r) + d(r, f_q) < d(q, r) + d(r, f_p) = d(q, f_p). \]
Since \( d(q, f_q) \leq d(q, r) + d(r, f_q) \), we have \( d(q, f_q) < d(q, f_p) \), a contradiction. \( \square \)

Consider a horizontal line segment \( h \) contained in \( F \). For each site \( s_i \), if \( s_i \in f_y^+(p_1) \) and \( s_i \in f_y^+(p_2) \) for \( p_1 \in h \) and \( p_2 \in h \), then \( s_i \in f_y^+(p_3) \) for every \( p_3 \in p_1 p_2 \). Since there are \( m \) sites, \( f_y^+ \) changes \( O(m) \) time along \( h \). This implies the following corollary.

**Corollary 2** Any horizontal line segment contained in \( F \) intersects \( O(m) \) cells in \( M_{y^+} \).

Using Corollaries 1 and 2, we analyze the complexity of \( M_{y^+} \) as follows. Note that each lower endpoint of a vertical edge of \( M_{y^+} \) appears on a horizontal line segment passing through a site or the top side of a rectangle. By Corollary 2, the maximal horizontal segment through the top side of a rectangle in \( R \) and contained in \( F \) intersects \( O(m) \) vertical edges of \( M_{y^+} \). Moreover, the maximal horizontal line segment through a site \( s \) and contained in \( F \) intersects \( O(1) \) lower endpoints of vertical edges on the boundary of the cell of \( s \). Since there are \( n \) rectangles in \( R \) and \( m \) sites in \( S \), \( M_{y^+} \) has \( O(nm + m) = O(nm) \) vertical edges. Every horizontal edge of \( M_{y^+} \) is a segment of a bisector or a \( b \)-edge, and it is incident to a side of a rectangle or another vertical edge. Since there are \( O(n) \) rectangle sides, and \( O(1) \) horizontal edges of \( M_{y^+} \) that are incident to each vertical edge, \( M_{y^+} \) has \( O(n + nm) = O(nm) \) horizontal edges. Thus, \( M_{y^+} \) has complexity \( O(nm) \).

Now we show that every farthest site \( s \in f(p) \) of a point \( p \) in \( F \) is one of the farthest sites of \( p \) in the four farthest-point maps. By the definition of the farthest-point maps, \( p \) is contained in a cell of \( M_{y^+}, M_{y^-}, M_{x^+} \) or \( M_{x^-} \). Since every geodesic path from a site to \( p \) is either \( y^+, y^-, x^+, \) or \( x^- \)-monotone by Lemma 1, \( s \in f(p) \) is one of the farthest sites of \( p \) in the four farthest-point maps. If \( p \) is contained in cells of two or more maps, we compare their distances to the farthest sites defining the cells and take the ones with the largest distance as the farthest sites of \( p \). Thus, once the four farthest-point maps are constructed, the farthest sites of a query point can be computed from the map.

### 4 Data Structure for Farthest-Neighbor Queries

We present an algorithm that constructs a data structure for farthest site queries. We denote \( m \) point sites of \( S \) by \( s_1, \ldots, s_m \) such that \( x(s_1) \leq \cdots \leq x(s_m) \), and \( n \) rectangular obstacles of \( R \) by \( R_1, \ldots, R_n \). The query data structure consists of four parts, each for one axis direction. Since the four parts can be constructed in the same way with respect to their directions, we focus on the part corresponding to the positive \( y \)-direction, and thus the query data structure corresponds to \( M_{y^+} \). We use \( Q_{y^+} \) to denote the query data structure.
By Corollary 1, we can find the farthest site of a query point using a vertical ray shooting query to the horizontal edges of $M_{y^+}$ and a binary search on the lower endpoints of vertical edges of $M_{y^+}$ lying on the horizontal edges of $M_{y^+}$. Thus, $Q_{y^+}$ consists of the horizontal edges of $M_{y^+}$ and the endpoints of vertical edges of $M_{y^+}$ lying on the horizontal edges of $M_{y^+}$.

A point $q$ lying on a horizontal segment $h$ of $Q_{y^+}$ is the lower endpoint of a vertical edge of $M_{y^+}$ if and only if there are two points $q_1 = (x(q) - \varepsilon, y(q))$ and $q_2 = (x(q) + \varepsilon, y(q))$ for sufficiently small $\varepsilon > 0$ satisfying $f_y^+(q_1) \cup f_y^+(q_2) = f_y^+(q)$ and $f_y^+(q_1) \neq f_y^+(q_2)$. We call each lower endpoint of vertical edges lying on $h$ a boundary point on $h$. See Fig. 3b.

We use a plane sweep algorithm with a horizontal sweep line $L$ to construct the horizontal line segments in $Q_{y^+}$. Note that $F \cap L$ consists of disjoint horizontal segments along $L$. The status of $L$ is the sequence of segments in $F \cap L$ along $L$. The status changes while $L$ moves upwards over the plane, at particular $y$-coordinates which we call events. To do such updates efficiently, we maintain three data structures for $L$: a balanced binary search tree $\mathcal{T}$ representing the status, a boundary list $B$, and a list $D$ of distance functions. The structures $B$ and $D$ are associated structures of $\mathcal{T}$.

We store the segments of $F \cap L$ in a balanced binary search tree $\mathcal{T}$ in increasing order of $x$-coordinate of their left endpoints. Each node $v$ of $\mathcal{T}$ corresponds to a horizontal line segment $h_v$ of $F \cap L$. We store $x_1(h_v)$ and $x_2(h_v)$, and an array $X_v$ of $m$ Boolean variables at $v$. We set $X_v[i] = T$ if at a point on $h_v$ is $y^+$-reachable from $s_i$ for $i = 1 \ldots, m$. Otherwise, we set $X_v[i] = F$. The range of $v$ is $[x_1(v), x_2(v)]$ for $x_1(v) = x_1(h_v)$ and $x_2(v) = x_2(h_v)$. There are at most $n + 1$ nodes in $\mathcal{T}$, and each node maintains an array of size $O(m)$. Thus, $\mathcal{T}$ uses $O(nm)$ space in total. See Fig. 5a.

The data structure $B$ consists of the boundary lists $B(v)$ for nodes $v$ of $\mathcal{T}$. Each node $v$ of $\mathcal{T}$ has a pointer to its boundary list $B(v)$, which is a doubly-linked list of the boundary points (including the endpoints of $h_v$) lying on $h_v$. Each boundary point in $B$ is the intersection of $L$ and a vertical edge of $M_{y^+}$, so there are $O(nm)$ boundary points in $B$.

The list $D$ consists of distance functions $d_v$ for nodes $v$ of $\mathcal{T}$. Let $p(r)$ denote a point on $L$ with $x(p(r)) = r$ for a real number $r$. Let $d_\delta(p) = d(s, p)$ for a site $s \in f_y^+(p)$ if $f_y^+(p) \neq \emptyset$ for $\delta \in \{y^+, y^-, x^+, x^-, x\}$. If $f_y^+(p) = \emptyset$, we set $d_\delta(p) = -\infty$. Each node $v$ of $\mathcal{T}$ has a pointer to its distance function $d_v(x) = d_{y^+}(p(x))$ for $x$ in $\mathcal{T}$.
the range \([x_1(v), x_2(v)]\) of \(v\). It is a piecewise linear function with pieces (segments) of slopes 1 or \(-1\). See Fig. 5b.

There are three types of events: (1) a site event, (2) a bottom-side event, and (3) a top-side event. A site event occurs when \(L\) encounters a site in \(S\). A bottom-side event occurs when \(L\) encounters the bottom side of a rectangle in \(R\). A top-side event occurs when \(L\) encounters the top side of a rectangle in \(R\). Thus, there are \(m\) site events, \(n\) bottom-side events, and \(n\) top-side events. See Fig. 6.

We maintain and update \(T, B\) and \(D\) during the plane sweep for those events. To handle events, we first sort the events in increasing order of \(y\)-coordinate, which takes \(O((n + m) \log(n + m)) = O(n \log n + m \log m)\) time. We update \(d_v(x)\) only at those events and keep it unchanged between two consecutive events. To reflect the distances from sites to \(p(x) \in h_v\) correctly, we assign an additive weight to \(d_v(x)\), which is the difference in the \(y\)-coordinates between the current event and the last event at which \(d_v(x)\) is updated.

Initially, when \(L\) is at the bottom side of \(B\), \(T\) consists of one node \(v\) with \(x_1(v) = x_1(B)\), \(x_2(v) = x_2(B)\), and \(X_v[i] = \emptyset\) for all \(i \in \{1, \ldots, m\}\). \(B(v)\) has no boundary point and \(d_v(x) = -\infty\) for all \(x\), since no points on \(L\) is \(y^+\)-reachable from any sites.

### 4.1 Handling a Site Event

When \(L\) encounters a site \(s_i \in S\), we find the node \(v \in T\) such that \(x_1(v) \leq x(s_i) \leq x_2(v)\). Every point on \(h_v\) is \(y^+\)-reachable from \(s_i\), so we set \(X_v[i] = T\). We can find \(v\) in \(O(\log n)\) time, and set \(X_v[i] = T\) in constant time. Thus, it takes \(O(\log n)\) time to update \(T\).

For any point \(p(x) \in h_v\), \(d(s_i, p(x)) = |x - x(s_i)|\). By Lemma 3, there is at most one maximal interval \(I \subset [x_1(v), x_2(v)]\) such that \(d_v(x) < d(s_i, p(x))\) for every \(x \in I\). Moreover, \(I\) is bounded from left by \(x_1(v)\) or from right by \(x_2(v)\) because \(d_v(x)\) is continuous and consists of pieces (segments) of slopes 1 or \(-1\), and \(d(s_i, p(x)) = |x - x(s_i)|\). We find the boundary point \(p(x^*) \in h_v\) induced by \(s_i\) such that \(d_v(x^*) = d(s_i, p(x^*))\). If \(I\) is bounded from left, we update \(d_v(x)\) to \(d_v(x) = d(s_i, p(x))\) for \(x \leq x^*\). If \(I\) is bounded from right, we update \(d_v(x)\) to \(d_v(x) = d(s_i, p(x))\) for \(x \geq x^*\).

If there is no point \(p(x^*) \in h_v\) induced by \(s_i\) such that \(d_v(x^*) = d(s_i, p(x^*))\), either \(d_v(x) < d(s_i, p(x))\) or \(d_v(x) > d(s_i, p(x))\) for all \(x\) with \(x_1(v) \leq x \leq x_2(v)\). If \(d_v(x) < d(s_i, p(x))\), we update \(d_v(x)\) to \(d_v(x) = d(s_i, p(x))\) for \(x_1(v) \leq x \leq x_2(v)\). If \(d_v(x) > d(s_i, p(x))\), we do not update \(d_v(x)\).
We update $B(v)$ by removing all the boundary points of $B(v)$ lying in the interior of $I$ in time linear to the number of the boundary points, and then inserting $p(x^*)$ into $B(v)$.

Since there are $m$ site events, it takes $O(m \log n)$ time in total to update $T$. The total time to remove the boundary points is linear to the total number of boundary points in $Q_y^+$, which is $O(nm)$.

**Lemma 4** We can handle all site events in $O(nm)$ time using $O(nm)$ space.

### 4.2 Handling a Bottom-Side Event

When $L$ encounters the bottom side of a rectangle $R \in R$, the line segment of $F \cap L$ incident to the bottom side is replaced by two line segments by the event. See Fig. 6b. Thus, we update $T$ by finding the node $v \in T$ with $x_1(v) \leq x_1(R) < x_2(R) \leq x_2(v)$, removing $v$ from $T$, and then inserting two new nodes $u$ and $w$, corresponding to the two line segments, into $T$. We set $[x_1(u), x_2(u)] = [x_1(v), x_1(R)]$, $[x_1(w), x_2(w)] = [x_2(R), x_2(v)]$, $X_u = X_v$, and $X_w = X_v$. This takes $O(\log n)$ time since $T$ is a balanced binary search tree. It takes $O(m)$ time to copy the Boolean values of $X_v$ to $X_u$ and $X_w$, and to remove $X_v$. Thus, it takes $O(m + \log n)$ time to update $T$.

We update $B$ by inserting two lists $B(u)$ and $B(w)$ into $B$, copying the boundary points of $B(v)$ to the lists, and then removing $B(v)$ from $B$. By Corollary 2, $h_v$ intersects $O(m)$ cells in $M_y^+$. Thus, $B(v)$ has $O(m)$ boundary points, and the update to $B(u)$ and $B(w)$ takes $O(m)$ time.

Since there are $n$ bottom-side events, it takes $O(nm + n \log n)$ time to update $T$ and $O(nm)$ time to update $B$ for all bottom-side events.

**Lemma 5** We can handle all bottom-side events in $O(nm + n \log n)$ time using $O(nm)$ space.

### 4.3 Handling a Top-Side Event

When $L$ encounters the top side of a rectangle $R \in R$, the two consecutive segments in $F \cap L$ incident to $R$ are replaced by one segment spanning them during the event. See Fig. 6c. We update $T$ by finding the two nodes $u, w \in T$ with $x_2(u) = x_1(R)$ and $x_1(w) = x_2(R)$, removing $u$ and $w$ from $T$, and then inserting a new node $v$ into $T$. We set $x_1(v) = x_1(u), x_2(v) = x_2(w)$, and $X_v[i] = X_u[i] \lor X_w[i]$ for each $i = 1, \ldots, m$. This takes $O(m + \log n)$ time.

We update the distance function $d_v(x)$ for $x$ with $x_1(v) \leq x \leq x_1(R)$ as follows. The geodesic path from any point $p(x) \in h_u$ to $s_i$ with $X_u[i] = F$ and $X_w[i] = T$ is $xy$-monotone by Lemma 1, and thus $d(s_i, p(x)) = y(p(x)) - y(s_i) + |x(s_i) - x|$. Also, we observe that $x(s_i) \geq x$ for any $x$. Thus, every $p(x)$ has the same site $s^*$ as its farthest site among the sites $s_i$ with $X_u[i] = F$ and $X_w[i] = T$. Then $d(s^*, p(x)) = y(p(x)) - y(s^*) + x(s^*) - x$. By Lemma 3, there is at most one maximal interval $I$ of $x \in [x_1(v), x_1(R)]$ such that $d_v(x) \leq d(s^*, p(x))$. Moreover, $I$ is bounded from left by $x_1(v)$. We find the boundary point $p(x^*) \in h_u$ such that $d_v(x^*) = d(s^*, p(x^*))$, and update $d_v(x)$ to $d(s^*, p(x))$ for $x \leq x^*$.  

 Springer
If there is no such point $p(x^*) \in h_u$, either $d_v(x) < d(s^*, p(x))$ or $d_v(x) > d(s^*, p(x))$ for all $x$ with $x_1(v) \leq x \leq x_1(R)$. If $d_v(x) < d(s^*, p(x))$, we update $d_v(x)$ to $d_v(x) = d(s^*, p(x))$ for $x_1(v) \leq x \leq x_1(R)$. If $d_v(x) > d(s^*, p(x))$, we do not update $d_v(x)$.

We update $B[x_1(v), x_1(R)]$, which is the part of $B(v)$ with range $[x_1(v), x_1(R)]$, by removing all the boundary points in the interior of $I$ in time linear to the number of the boundary points, and then inserting $p(x^*)$ as a boundary point. We can handle the case of $x$ with $x_2(R) \leq x \leq x_3(v)$, and update the part $B[x_2(R), x_2(v)]$ of $B(v)$ with range $[x_2(R), x_2(v)]$ analogously.

### 4.3.1 Computing Distance Functions for a Top Side

We show how to compute $d_v(x)$ for $x \in [x_1(R), x_2(R)]$ and update the part $B[x_1(R), x_2(R)]$ of $B(v)$ with range $[x_1(R), x_2(R)]$ efficiently. Lemma 1 implies the following observation.

**Observation 1** For any point $p$ on the top side of a rectangle $R \in R$ and any site $s \in S$ from which $p$ is $y^+$-reachable, every geodesic path from $p$ to $s$ passes through the top-left corner or the top-right corner of $R$.

For an index $k$, let $\alpha_k$ and $\beta_k$ denote the top-left corner and the top-right corner of rectangle $R_k \in R$, and let $S_k$ denote the set of the sites that lie below the polygonal curve consisting of $\pi_{dl}(\alpha_k)$, the top side of $R_k$, and $\pi_{dl}(\beta_k)$.

For the top-side event of $R = R_k$, let $\alpha = \alpha_k$ and $\beta = \beta_k$. Note that $x(\alpha) = x_1(R)$ and $x(\beta) = x_2(R)$. Let $S^\alpha$ be the set of the sites $s_i$ with $X_v[i] = T$ for all $i = 1, \ldots, m$. We partition $S^\alpha$ into three disjoint subsets, $S_k$, $S(\alpha)$, and $S(\beta)$ such that $S(\alpha) = \{s_i \in S^\alpha \mid x(s_i) \leq x_1(R)\}$ and $S(\beta) = \{s_i \in S^\alpha \setminus S_k \mid x(s_i) \geq x_2(R)\}$. See Fig. 7.

Every geodesic path from any site in $S(\alpha)$ or $S(\beta)$ to any point on the top side of $R$ is $xy$-monotone. Thus for any point $p(x)$ lying on the top side of $R$, we can compute $d(s^\alpha, p(x))$ and $d(s^\beta, p(x))$, where $s^\alpha$ and $s^\beta$ are the farthest sites of $p(x)$ among sites in $S(\alpha)$ and among sites in $S(\beta)$, respectively, as we did for $B[x_1(v), x_1(R)]$ or $B[x_2(R), x_2(v)]$.

Let $R_a$ be the rectangle first hit by the vertical ray emanating from $\alpha$ going downwards, and let $R_b$ be the rectangle first hit by the vertical ray emanating from $\beta$ going downwards. See Fig. 7.

We compute $d(\alpha, s)$ and $d(\beta, s)$ for each $s \in S_k$, and then compute $d_v(x)$ for $x$ with $x_1(R) \leq x \leq x_2(R)$, where $v$ is the node of $T$ corresponding to the top-side event of $R$. The top-side events for $R_a$ and $R_b$ were handled before the top-side event of $R$, and thus we have $d(\alpha_a, s)$ and $d(\beta_a, s)$ for sites $s \in S_a$, and $d(\alpha_b, s')$ and $d(\beta_b, s')$ for sites $s' \in S_b$. By Observation 1, we can compute $d(\alpha, s)$ and $d(\beta, s)$ for a site $s \in S_k$ as follows.

- If $s \notin S_a$, $\pi(\alpha, s)$ is $xy$-monotone by Lemma 1, and thus $d(\alpha, s) = |x(\alpha) - x(s)| + |y(\alpha) - y(s)|$. If $s \in S_a$, $d(\alpha, s) = \min\{d(\alpha, \alpha_a) + d(\alpha_a, s), d(\alpha, \beta_a) + d(\beta_a, s)\}$.
- If $s \notin S_b$, $\pi(\beta, s)$ is $xy$-monotone by Lemma 1, and thus $d(\beta, s) = |x(\beta) - x(s)| + |y(\beta) - y(s)|$. If $s \in S_b$, $d(\beta, s) = \min\{d(\beta, \alpha_b) + d(\alpha_b, s), d(\beta, \beta_b) + d(\beta_b, s)\}$.
Lemma 6

By Observation 1, every geodesic path from $s$ to $p(x)$ passes through either $\alpha$ or $\beta$. We denote by $d^\alpha(i, x) = d(\alpha, s_i) + x - x(\alpha)$ the length of a geodesic path from a site $s_i$ to $p(x)$ passing through $\alpha$, and denote by $d^\beta(i, x) = d(\beta, s_i) + x(\beta) - x$ the length of a geodesic path from $s_i$ to $p(x)$ passing through $\beta$. Let $D(x) = \max_{s_j \in S_k} \{d^\alpha(i, x), d^\beta(i, x)\}$ for all $x$ with $x(\alpha) \leq x \leq x(\beta)$. For ease of description, let $d^\alpha(i) = d(\alpha, s_i)$ and $d^\beta(i) = d(\beta, s_i) + x(\beta) - x(\alpha)$. Note that $d^\beta(i)$ denotes the length of a geodesic path from $s_i$ to $\alpha$ passing through $\beta$. Therefore, $d^\alpha(i) \leq d^\beta(i)$ holds for every $s_i \in S_k$. Let $d_{\beta\alpha}(i) = d^\beta(i) - d^\alpha(i)$.

Our goal is to compute $D(x)$ in $O(m)$ time. To achieve this, we consider two cases, either (1) $d_{\beta\alpha}(a) \geq d_{\beta\alpha}(b)$ for every indices $a$ and $b$ with $1 \leq a < b \leq m$, or (2) $d_{\beta\alpha}(a) < d_{\beta\alpha}(b)$ for some indices $a$ and $b$ with $1 \leq a < b \leq m$. The following two lemmas show how to compute $D(x)$ in $O(m)$ time for these two cases.

**Lemma 6** If $d_{\beta\alpha}(a) \geq d_{\beta\alpha}(b)$ for every two indices $a$ and $b$ with $1 \leq a < b \leq m$, we can compute $D(x)$ in $O(m)$ time.

**Proof** Let $s_c \in f_y^+(\alpha)$ be the farthest site from $\alpha$ with the smallest index $c$. By Lemma 3 and $x(s_a) \leq x(s_b)$ for every two indices $a$ and $b$ with $1 \leq a < b \leq m$, no site $s_i \notin f_y^+(\alpha)$ for any $i = c + 1, \ldots, m$ is in $f_y^+(p(x))$ for any point $p(x)$ on the top side of $R$. Observe that $d^\alpha(c) + t \leq d^\beta(c) - t$ for $0 \leq t \leq d_{\beta\alpha}(c)/2$. Also, for every index $i = 1, \ldots, c - 1$, $d^\alpha(i) + t \leq d^\beta(i) - t$ still holds for $0 \leq t \leq d_{\beta\alpha}(c)/2$ since $d_{\beta\alpha}(i) \geq d_{\beta\alpha}(c)$.

Let $c'$ be the smallest index satisfying $d^\alpha(c') = \max_{i \in [1, \ldots, c-1]} d^\alpha(i)$. Then $s_{c'}$ is in $f_y^+(p(x(\alpha) + t))$ for $t \geq (d^\beta(c) - d^\alpha(c'))/2$. For $d_{\beta\alpha}(c)/2 \leq t \leq (d^\alpha(c) - d^\alpha(c'))/2$, $s_c \in f_y^+(p(x(\alpha) + t))$. Therefore, $p((d^\alpha(c) - d^\alpha(c'))/2)$ becomes a boundary point. See Fig. 8. Using $c'$, we compute the smallest index $c''$ satisfying $d^\alpha(c'') = \max_{i \in [1, \ldots, c'-1]} d^\alpha(i)$ and find all farthest sites $f_y^+(p(x(\alpha) + t))$ for all $t$ recursively.
We find sites $s_c = \max_{i \in \{1, \ldots, m\}} d^\alpha(i)$, $s'_c = \max_{i \in \{1, \ldots, c-1\}} d^\alpha(i)$, and $s''_c = \max_{i \in \{1, \ldots, c'-1\}} d^\alpha(i)$ recursively. They are the farthest sites from $p(x)$ moving rightwards. a The graph with respect to indices of sites. b The graph for $x$ with $x_1(R) \leq x \leq x_2(R)$.

We use a stack storing indices of sites to compute $c'$ and $c''$ recursively. We can find $s_c$ in $O(m)$ time. Let top be the top element (site) of the stack. Initially, the stack contains $c$. For each index $i$ decreasing from $c - 1$ to 1, we pop top from the stack repeatedly while $d^\alpha(i) \geq d^\alpha(\text{top}(A))$, and then push $i$ into the stack. Observe that the stack never be empty since $d^\alpha(c) > d^\alpha(i)$ for all $i \in \{1, \ldots, c-1\}$.

The pseudocode of the algorithm is given in Algorithm 1. We repeat this until we compute $D(x)$ and all boundary points. This takes $O(m)$ time. □

### Algorithm 1 Computing Farthest Sites on Rectangle

```plaintext
procedure FARDESTsites([s_1, \ldots, s_m], R)
    c ← arg max_{i \in \{1, \ldots, m\}} d^\alpha(i)
    stack A ← \{c\}
    for i ≡ c - 1 down to 1 do
        while d^\alpha(i) ≥ d^\alpha(\text{top}(A)) do
            pop top(A) from A
        end while
        push i into A
    end for
    sort A in reverse order
    x ← x(\alpha)
    set p(x) as a boundary point
    while |A| > 1 do
        i ← \text{top}(A)
        pop top(A) from A
        x' ← (d^\alpha(i) - d^\alpha(\text{top}(A)))/2
        D(t) ← \min\{d(\alpha, s_i) + t - x(\alpha), d(\beta, s_i) + x(\beta) - t\} for x ≤ t ≤ x'
        x ← x'
        set p(x) as a boundary point
    end while
    x' ← x(\beta)
    D(t) ← \min\{d(\alpha, s_{\text{top}(A)}) + t - x(\alpha), d(\beta, s_{\text{top}(A)}) + x(\beta) - t\} for x ≤ t ≤ x'
    set p(x') as a boundary point
    return D(x) and boundary points
end procedure
```

If $d^\alpha_{\beta\alpha}(a) < d^\alpha_{\beta\alpha}(b)$ for some indices $a$ and $b$ with $a < b$, we can remove either $s_a$ or $s_b$ by the following lemma.
Lemma 7  If there are two indices \( a \) and \( b \) with \( a < b \) such that \( d_{\beta a}(a) < d_{\beta a}(b) \), either \( s_a \) or \( s_b \) is a farthest site from no point \( p(x) \) for \( x \) with \( x_1(R) \leq x \leq x_2(R) \).

Proof  The proof is similar to that of Lemma 3. First, we show that if there are two geodesic paths \( \pi(\alpha, s_a) \) and \( \pi(\beta, s_a) \) that intersect each other, then \( d_{\beta a}(a) \geq d_{\beta a}(b) \). Let \( q \) be a point in the intersection of the paths. Then

\[
d(\alpha, s_b) = d(\alpha, q) + d(q, s_b) \quad \text{and} \quad d(\beta, s_a) = d(\beta, q) + d(q, s_a).
\]

We observe that \( d(\alpha, s_a) \leq d(\alpha, q) + d(q, s_b) \) and \( d(\beta, s_b) \leq d(\beta, q) + d(q, s_b) \). Adding these two inequalities, we obtain

\[
d(\alpha, s_a) + d(\beta, s_b) \leq d(\alpha, q) + d(q, s_b) + d(\beta, q) + d(q, s_a)
\]

and thus \( d(\beta, s_b) - d(\alpha, s_b) \leq d(\beta, s_a) - d(\alpha, s_a) \). Since \( d(\beta, s_b) - d(\alpha, s_b) = d_\beta^a(b) + C - d_\alpha^a(b) \) and \( d(\beta, s_a) - d(\alpha, s_a) = d_\beta^a(a) + C - d_\alpha^a(a) \), we have \( d_\beta^a(b) - d_\alpha^a(b) \leq d_\beta^a(a) - d_\alpha^a(a) \), where \( C = x(\beta) - x(\alpha) \). Therefore, \( d_{\beta a}(a) \geq d_{\beta a}(b) \).

By contraposition, if \( d_{\beta a}(a) < d_{\beta a}(b) \), no two geodesic paths \( \pi(\alpha, s_b) \) and \( \pi(\beta, s_a) \) intersect each other. Since \( y(\alpha) = y(\beta) \), \( x(s_a) \leq x(s_b) \), and all geodesic paths from the sites we consider are \( y^+ \)-monotone, we have \( y(s_a) \neq y(s_b) \). We show that \( s_b \) is a farthest site from no point \( p(x) \) if \( y(s_a) < y(s_b) \), and \( s_a \) is a farthest site from no point \( p(x) \) if \( y(s_a) > y(s_b) \).

Consider the case that \( y(s_a) < y(s_b) \). Then \( \pi_{\alpha}(s_a) \) intersects \( \pi(\beta, s_a) \) at a point, say \( q \). Since \( x(s_a) \leq x(s_b) \leq x(q) \), \( y(s_a) < y(s_b) \leq y(q) \), and \( \pi(s_b, q) \) is \( xy \)-monotone, we have \( d(q, s_a) > d(q, s_b) \). Then

\[
d(\beta, s_b) \leq d(\beta, q) + d(q, s_b) < d(\beta, q) + d(q, s_a) = d(\beta, s_a) \quad \text{and} \quad d(\beta, s_b) - d(\beta, s_a) < 0.
\]

Then \( d_\beta^a(b) - d_\alpha^a(a) < 0 \) because \( d_\beta^a(b) = d(\beta, s_b) + C \) and \( d_\alpha^a(a) = d(\beta, s_a) + C \), where \( C = x(\beta) - x(\alpha) \). Since \( d_{\beta a}(a) < d_{\beta a}(b) \), we have \( d_\alpha^a(b) - d_\alpha^a(a) < d_\beta^a(b) - d_\beta^a(a) < 0 \). This implies that

\[
\min\{d_\alpha^a(b) + t, d_\beta^a(b) - t\} < \min\{d_\alpha^a(a) + t, d_\beta^a(a) - t\}
\]

for any \( t \in [0, x(\beta) - x(\alpha)] \). Thus, \( s_b \) is a farthest site from no point \( p(x) \) for \( x \) with \( x_1(R) \leq x \leq x_2(R) \).

Now consider the case that \( y(s_a) > y(s_b) \). Then \( \pi_{\beta}(s_a) \) intersects \( \pi(\alpha, s_b) \) at a point, say \( q' \). Since \( x(q') \leq x(s_a) \leq x(s_b) \), \( y(q') \geq y(s_a) > y(s_b) \), and \( \pi(q', s_a) \) is \( xy \)-monotone, we have \( d(q', s_b) > d(q', s_a) \). Thus,

\[
d_\alpha^a(a) \leq d(\alpha, q') + d(q', s_a) < d(\alpha, q') + d(q', s_b) = d_\alpha^a(b) \quad \text{and} \quad d_\alpha^a(b) - d_\alpha^a(a) > 0.
\]
Since $d_{\beta\alpha}(a) < d_{\beta\alpha}(b)$, we have $0 < d^a(b) - d^a(a) < d_{\beta}^a(b) - d_{\beta}^a(a)$. Then $d_{\beta}^a(b) > d_{\beta}^a(a)$ and $d^a(b) > d^a(a)$ which implies that
\[
\min[d^a(b) + t, d_{\beta}^a(b) - t] > \min[d^a(a) + t, d_{\beta}^a(a) - t]
\]
for any $t \in [0, x(\beta) - x(\alpha)]$. Therefore, $s_a$ is a farthest site from no point $p(x)$ for $x$ with $x_1(R) \leq x \leq x_2(R)$.

\[\square\]

\textbf{Algorithm 2} Pruning Sites on Rectangle

\begin{algorithm}
\begin{algorithmic}
\Procedure{PruningSites}{$[s_1, \ldots, s_m], R$}
\State stack $A \leftarrow \emptyset$
\For{$i = 1$ to $m$}
\While{$\text{True}$}
\If{$A = \emptyset$ or $d(\alpha, s_i) - d(\alpha, s_i) \geq d(\beta, \text{top}(A)) - d(\alpha, \text{top}(A))$}
\State push $s_i$ into $A$
\State break
\EndIf
\If{$d(\alpha, s_i) \leq d(\alpha, \text{top}(A))$}
\State ignore $s_i$
\State break
\EndIf
\If{$d(\alpha, s_i) > d(\alpha, \text{top}(A))$}
\State pop top$(A)$ from $A$
\EndIf
\EndWhile
\EndFor
\State return $A$
\EndProcedure
\end{algorithmic}
\end{algorithm}

Lemmas 6 and 7 imply that the complexity of $D(x)$ is $O(m)$. By Lemma 7, we can remove the sites which are farthest from no point $p(x)$ for $x$ with $x_1(R) \leq x \leq x_2(R)$ by comparing $d_{\beta\alpha}(a)$ and $d_{\beta\alpha}(b)$ for two indices $a$ and $b$. This can be done in $O(m)$ time by Algorithm 2. After pruning, $d_{\beta\alpha}(a) \geq d_{\beta\alpha}(b)$ for every pair of remaining sites $s_a$ and $s_b$ with $a < b$. Therefore, we can compute $D(x)$ in $O(m)$ time by Lemma 6. Then we can compute $d_v(x) = \max\{d(s^\alpha, p(x)), D(x), d(s^\beta, p(x))\}$ in $O(m)$ time. We update $B[x_1(R), x_2(R)]$ in $O(m)$ time using $d_v(x)$.

For each top-side event of $R_k$, we first identify $R_a$ and $R_b$ using ray shooting queries in $O(\log n)$ time. Then we compute $d(\alpha, s)$ and $d(\beta, s)$ for every $s \in S^T$ and store them in $O(m)$ time. Using Algorithms 1 and 2 we compute $D(x)$ on the top side of $R_k$. There are $n$ top-side events and we can handle them in $O(nm + n \log n)$ time. In addition, we compute the distances from $O(m)$ sites to each corner of $O(n)$ rectangles, and store them using $O(nm)$ space in total. Therefore, we have the following lemma.

\textbf{Lemma 8} We can handle all top-side events in $O(nm + n \log n)$ time using $O(nm)$ space.

\section{4.4 Constructing the Query Data Structure}

Initially, $Q_{y^+} = \emptyset$. For each site event and top-side event, we update $d_v(x)$ and $B(v)$ for node $v$ of $T$ corresponding to the event. We insert a horizontal segment $h$.
corresponding to each interval which is updated at the event into $Q_{y^+}$, and copy the boundary points into $h$. For each site event, at most one horizontal line segment $h$ is inserted. There is no boundary point in the interior of $h$, so we can insert $h$ with two endpoints in $O(1)$ time. For each top-side event, at most three horizontal line segments are inserted. They have $O(m)$ boundary points by Lemma 3, so we can copy them in $O(m)$ time. Thus, a farthest-neighbor query takes $O(\log(n+m))$ time in total.

Once the farthest sites of $q$ for each of the four data structures is found, we take the sites with the largest distance among them as the farthest sites $f(q)$ of $S$ from $q$. Combining Lemmas 4, 5 and 8 with query time, we have the following theorem.

**Theorem 1** We can construct a data structure for $m$ point sites in the presence of $n$ axis-aligned rectangular obstacles in the plane in $O(nm + n \log n + m \log m)$ time and $O(nm)$ space that answers any $L_1$ farthest-neighbor query in $O(\log(n+m))$ time.

5 Computing the Explicit Farthest-point Voronoi Diagram

We construct the explicit farthest-point Voronoi diagram $FVD = FVD(S, R)$ of a set $S$ of $m$ point sites in the presence of a set $R$ of $n$ rectangular obstacles in the plane. It is known that $FVD$ requires $\Omega(nm)$ space [7]. It takes $\Omega(n \log n)$ time to compute the geodesic distance between two points in $F$ [12]. By a reduction from the sorting problem, it can be shown to take $\Omega(m \log m)$ time for computing the farthest-point Voronoi diagram of $m$ point sites in the plane. We present an $O(nm + n \log n + m \log m)$-time algorithm using $O(nm)$ space that matches the time and space lower bounds. This is the first optimal algorithm for constructing the farthest-point Voronoi diagram of points in the presence of obstacles in the plane in both time and space.

We construct $Q_{y^+}$ using the plane sweep in Sect. 4. During the plane sweep, we find all horizontal edges of $M_{y^+}$ and insert them into $Q_{y^+}$ as segments. We find all
the lower endpoints of the vertical edges of $M_{y^+}$ and insert them as boundary points in $B$. We also find the upper endpoints of vertical edges of $M_{y^+}$. By connecting those endpoints using vertical segments appropriately, we can construct $M_{y^+}$ from $Q_{y^+}$ in a doubly connected edge list without increasing the time and space complexities. The other three maps can also be constructed in the same way in the same time and space.

We construct the farthest-point Voronoi diagram $FVD$ using the four maps explicitly. Note that $f(p) = f_{y^+}(p)$ for any point $p$ lying on the top side of $B$. Thus, it suffices to compute $FVD$ in $F \cap B$. For ease of description, we assume that the $x$-coordinates of the vertical sides of the rectangles in $R$ are all distinct. We consider a vertical decomposition $F_V$ obtained by drawing maximal vertical line segments contained in $F \cap B$, each of which extends from a vertical side of a hole of $F$. Let $V$ be the set of those vertical line segments. Then $F \setminus \bigcup_{\ell \in V} \ell$ consists of $O(n)$ connected faces. Each face is a rectangle since each hole of $F$ is a rectangle and $F$ is bounded by $B$. See Fig. 9a.

Any two farthest-point maps $M_1, M_2$ have a bisector which consists of the points in $F$ having the same distance to their farthest sites in $M_1$ and to their farthest sites in $M_2$. The four maps define six bisectors. In a face of $F_V$, the six bisectors and some axis-aligned segments partition the face into zones such that $FVD$ restricted to one zone coincides with the diagram in the corresponding region of a farthest-point map. Thus, we compute the bisectors between maps in each face of $F_V$, partition the face into zones, find the region of a farthest-point map corresponding to each zone, and then glue the regions and faces to compute $FVD$ completely.

5.1 Bisectors of farthest-point maps

We define the bisector between $M_\delta$ and $M_{\delta'}$ as $B(\delta, \delta') = \{q \in F \mid d_\delta(q) = d_{\delta'}(q)\}$ for any two distinct $\delta, \delta' \in \{y^+, y^-, x^+, x^-\}$. Note that a bisector may contain a two-dimensional region if there is a vertex of an obstacle equidistant from two distinct sites. It may consist of one or more connected components if there are points that are not $\delta$-reachable for $\delta \in \{y^+, y^-, x^+, x^-\}$. We show some structural and combinatorial properties of the bisectors between two farthest-point maps.

Lemma 9 Any vertical line intersects $B(y^+, y^-)$ in at most one point.
Proof Imagine that a point $p$ moves vertically upwards. We show that $d_{y^+}(p)$ increases as $p$ moves. By Corollary 1, $f_{y^+}(p)$ does not change until $p$ meets a horizontal edge of $M_{y^+}$, or a rectangle of $R$, and thus $d_{y^+}(p)$ increases as $p$ moves. When $p$ meets a horizontal edge of $M_{y^+}$, $f_{y^+}(p)$ changes, but $d_{y^+}(p)$ still increases by the definition of $f_{y^+}(p)$.

Consider the case that $p$ meets the bottom side of a rectangle $R \in R$. See Fig. 10. Let $p'$ be the point on the top side of $R$ with $x(p') = x(p)$. Then there is a geodesic path $\pi(p', f_{y^+}(p))$ that passes through the top-left (or the top-right) corner of $R$. Without loss of generality, assume that it passes through the top-left corner of $R$. By the definition of $f_{y^+}(p)$, $\pi_{d}(p)$ intersects $\pi(p', f_{y^+}(p))$ at a point, say $q$. Then,

$$d(p, f_{y^+}(p)) \leq d(p, q) + d(q, f_{y^+}(p)) < d(p', f_{y^+}(p)) \leq d(p', f_{y^+}(p')).$$

Therefore, $d(p, f_{y^+}(p))$ still increases as $p$ jumps to $p'$. Likewise, $d(p, f_{y^-}(p))$ decreases as $p$ moves vertically upwards. Then $d(p, f_{y^+}(p)) = d(p, f_{y^-}(p))$ occurs at most once at a moment for $p$ moving vertically upwards. Thus, any vertical line intersects $B(y^+, y^-)$ in at most one point. $\square$

By Lemma 9, $B(y^+, y^-)$ is $x$-monotone and it consists of segments of slopes $0$, $+1$, or $−1$.

Lemma 10 For any vertical line segment $\ell$ contained in $F$, $\ell \cap B(y^+, x^+)$ consists of at most one connected component.

Proof The proof is similar to the one for Lemma 9. Consider a vertical line segment $\ell$ contained in $F$. Imagine that a point $p$ moves vertically upwards from the lower endpoint of $\ell$ to the upper endpoint. We show that $d_{1}(p) = d_{y^+}(p) − y(p)$ does not decrease. By Corollary 1, $f_{y^+}(p)$ does not change until $p$ meets a horizontal edge of $M_{y^+}$, or a rectangle of $R$, and thus $d_{1}(p)$ remains the same as $p$ moves. When $p$ meets a horizontal edge of $M_{y^+}$, $f_{y^+}(p)$ changes and $d_{1}(p)$ increases by the definition of $f_{y^+}(p)$. 

![Fig. 10](image-url)
Fig. 11 Three bisectors of $M_{y^+}$ (red) and the other three farthest-point maps (blue). a $B(y^+, y^-)$. b $B(y^+, x^-)$. c $B(y^+, x^-)$. Bisectors may contain a two-dimensional region (Color figure online)

We also show that $d_2(p) = d_{x^+}(p) - y(p)$ does not increase. Recall that every distance function in $D$ consists of pieces of slopes 1 or $-1$ during the plane sweep in Sect. 4. As $d_1$ increases and $d_2$ does not increase, there is at most one connected component satisfying $d_{y^+}(p) = d_{x^+}(p)$ in $\ell$.

We can show that Lemma 10 also holds for $B(y^+, x^-)$. See Fig. 11 for bisectors. Lemmas 9 and 10 imply that $f(p') = f_{y^+}(p')$ if $p'p$ is a vertical line segment contained in $F$ with $y(p') > y(p)$, and $f(p) = f_{y^+}(p)$. Thus, these three bisectors contained in a face of $F_V$ are $x$-monotone.

For a face $f$ of $F_V$, the trace $T_f(y^+, y^-)$ of $f$ for $y^+$ and $y^-$ is an $x$-monotone curve contained in $f$ that connects the two vertical sides in $f$, and satisfying the following condition: any point $p \in f$ lying above $T_f(y^+, y^-)$ satisfies $d_{y^+}(p) \geq d_{y^-}(p)$, and any point $q \in f$ lying below the trace satisfies $d_{y^+}(q) < d_{y^-}(q)$. Observe that $T_f(y^+, y^-)$ is piecewise linear with pieces (segments) of slopes 1, 0 or $-1$. Each piece is a boundary segment of $B(y^+, y^-) \cap f$, or a $b$-edge of $M_{y^+}$ or $M_{y^-}$.

We compute $T_f(y^+, y^-)$ incrementally, from the leftmost segment to the rightmost segment. First, we compute the point $p$ of $T_f(y^+, y^-)$ that lies on the left side of $f$ as follows. There are $O(m)$ intersections of the left side of $f$ with the horizontal segments of $Q_{y^+}$ and $Q_{y^-}$ as any vertical line segment contained in $F$ intersects $O(m)$ horizontal segments of them. For each intersection point $q$, we compute $d_{y^+}(q)$ and $d_{y^-}(q)$, and find two consecutive points $q_1$ and $q_2$ among the intersection points by $y$-coordinate such that $d_{y^+}(q_1) \geq d_{y^-}(q_1)$ and $d_{y^+}(q_2) < d_{y^-}(q_2)$. We can compute $q_1$ and $q_2$ in $O(m)$ time using $Q_{y^+}$ and $Q_{y^-}$. Then we compute $p$ lying on $q_1q_2$.

The segment $H$ of $T_f(y^+, y^-)$ incident to the left side of $f$ is a boundary segment of $B(y^+, y^-) \cap f$, or a $b$-edge of $M_{y^+}$ or $M_{y^-}$. We compute the right endpoint of $H$ as follows. For the case that $H$ is a boundary segment of $B(y^+, y^-) \cap f$, the slope of $H$ is determined by using the distance functions $d_{y^+}$ and $d_{y^-}$. The right endpoint of $H$ is the first point $p$ on the ray from the left endpoint with the slope of $H$ at which the slope of $d_{y^+}(p)$ or $d_{y^-}(p)$ changes. For the case that $H$ is a $b$-edge of $M_{y^+}$ or $M_{y^-}$, $H$ is horizontal. The right endpoint of $H$ is the leftmost point $p$ on the $b$-edge with $d_{y^+}(p) = d_{y^-}(p)$.

The slope of $d_{y^+}(p)$ (and $d_{y^-}(p)$) changes at most once within a cell of $M_{y^+}$ (and $M_{y^-}$) for $p$ moving from the left endpoint to the right endpoint of $H$. Thus, we can find

\[\therefore\] Springer
the right endpoint of $H$ in time linear to the total complexity of the cells of $M_{y^+}$ and $M_{y^-}$ intersected by $H$. By repeating this procedure until $L$ hits the right side of $f$, we can compute $T_f(y^+, y^-)$. Observe that each cell $C$ of $M_{y^+}$ (and $M_{y^-}$) contained in $f$ is $y$-monotone and each horizontal edge of $C$ is a $b$-edge. Since $T_f(y^+, y^-)$ never crosses a $b$-edge, $C \cap T_f(y^+, y^-)$ is connected. Thus, we can compute $T_f(y^+, y^-)$ in time linear to the total complexity of the cells of $M_{y^+}$ and $M_{y^-}$ intersected by $T_f(y^+, y^-)$. Since there are no obstacles in $f$, the total complexity of the cells of $M_{y^+}$ (and $M_{y^-}$) contained in $f$ is $O(m)$. We compute $T_f(y^+, x^+)$ for $y^+$ and $x^+$, and $T_f(y^+, x^-)$ for $y^+$ and $x^-$ in a similar way in $O(m)$ time. We do this for every face of $F_V$ and compute the traces in $O(nm)$.

We compute the distance functions using $Q_{y^+}, Q_{y^-}, Q_{x^+}$, and $Q_{x^-}$ which consist of $O(n+m)$ line segments and support $O(\log(n+m))$ query time. After computing those distance functions, the traces can be constructed in time linear to the total of complexity of the cells contained in every faces. Thus, in total it takes $O(nm+n\log n+m\log m)$ time to construct the traces for all faces.

### 5.2 Partitioning $f$ into Zones

With the three traces $T_f(y^+, y^-), T_f(y^+, x^+), T_f(y^+, x^-)$ in $f$, we compute the zone $Z_{y^+}$ in $f$ corresponding to $M_{y^+}$ in $f$. Let $T$ be an upper envelope of $T_f(y^+, y^-), T_f(y^+, x^+)$ and $T_f(y^+, x^-)$. Then $Z_{y^+}$ is the set of points lying above $T$ in $f$. See Fig. 9b. The following lemma can be shown using Lemmas 9 and 10 in Sect. 5.1.

**Lemma 11** For any point $p \in Z_{y^+}$, $f(p) = f_{y^+}(p)$.

Similarly, we define the other three zones $Z_{y^-}, Z_{x^+}$, and $Z_{x^-}$. Note that $d_\delta(p) > d_{\delta'}(p)$ for every point $p \in Z_\delta$ for distinct $\delta, \delta' \in \{y^+, y^-, x^+, x^-, x^-, x^-, x^-\}$. By Lemma 11, $FVD \cap Z_{y^+}$ coincides with $M_{y^+}$. We copy the corresponding farthest-point map of $\delta$ into $Z_\delta$ for each $\delta \in \{y^+, y^-, x^+, x^-, x^-, x^-, x^-\}$.

We call $f \setminus (Z_{y^+} \cup Z_{y^-} \cup Z_{x^+} \cup Z_{x^-})$ the bisector zone. Every point $p$ in the bisector zone lies on a bisector of two or more maps. Thus, for each bisector of two maps, we copy one of the maps into the corresponding zone.

### 5.3 Gluing Along Boundaries

We first glue the zones along their boundaries in each face of $F_V$. For each edge $e$ incident to two zones, we check whether the two cells incident to the edge have the same farthest site or not. If they have the same farthest site, $e$ is not a Voronoi edge of FVD. Then we remove the edge and merge the cells into one. If they have different farthest sites, $e$ is a Voronoi edge of FVD. This takes $O(nm)$ time in total, which is linear to the number of Voronoi edges and cells in FVD.

After gluing zones in every face, we glue the faces of $F_V$ along their boundaries. Since $e$ is a vertical line segment and incident to more than two cells, we divide $e$ into pieces such that any point in the same piece $e'$ is incident to the same set of two cells. If both cells incident to $e'$ have the same farthest site, $e'$ is not a Voronoi edge of FVD. Then we remove the edge and merge the cells. If they have different farthest sites, $e'$ is
a Voronoi edge of FVD. There are $O(n)$ vertical line segments in $V$ and each of them intersects $O(m)$ cells of FVD. Thus, this takes $O(nm)$ time in total. Then we obtain the geodesic $L_1$ farthest-point Voronoi diagram FVD explicitly. See Fig. 9c.

**Theorem 2** We can compute the $L_1$ farthest-point Voronoi diagram of $m$ point sites in the presence of $n$ axis-aligned rectangular obstacles in the plane in $O(nm + n \log n + m \log m)$ time and $O(nm)$ space.

By computing FVD, we can also compute the $L_1$ geodesic center. Note that the $L_1$ geodesic center is a Voronoi vertex of FVD. While computing FVD, we also store $d_v$ for each cell of FVD. Using $d_v$, we can compute the geodesic distance between each Voronoi vertex of FVD and the farthest site from it in $O(1)$ time. Therefore, by using $O(nm)$ additional time, we can find the $L_1$ geodesic center among the vertices.

**Corollary 3** We can compute the $L_1$ geodesic center of $m$ point sites in the presence of $n$ axis-aligned rectangular obstacles in the plane in $O(nm + n \log n + m \log m)$ time and $O(nm)$ space.

### 6 Concluding Remarks

We present an optimal algorithm for computing the farthest-point Voronoi diagram of point sites in the presence of rectangular obstacles. However, our algorithm may not work for more general obstacles as it is, because some properties we use for the axis-aligned rectangles including their convexity may not hold any longer. Our results, however, may serve as a stepping stone to closing the gap to the optimal bounds.

### Declarations

**Conflict of interest** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

### References

1. Aggarwal, A., Guibas, L.J., Saxe, J., Shor, P.W.: A linear-time algorithm for computing the Voronoi diagram of a convex polygon. Discrete Comput. Geom. 4(6), 591–604 (1989)
2. Alt, H., Cheong, O., Vigneron, A.: The Voronoi diagram of curved objects. Discrete Comput. Geom. 34(3), 439–453 (2005)
3. Aronov, B.: On the geodesic Voronoi diagram of point sites in a simple polygon. Algorithmica 4(1), 109–140 (1989)
4. Aronov, B., Fortune, S., Wilfong, G.: The furthest-site geodesic Voronoi diagram. Discrete Comput. Geom. 9(3), 217–255 (1993)
5. Bae, S.W., Chwa, K.-Y.: The geodesic farthest-site Voronoi diagram in a polygonal domain with holes. In: Proceedings of the 25th annual symposium on computational geometry (SoCG), pp. 198–207 (2009)
6. Ben-Moshe, B., Bhattacharya, B.K., Shi, Q.: Farthest neighbor Voronoi diagram in the presence of rectangular obstacles. In: Proceedings of the 13th Canadian conference on computational geometry (CCCG), pp. 243–246 (2005)
7. Ben-Moshe, B., Katz, M.J., Mitchell, J.S.B.: Farthest neighbors and center points in the presence of rectangular obstacles. In: Proceedings of the 17th annual symposium on computational geometry (SoCG), pp. 164–171 (2001)

@Springer
8. Cheong, O., Everett, H., Glisse, M., Gudmundsson, J., Hornus, S., Lazard, S., Lee, M., Na, H.-S.: Farthest-polygon Voronoi diagrams. Comput. Geom. 44(4), 234–247 (2011)

9. Chew, L.P., Dyrsdale, R.L.: III. Voronoi diagrams based on convex distance functions. In: Proceedings of the 1st annual symposium on Computational geometry (SoCG), pp. 235–244 (1985)

10. Choi, J., Shin, C.-S., Kim, S.K.: Computing weighted rectilinear median and center set in the presence of obstacles. In: International symposium on algorithms and computation, pp. 30–40. Springer, Berlin (1998)

11. Choi, J., Yap, C.: Monotonicity of rectilinear geodesics in d-space. In: Proceedings of the 12th annual symposium on computational geometry (SoCG), pp. 339–348 (1996)

12. De Rezende, P.J., Lee, D.-T., Wu, Y.-F.: Rectilinear shortest paths with rectangular barriers. In: Proceedings of the 1st annual symposium on computational geometry (SoCG), pp. 204–213 (1985)

13. Edelsbrunner, H., Guibas, L.J., Stolfi, J.: Optimal point location in a monotone subdivision. SIAM J. Comput. 15(2), 317–340 (1986)

14. Edelsbrunner, H., Seidel, R.: Voronoi diagrams and arrangements. Discrete Comput. Geom. I(1), 25–44 (1986)

15. Fortune, S.: A sweepline algorithm for Voronoi diagrams. Algorithmica 2(1), 153–174 (1987)

16. Hershberger, J., Suri, S.: An optimal algorithm for Euclidean shortest paths in the plane. SIAM J. Comput. 28(6), 2215–2256 (1999)

17. Klein, R.: Abstract Voronoi diagrams and their applications. In: Proceedings of the 4th international workshop on computational geometry (EuroCG), pp. 148–157. Springer, Berlin (1988)

18. Lee, D.-T.: Two-dimensional Voronoi diagrams in the $L_p$-$L_p$-metric. J. ACM 27(4), 604–618 (1980)

19. Mitchell, J.S.B.: $L_1$-$L_1$ shortest paths among polygonal obstacles in the plane. Algorithmica 8(1–6), 55–88 (1992)

20. Oh, E.: Optimal algorithm for geodesic nearest-point Voronoi diagrams in simple polygons. In: Proceedings of the 30th annual ACM-SIAM symposium on discrete algorithms (SODA), pp. 391–409 (2019)

21. Oh, E., Ahn, H.-K.: Voronoi diagrams for a moderate-sized point-set in a simple polygon. Discrete Comput. Geom. 63(2), 418–454 (2020)

22. Oh, E., Barba, L., Ahn, H.-K.: The geodesic farthest-point Voronoi diagram in a simple polygon. Algorithmica 82(5), 1434–1473 (2020)

23. Papadopoulou, E., Dey, S.K.: On the farthest line-segment Voronoi diagram. Int. J. Comput. Geom. Appl. 23(06), 443–459 (2013)

24. Papadopoulou, E., Lee, D.T.: The $L_\infty$-$L_0$ Voronoi diagram of segments and VLSI applications. Int. J. Comput. Geom. Appl. 11(05), 503–528 (2001)

25. Sarnak, N., Tarjan, R.E.: Planar point location using persistent search trees. Commun. ACM 29(7), 669–679 (1986)

26. Shamos, M.I., Hoey, D.: Closest-point problems. In: Proceedings of the 16th IEEE annual symposium on foundations of computer science (FOCS), pp. 151–162 (1975)

27. Wang, H.: An optimal deterministic algorithm for geodesic farthest-point Voronoi diagrams in simple polygons. In: Proceedings of the 37th international symposium on computational geometry (SoCG), pp. 59:1–59:15 (2021)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.