Minimum-Link Shortest Paths for Polygons amidst Rectilinear Obstacles*

Mincheol Kim† Hee-Kap Ahn‡

Abstract

Consider two axis-aligned rectilinear simple polygons in the domain consisting of axis-aligned rectilinear obstacles in the plane such that the bounding boxes, one for each obstacle and one for each polygon, are disjoint. We present an algorithm that computes a minimum-link rectilinear shortest path (a rectilinear shortest path with the minimum number of line segments) connecting the two polygons in $O((N+n)\log(N+n))$ time using $O(N+n)$ space, where n is the number of vertices in the domain and N is the total number of vertices of the two polygons.

1 Introduction

The problem of finding paths connecting two objects amidst obstacles has been studied extensively in the past. It varies on the underlying metric (Euclidean, rectilinear, etc.), types of obstacles (simple polygons, rectilinear polygons, rectangles, etc.), and objective functions (minimum length, minimum number of links, or their combinations). See the survey in Chapter 31 of Handbook of Discrete and Computational Geometry [17] on various approaches to this problem and results.

For two points p and q contained in the plane, possibly with rectilinear polygonal obstacles (i.e., a *rectilinear domain*), a rectilinear shortest path from p to q is a rectilinear path from p to q with minimum total length that avoids the obstacles. In the rest of the paper, we say a *shortest path* to refer to a rectilinear shortest path unless stated otherwise. A rectilinear path consists of horizontal and vertical segments, each of which is called *link*. Among all shortest paths from p to q, we are interested in a *minimum-link* shortest path from p to q, that is, a shortest path with the minimum number of links (or one with the minimum number of bends). There has been a fair amount of work on finding minimum-link shortest paths connecting two points amidst rectilinear obstacles in the plane [1, 9, 18, 19, 21].

These definitions are naturally extended to two more general objects contained in the domain. A shortest path connecting the objects is one with minimum path length among all shortest paths from a point of one object to a point of the other object. A minimum-link shortest path connecting the objects is a minimum-link path among all shortest paths.

In this paper, we consider the problem of finding minimum-link shortest paths connecting two objects in a rectilinear domain, which generalizes the case of connecting two points, in some modest environment. The rectilinear polygonal obstacles are considered as open sets. Two axis-aligned rectilinear polygons are said to be *box-disjoint* if the axis-aligned bounding boxes, one for each rectilinear polygon, are disjoint in their interiors. A set of axis-aligned rectilinear polygons is box-disjoint if the polygons of the set are pairwise box-disjoint. The rectilinear domain induced by a set of box-disjoint rectilinear polygons in the plane is called a *box-disjoint rectilinear domain.* We require the input objects and the obstacles in the domain to be also pairwise box-disjoint, unless stated otherwise.

Problem definition. Given two axis-aligned rectilinear simple polygons S and T in a rectilinear domain in the plane such that S, T, and the obstacles in the domain are pairwise box-disjoint, find a minimum-link rectilinear shortest path from S to T.

Related Works. Computing shortest paths or minimum-link paths in a polygonal domain has been studied extensively. When obstacles are all rectangles, Rezende et al. [6] presented an algorithm with $O(n \log n)$ time and $O(n)$ space to compute a shortest path connecting two points amidst n rectangles. For a rectilinear domain with n vertices, Mitchell [11] gave an algorithm with $O(n \log n)$ time and $O(n)$ space to compute a shortest path connecting two points using a method based on the continuous Dijkstra paradigm [10]. Later, Chen and Wang [2] improved the time complexity to $O(n + h \log h)$ for a triangulated polygonal domain with h holes.

Computing a minimum-link path, not necessarily

*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)).

†Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea. rucatia@postech.ac.kr

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

shortest, in a polygonal domain has also been studied well. For a minimum-link rectilinear path connecting two points in a rectilinear domain with n vertices, Imai and Asano [8] gave an algorithm with $O(n \log n)$ time and space. Then a few algorithms improved the space complexity to $O(n)$ without increasing the running time [4, 12, 15]. Very recently, Mitchell [13] gave an algorithm with $O(n + h \log h)$ time and $O(n)$ space for triangulated rectilinear domains with h holes.

Yang et al. [19] considered the problem of finding a rectilinear path connecting two points amidst rectilinear obstacles under a few optimization criteria, such as a minimum-link shortest path, a shortest minimum-link path, and a least-cost path (a combination of link cost and length cost). By constructing a path-preserving graph, they gave a unified approach to compute such paths in $O(ne+n \log n)$ time, where n is the total number of polygon edges and e is the number of polygon edges connecting two convex vertices. The space complexity is $O(ne)$ due to the path-preserving graph of size $O(ne)$. Since e is $O(n)$, the running time becomes $O(n^2)$ in the worst case, even for convex rectilinear polygons (obstacles). A few years later, they gave two algorithms on the problem [21], improving their previous result, one with $O(n \log^2 n)$ time and $O(n \log n)$ space and the other with $O(n \log^{3/2} n)$ time and $O(n \log^{3/2} n)$ space using a combination of a graph-based approach and the continuous Dijkstra approach. It is claimed in [9] that a minimum-link shortest path can be computed in $\Theta(n \log n)$ time and $O(n)$ space when obstacles are rectangles by the algorithm in another paper [20] by the same authors, but the paper [20] is not available.

Later, Chen et al. [1] gave an algorithm improving the previous results for finding a minimum-link shortest path connecting two points in $O(n \log^{3/2} n)$ time and $O(n \log n)$ space using an implicit representation of a reduced visibility graph, instead of computing the whole graph explicitly. Very recently, Wang [18] pointed out a flaw in the algorithm in [1] and claimed that to make it work, each vertex of the graph must store a constant number of nonlocal optimum paths together with local optimum paths. Wang gave an algorithm with $O(n + h \log^{3/2} h)$ time and $O(n+h \log h)$ space using a reduced path-preserving graph from the corridor structure [13] and the histogram partitions [16], where h is the number of holes (obstacles) in the rectilinear domain.

However, we are not aware of any result on computing the minimum-link shortest path connecting two objects other than points.

Our Results. We consider the minimum-link shortest path problem for two axis-aligned rectilinear polygons S and T in a box-disjoint rectilinear domain. This generalizes the two-point shortest path problem to two-polygon shortest path problem. The theorem below summarizes our results.

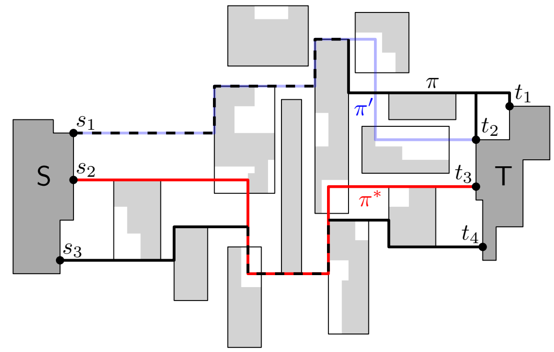


Figure 1: The box-disjoint rectilinear domain with S, T. Light gray rectilinear polygons are obstacles. There are six pairs of points, (s_1, t_1), (s_1, t_2), (s_2, t_3), (s_2, t_4), (s_3, t_3), and (s_3, t_4), that determine the length of a shortest path from S to T. The path π from s_1 to t_1 (or t_2) is a minimum-link shortest path from S to T with eight links among all shortest paths without intersecting the interiors of bounding boxes of obstacles. However, the blue path π' from s_1 to t_2 has seven links and the red path π^* from s_2 to t_3 has five links, which is optimal.

Theorem 1 *Let S and T be two axis-aligned rectilinear simple polygons with N vertices in a rectilinear domain with n vertices in the plane such that S, T, and the obstacles in the domain are pairwise box-disjoint. We can compute a minimum-link shortest path from S to T in $O((N+n) \log(N+n))$ time using $O(N+n)$ space.*

The main difficulty lies in computing a shortest path from S to T. The length of a shortest path from S to T is determined by a pair of points, one lying on the boundary of S and one lying on the boundary of T. Such a point is a vertex of S or T, or the point on the boundary of S or T where a horizontal or vertical ray emanating from a vertex in the domain hits first. Since the domain has $O(N+n)$ vertices, there are $O(N+n)$ such points on the boundaries of S and T, and $O((N+n)^2)$ pairs of points, one from S and the other from T, to consider in order to determine the length of a shortest path. Thus, if we use a naive approach that computes a minimum-link shortest path for each point pair, it may take $\Omega((N+n)^2)$ time. Theorem 1 shows that our algorithm computes a minimum-link shortest path from S to T efficiently. Also, a minimum-link shortest path may intersect the bounding box of an obstacle, although S, T, and obstacles are pairwise box-disjoint. See Figure 1.

We first consider a simpler problem for an axis-aligned line segment S and a point t contained in the domain consisting of axis-aligned rectangular obstacles. We partition the domain into at most eight regions using eight xy-monotone paths from S. We observe that every shortest path from S to a point in a region is either x-, y-, or xy-monotone [6]. Moreover, we define a set of $O(n)$ *baselines* for each region, and show that there is a minimum-link shortest path from S to t consisting of segments contained in the baselines. Based on these observations, our algorithm applies a plane sweep technique with a sweep line moving from S to t and computes the minimum numbers of links from S to the intersections of the baselines and the sweep line efficiently. After the sweep line reaches t, our algorithm reports a minimum-link shortest path that can be obtained from a reverse traversal from t using the number of links stored in baselines. During the sweep, our algorithm maintains a data structure storing baselines (and their minimum numbers of links) and updates the structure for the segments (events) on the boundary of the region.

It takes, however, $O(n^2)$ time using $O(n)$ space. To reduce the time complexity without increasing the space complexity, our algorithm maintains another data structure, a *balanced binary search tree,* each node of which corresponds to a set of consecutive baselines. This tree behaves like a segment tree [5]. Instead of updating the minimum numbers of links of $O(n)$ baselines at each event of the plane sweep algorithm, we update $O(\log n)$ nodes of the tree that together correspond to the baselines. This improves the time for handling each sweep-line event from $O(n)$ to $O(\log n)$, and thus improving the total time complexity to $O(n \log n)$.

Then we extend our algorithm to handle a line segment T (not a point t) and box-disjoint rectilinear obstacles (not necessarily rectangles). We observe that every shortest path contained in a region from S to any point of T is either x-, y-, or xy-monotone, so our algorithm partitions the domain into at most eight regions again. Then T intersects at most five regions. Our algorithm computes a minimum-link shortest path from S to T' for the portion T' of T contained in each region, and then returns the minimum-link shortest path among the paths. Also, we consider that the input objects are rectilinear simple polygons S and T with N vertices. Recall that there are $O((N+n)^2)$ pairs of points that determine the length of a shortest path from S to T. To handle them efficiently, we add $O(N)$ additional baselines and $O(N)$ events induced by S and T during the plane sweep algorithm. Then the number of events becomes $O(N+n)$ and the time to handle each event takes $O(\log(N+n))$, so we obtain Theorem 1.

Due to lack of space, some of the proofs and details are omitted. They are available in Appendix.

2 Preliminaries

Let R be a set of n disjoint axis-aligned rectangles in $\mathbb{R}^2$. Each rectangle $R \in \mathsf{R}$ is considered as an open set and plays as an obstacle in computing a minimum-link shortest path in the plane. We let $\mathsf{D} := \mathbb{R}^2 - \cup_{R \in \mathsf{R}} R$ and call it the *rectangular domain* induced by R in the plane. For two points p and q in D, $d(p,q)$ denotes the L_1 distance (or the Manhattan distance) from p to q in D, that is, the length of a shortest path from p to q avoiding the obstacles. A path is *x-monotone* if the intersection of the path with any line perpendicular to the x-axis is connected. Likewise, a path is *y-monotone* if the intersection of the path with any line perpendicular to the y-axis is connected. If a path is x-monotone and y-monotone, the path is *xy-monotone*.

For two objects S and T in D, $d(\mathsf{S},\mathsf{T}) = \min_{p \in \mathsf{S}, q \in \mathsf{T}} d(p,q)$. A shortest path from S to T is a path in D from a point $p \in \mathsf{S}$ to a point $q \in \mathsf{T}$ of length $d(\mathsf{S},\mathsf{T})$. A *minimum-link shortest path* from S to T is a path that has the minimum number of links among all shortest paths from S to T in D, and we use $\lambda(\mathsf{S},\mathsf{T})$ to denote the number of links of a minimum-link shortest path from S to T. We call a pair (p,q) of points with $p \in \mathsf{S}$ and $q \in \mathsf{T}$ such that $d(\mathsf{S},\mathsf{T}) = d(p,q)$ a *closest pair* of points of S and T. We say p is a closest point of S from T, and q is a closest point of T from S. Note that there can be more than one closest pair of points of S and T.

We make an assumption that the rectangles are in *general position*, that is, no two rectangles in R have corners, one corner from each rectangle, with the same x- or y-coordinate. A horizontal line segment H can be represented by the two x-coordinates $x_1(H)$ and $x_2(H)$ of its endpoints ($x_1(H) < x_2(H)$) and the y-coordinate $y(H)$ of them. Likewise, a vertical line segment V can be represented by the two y-coordinates $y_1(V)$ and $y_2(V)$ of its endpoints ($y_1(V) < y_2(V)$) and the x-coordinate $x(V)$ of them.

2.1 Eight disjoint regions of a rectangular domain

Given a rectangular domain D and a vertical segment S, we partition D into at most eight disjoint regions by using eight xy-monotone paths from the endpoints of S in a way similar to the one by Choi and Yap [3]. Consider a horizontal ray from a point $p = p_1$ on S going rightwards. The ray stops when it hits a rectangle $R \in \mathsf{R}$ 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 $\pi_{\mathsf{ru}}(p) = (p = p_1 p_1' p_2 p_2' \ldots)$. In other words, $\pi_{\mathsf{ru}}(p)$ is an xy-monotone path from p that alternates going *rightwards* (until hitting a rectangle) and going *upwards* (to the top-left corner of the rectangle).

By choosing two directions, one going either rightwards or leftwards horizontally, and one going either upwards or downwards vertically, and order-

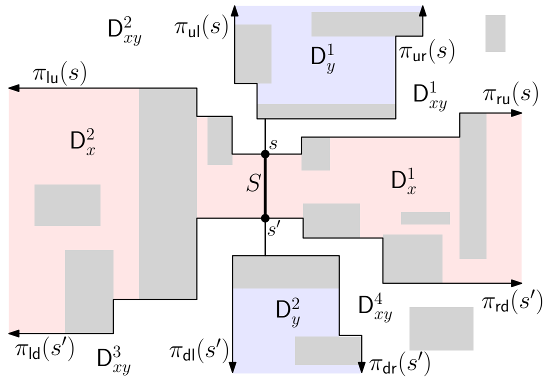


Figure 2: Eight disjoint regions of D by eight xy-monotone paths from s or s'. Gray rectangles are obstacles.

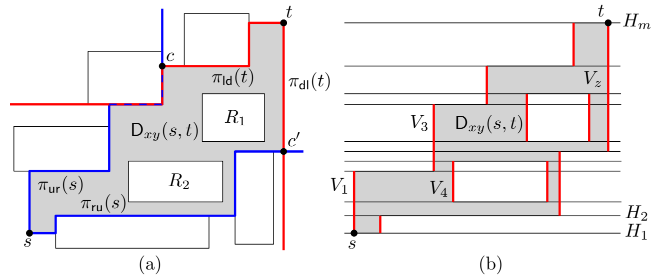


Figure 3: (a) $\mathsf{D}_{xy}(s,t)$ is the region of D_{xy} enclosed by the closed curve composed of $\pi_{\mathsf{ur}}(s,c)$, $\pi_{\mathsf{ld}}(t,c)$, $\pi_{\mathsf{ru}}(s,c')$, and $\pi_{\mathsf{dl}}(t,c')$. R_1 and R_2 are the holes of $\mathsf{D}_{xy}(s,t)$. (b) Horizontal baselines $H_1, H_2, \ldots, H_m$ of $\mathsf{D}_{xy}(s,t)$ and vertical segments (red) $V_1, V_2, \ldots, V_z$ on the boundary of $\mathsf{D}_{xy}(s,t)$.

ing the chosen directions, we define eight rectilinear xy-monotone paths with directions: rightwards-upwards (ru), upwards-rightwards (ur), upwards-leftwards (ul), leftwards-upwards (lu), leftwards-downwards (ld), downwards-leftwards (dl), downwards-rightwards (dr), and rightwards-downwards (rd). We use $\pi_\alpha(p)$ to denote them, where α is one in $\{\mathsf{ru}, \mathsf{ur}, \mathsf{ul}, \mathsf{lu}, \mathsf{ld}, \mathsf{dl}, \mathsf{dr}, \mathsf{rd}\}$. Also, we use $\pi_\alpha(p,q)$ to denote the subpath of $\pi_\alpha(p)$ from p to $q \in \pi_\alpha(p)$.

Figure 2 illustrates these eight xy-monotone paths, four upward paths from the upper endpoint s of S and four downward paths from the lower endpoint s' of S. Observe that for a point $p \in \mathsf{D}$, the eight paths $\pi_\alpha(p)$ do not cross each other. Thus, by the eight paths, D is partitioned into eight regions. See Figure 2. We denote by D^1_{xy} (and D^2_{xy}, D^3_{xy}, D^4_{xy}) the region bounded by $\pi_{\mathsf{ru}}(s)$ and $\pi_{\mathsf{ur}}(s)$ (and by $\pi_{\mathsf{ul}}(s)$ and $\pi_{\mathsf{lu}}(s)$, by $\pi_{\mathsf{ld}}(s')$ and $\pi_{\mathsf{dl}}(s')$, by $\pi_{\mathsf{dr}}(s')$ and $\pi_{\mathsf{rd}}(s')$). We denote by D^1_x (and D^2_x) the region bounded by $\pi_{\mathsf{ru}}(s)$ and $\pi_{\mathsf{rd}}(s')$ (and by $\pi_{\mathsf{lu}}(s)$ and $\pi_{\mathsf{ld}}(s')$), and denote by D^1_y (and D^2_y) the region bounded by $\pi_{\mathsf{ur}}(s)$ and $\pi_{\mathsf{ul}}(s)$ (and by $\pi_{\mathsf{dl}}(s')$ and $\pi_{\mathsf{dr}}(s')$).

Lemma 2 *For a point $t \in \cup_{1\le i\le 4}\mathsf{D}^i_{xy}$, every shortest path from S to t is xy-monotone. For a point $t \in \cup_{1\le i\le 2}\mathsf{D}^i_x$, every shortest path from S to t is x-monotone. For a point $t \in \cup_{1\le i\le 2}\mathsf{D}^i_y$, every shortest path from S to t is y-monotone.*

From now on we simply use D_{xy}, D_x and D_y to denote D^1_{xy}, D^1_x and D^1_y, respectively, and assume that t lies in a region D' of the regions. The case that t lies in other regions can be handled analogously. For each horizontal side of the rectangles incident to D', we call the horizontal line containing the side a *horizontal baseline* of D'. Similarly, for each vertical side of the rectangles incident to D', we call the vertical line containing the side a *vertical baseline* of D'. The two vertical lines through S and t, and the three horizontal lines through s, s' and t are also regarded as vertical and horizontal baselines of D', respectively. We say a minimum-link shortest path π is *aligned to the baselines* if every segment of π is contained in a baseline of the corresponding region. By using Lemma 3, we find a minimum-link shortest path aligned to the baselines of each region.

Lemma 3 *There is a minimum-link shortest path from S to t that is aligned to the baselines of D'.*

3 t lies in D_{xy}

We consider the case that t lies in D_{xy}. By Lemma 2, every shortest path from S to t is xy-monotone and connects the upper endpoint s of S and t. Let c be the point with the maximum x-coordinate and the maximum y-coordinate among the points in $\pi_{\mathsf{ur}}(s) \cap \pi_{\mathsf{ld}}(t)$. Observe that c is defined uniquely as $\pi_{\mathsf{ur}}(s) \cap \pi_{\mathsf{ld}}(t)$ is connected and xy-monotone by the definition. Likewise, let c' be the point with the maximum x-coordinate and the maximum y-coordinate among the points in $\pi_{\mathsf{ru}}(s) \cap \pi_{\mathsf{dl}}(t)$. Then we use $\mathsf{D}_{xy}(s,t)$ to denote the region of D_{xy} enclosed by the closed curve composed of $\pi_{\mathsf{ur}}(s,c)$, $\pi_{\mathsf{ld}}(t,c)$, $\pi_{\mathsf{ru}}(s,c')$, and $\pi_{\mathsf{dl}}(t,c')$. We denote by $\partial_{xy}(s,t)$ the rectilinear chain of the outer boundary of $\mathsf{D}_{xy}(s,t)$ from s to t in clockwise, and denote by $\partial_{xy}(t,s)$ the rectilinear chain of the outer boundary of $\mathsf{D}_{xy}(s,t)$ from t to s in clockwise. See Figure 3(a) for an illustration. By Lemma 2, every shortest path from s to t is contained in $\mathsf{D}_{xy}(s,t)$, and therefore every minimum-link shortest path from s to t is also contained in $\mathsf{D}_{xy}(s,t)$.

We focus on the baselines of D_{xy} that are defined by s, t, and the rectangles incident to $\mathsf{D}_{xy}(s,t)$, which we call the baselines of $\mathsf{D}_{xy}(s,t)$. Figure 3(b) shows the horizontal baselines of $\mathsf{D}_{xy}(s,t)$. Note that a baseline may cross rectangles incident to $\mathsf{D}_{xy}(s,t)$. Let $H_1, H_2, \ldots, H_m$ be the m horizontal baselines of $\mathsf{D}_{xy}(s,t)$ such that $y(H_1) < y(H_2) < \ldots < y(H_m)$. Note that s is on H_1 and t is on H_m.

3.1 Computing the minimum number of links

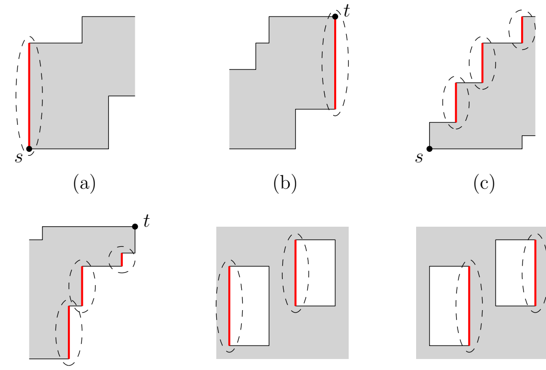
Consider a minimum-link shortest path aligned to the baselines of $\mathsf{D}_{xy}(s,t)$. For the rightmost vertical segment V of $\mathsf{D}_{xy}(s,t)$, we have $y_2(V) = y(t)$ and $y_1(V) = y(H_{m'})$ for some horizontal baseline $H_{m'}$ with $m' < m$. We can compute a minimum-link shortest path once we have a minimum-link shortest path from s to the intersection point c_i of V and H_i for each $i = m', m'+1, \ldots, m$, since t is the endpoint of V.

We compute $\lambda(s,t)$ by applying the plane sweep algorithm, and then report a minimum-link shortest path aligned to the baselines of $\mathsf{D}_{xy}(s,t)$ that can be obtained from a reverse traversal from t using $\lambda(s,t)$.

Imagine a vertical line L sweeping $\mathsf{D}_{xy}(s,t)$ rightwards. Our plane sweep algorithm maintains a data structure storing horizontal baselines and their minimum numbers of links among shortest paths from s to intersections of baselines and L such that the line segments incident to the intersections of those shortest paths are horizontal. The algorithm updates their status and minimum numbers of links when L encounters the vertical segments (vertical baselines) on the boundary of $\mathsf{D}_{xy}(s,t)$.

We define the status for each horizontal baseline as follows. For the intersection point $c_i = H_i \cap L$ for each $i = 1, \ldots, m$, if $c_i \in \mathsf{D}_{xy}(s,t)$, then H_i is *active*. Otherwise, H_i is *inactive*. Observe that a baseline may switch its status between active and inactive, depending on the position of L, and these switches occur only when L encounters a vertical segment on the boundary of $\mathsf{D}_{xy}(s,t)$. During the sweep, we maintain the active baselines of $\mathsf{D}_{xy}(s,t)$ in a set of ranges with respect to their indices in a range tree $\mathcal{T}_{\mathsf{ran}}$. A range $[a,b]$ contained in $\mathcal{T}_{\mathsf{ran}}$ represents a set of active baselines $H_a, H_{a+1}, \ldots, H_b$, consecutive in their indices from a to b. Every range $[a,b]$ in $\mathcal{T}_{\mathsf{ran}}$ is *maximal* in the sense that H_{a-1} and H_{b+1} are inactive or not defined in $\mathsf{D}_{xy}(s,t)$. We use $M(i)$ to denote the minimum number of links among all shortest paths from s to c_i whose segment incident to c_i is horizontal.

We maintain $M(i)$'s for horizontal baselines during the plane sweep as follows. There are vertical line segments $V_1, V_2, \ldots, V_z$ on the boundary of $\mathsf{D}_{xy}(s,t)$, satisfying $x(V_1) < x(V_2) < \ldots < x(V_z)$. Note that the lower endpoint of V_1 is s and the upper endpoint of V_z is t. We consider each vertical segment V_j ($1 \le j \le z$) of $\mathsf{D}_{xy}(s,t)$ as an event, denoted by E_j, because we compute a minimum-link shortest path aligned to the baselines of $\mathsf{D}_{xy}(s,t)$, so $M(i)$ changes only when L encounters a vertical segment. For each E_j, we use $\alpha(j)$ and $\beta(j)$ (with $\alpha(j) < \beta(j)$) to denote the indices such that $y_1(V_j) = y(H_{\alpha(j)})$ and $y_2(V_j) = y(H_{\beta(j)})$, respectively. E_j belongs to one of the following six types depending on the boundary part of $\mathsf{D}_{xy}(s,t)$ that V_j lies on. See Figure 4 for an illustration of each type.



(a) (b) (c) (d) (e) (f)

Figure 4: Six types of events of the plane sweep algorithm. (a) originate event. (b) terminate event. (c) attach event. (d) detach event. (e) split event. (f) merge event.

- E_1 belongs to type originate and E_z belongs to type terminate.
- E_j for each $j = 2, \ldots, z-1$ belongs to type attach if V_j lies on $\partial_{xy}(s,t)$, and to type detach if V_j lies on $\partial_{xy}(t,s)$.
- E_j belongs to type split if V_j is the left side of a hole of $\mathsf{D}_{xy}(s,t)$, and to type merge if V_j is the right side of a hole.

The events are sorted by their x-coordinates. During the sweep, L encounters E_j when $x(L) = x(V_j)$. Initially, the tree $\mathcal{T}_{\mathsf{ran}}$ contains no range, and $M(i)$ is set to ∞ for all horizontal baselines H_i. When L encounters E_1, which is the originate event with $\alpha(1) = 1$, we update $M(\alpha(1)) := 1$ and $M(k) := 2$ for each $k \in [\alpha(1)+1, \beta(1)]$, and insert the range $[\alpha(1), \beta(1)]$ into $\mathcal{T}_{\mathsf{ran}}$.

If E_j is an attach event, the inactive baselines H_i for i from $\alpha(j)+1$ to $\beta(j)$ become active. Observe that there always exists a range $[a', \alpha(j)]$ in $\mathcal{T}_{\mathsf{ran}}$ with $a' < \alpha(j)$. Thus, we remove $[a', \alpha(j)]$ from $\mathcal{T}_{\mathsf{ran}}$ and insert $[a', \beta(j)]$ into $\mathcal{T}_{\mathsf{ran}}$. Then we update $M(i) := \min_{k \in [a', \alpha(j)]}\{M(k)+2\}$ for each $i \in [\alpha(j)+1, \beta(j)]$.

If E_j is a detach event, the active baselines H_i for i from $\alpha(j)$ to $\beta(j)-1$ become inactive. Observe that there always exists a range $[\alpha(j), b']$ in $\mathcal{T}_{\mathsf{ran}}$ with $\beta(j) < b'$. Thus, we remove $[\alpha(j), b']$ from $\mathcal{T}_{\mathsf{ran}}$, and insert $[\beta(j), b']$ into $\mathcal{T}_{\mathsf{ran}}$. Then we update $M(i) := \min\{M(i), \min_{k \in [\alpha(j), \beta(j)-1]}(M(k)+2)\}$ for each $i \in [\beta(j), b']$.

If E_j is a split event, the active baselines lying in between $H_{\alpha(j)}$ and $H_{\beta(j)}$ become inactive. If there is such a baseline, there always exists a range $[a', b']$ in $\mathcal{T}_{\mathsf{ran}}$ with $a' < \alpha(j)$ and $\beta(j) < b'$. In this case, we remove $[a', b']$ from $\mathcal{T}_{\mathsf{ran}}$, insert $[a', \alpha(j)]$ and $[\beta(j), b']$ into $\mathcal{T}_{\mathsf{ran}}$, and update for each $i \in [\beta(j), b']$

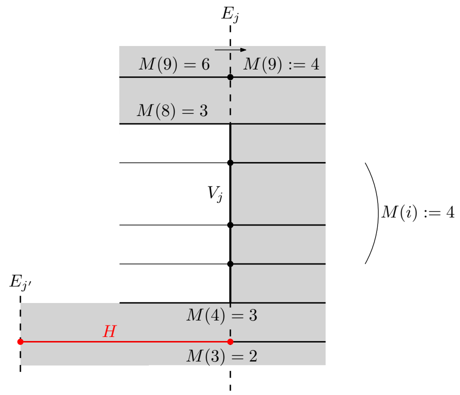


Figure 5: A merge event E_j. $M(3) = \min_{k \in [a', \alpha(j)]} M(k)$, which was updated from $E_{j'}$. The baselines H_i for $i = 5, \ldots, 7$ become active, and $M(i)$ is updated to 4 by $M(3)$ (Equation 1). $M(9)$ is also updated by $M(3)$ (Equation 1). H is the canonical segment for E_j.

$M(i) := \min\{M(i), \min_{k \in [a', \beta(j)-1]}\{M(k)+2\}\}$ for each $i \in [\beta(j), b']$.

If E_j is a merge event, the inactive baselines lying in between $H_{\alpha(j)}$ and $H_{\beta(j)}$ become active. If there is such a baseline, there always exist two ranges $[a', \alpha(j)]$ and $[\beta(j), b']$ in $\mathcal{T}_{\mathsf{ran}}$ with $a' < \alpha(j)$ and $\beta(j) < b'$. In this case, we remove $[a', \alpha(j)]$ and $[\beta(j), b']$ from $\mathcal{T}_{\mathsf{ran}}$, insert $[a', b']$ into $\mathcal{T}_{\mathsf{ran}}$, and update

$$M(i) := \begin{cases} \min_{k \in [a', \alpha(j)]} M(k) + 2 \\ \qquad\qquad \text{for } i \in [\alpha(j)+1, \beta(j)-1], \\ \min\{M(i), \min_{k \in [a', \alpha(j)]} M(k) + 2\} \\ \qquad\qquad \text{for } i \in [\beta(j), b']. \end{cases} \tag{1}$$

Our algorithm eventually finds $\lambda(s,t)$ when L encounters the terminate event E_z with $\beta(z) = m$. Then $\mathcal{T}_{\mathsf{ran}}$ has exactly one range $[\alpha(z), \beta(z)]$, and we remove it from $\mathcal{T}_{\mathsf{ran}}$. We take $\lambda(s,t) = \min\{M(m), \min_{k \in [\alpha(z), \beta(z)-1]} M(k) + 1\}$.

3.2 Computing a minimum-link shortest path

We compute a minimum-link shortest path from s to t aligned to the baselines of $\mathsf{D}_{xy}(s,t)$ using $\lambda(s,t)$. To do this, we add a horizontal line segment at each event, which we call a *canonical segment*. Then we report a minimum-link shortest path using these canonical segments.

For instance, consider a merge event E_j. Recall that $\mathcal{T}_{\mathsf{ran}}$ has two disjoint ranges $[a', \alpha(j)]$ and $[\beta(j), b']$ with $a' < \alpha(j)$ and $\beta(j) < b'$. We update $M(i)$ using Equation 1. Let $k^* \in [a', \alpha(j)]$ be the smallest index such that $M(k^*)$ equals $\min_{k \in [a', \alpha(j)]} M(k)$. Assume that $M(k^*)$ was updated lately to the current value at an event $E_{j'}$ before L encounters E_j. Obviously, $x(V_{j'}) < x(V_j)$. We add a horizontal line segment H, which we call a canonical segment for E_j with $x_1(H) = x(V_{j'})$, $x_2(H) = x(V_j)$ and $y(H) = y(H_{k^*})$. See Figure 5.

We add one canonical segment for the merge event E_j. Likewise, we add one canonical segment per event of other types, except for the originate event. Since the x-coordinates of the events are distinct by the general position assumption, the right endpoints of the canonical segments we add are also distinct. Once the plane sweep algorithm is done, by following lemma, we can report a shortest path that has $\lambda(s,t)$ links.

Lemma 4 *There is a minimum-link shortest path from s to t whose horizontal line segments are all canonical segments.*

$\mathsf{D}_{xy}(s,t)$ can be obtained by ray shooting queries, each taking $O(\log n)$ time, using the data structure of Giora and Kaplan [7] with $O(n \log n)$ preprocessing time. Let h be the number of holes in $\mathsf{D}_{xy}(s,t)$, and o be the complexity of the outer boundary of $\mathsf{D}_{xy}(s,t)$. We can construct $\mathsf{D}_{xy}(s,t)$ in $O((o+h)\log n)$ time using $O(o+h)$ space.

Let z denote the number of events occurring during the sweep. At each of the z events, we remove and insert some ranges. Because the ranges in $\mathcal{T}_{\mathsf{ran}}$ are disjoint by the definition of $\mathcal{T}_{\mathsf{ran}}$, we can insert and remove a range in $O(\log m)$ time by using a simple balanced binary search tree for $\mathcal{T}_{\mathsf{ran}}$. We also set or update some $M(i)$'s at each event. For each event, if we know $\min_{k \in [a_1, b_1]} M(k)$ for a range $[a_1, b_1]$, we can update $M(i)$ for $i \in [a_2, b_2]$ (with $b_1 < a_2$) in time linear to the number of consecutive baselines from H_{a_2} to H_{b_2}, and the number of $M(i)$'s is $O(m)$. Therefore, it takes $O(m)$ time to handle an event. We use $O(m)$ space to maintain $\mathcal{T}_{\mathsf{ran}}$ and $M(i)$'s. In total, we use $O(o+h+m)$ space to compute $\lambda(s,t)$. We can report a minimum-link shortest path using $O(z)$ canonical segments. Thus, our algorithm takes $O((n+o+h)\log n + mz) = O(n^2)$ time and $O(o+h+m+z) = O(n)$ space.

Reducing the time complexity. To reduce the time complexity of our algorithm to $O(\log m)$ for handling each event while keeping the space complexity to $O(n)$ space, we build another balanced binary search tree $\mathcal{T}_{\mathsf{seg}}$, a variant of a segment tree in [5]. The idea is to use $\mathcal{T}_{\mathsf{seg}}$ together with $\mathcal{T}_{\mathsf{ran}}$ to maintain $O(\log m)$ nodes corresponding $O(m)$ $M(i)$'s efficiently, instead of updating $M(i)$'s for each event immediately. For each event, we have the range of indices of baselines inserted into (or removed from) $\mathcal{T}_{\mathsf{ran}}$. Using the range, we find $O(\log m)$ nodes in $O(\log m)$ time and update information for each node in constant time. The details can be found in Appendix.

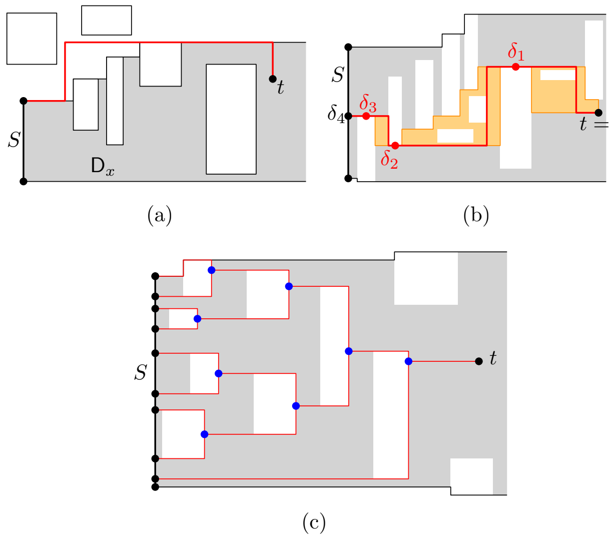


Figure 6: (a) A minimum-link shortest path from S to t. It is not contained in D_x. (b) Every shortest path from S to t passes through δ_i for $i = 0, \ldots, 4$ and is contained in $\bigcup_{i=0,\ldots,3} \mathsf{D}_{xy}(\delta_i, \delta_{i+1})$. (c) Shortest paths from S to t such that the closest pair of S and t is not unique.

Lemma 5 *For a point t in D_{xy}, we can compute a minimum-link shortest path from S to t in $O(n \log n)$ time using $O(n)$ space.*

4 t lies in D_x or D_y

In this section, we assume that t lies in D_x. Then every shortest path from S to t is x-monotone by Lemma 2. In case that t lies in D_y, every shortest path is y-monotone and we can handle the case in a similar way. Unlike the case of $t \in \mathsf{D}_{xy}$, there can be a shortest path from S to t not contained in D_x. See Figure 6(a). However, we can compute a minimum-link shortest path from S to t using the algorithm in Section 3 as a subprocedure.

Let π denote a minimum-link shortest path from S to t aligned to the baselines, and let s^* and t be the two endpoints of π with $s^* \in S$. By definition, (s^*, t) is a closest pair of S and t. Since π is x-monotone, it is a concatenation of xy-monotone paths such that every two consecutive xy-monotone subpaths of π change their directions between monotone increasing and monotone decreasing on a horizontal segment, which we call a *winder*, of π.

Lemma 6 *Every winder of a shortest path from S to t contains one entire horizontal side of a rectangle in* R *incident to D_x.*

Consider the horizontal sides of rectangles contained in the winders of a minimum-link shortest path π. Let g be the number of winders of π, and let δ_i be the midpoint of the horizontal side contained in the ith winder in order along π from t to s^*. We call such a midpoint a *divider* of π. For convenience, we let $t = \delta_0$ and $s^* = \delta_{g+1}$. Then the subpath from δ_i to δ_{i+1} for $0 \le i \le g$ of π is xy-monotone by the definition of the winders of π.

By Lemma 6, every winder of a minimum-link shortest path from S to t contains a horizontal side of a rectangle incident to D_x. Therefore, in the following we compute the dividers of a minimum-link shortest path among the midpoints of the top and bottom sides of each rectangle incident to D_x, and compute the xy-monotone paths connecting the dividers, in order, which together form a minimum-link shortest path.

We compute $d(S, t)$ by a plane sweep algorithm and find the dividers $\delta_1, \ldots, \delta_g$ of a minimum-link shortest path π as follows. For a rectangle $R \in \mathsf{R}$ incident to D_x, let $\delta(R)$ and $\delta'(R)$ denote the midpoints of the top and bottom sides of R, respectively. Then $\delta(R)$ and $\delta'(R)$ are candidates of the dividers of π. We consider each midpoint as an event during the sweep.

While sweeping D_x with a vertical line L moving rightwards, L encounters $\delta(R)$ and $\delta'(R)$ of a rectangle R at the same time. Consider two horizontal rays, one from $\delta(R)$ and one from $\delta'(R)$, going leftwards. We show how to handle the ray γ from $\delta(R)$. The ray from $\delta'(R)$ can be handled similarly. Let p_γ be the point of the vertical segment on the boundary of D_x which γ hits first. If $p_\gamma \in S$, the shortest path from S to $\delta(R)$ is simply $p_\gamma\delta(R)$. If p_γ lies in $\pi_{\mathsf{ru}}(s)$ (or $\pi_{\mathsf{rd}}(s')$), every shortest path from S to $\delta(R)$ is xy-monotone. For these two cases, we store at $\delta(R)$ the distance $d(S, \delta(R))$ and the closest point (one of p_γ, s, or s') of S from $\delta(R)$. Consider the case that p_γ is in the right side of a rectangle $R' \in \mathsf{R}$ incident to D_x. We already have $d(S, \delta(R'))$ stored at $\delta(R')$ and $d(S, \delta'(R'))$ stored at $\delta'(R')$ during the plane sweep. Observe that every shortest path from $\delta(R)$ to $\delta(R')$ or to $\delta'(R')$ is xy-monotone, and the closest point s^o of S from $\delta(R)$ is the closest point of S from $\delta(R')$ or from $\delta'(R')$. Since $d(s^o, \delta(R)) = \min\{d(S, \delta(R')) + d(\delta(R'), \delta(R)), d(S, \delta'(R')) + d(\delta'(R'), \delta(R))\}$ by definition, we can compute s^o and $d(s^o, \delta(R))$ in constant time.

When L encounters t, we again consider a horizontal ray γ from t going leftwards and the point p_γ of the vertical segment on the boundary of D_x which γ hits first. If $p_\gamma \in S$, the minimum-link shortest path is simply $p_\gamma t$ and we are done. For p_γ lying on a side of a rectangle $R \in \mathsf{R}$ incident to D_x, if $d(t, \delta(R)) + d(S, \delta(R)) < d(t, \delta'(R)) + d(S, \delta'(R))$ (or the other way around without equality), we conclude there is no shortest path from S to t passing through $\delta'(R)$ (or through $\delta(R)$). Assume that π passes through $\delta(R)$. By the general position assumption, $y(\delta(R)) > y(t)$. Let R' be the rectangle incident to D_x that the horizontal ray from $\delta(R)$ going leftwards hits first. If $d(S, \delta(R')) + d(\delta(R'), \delta(R)) > d(S, \delta'(R')) + d(\delta'(R'), \delta(R))$ or there is no such rectangle R', $\delta(R)$ is a divider of π. Moreover, $\delta(R)$ is the first divider δ_1 of π from t, and thus every shortest path from $\delta_1 = \delta(R)$ to t is xy-monotone. Therefore, we construct $\mathsf{D}_{xy}(\delta(R), t)$ and apply the algorithm in Section 3. Then we apply this procedure from $\delta(R)$, recursively, and compute every xy-monotone subpath of π using canonical segments by Lemma 4, and glue them into one to form π. See Figure 6(b). Finally we obtain $g+1$ xy-monotone paths with dividers $\delta_0 = t, \delta_1, \ldots, \delta_{g+1} = s^* \in S$.

During the plane sweep, we find in $O(\log n)$ time the first rectangle hit by the horizontal ray γ emanating from a midpoint of a rectangle going leftwards using the data structure supporting ray shooting queries by Giora and Kaplan [7]. Thus, it takes $O(n \log n)$ time for ray shootings from midpoints in total. It takes $O(K_i)$ time to find a divider δ_i, where K_i is the number of the recursion depth of the algorithm to compute δ_i from δ_{i-1}. As shown in Section 3, computing an xy-monotone minimum-link shortest path from δ_i to δ_{i-1} takes $O(D_i \log D_i)$ time with $O(D_i)$ space after $O(n \log n)$-time preprocessing, where D_i is the number of the baselines defined by the rectangles incident to $\mathsf{D}_{xy}(\delta_i, \delta_{i-1})$. Observe that $\Sigma_{1 \le i \le g+1} K_i = O(n)$, and $\Sigma_{1 \le i \le g+1} D_i = O(n)$ because the regions $\mathsf{D}_{xy}(\delta_i, \delta_{i-1})$'s are disjoint in their interiors. Thus, the total time complexity is $O(n \log n)$ and the total space complexity is $O(n)$.

Handling degenerate cases. There can be two shortest paths from S to t, one passing through $\delta(R)$ and one passing through $\delta'(R)$ for a rectangle R. In this case, we have $d(t, \delta(R)) + d(S, \delta(R)) = d(t, \delta'(R)) + d(S, \delta'(R))$, which can be found in handling the midpoints of R during the plane sweep. Observe that this equality may occur multiple times in finding dividers of a minimum-link shortest path. Thus we need to devise an efficient way of maintaining all sequences of dividers, each of which may define a shortest path. See Figure 6(c). we show how to maintain these sequences of dividers and to find a minimum-link shortest path without increasing the time and space complexities in Lemma 7. The details can be found in Appendix.

Lemma 7 *For a point t in $\mathsf{D}_x \cup \mathsf{D}_y$, we can compute a minimum-link shortest path from S to t in $O(n \log n)$ time using $O(n)$ space.*

5 Extending to a line segment T

Consider the case that the target is not just a point but an axis-aligned line segment T. We explain how the algorithm presented in previous sections works for T. Assume that T is a vertical line segment and $x(S) < x(T)$. We partition the domain D into eight regions using the eight monotone paths π_α's from S defined in Section 2.1. Then T intersects at most five regions D_x^1, D_{xy}^1, D_{xy}^4, D_y^1, and D_y^2. For the portion T' of T contained in each region, we compute a minimum-link shortest path from S to T'.

For the portion of T contained in a region of $\mathsf{D}_{xy}^1, \mathsf{D}_{xy}^4, \mathsf{D}_y^1$ and D_y^2, the closest point of S from T' is an endpoint of S and the closest point in T' from S is an endpoint of T' by Lemma 2. Thus we just apply the algorithms in Sections 3 and 4 for the corresponding endpoints of S and T'.

Consider the case that $T' \subset \mathsf{D}_x^1$. A minimum-link shortest path from S to T' connects S and an endpoint of T' or the intersection point t' of T' with a horizontal baseline of D_x. We can compute the distance from S to two endpoints of T' using the algorithm in Section 4. There are $O(n)$ intersection points on T' with horizontal baselines of D_x. During the plane sweep, we have $d(S, \delta(R))$ and $d(S, \delta'(R))$ for each hole R of D_x such that the horizontal baselines defined by R intersects T'. Thus, we can compute the distance from S to each intersection point t' on T' after the plane sweep. Then we obtain all the closest pairs of S and T'.

If there is only one closest pair, or the closest point of T' from S is the same for all closest pairs, we can compute a minimum-link shortest path from S to T' as we do in Section 4. Otherwise, we can compute a minimum-link shortest path from S to T' using technical lemmas in Appendix.

We can compute the portions T' of T contained in each of the five regions in $O(\log n)$ time using binary search along each path π_α and computing an intersection of T and π_α. For $T' \subset \mathsf{D}_x^1$, we can find the closest pairs in $O(n \log n)$ time if we use the ray shooting structure of Giora and Kaplan [7]. For each T' we use our algorithm in Sections 3 and 4 with $O(n \log n)$ time and $O(n)$ space, and eventually find a minimum-link shortest path from S to T by choosing $\min \lambda(S, T')$ for all T'.

Lemma 8 *Given two axis-aligned line segments S and T in a rectangular domain with n disjoint rectangular obstacles in the plane, we can compute a minimum-link shortest path from S to T in $O(n \log n)$ time using $O(n)$ space.*

6 Extending to box-disjoint rectilinear polygons

We show how to extend our algorithm in previous sections so that it handles box-disjoint rectilinear polygons. Let R_P be a set of box-disjoint rectilinear polygons, and let $B(P)$ denote the bounding box of a polygon $P \in \mathsf{R}_P$. We use $\mathsf{C} := \mathbb{R}^2 - \cup_{P \in \mathsf{R}_P} P$ to denote a *box-disjoint rectilinear domain* induced by R_P in the plane. A set Q is *rectilinear convex* if and only if any line parallel to the x- or y-axis intersects Q in at most one connected component. The rectilinear convex hull of P, denoted by $\mathsf{CH}(P)$, is the common intersection of all rectilinear convex sets containing P.

We assume that both S and T are disjoint from the rectangles $B(P)$ for $P \in \mathsf{R}_P$. Then no shortest path intersects the interior of $\mathsf{CH}(P)$ for $P \in \mathsf{R}_P$. If there is a shortest path π intersecting the interior of $\mathsf{CH}(P)$ for a rectilinear polygon $P \in \mathsf{R}_P$, π can be shortened by replacing each connected portion of π contained in the interior with the boundary curve of $\mathsf{CH}(P)$ between the endpoints of the portion, a contradiction. Thus, we replace each polygon P with $\mathsf{CH}(P)$ and find a minimum-link shortest path from S to T avoiding $\mathsf{CH}(P)$'s. We assume that each polygon $P \in \mathsf{R}_P$ is rectilinear convex in this subsection. If there is a shortest path π from S to T intersecting $B(P)$ for $P \in \mathsf{R}_P$, the subpath $\pi \cap B(P)$ can be replaced with a subpath along the boundary of $B(P)$ without increasing the length. This implies that there is a shortest path from S to T avoiding $B(P)$ for all $P \in \mathsf{R}_P$. From Lemma 2, every shortest path from S to T avoiding $B(P)$ for all $P \in \mathsf{R}_P$ is either x-, y-, or xy-monotone. The two subpaths have same length and endpoints, so they have the same monotonicity: One is X-monotone if and only if the other is X-monotone, for $X \in \{x, y, xy\}$. Therefore, every shortest path from S to T contained in C is either x-, y-, or xy-monotone.

Here we partition the domain into eight disjoint regions using eight xy-monotone paths as follows. We define the eight xy-monotone paths from S in a way slightly different to the one in Section 2.1. Consider the horizontal ray emanating from $s = p_1$ going rightwards. Let $P \in \mathsf{R}_P$ be the polygon such that $B(P)$ is the first rectangle hit by the ray among the rectangles, at point b on its left side. If the upper endpoint q of the leftmost vertical side of P lies above b, we set p'_1 to b and continue with the vertical ray from p'_1 to q, and continue along the boundary chain of P from q to the left endpoint p_2 of the topmost side of P in clockwise direction. Otherwise, the horizontal ray continues going rightwards until it hits P at a point b'. Then we set p'_1 to b' and continue along the boundary chain of P from p'_1 to the left endpoint p_2 of the topmost side of P in clockwise direction. We repeat this process by taking the horizontal ray from p_2 going rightwards. Then we obtain an xy-monotone path $\pi_{\mathsf{ru}}(p) = (p = p_1, p'_1, p_2, p'_2, \ldots)$, by following the boundary chain of P from p'_i to p_{i+1} in clockwise direction. Thus, $\pi_{\mathsf{ru}}(p)$ is an xy-monotone path from p that alternates going horizontally rightwards and going vertically upwards. We define eight xy-monotone paths $\pi_\alpha(p)$ as in Section 2.1. Using these eight xy-monotone paths, we construct at most eight disjoint regions.

Using those regions, we compute a minimum-link shortest path from S to the portion of T contained in each region. Let T' be the portion of T contained in D_{xy}. The closest pair (s,t) of S and T' consists of their endpoints. We compute $\mathsf{D}_{xy}(s,t)$ using the method in Section 3. Observe that every shortest path from S to T' is contained in $\mathsf{D}_{xy}(s,t)$. With $O(n)$ baselines defined by the sides of $B(P)$ and the boundary segments of P incident to D_{xy} for all $P \in \mathsf{R}_P$, we can show that there is a minimum-link shortest path from S to T' which is aligned to the baselines using an argument similar to the proof of Lemma 3. Hence, we can compute a minimum-link shortest path from S to T' in the same time and space as in Lemma 5. Similarly, we can compute a minimum-link shortest path from S to T' for the portions T' of T contained in other regions. When T' is contained in D_x, a minimum-link shortest path may have some winders, each of which contains the topmost or the bottommost side of P for a rectilinear polygon $P \in \mathsf{R}_P$. This can be shown by an argument similar to the proof of Lemma 6. Thus we can compute $d(S, T')$ using the same plane sweep algorithm on D_x, and find the dividers which are midpoints of the topmost or the bottommost side of P as we do in Section 4 in the same time and space stated in Lemma 7.

S or T intersects bounding boxes. When S or T intersects some bounding boxes of obstacles, we consider each portion of S or T contained in a bounding box independently. The portion not contained in any bounding box can be handled as we do for segments disjoint from the boxes. For the portion contained in a bounding box $B(P)$ for a rectilinear polygon P, every minimum-link shortest path from S to T is the concatenation of a subpath contained in $B(P)$ and the subpath not contained in $B(P)$ such that both subpaths are minimum-link shortest paths sharing one point on the boundary of $B(P)$. Using that property, we can compute a minimum-link shortest path from S to T. The overall running time remains to be $O(n \log n)$ time using $O(n)$ space. See Appendix for details.

Lemma 9 *For two axis-aligned line segments S and T in C such that both S and T are disjoint from $B(P)$ for all $P \in \mathsf{R}_P$, we can compute a minimum-link shortest path from S to T in C in $O(n \log n)$ time using $O(n)$ space.*

7 Extending to two polygons S and T

Now we consider two rectilinear polygons S and T with N vertices in C. We can compute a minimum-link shortest path from S to T using our algorithms in previous sections. Since S, T, and obstacles are pairwise box-disjoint, the distance $d(\mathsf{S},\mathsf{T})$ between S and T can be represented as $d(\mathsf{S},\mathsf{T}) = \min_{s \in B(\mathsf{S}), t \in B(\mathsf{T})}\{d(s,t) + \min_{s' \in \mathsf{S}} d(s,s') + \min_{t' \in \mathsf{T}} d(t,t')\}$. If we construct the L_1 Voronoi diagram of N boundary segments of S (or T) [14] in $O(N \log N)$ time using $O(N)$ space, we can

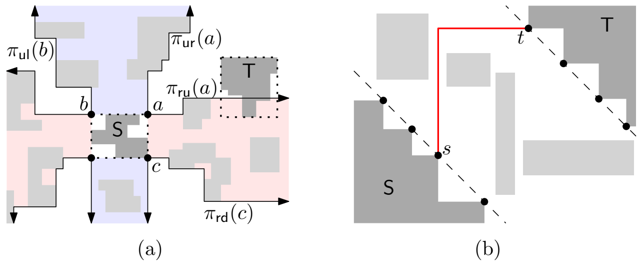


Figure 7: (a) Eight regions of C by eight xy-monotone paths from four corners of $B(\mathsf{S})$ with box-disjoint obstacles. T intersects at most five regions. (b) There are nine closest pairs of S and T. Among all paths connecting closest pairs, the minimum-link shortest path from s to t is the optimal.

maintain and report $\min_{s' \in \mathsf{S}} d(s, s')$ and $\min_{t' \in \mathsf{T}} d(t, t')$ for any $s \in B(\mathsf{S})$ and $t \in B(\mathsf{T})$ in $O(\log N)$ query time. From this observation, together with Lemma 2, we have the following lemma.

Lemma 10 *If there is an x-monotone shortest path from S to T, then every shortest path from S to T is x- or xy-monotone. If there is a y-monotone shortest path from S to T, then every shortest path from S to T is y- or xy-monotone.*

From Lemma 10, we can partition the box-disjoint rectilinear domain into eight disjoint regions using eight xy-monotone paths from $B(\mathsf{S})$ as done in Section 6. See Figure 7(a). There are $O(N)$ vertical and horizontal baselines defined by the boundary segments of S and T, and Lemma 3 also holds. Thus, we compute a minimum-link shortest path aligned to the baselines of each region in which the portion of T is contained.

Let T' be the portion of T contained in D_{xy}. Since S and T' are rectilinear polygons, there can be more than one closest pair of points for S and T'. Moreover, the points appearing in the closest pairs are on line segments with slopes ± 1. See Figure 7(b). If we compute $\mathsf{D}_{xy}(s,t)$ for every closest pair (s,t) of S and T', the time and space complexities may increase. Instead, we modify the plane sweep algorithm in Section 3 slightly. There can be more than one originate and terminate events during the plane sweep because there can be more than one closest pair of S and T'. Also, there are no attach and detach events since we do not compute $\mathsf{D}_{xy}(s,t)$. For each event E_j in Section 3, however, we use $\alpha(j)$, $\beta(j)$ and $\mathcal{T}_{\mathsf{ran}}$ to maintain active baselines. Since we have all points in closest pairs of S and T, we set two horizontal baselines with the smallest and largest y-coordinate from those points, respectively. The horizontal baselines between the two baselines are used for inserting the range, which represents active baselines, into $\mathcal{T}_{\mathsf{ran}}$ of each event. Since there are $O(N+n)$ baselines between the two baselines, the time to handle an event takes $O(\log(N+n))$ time. Also, there are additional $O(N)$ originate and terminate events with $O(n)$ the other events, so we can compute a minimum-link shortest path from S to T' in $O((N+n)\log(N+n))$ time using $O(N+n)$ space.

Let T' be the portion of T contained in D_x. Every shortest path from S to T' is x-monotone, so we can compute the closest pairs of S and T' as the sweep line encounters each vertical line segments of T' using the plane sweep algorithm in Section 4. Then we can compute dividers in the same way without modifying the algorithm in Section 4. Lemmas related to dividers in Section 4 still hold, so we can compute a minimum-link shortest path connecting dividers similarly. As above, we can compute a minimum-link shortest path from S to a divider (or from a divider to T'). This implies we obtain a minimum-link shortest path from S to T'. We omit the details.

Therefore, we have Theorem 1.

8 Conclusion

We present an algorithm to compute a minimum-link shortest path connecting two rectilinear polygons in the box-disjoint rectilinear domain efficiently. Our algorithm computes a minimum-link shortest path from a point to the line segment using plane sweep, based on the monotonicity of the optimal path. Then we can extend objects to rectilinear polygons and apply a slightly modified algorithm.

Still there are quite a few problems to study. One typical problem is to compute a minimum-link shortest path connecting two objects in a general rectilinear domain such that the obstacles in the domain are not necessarily box-disjoint. There is a previous work in a general rectilinear domain, but there seem some gaps to the optimal time and space complexities.

References

[1] D.Z. Chen, O. Daescu, and K.S. Klenk. On geometric path query problems. *International Journal of Computational Geometry & Applications*, 11(6):617–645, 2001.

[2] D.Z. Chen and H. Wang. L_1 shortest path queries among polygonal obstacles in the plane. In *30th International Symposium on Theoretical Aspects of Computer Science*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[3] J. Choi and C. Yap. Monotonicity of rectilinear geodesics in d-space. In *Proceedings of the Annual Symposium on Computational Geometry*, pages 339–348, 1996.

[4] G. Das and G. Narasimhan. Geometric searching and link distance. In *Workshop on Algorithms and Data Structures*, pages 261–272. Springer, 1991.

[5] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd edition, 2008.

[6] P.J. De Rezende, D.-T. Lee, and Y.-F. Wu. Rectilinear shortest paths in the presence of rectangular barriers. *Discrete & Computational Geometry*, 4:41–53, 1989.

[7] Y. Giora and H. Kaplan. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. *ACM Transactions on Algorithms*, 5(3):28:1–51, 2009.

[8] H. Imai and T. Asano. Efficient algorithms for geometric graph search problems. *SIAM Journal on Computing*, 15(2):478–494, 1986.

[9] D.-T. Lee, C.-D. Yang, and C.K. Wong. Rectilinear paths among rectilinear obstacles. *Discrete Applied Mathematics*, 70(3):185–215, 1996.

[10] J.S.B. Mitchell. An optimal algorithm for shortest rectilinear paths among obstacles in the plane. In *Abstracts of the 1st Canadian Conference on Computational Geometry*, volume 22, 1989.

[11] J.S.B. Mitchell. L_1 shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8(1–6):55–88, 1992.

[12] J.S.B. Mitchell, V. Polishchuk, and M. Sysikaski. Minimum-link paths revisited. *Computational Geometry*, 47(6):651–667, 2014.

[13] J.S.B. Mitchell, V. Polishchuk, M. Sysikaski, and H. Wang. An optimal algorithm for minimum-link rectilinear paths in triangulated rectilinear domains. *Algorithmica*, 81(1):289–316, 2019.

[14] E. Papadopoulou and D.T. Lee. The L_∞ Voronoi diagram of segments and VLSI applications. *International Journal of Computational Geometry & Applications*, 11(05):503–528, 2001.

[15] M. Sato, J. Sakanaka, and T. Ohtsuki. A fast line-search method based on a tile plane. In *IEEE International Symposium on Circuits and Systems*, volume 5, pages 588–591, 1987.

[16] S. Schuierer. An optimal data structure for shortest rectilinear path queries in a simple rectilinear polygon. *International Journal of Computational Geometry & Applications*, 6(02):205–225, 1996.

[17] C.D. Toth, J. O'Rourke, and J.E. Goodman. *Handbook of discrete and computational geometry*. CRC press, 3rd edition, 2017.

[18] H. Wang. Bicriteria rectilinear shortest paths among rectilinear obstacles in the plane. *Discrete & Computational Geometry*, 62:525–582, 2019.

[19] C.-D. Yang, D.-T. Lee, and C.K. Wong. On bends and lengths of rectilinear paths: a graph-theoretic approach. *International Journal of Computational Geometry & Applications*, 2(01):61–74, 1992.

[20] C.-D. Yang, D.-T. Lee, and C.K. Wong. On minimum-bend shortest recilinear path among weighted rectangles. In *Tech. Report 92-AC-122*. Dept. of EECS, Northwestern Univ, 1992.

[21] C.-D. Yang, D.-T. Lee, and C.K. Wong. Rectilinear path problems among rectilinear obstacles revisited. *SIAM Journal on Computing*, 24(3):457–472, 1995.

Appendix

9 Proof of Lemma 2

Proof. We claim that every shortest path from S to t connects the upper endpoint s of S and t for a point $t \in \mathsf{D}^1_y \cup \mathsf{D}^1_{xy} \cup \mathsf{D}^2_{xy}$. Assume to the contrary that a shortest path π from S to t does not pass through s. Then π crosses $\pi_{\mathsf{ru}}(s)$ (or $\pi_{\mathsf{lu}}(s)$) at a point t'. By replacing the portion of π from S to t' with the portion of $\pi_{\mathsf{ru}}(s)$ from s to t', we can get a shorter path, a contradiction. By a similar argument, we observe that every shortest path from S to t connects the lower endpoint s' of S and t if $t \in \mathsf{D}^2_y \cup \mathsf{D}^3_{xy} \cup \mathsf{D}^4_{xy}$.

Rezende et al. [6] showed that every shortest path connecting two points in D is x-, y-, or xy-monotone. Choi and Yap [3] gave a classification that for a point $t \in \cup_{1\le i\le 2}\mathsf{D}^i_y$ every shortest path from S to t is y-monotone, and for a point $t \in \cup_{1\le i\le 4}\mathsf{D}^i_{xy}$ every shortest path from S to t is xy-monotone. Assume that $t \in \mathsf{D}^1_x$. Both $\pi_{\mathsf{ul}}(t)$ and $\pi_{\mathsf{ur}}(t)$ intersect $\pi_{\mathsf{ru}}(s)$, and both $\pi_{\mathsf{dl}}(t)$ and $\pi_{\mathsf{dr}}(t)$ intersect $\pi_{\mathsf{rd}}(s')$. This implies that every shortest path from a point in S to t is x- or xy-monotone by the classification of Choi and Yap [3]. Hence every shortest path from S to t is x-monotone. The case for $p \in \mathsf{D}^2_x$ can be shown similarly. □

10 Proof of Lemma 3

Proof. Assume that a minimum-link shortest path π has a horizontal line segment H which is not contained in any baseline of D'. Clearly, H is not incident to t, because there is a horizontal baseline through t. If H is incident to S not at its endpoints, we can move H vertically and get a shorter path, a contradiction. If both vertical segments of π incident to H are contained in one side of the line through H, then we can get a path shorter than π by moving H towards the side and shortening the two vertical segments incident to H, a contradiction. This also applies to a vertical segment of π not contained in any baseline.

Now assume that π is xy-monotone and H is incident to neither S nor t. Let π' be a maximal subpath of π such that π' contains H, and no horizontal baseline of D' intersects π' except at its two endpoints p_1 and p_2 with $y(p_1) < y(p_2)$. See Figure 8(a). We show that the axis-aligned rectangle R with corners at p_1 and p_2, is contained in D'. Assume to the contrary that R is not contained in D', that is, there is a rectangle $R' \in \mathsf{R}$ incident to D' that intersects R. Then there is a horizontal baseline of D' through a side of R' that intersects π'. This contradicts the definition of π', so R is contained in D'. See Figure 8(a).

Thus, we can replace the subpath π' with a horizontal side and a vertical side of R without increasing the length of π. See Figure 8(b). The resulting path has the number of links smaller than or equal to that of π. By applying the procedure above for every horizontal line segment not contained in a horizontal baseline of D', we can get a minimum-link shortest path π^* from s^* to t such that every horizontal line segment of π^* is contained in a horizontal baseline of D'.

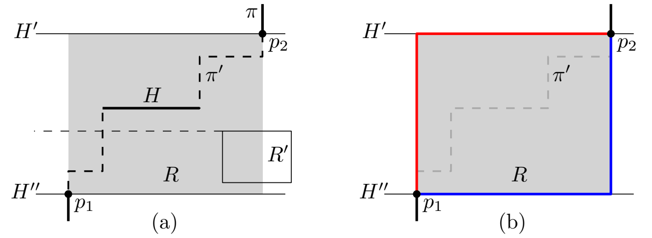


Figure 8: Proof of Lemma 3. (a) When π' intersects two baselines H' and H'' at its two endpoints p_1, p_2, no rectangle intersects the rectangle R (gray) with corners at p_1, p_2. (b) By replacing π' with the subpath (red or blue) along the boundary of R, we obtain a path with a smaller or the same number of links.

Similarly, we can replace every vertical segment of π' not contained in a vertical baseline with one contained in a baseline without increasing the length of the path. □

11 Proof of Lemma 4

Proof. At the terminate event E_z, we have $\lambda(s,t) := \min\{M(m), M(k^*)+1\}$, where k^* is the index satisfying $M(k^*) = \min_{k\in[\alpha(z),\beta(z)-1]} M(k)$. If $M(m) \le M(k^*)$, there is a canonical segment H incident to t, so H becomes the horizontal segment of π that is incident to t. Otherwise, there is a canonical segment H incident to V_z with $y(H) = y(H_{k^*})$, and thus H and V'_z form a subpath of π, where V'_z is the portion of V_z with $y_1(V'_z) = y(H_{k^*})$ and $y_2(V'_z) = y(t)$. For both cases, we can find the left endpoint of H such that $x_1(H) = x(V_j)$. We know there exists a canonical segment H' for the event E_j, so we do the above process for H' and the V'_j to form a subpath of π recursively, where V'_j is the vertical line segment connecting the left endpoint of H and the right endpoint of H'. See Figure 9. At the originate event E_1, we have a canonical segment H^0 with $x(H^0) = x(s)$. Then the vertical line segment connecting s and the left endpoint of H^0 forms a subpath of π. Gluing all subpaths formed from above recursive process, we finally obtain π whose horizontal line segments are all canonical segments. □

12 Proof of Lemma 6

Proof. Let π be a shortest path from S to t. Assume that π has a winder H, which does not contain a horizontal side of a rectangle. Since H is a winder, the two

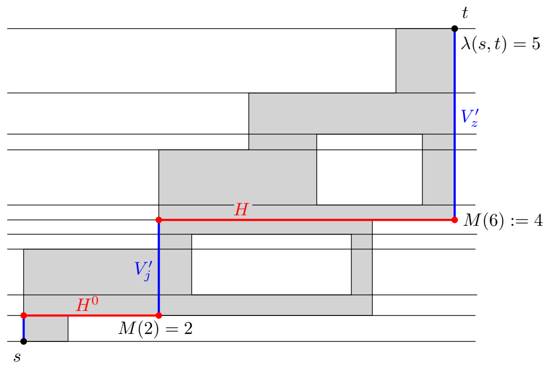


Figure 9: A minimum-link shortest path from s to t consisting of three vertical segments (blue) and two horizontal segments (red). The red horizontal segments are canonical segments in $\mathsf{D}_{xy}(s,t)$. $\lambda(s,t)$ is computed by $M(6)$ in the terminate event. $M(6)$ was updated in E_j so there is a canonical segment H for E_z. H and V'_z form a subpath of π.

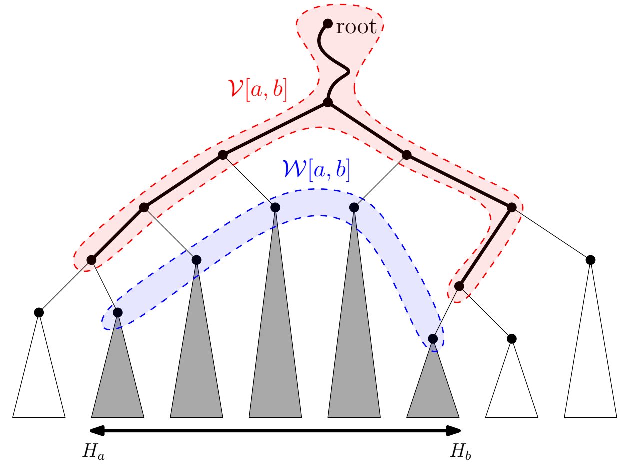


Figure 10: A balanced binary search tree $\mathcal{T}_{\mathsf{seg}}$. A range $[a,b]$ can be represented by the $O(\log m)$ nodes in $\mathcal{W}[a,b]$. For each node $v \in \mathcal{V}[a,b]$, $[\alpha(v),\beta(v)] \cap [a,b] \neq \emptyset$ and $[\alpha(v),\beta(v)] \not\subseteq [a,b]$.

consecutive xy-monotone subpaths of π sharing H lie in one side of the line containing H. Without loss of generality, assume that both subpaths lie above the line containing H. Then we can drag H upward while shortening the vertical segments of π incident to H, which results in a shorter path, a contradiction.

Now assume to the contrary that there is a winder H of π containing a horizontal side of a rectangle not incident to D_x. By the general position, H does not contain a horizontal side of a rectangle incident to D_x. Then the subpath of $\pi \backslash \mathsf{D}_x$ containing H is not a shortest path connecting its two endpoints incident to D_x. Thus, π is not a shortest path from S to t, a contradiction. □

13 Reducing the time complexity in Section 3

To reduce the time complexity of our algorithm to $O(\log m)$ for handling each event while keeping the space complexity to $O(n)$ space, we build another balanced binary search tree $\mathcal{T}_{\mathsf{seg}}$, a variant of a segment tree in [5]. The idea is to use $\mathcal{T}_{\mathsf{seg}}$ together with $\mathcal{T}_{\mathsf{ran}}$ to maintain $O(\log m)$ nodes corresponding to $O(m)$ $M(i)$'s efficiently, instead of updating $M(i)$'s for each event immediately.

Each node w of $\mathcal{T}_{\mathsf{seg}}$ corresponds to a sequence of baselines consecutive in their indices, say from $\alpha(w)$ to $\beta(w)$ with $\alpha(w) \leq \beta(w)$. Let $\ell_c(w)$ and $r_c(w)$ be the left child and the right child of w, respectively. A leaf node w corresponds to one baseline H_i, hence $\alpha(w) = \beta(w) = i$. A nonleaf node w corresponds to a sequence of baselines corresponding to the leaf nodes in the subtree rooted at w, and thus $\alpha(w) = \alpha(\ell_c(w))$ and $\beta(w) = \beta(r_c(w))$. We say a node w of $\mathcal{T}_{\mathsf{seg}}$ is *inactive* if all baselines with indices from $\alpha(w)$ to $\beta(w)$ are inactive. Node w is *active* otherwise.

We can represent any range of indices using $O(\log m)$ nodes of $\mathcal{T}_{\mathsf{seg}}$ whose ranges are disjoint. We use $\mathcal{W}[a,b]$ to denote the set of nodes of $\mathcal{T}_{\mathsf{seg}}$ such that $[a,b] = \cup_{w\in\mathcal{W}[a,b]}[\alpha(w),\beta(w)]$ and $[\alpha(w),\beta(w)] \cap [\alpha(w'),\beta(w')] = \emptyset$ for any two nodes $w, w' \in \mathcal{W}[a,b]$. We define another set $\mathcal{V}[a,b]$ of nodes v of $\mathcal{T}_{\mathsf{seg}}$ such that $[\alpha(v),\beta(v)] \cap [a,b] \neq \emptyset$ and $[\alpha(v),\beta(v)] \not\subseteq [a,b]$. Observe that the number of nodes in $\mathcal{V}[a,b]$ is also $O(\log m)$. See Figure 10 for an illustration of $\mathcal{W}[a,b]$ and $\mathcal{V}[a,b]$.

For a node $w \in \mathcal{T}_{\mathsf{seg}}$, we define two values, $\lambda(w)$ and $U(w)$ as $\lambda(w) = \min_{i\in[\alpha(w),\beta(w)]} M(i)$ and $U(w) = \max_{i\in[\alpha(w),\beta(w)]} M(i)$. We need $\lambda(w)$ in computing a minimum-link shortest path, while $U(w)$ is used for updating $\mathcal{T}_{\mathsf{seg}}$. We initialize both $\lambda(w)$ and $U(w)$ to ∞ for every node w in $\mathcal{T}_{\mathsf{seg}}$. We update these values stored at some nodes of $\mathcal{T}_{\mathsf{seg}}$ at an event during the plane sweep. At originate and terminate events, we update $\lambda(w)$ and $U(w)$ for the leaf nodes of subtrees rooted at $w \in \mathcal{W}[a,b]$ for a range $[a,b]$, and also update for all w in $\mathcal{T}_{\mathsf{seg}}$ using values of leaf nodes in bottom-up manner. We process other types of events in the following way: we find $\lambda^* = \min_{w\in\mathcal{W}[a_1,b_1]} \lambda(w)$ for a range $[a_1,b_1]$, and then update $\lambda(u)$ and $U(u)$ for $u \in \mathcal{W}[a_2,b_2]$ for another range $[a_2,b_2]$ disjoint from $[a_1,b_1]$. There are three cases. (1) If u becomes inactive at the event, we set both $\lambda(u)$ and $U(u)$ to ∞. (2) If u becomes active at the event, we set both $\lambda(u)$ and $U(u)$ to λ^*+2. Observe that all baselines corresponding to u become active at the event since u is in $\mathcal{W}[a_2,b_2]$. (3) If there is no status change in u, we set $U(u) := \min\{U(u), \lambda^*+2\}$ and $\lambda(u) := \min\{\lambda(u), \lambda^*+2\}$. Once $\lambda(u)$ and $U(u)$ are updated, we also update $\lambda(v)$ and $U(v)$ for $v \in \mathcal{V}[a_2,b_2]$ in bottom-up manner.

Observe that we update neither λ nor U values of the children of u during the update of $\lambda(u)$ and $U(u)$ for $u \in \mathcal{W}[a_2, b_2]$. Some nodes may have their λ and U values outdated when they are used for finding λ^* and updating λ an U values of other nodes. To resolve this problem, we update $\lambda(v')$ and $U(v')$ for the children v' of each node $v \in \mathcal{V}[a,b]$ when we find $\mathcal{W}[a,b]$ for every range $[a,b]$. Note that to compute $\mathcal{W}[a,b]$, we must compute $\mathcal{V}[a,b]$. By the definition of $\lambda(v)$ and $U(v)$, we have $\lambda(v) = \min\{\lambda(\ell_c(v)), \lambda(r_c(v))\}$ and $U(v) = \max\{U(\ell_c(v)), U(r_c(v))\}$, and $\lambda(v) \leq U(v)$.

We update $\lambda(v')$ and $U(v')$ for two children $v' \in \{\ell_c(v), r_c(v)\}$ of v. If $\lambda(v) = U(v)$, by the definition of $\lambda(v)$ and $U(v)$, we have $\lambda(v') = \lambda(v)$ and $U(v') = U(v)$ for all v'. Therefore, we set $\lambda(v') := \lambda(v)$ and $U(v') := U(v)$ for all v'.

If $\lambda(v) < U(v)$, there are four subcases: (1) $\lambda(v) \neq \min_{v'} \lambda(v')$ (2) $\lambda(v) = \min_{v'} \lambda(v')$ but $U(v) > \max_{v'} U(v')$, (3) $\lambda(v) = \min_{v'} \lambda(v')$ but $U(v) < \max_{v'} U(v')$, and (4) $\lambda(v) = \min_{v'} \lambda(v')$ and $U(v) = \max_{v'} U(v')$. For the cases (1) and (2), we set $\lambda(v') := \lambda(v)$ and $U(v') := U(v)$ for all v' because they are outdated. For the case (3), for v' satisfying $U(v) < U(v')$, we set $U(v') := \min\{U(v'), U(v)\}$ and update $\lambda(v')$ compared with $U(v')$. For the case (4), we already use $\lambda(v')$ and $U(v')$ to update $\lambda(v)$ and $U(v)$ in bottom-up manner, so they are not outdated and we do not change any values.

Recall that the number of nodes in $\mathcal{W}[a,b]$ and $\mathcal{V}[a,b]$ for a range $[a,b]$ is $O(\log m)$, and we can find them in $O(\log m)$ time since $\mathcal{T}_{\mathsf{seg}}$ is a balanced binary search tree with height $O(\log m)$. See Chapter 10 in [5]. For each node $w \in \mathcal{W}[a,b] \cup \mathcal{V}[a,b]$, only a constant number of nodes are affected by an update above, and $\lambda(u)$ or $U(u)$ for such node u can be computed in constant time. Thus, each query in $\mathcal{T}_{\mathsf{seg}}$ takes $O(\log m)$ time, and we can find one canonical segment for each event in the same time because we have λ^*. By using this data structure, we can reduce the time complexity from $O(m)$ to $O(\log m)$ per event. Since $\mathcal{T}_{\mathsf{seg}}$ uses $O(m)$ space, the total space complexity remains to be $O(n)$. Thus, we have Lemma 5.

14 Handling degenerate cases in Section 4

There can be two shortest paths from S to t, one passing through $\delta(R)$ and one passing through $\delta'(R)$ for a rectangle R. In this case, we have $d(t, \delta(R)) + d(S, \delta(R)) = d(t, \delta'(R)) + d(S, \delta'(R))$, which can be found in handling the midpoints of R during the plane sweep. Observe that this equality may occur multiple times in finding dividers of a minimum-link shortest path. Thus we need to devise an efficient way of maintaining all sequences of dividers, each of which may define a shortest path. In this section, we show how to maintain these sequences of dividers and how to find a minimum-link shortest path without increasing the time and space complexities in Lemma 7.

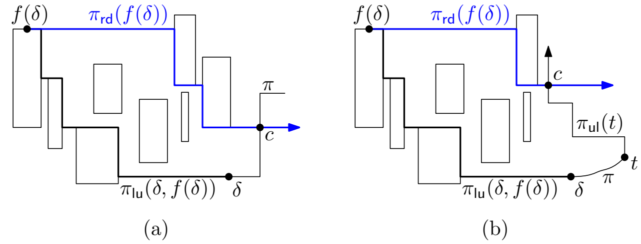


Figure 11: Proof of Lemma 12.

We say two shortest paths, π_1 and π_2, from S to t are *combinatorially distinct* if the sequence of dividers for π_1 and the sequence of dividers for π_2 are different. Let Π be the set of all combinatorially distinct sequences of dividers from t to S for shortest paths, since we let $t = \delta_0$ for convenience. Assume that a divider δ appearing in a sequence of Π is the midpoint of the bottom side of a rectangle. Observe that $\pi_{\mathsf{lu}}(\delta)$ passes through dividers including δ that are consecutive in a sequence of Π. We denote by $f(\delta)$ one with smallest x-coordinate among these dividers. Observe that $f(\delta)$ is uniquely defined for δ with $x(f(\delta)) < x(\delta)$, and it is the midpoint of the top side of another rectangle. We construct $\mathsf{D}_{xy}(f(\delta), \delta)$ to compute the subpath of a minimum-link shortest path from $f(\delta)$ to δ.

Lemma 11 *For any point p in $\mathsf{D}_{xy}(f(\delta), \delta)$, there are an xy-monotone path from $f(\delta)$ to p and an xy-monotone path from p to δ, which are shortest among paths connecting the points.*

Proof. For any point p in $\mathsf{D}_{xy}(f(\delta), \delta)$, let c be a point in the intersection $\pi_{\mathsf{lu}}(\delta) \cap \pi_{\mathsf{lu}}(p)$. Then the path obtained by concatenating the subpath of $\pi_{\mathsf{lu}}(\delta)$ from $f(\delta)$ to c and the subpath of $\pi_{\mathsf{lu}}(p)$ from c to p is xy-monotone, and it is shortest among all paths from $f(\delta)$ to p. Similarly, let c' be a point in the intersection $\pi_{\mathsf{lu}}(\delta) \cap \pi_{\mathsf{dr}}(p)$. Then the path obtained by concatenating the subpath of $\pi_{\mathsf{dr}}(p)$ from p to c and the subpath of $\pi_{\mathsf{lu}}(\delta)$ from c' to δ is xy-monotone, and it is shortest among all paths from p to δ. $\square$

Lemma 12 *$\partial_{xy}(\delta, f(\delta))$ is $\pi_{\mathsf{lu}}(\delta, f(\delta))$ and $\partial_{xy}(f(\delta), \delta)$ is $\pi_{\mathsf{rd}}(f(\delta), \delta)$.*

Proof. By the definitions of $f(\delta)$ and $\mathsf{D}_{xy}(f(\delta), \delta)$, $\partial_{xy}(\delta, f(\delta))$ is $\pi_{\mathsf{lu}}(\delta, f(\delta))$.

Let π be a shortest path from S to t that contains $\pi_{\mathsf{lu}}(\delta, f(\delta))$ as a subpath. Assume that $\pi_{\mathsf{rd}}(f(\delta))$ does not pass through δ. If $\pi_{\mathsf{rd}}(f(\delta)) \cap \pi \neq \emptyset$ except for $f(\delta)$, let c be the last point of $\pi_{\mathsf{rd}}(f(\delta)) \cap \pi$ along $\pi_{\mathsf{rd}}(f(\delta))$

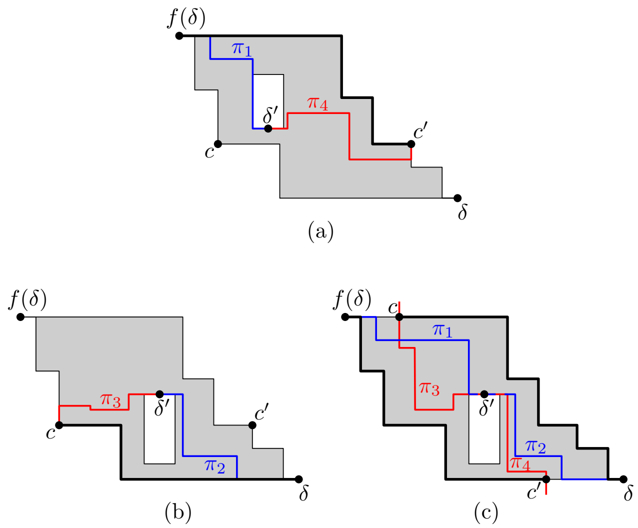


Figure 12: Proof of Lemma 13. Blue paths π_1 and π_2 are subpaths of π, and red paths π_3 and π_4 are subpaths of π^*. (a) If δ' is the midpoint of the bottom side of a rectangle, an xy-monotone path from $f(\delta)$ to c' along $\pi_{\mathsf{rd}}(f(\delta))$ (thick path) is shorter than the concatenation of π_1 and π_4 (subpath of π^+). (b) If δ' is the midpoint of the top side of a rectangle, an xy-monotone path from c to δ along $\pi_{\mathsf{lu}}(\delta)$ (thick path) is shorter than the concatenation of π_3 and π_2 (subpath of π^-). (c) One of thick paths is shorter than one of two concatenations: one is of π_1 and π_4 (subpath of π^+), the other is of π_3 and π_2 (subpath of π^-).

from $f(\delta)$. Since $\pi_{\mathsf{rd}}(f(\delta))$ does not pass through δ, $\pi_{\mathsf{rd}}(f(\delta))$ does not intersect the portion of π from $f(\delta)$ to δ. Thus, c is on the portion of π from δ to t. See Figure 11(a). If $\pi_{\mathsf{rd}}(f(\delta)) \cap \pi = \emptyset$ except for $f(\delta)$, let c be the last point of $\pi_{\mathsf{rd}}(f(\delta)) \cap \pi_{\mathsf{ul}}(t)$ along $\pi_{\mathsf{rd}}(f(\delta))$ from $f(\delta)$. See Figure 11(b). Since $\pi_{\mathsf{rd}}(f(\delta), c)$ is xy-monotone, it is shorter than the portion of π from $f(\delta)$ to c in both cases. Thus, we can get a path from S to t shorter than π by replacing the portion of π from $f(\delta)$ to c with $\pi_{\mathsf{rd}}(f(\delta), c)$, a contradiction. In other words, $\pi_{\mathsf{rd}}(f(\delta))$ pass through δ, so it implies that $\partial_{xy}(f(\delta), \delta)$ is $\pi_{\mathsf{rd}}(f(\delta), \delta)$. $\square$

Lemma 13 *There is no divider on the inner boundary of* $\mathsf{D}_{xy}(f(\delta), \delta)$.

Proof. Assume to the contrary that there is a divider δ' on the inner boundary of $\mathsf{D}_{xy}(f(\delta), \delta)$. Since there are two xy-monotone paths, one from $f(\delta)$ to δ' and one from δ' to δ by Lemma 11, there is an xy-monotone path π' from $f(\delta)$ to δ that passes through δ'. Thus, there is a shortest path π from S to t that contains π' as a subpath.

Since there is a sequence of dividers containing δ' in Π, there is a shortest path π^* from S to t that uses δ' as a divider, that is, two xy-monotone subpaths of π^* change their directions at δ'. Let π'' be the subpath of $\pi^* \cap \mathsf{D}_{xy}(f(\delta), \delta)$ that passes through δ', and let c and c' be the endpoints of π'' with $x(c) \leq x(c')$. If both c and c' are on $\pi_{\mathsf{lu}}(\delta)$, then by replacing π'' of π^* by the portion of $\pi_{\mathsf{lu}}(\delta)$ between c and c' we can get a path from S to t shorter than π^*, a contradiction. Similarly, for the case that both c and c' are on $\pi_{\mathsf{rd}}(f(\delta))$ we can get a shorter path both by replacing π'' of π^* by the portion of $\pi_{\mathsf{rd}}(f(\delta))$ between c and c'.

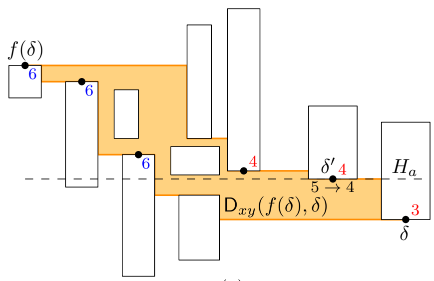
Consider the case that c is on $\pi_{\mathsf{lu}}(\delta)$ and c' is on $\pi_{\mathsf{rd}}(f(\delta))$. Let π^+ be the concatenation of the subpath of π from S to δ' and the subpath of π^* from δ' to t, and π^- be the concatenation of the subpath of π^* from S to δ' and the subpath of π from δ' to t. Observe that π^+ and π^- should be also shortest paths from S to t. If δ' is the midpoint of the bottom side of a rectangle, by replacing the subpath from $f(\delta)$ to c' of π^+ with an xy-monotone path from $f(\delta)$ to c' along $\pi_{\mathsf{rd}}(f(\delta))$, we can get a path from S to t shorter than π^+. See Figure 12(a). If δ' is the midpoint of the top side of a rectangle, by replacing the subpath from c to δ of π^- with an xy-monotone path from c to δ along $\pi_{\mathsf{lu}}(\delta)$, we can get a path from S to t shorter than π^-. See Figure 12(b).

Consider the case that c is on $\pi_{\mathsf{rd}}(f(\delta))$ and c' is on $\pi_{\mathsf{lu}}(\delta)$. Since π'' also uses δ' as a divider, observe that the subpath of π^* from c to c' is not xy-monotone. By replacing the subpath from $f(\delta)$ to c' of π^+ with an xy-monotone path from $f(\delta)$ to c' along $\pi_{\mathsf{lu}}(\delta)$, we can get a path from S to t and we let d_1 be the length of the path. By replacing the subpath from δ to c of π^- with an xy-monotone path from δ to c along $\pi_{\mathsf{rd}}(f(\delta))$, we can get a path from S to t and we let d_2 be the length of the path. Then $\min\{d_1, d_2\} < d(S, t)$, so we can get a path shorter than either π^+ or π^-. See Figure 12(c). $\square$

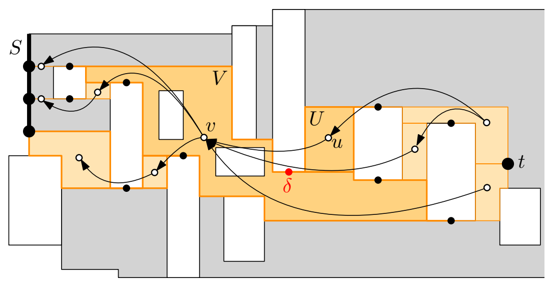
Lemma 14 *If there is a divider δ' lying on $\partial_{xy}(f(\delta), \delta)$, $f(\delta')$ and $f(\delta)$ are the same.*

Proof. Let δ' be a divider lying on $\partial_{xy}(f(\delta), \delta)$. Observe that δ' is the midpoint of the bottom side of a rectangle by Lemma 12. By definition, there is a shortest path π from S to t passing through $f(\delta')$ and δ'. Since δ' lies on $\partial_{xy}(f(\delta), \delta)$, $\pi_{\mathsf{lu}}(\delta')$ passes through $f(\delta)$ and $f(\delta')$ by Lemma 12.

Assume to the contrary that $f(\delta') \neq f(\delta)$. By definition, $f(\delta')$ does not lie on $\partial_{xy}(\delta, f(\delta))$. By Lemma 13, $f(\delta')$ does not lie on the inner boundary of $\mathsf{D}_{xy}(f(\delta), \delta)$. Thus $f(\delta')$ is not incident to $\mathsf{D}_{xy}(f(\delta), \delta)$. Since $\pi_{\mathsf{lu}}(\delta')$ passes through both $f(\delta)$ and $f(\delta')$, $d(f(\delta'), \delta') = d(f(\delta'), f(\delta)) + d(f(\delta), \delta')$. We observe that $d(S, f(\delta)) < d(S, f(\delta')) + d(f(\delta'), f(\delta))$ since otherwise $f(\delta')$ and δ are consecutive in a sequence of Π, so it violates the definition of $f(\delta)$.



(a)



(b)

Figure 13: (a) $\mathsf{D}_{xy}(f(\delta),\delta)$ where δ is the divider such that $g(f(\delta)) = \delta$. Points including δ and δ' lying on $\partial_{xy}(f(\delta),\delta)$ are dividers as originate events, and points including $f(\delta)$ lying on $\partial_{xy}(\delta, f(\delta))$ are dividers as terminate events. During the plane sweep, $M(a) = 5$ before the sweep line encounters δ'. When the line encounters δ', $M(a)$ is updated since we know $\lambda(\delta,t)$ is 4. (b) A directed acyclic graph with constructed xy-monotone subregions. Each xy-monotone subregions correspond to vertices of the graph represented as circles. Both the outer boundaries of the two regions U and V corresponded to u and v contains δ. δ is an terminate event in U and an originate event in V. To apply the plane sweep algorithm in V, $\lambda(\delta,t)$ should be computed first, so there exists a directed edge (u,v).

Adding $d(f(\delta),\delta')$ to the both sides of the inequality, we have $d(S,\delta') < d(S,f(\delta')) + d(f(\delta'),\delta')$. This contradicts that π is a shortest path from S to t. $\square$

For a fixed divider δ, there are dividers each of which appear before δ consecutively in a sequence in Π. Among those dividers, we let $g(\delta)$ be the divider with the largest x-coordinate. Let δ be a divider satisfying $g(f(\delta)) = \delta$. By Lemma 12, $\mathsf{D}_{xy}(f(\delta),\delta)$ is bounded by $\pi_{\mathsf{lu}}(\delta, f(\delta))$ and $\pi_{\mathsf{rd}}(f(\delta),\delta)$. By Lemmas 12 and 14, $\mathsf{D}_{xy}(f(\delta),\delta)$ contains all $\mathsf{D}_{xy}(\delta_i,\delta_j)$'s, where δ_i is a divider lying on $\pi_{\mathsf{lu}}(\delta, f(\delta))$ and δ_j is a divider lying on $\pi_{\mathsf{rd}}(f(\delta),\delta)$ such that δ_i and δ_j are consecutive in a sequence of Π. After constructing $\mathsf{D}_{xy}(f(\delta),\delta)$, we can compute a minimum-link shortest path among all shortest paths from δ_i to δ_j using the plane sweep algorithm in Section 3.

We first find a divider δ such that $g(f(\delta)) = \delta$ appearing in a sequence of Π. By choosing a divider in decreasing order of the x-coordinate, we can easily find such δ. We find $f(\delta)$ and construct $\mathsf{D}_{xy}(f(\delta),\delta)$. In $\mathsf{D}_{xy}(f(\delta),\delta)$, we apply the algorithm in Section 3. In the algorithm, each divider, including δ, lying on $\partial_{xy}(f(\delta),\delta)$ is considered as an originate event, and each divider, including $f(\delta)$, lying on $\partial_{xy}(\delta, f(\delta))$ is considered as a terminate event. See Figure 13(a).

To apply the plane sweep algorithm in Section 3, we must have $\lambda(\delta',t)$ in advance for each divider δ' considered as an originate event. When the sweep line encounters δ', we update $M(a) := \min(M(a), \lambda(\delta',t))$, where a is the index of the horizontal baseline incident to δ'. When the sweep line encounters δ' considered as a terminate event, we can compute $\lambda(\delta',t) := M(a)$, where a is an index of the horizontal baseline incident to δ'. Then $\lambda(\delta',t)$ can be used when δ is considered as an originate event in other xy-monotone subregions.

Recall that t and the closest points of S from t are not divider, but they also construct xy-monotone subregions. If $t = \delta$, Lemma 12 does not hold, but $\mathsf{D}_{xy}(f(t),t)$ has no divider on $\partial_{xy}(f(t),t)$ except $f(t)$ and t. Therefore, we do not have to change the originate event in $\mathsf{D}_{xy}(f(t),t)$. If $f(\delta)$ is the closest point s of S from t, Lemma 12 does not hold, but $\mathsf{D}_{xy}(s,g(s))$ has no divider on $\partial_{xy}(g(s),s)$ except $g(s)$ and s. Therefore, we do not have to change the terminate event in $\mathsf{D}_{xy}(s,g(s))$.

Therefore, to compute $\lambda(\delta',t)$ at the terminate event, where δ' lies on $\partial_{xy}(\delta, f(\delta))$, we have to know $\lambda(\delta'',t)$ at the originate event, where δ'' lies on $\partial_{xy}(f(\delta),\delta)$. It implies that there is an order among xy-monotone subregions to compute a minimum-link shortest path correctly. With the order, we can construct a directed acyclic graph, which is a dual graph of the xy-monotone subregions. Each node v of the graph corresponds to $\mathsf{D}_{xy}(f(\delta),\delta)$. We connect a directed edge from u to v if the two subregions corresponding to u and v are adjacent, and the subregion corresponding to u has a divider δ as a terminate event, and the subregion corresponding to v has δ as an originate event. See Figure 13(b).

Then we can compute $\lambda(S,t)$ using a sequence of xy-monotone subregions corresponding to a path in the dual graph. Recall that during the plane sweep for each xy-monotone subregion, we construct canonical segments to find a minimum-link shortest path whose horizontal line segments are all canonical segments. Since the xy-monotone subregions are disjoint in their interiors, we can report an xy-monotone path using canonical segments by Lemma 4, and glue them to get a minimum-link shortest path from S to t.

Since one rectangle has at most two dividers, there are $O(n)$ dividers and $O(n)$ closest pairs of S and t. By Lemmas 12 and 14 with the property of $\mathsf{D}_{xy}(f(\delta),\delta)$, there are at most two xy-monotone subregions incident to a divider that we construct during the plane sweep. Thus, there are four such subregions incident to a rectangle. Also, those subregions are disjoint in their interiors by Lemma 13.

Lemma 15 *During the plane sweep, we construct $O(n)$ xy-monotone subregions defined by pairs of dividers whose total complexity is $O(n)$. By using these subregions, we can compute a minimum-link shortest path in $O(n \log n)$ time using $O(n)$ space.*

15 S or T intersects bounding boxes in Section 6

Each horizontal or vertical line segment contained in C intersects at most two bounding boxes of polygons in R_P, and thus it can be partitioned into at most three pieces, one disjoint from the rectangles of R, and the other two, each contained in the bounding box of a polygon in R_P. This applies to S and T. Thus, in order to find a minimum-link shortest path from S to T, we need to consider at most 9 pairs, each consisting of one piece of S and one piece of T, and find a minimum-link shortest path for each pair. We can handle the pair consisting of the pieces of S and T disjoint from the rectangles of R using the method in Section 6. In this section we show how to handle the remaining 8 pairs. Each such pair has at least one piece of S or T that is contained in the bounding box of a polygon in R_P.

Without loss of generality, we assume that S is contained in $B(P)$ of $P \in \mathsf{R}_P$ in the following. Let C_S be the the component among the connected components of $B(P) \setminus \mathrm{cl}(P)$ that contains S, where $\mathrm{cl}(P)$ is the closure of P.

15.1 T intersecting C_S

We first consider the case that $T \cap C_S \neq \emptyset$. We assume that T is a vertical line segment. The case that T is a horizontal line segment can be handled analogously. Observe that any closest point of T from S lies on $T' = T \cap C_S$, that is, the problem reduces to computing a minimum-link shortest path from S to T' in the rectilinear polygon C_S. To ease the description, we simply assume that T is contained in C_S. There exists a shortest path from S to T which is not x-, y-, or xy-monotone. However, C_S is the rectilinear polygon without holes, so we can use the algorithm of Schuierer [16], which computes a minimum-link shortest path connecting two points in a rectilinear polygon.

Lemma 16 *If no axis-aligned line segment contained in C_S connects S and T, the closest pair of S and T is unique.*

Proof. We show that the closest points of S from any points of T are the same. Then by symmetry, the closest points of T from any points of S are the same, and thus the lemma holds. Assume to the contrary that there are two distinct closest points s_1 and s_2 in S from two closest points t_1 and t_2 in T, possibly $t_1 = t_2$, respectively. Let π_1 and π_2 be the shortest paths such that π_1 connects s_1 and t_1, and π_2 connects s_2 and t_2. Clearly, both π_1 and π_2 are contained in C_S.

Since S is vertical, the segments of π_1 and π_2 incident to s_1 and s_2 are horizontal, respectively. Let s' be the point in S such that $y(s') = (y(s_1) + y(s_2))/2$, and H be the maximal horizontal segment contained in C_S that contains s'. Since no axis-aligned line segment contained in C_S connects S and T, H does not intersect T but it intersects π_1 or π_2 at a point p. Then we can get a shorter path from S to T by replacing the subpath from s_1 to p of π_1 (or from s_2 to p of π_2) with the segment $s'p$, a contradiction. □

If there is an axis-aligned line segment in C_S connecting S and T, the line segment is a minimum-link shortest path. Otherwise, for a point $t \in T$, we find $d(s,t)$ for every intersection point s of S and the horizontal baselines of C. Lemma 3 also holds in C, so one of those intersection points is the closest point of S from T. Let $s^* \in S$ be the point achieving $d(s^*, t) = \min_s d(s,t)$. Then s^* is the closest point of S from T by Lemma 16. From s^*, we find the point $t^* \in T$ achieving $d(s^*, t^*) = \min_t d(s^*, t)$ among all intersection points t of T and the horizontal baselines of C. Finally we find two points s^* and t^*, so we can compute $\lambda(s^*, t^*)$ using the data structure of Schuierer [16] directly.

We compute the bounding boxes of the polygons in R_P and C_S in $O(n)$ time. We construct the data structure of Schuierer [16] with $O(n)$ time and space for a rectilinear polygon Q with n edges that given two points p and q in Q, reports $d(p,q)$ and $\lambda(p,q)$ in $O(\log n)$ query time, and a minimum-link shortest path from p and q in $O(\log n + K)$ time, where K is the number of links of the path. Since there are $O(n)$ baselines in C, we can find s^* and t^* in $O(n \log n)$ time using the data structure, and a minimum-link shortest path from s^* to t^* in $O(n)$ time since $K = O(n)$.

15.2 T disjoint from C_S

Consider the case that T is disjoint from C_S. The portion of the boundary of C_S which is not incident to P consists of a horizontal segment H_S and a vertical segment V_S. We assume that T is also contained in a connected component C_T of $B(P') \setminus \mathrm{cl}(P')$ for a polygon $P' \in \mathsf{R}_P$. Let H_T and V_T for T be the horizontal segment and a vertical segment of the portion of the boundary of C_T which is not incident to P'.

We compute minimum-link shortest paths from S to T passing through $H_S \cup V_S$ and $H_T \cup V_T$, and then we choose the optimal path among them. In the following, we show how to compute a minimum-link shortest path from S to T passing through V_S and V_T. The other cases can be handled analogously. If no axis-aligned line segment contained in C_S connects S and V_S, the closest pair (s^*, v^*) of S and V_S is unique by Lemma 16. Thus, $d(S, v) = d(s^*, v) = d(s^*, v^*) + d(v^*, v)$ for any point $v \in V_S$. Similarly, the closest pair (t^*, u^*) of T and V_T is also unique if no axis-aligned line segment contained in C_T connects T and V_T. In this case we have $d(T, u) = d(t^*, u) = d(t^*, u^*) + d(u^*, u)$ for any point $u \in V_T$.

Lemma 17 *If the closest pair (s^*, v^*) of S and V_S is unique, and there is a shortest path from S to T passing through V_S, there is a shortest path from S to T passing through v^*.*

Proof. Let π be a shortest path from S to T that passes through a point $v \in V_S \setminus \{v^*\}$. Since $d(s^*, v) = d(s^*, v^*) + d(v^*, v)$, we have $|\pi| = d(S, T) = d(s^*, v) + d(v, T) = d(s^*, v^*) + d(v^*, v) + d(v, T)$. Let π^* be a path from S to T consisting of a shortest path from s^* to v^* and a shortest path from v^* to T. Since $d(v^*, T) \leq d(v^*, v) + d(v, T)$, we have $|\pi^*| = d(s^*, v^*) + d(v^*, T) \leq |\pi| = d(S, T)$. Thus, π^* is also a shortest path from S to T. □

We compute a minimum-link shortest path from S to T as follows. Let Q_S be the set of intersection points of V_S with the horizontal baselines in C, and Q_T be the set of intersection points of V_T with the horizontal baselines in C. Let $\lambda_H(X, Y)$ is the minimum number of links of all shortest paths connecting two sets X and Y whose segments incident to Y are horizontal. We first compute $d(S, v)$ and $\lambda_H(S, v)$ for every point $v \in Q_S$. We also compute $d(T, u)$ and $\lambda_H(T, u)$ for every point $u \in Q_T$. By Lemma 17, once we have the unique closest pairs (s^*, v^*) and (u^*, t^*), their distances $d(s^*, v^*), d(v^*, u^*), d(u^*, t^*)$, and their minimum numbers of links $\lambda_H(s^*, v^*), \lambda_H(v^*, t^*)$, we can compute a minimum-link shortest path π from S to T passing through s^*, v^*, u^* and t^* in order. Note that we do not guarantee that π is a minimum-link shortest path from S to T. However, we can compute a minimum-link shortest path while we compute $\lambda_H(v^*, t^*)$ as follows.

Once we have v^*, u^*, and $d(v^*, u^*)$, we apply the algorithm in Section 6. In the algorithm, we construct xy-monotone subregions. Let $\mathsf{D}(v^*)$ and $\mathsf{D}(u^*)$ be the xy-monotone subregions incident to v^* and u^*, respectively. We may have $\mathsf{D}(v^*) = \mathsf{D}(u^*) = \mathsf{D}_{xy}(v^*, u^*)$ if a shortest path from v^* to u^* is xy-monotone.

Consider a point $v \in Q_S$ that is incident to $\mathsf{D}(v^*)$. Then $d(s^*, v) = d(s^*, v^*) + d(v^*, v)$, v lies on the outer boundary of $\mathsf{D}(v^*)$, and $d(v^*, v) + d(v, t^*) = d(v^*, t^*)$. Thus, we have $d(s^*, v) + d(v, t^*) = d(s^*, v^*) + d(v^*, t^*)$. Once $\lambda_H(v, t^*)$ is computed for every point v of Q_S that is incident to $\mathsf{D}(v^*)$, we can find a minimum-link shortest path from S to T. If $v \in Q_S$ is not incident to $\mathsf{D}(v^*)$, we have $d(v^*, v) + d(v, t^*) > d(v^*, t^*)$, and thus no shortest path from S to T passes through v. This observation can also be applied for points in Q_T that are incident to $\mathsf{D}(u^*)$. The plane sweep algorithm starts with updating $M(i)$'s for the horizontal baselines H_i intersecting the vertical line segment V of the outer boundary of $\mathsf{D}(u^*)$ corresponding to the originate event. At the originate event, those $M(i)$'s are initialized to $\lambda_H(t^*, V \cap H_i)$. Observe that every intersection point $V \cap H_i$ is in Q_T. It also computes $M(i)$'s for the horizontal baselines H_i intersecting the vertical line segment of the outer boundary of $\mathsf{D}(v^*)$ corresponding to the terminate event. Hence one of $M(i)$'s corresponds to $\lambda_H(t^*, v^*)$ at the terminate event. By choosing the minimum of $\lambda_H(s^*, v) + \lambda_H(v, t^*) - 1$ for all $v \in Q_S$ incident to $\mathsf{D}(v^*)$, we finally obtain $\lambda(S, T)$, and compute a minimum-link shortest path from S to T. Recall that to reduce the time complexity to $O(n \log n)$, we do not maintain $M(i)$'s explicitly, but focus on the minimum of $M(i)$'s using $O(\log n)$ nodes of $\mathcal{T}_{\mathsf{seg}}$ as we do in Section 13. However, we observe that $\lambda_H(s^*, v^*) + c = \lambda_H(s^*, v)$ for every point v of Q_S, where $c \in \{0, 1, 2\}$. Therefore, by storing for each node w of $\mathcal{T}_{\mathsf{seg}}$, $\lambda(w), U(w)$, and the second minimum and the third minimum of $M(i)$'s for $i \in [\alpha(w), \beta(w)]$, we can compute a minimum-link shortest path from S to T without increasing time and space complexities.

If the closest pair of S and V_S is not unique, there is a maximal line segment $V' \subseteq V_S$ such that for every point $p \in V'$, the shortest path from S to p is a horizontal line segment in C_S. Recall that our algorithm uses the point v^* in V_S if the closest pair (s^*, v^*) of S and V_S is unique. Hence, instead of using v^*, we apply the algorithm using V' and then we can compute a minimum-link shortest path.

Again using the data structure of Schuierer [16], we can compute $d(S, v)$ and $\lambda_H(S, v)$ for all $v \in Q_S$ (and $d(T, u)$ and $\lambda_H(T, u)$ for all $u \in Q_T$) in $O(n \log n)$ time. Then we use the algorithms in Section 6 based on the methods in Sections 3 and 4 to compute $d(v^*, u^*)$. Observe that the time and space complexities remain the same as stated in Lemma 9. The initialization of $M(i)$'s at the originate event of $\mathsf{D}(u^*)$, and the computation of $\lambda_H(T, u^*)$ using $M(i)$'s at the terminate event of $\mathsf{D}(v^*)$ do not affect the time and space complexities asymptotically. Therefore, we have the following theorem.

Theorem 18 *Given two axis-aligned line segments S and T in a box-disjoint rectilinear domain with n vertices in the plane, we can compute the minimum-link shortest path from S to T in $O(n \log n)$ time using $O(n)$ space.*