Approxiamte Shortest Homotopic Paths in Weighted Regions[*]

Siu-Wing Cheng[†] Jiongxin Jin[†] Antoine Vigneron[‡] Yajun Wang[§]

Abstract

Let P be a path between two points s and t in a polygonal subdivision $\mathcal{T}$ with obstacles and weighted regions. Given a relative error tolerance $\varepsilon \in (0,1)$, we present the first algorithm to compute a path between s and t that can be deformed to P without passing over any obstacle and the path cost is within a factor $1+\varepsilon$ of the optimum. The running time is $O(\frac{h^3}{\varepsilon^2} kn \,\mathrm{polylog}(k, n, \frac{1}{\varepsilon}))$, where k is the number of segments in P and h and n are the numbers of obstacles and vertices in $\mathcal{T}$, respectively. The constant in the running time of our algorithm depends on some geometric parameters and the ratio of the maximum region weight to the minimum region weight.

1 Introduction

Given a path P in the plane, the shortest homotopic path problem is to find a minimum-cost path that can be deformed to P without crossing any obstacle. The problem originates from research in VLSI [4, 9, 15]. In some planning system, a user makes a path sketch for vehicles or people and then the system generates the detailed optimized path homotopic to the sketch [7]. It is natural to consider non-Euclidean cost models because different regions incur different costs; for example, traveling in swamps is harder than traveling on roads. Besides applications, the shortest homotopic path problem is a natural variant of the classical shortest path problem.

With Euclidean cost, the shortest path problem in the plane can be solved in optimal time using the algorithm of Hershberger and Suri [11]. Given a homotopy constraint (specified by an input path P with k segments), Hershberger and Snoeyink showed how to compute the shortest homotopic path in $O(kn)$ time after triangulating the space, where n is the number of obstacle vertices [10]. Efrat, Kobourov, and Lubiw reduced the running time when P is simple: $O(k_{\text{out}} + k \log n + n\sqrt{n})$ worst-case time and $O(k_{\text{out}} + k \log n + n \log^{1+\varepsilon} n)$ expected time for any $\varepsilon > 0$, where k_{out} is the output size [5]. Bespamyatnikh improved it further: $O(k_{\text{out}} + k \log n + n \log^{1+\varepsilon} n)$ time when P is simple and $O(k \log^2 n + n^{2+\varepsilon})$ time otherwise [2].

The *weighted region* model is the first non-Euclidean cost model and there has been much work on it [1, 14, 16, 17]. The environment is a polygonal subdivision, each region f has a weight w_f, and the subpath cost within a region f is w_f times the subpath length. Computing the exact shortest path seems hard and only approximation algorithms are known so far. The first algorithm of Mitchell and Papadimitriou runs in $O(n^8 \log \frac{nN\rho}{\varepsilon})$ time, where n is the number

[*]Research of the first and second authors are partly supported by the Research Grant Council, Hong Kong, China (project no. 612107).

[†]Department of Computer Science and Engineering, HKUST, Hong Kong. Email: {scheng,jamesjjx}@cse.ust.hk

[‡]Applied Mathematics and Informatics Department, INRA, Jouy-en-Josas, France. Email: antoine.vigneron@jouy.inra.fr

[§]Microsoft Research Asia, Beijing, China. Email: yajunw@microsoft.com of subdivision vertices, the vertices have integer coordinates in $[0, N]$, and ρ is the ratio of the maximum region weight to the minimum region weight [16]. Subsequently, other algorithms have been proposed whose running times have a lower dependence on n. The most notable approach is to compute the shortest path in a graph obtained by discretizing the input subdivision, so as to approximate the true shortest path [1, 17]. Sun and Reif gave an algorithm that runs in $O(\frac{n}{\varepsilon}\log\frac{n}{\varepsilon}\log\frac{1}{\varepsilon})$ time, where the hidden constant depends on some geometric parameters [17]. Aleksandrov et al. achieved the best dependence on n and ε with a running time of $O(\frac{n}{\sqrt{\varepsilon}}\log\frac{n}{\varepsilon}\log\frac{1}{\varepsilon})$, where the hidden constant depends on ρ and some geometric parameters [1]. No result is known so far on the shortest homotopic path problem in weighted regions.

The main result in this paper is a $(1+\varepsilon)$-approximate shortest homotopic path algorithm for any $\varepsilon \in (0, 1)$ in weighted regions. Let P be a path between two points s and t in a polygonal subdivision $\mathcal{T}$ with obstacles and weighted regions. Self-intersections in P are allowed. Given $\varepsilon \in (0, 1)$, our algorithm computes a path between s and t that can be deformed to P without passing over any obstacle and the path cost is within a factor $1+\varepsilon$ of the optimum. The running time is $O(\frac{h^3}{\varepsilon^2}kn \operatorname{polylog}(k, n, \frac{1}{\varepsilon}))$, where k is the number of segments in P and h and n are the numbers of obstacles and vertices in $\mathcal{T}$, respectively. The hidden constant in our running time depends on ρ and some geometric parameters. These geometric parameters and the dependence on them are of the same kind as in the work of Sun and Reif [17] as we use their result as a subroutine.

2 Preliminaries

We denote the input polygonal subdivision by $\mathcal{T}$, which consists of vertices, edges, and polygonal faces. Some polygonal faces are marked as inaccessible and each connected component of inaccessible faces forms an obstacle. The remaining polygonal faces are accessible and they are called the *regions* of $\mathcal{T}$. Each region f is associated with a positive weight $w_f > 0$. Without loss of generality, we assume that $\mathcal{T}$ is connected, every obstacle is a simple polygon, every region is a triangle, and the minimum region weight is equal to 1. We use ρ to denote the maximum region weight in $\mathcal{T}$.

Consider a line segment pq and a region f. Let $|pq|$ denote the length of pq. We use $\operatorname{int}(\cdot)$ to denote the interior of the operand. If $\operatorname{int}(pq) \subset \operatorname{int}(f)$ or pq is contained in an edge adjacent to f only, we define $\operatorname{cost}_{\mathcal{T}}(pq) = w_f|pq|$. If pq is contained in an edge shared between f and another region g, we define $\operatorname{cost}_{\mathcal{T}}(pq) = \min\{w_f, w_g\} \cdot |pq|$. A *polygonal path* Q is a polyline in $\mathcal{T}$ with finitely many segments. A *link* of Q is a maximal segment in Q that lies in a region of $\mathcal{T}$. An endpoint of a link is called a *node*. We use $|Q|$ to denote the length of Q. We use $\operatorname{cost}_{\mathcal{T}}(Q)$ to denote the sum of the costs of its links. Notice that $|Q| \le \operatorname{cost}_{\mathcal{T}}(Q) \le \rho|Q|$.

We use P to denote the input polygonal path. We use s and t to denote the endpoints of P and we enforce them to be vertices of $\mathcal{T}$ by splitting regions if necessary. Two paths with the same endpoints are *homotopic* if one can be deformed to the other without passing over any obstacle.

3 Overview

We present a simplified version of our strategy to highlight the main ideas. This simplified strategy cannot be turned into an effective algorithm, for instance, because no algorithm is known for computing an exact shortest path in weighted regions.

We are given a triangulated domain with obstacles, and we want to find a shortest path

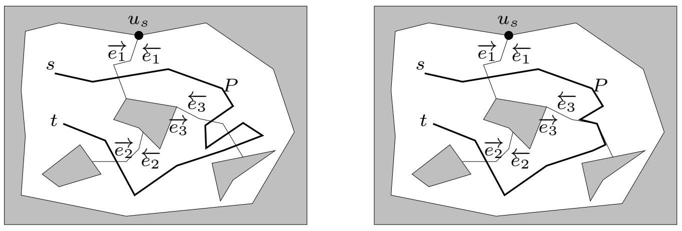


Figure 1: The obstacles are shaded. After canceling one $\overleftarrow{e_3}$ and one $\overrightarrow{e_3}$, the path P becomes a new path P' that crosses the edge e_3 once.

homotopic to a given input path P, with endpoints s and t. We first need to encode the homotopy of P. To this end, we build a spanning tree of the obstacles, with an extra edge connecting it to a point u_s on the outer face of our domain. The edges of this spanning tree are denoted by $e_1, e_2, \ldots$. We choose each such edge e_i to be a shortest path between two points lying on obstacles, or between u_s and a point lying on an obstacle.

We follow P from s to t to trace the edges that it crosses as well the crossing directions (determined with respect to an arbitrarily chosen orientation of the e_i's). In Fig. 1, the trace is $\overrightarrow{e_1}\overleftarrow{e_3}\overrightarrow{e_3}\overleftarrow{e_3}\overleftarrow{e_2}$, where $\overleftarrow{e_i}$ means crossing e_i from right to left and $\overrightarrow{e_i}$ means crossing e_i from left to right. We call it the *crossing sequence* of P. If $\overleftarrow{e_i}$ and $\overrightarrow{e_i}$ appear consecutively in the crossing sequence, we can cancel them. This corresponds to making a shortcut along e_i between the two crossings as illustrated in Fig. 1. The important point is that the above cancellation does not change the homotopy of the path. When all cancellations are done, we obtain the *canonical crossing sequence*, a unique encoding of the homotopy of P. Indeed, two paths P and Q with the same endpoints are homotopic iff their canonical crossing sequences are identical.

Since the tree edges are shortest paths, a shortcut (canceling two adjacent symbols in the crossing sequence) does not increase the path cost. This is ideal because it means that for any path P, there is a shortest path P^* homotopic to P that crosses the spanning tree as dictated by the canonical crossing sequence of P. The path P^* makes no redundant crossings. A natural approach to compute such a shortest path is as follows. Assume that the canonical crossing sequence starts with $\overrightarrow{e_1}\overleftarrow{e_3}\overleftarrow{e_2}\ldots$. We know that P^* will first reach e_1 from the left. As we do not know at which point of e_1 it arrives, we can discretize e_1 by placing many vertices along it. For each of these vertices, we compute an approximate shortest path from s, treating the edges e_i of our tree as obstacles. As these paths avoid our spanning tree, they lie in a simply connected region, so we do not need to consider their homotopy class. Thus, we can apply known algorithms for approximate shortest paths in weighted regions.

After crossing e_1, we know that P^* will reach e_3 from the right. So we perform a second round of approximate shortest paths computation (where the paths are not allowed to cross our spanning tree). We perform this computation with multiple sources, each source being one of the vertices placed on e_1, and each such vertex having an additive weight which is the approximate shortest distance from s to this vertex. The target points, again, are the vertices placed densely along e_3. We repeat this process for each symbol in the canonical crossing sequence, and we obtain an approximate shortest path homotopic to P.

Our actual algorithm follows similar ideas, but there are important differences as we face several difficulties. The most obvious one is that no algorithm is known for computing an exact shortest path in weighted regions. Second, the spanning tree calls for repeated shortest path computations in order to connect the obstacles, which is rather wasteful. So we replace the spanning tree above by another tree, the *anchor tree*, which is basically an approximate shortest path tree from u_s to one vertex of each obstacle. The homotopy encoding is still based on the crossings between P and the anchor tree, but we change it slightly for technical convenience. Since the paths in the anchor tree are not exact shortest paths, we cannot expect a shortest path homotopic to P to cross the anchor tree exactly as dictated by the canonical crossing sequence. To eliminate the redundant crossings, we have to reroute the optimal path along the anchor tree in the analysis. This demands a careful construction of the anchor tree so that the rerouting error is small. Another major efficiency issue is that we need to keep the canonical crossing sequence short because the running time of our algorithm is directly related to it. Finally, to make our algorithm run faster, we will not discretize the anchor tree. We will still run one round of approximate shortest paths computation for each symbol in the canonical crossing sequence, but in the absence of vertices on the anchor tree, multiple crossings of the anchor tree (instead of just one) may have to be taken at the end of a round. We need to do this quickly while conforming to the canonical crossing sequence. The rest of this paper explains how to handle these difficulties.

4 The subdivision $\mathcal{S}$ and the graph H_ε

We introduce a graph H_ε which is the discretization of some subset of $\mathcal{T}$ based on the scheme of Sun and Reif [17]. We briefly review their construction below. Given a subdivision $\mathcal{K}$ with triangular regions, Sun and Reif place $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ Steiner points on each edge of $\mathcal{K}$, where the hidden constant depends on some geometric parameters. The vertices of $\mathcal{K}$ and these Steiner points form the vertex set of a graph which we denote by $G_\varepsilon(\mathcal{K})$. Every two vertices p and q of $G_\varepsilon(\mathcal{K})$ on the boundary of a region are connected by the edge pq with weight $\mathrm{cost}_{\mathcal{K}}(pq)$. There are $O(\frac{1}{\varepsilon}|\mathcal{K}|\log\frac{1}{\varepsilon})$ vertices and $O(\frac{1}{\varepsilon^2}|\mathcal{K}|\log^2\frac{1}{\varepsilon})$ edges in $G_\varepsilon(\mathcal{K})$. So Dijkstra's algorithm returns a shortest path or a shortest path tree in $G_\varepsilon(\mathcal{K})$ in $O(\frac{1}{\varepsilon^2}|\mathcal{K}|\log\frac{|\mathcal{K}|}{\varepsilon}\log\frac{1}{\varepsilon})$ time [8]. A shortest path in $G_\varepsilon(\mathcal{K})$ is a $(1+\varepsilon)$-approximate shortest path in $\mathcal{K}$. Sun and Reif gave a faster shortest path algorithm that avoids generating the edges of $G_\varepsilon(\mathcal{K})$, but we do not need this as other tasks will prove to be more time-consuming. Aleksandrov et al. has a related construction, which places Steiner points in the interior of each triangle [1]. It has better dependence on ε, but we cannot use it due to some technical difficulties.

The graph H_ε is $G_\varepsilon(\mathcal{S})$ for some refinement $\mathcal{S}$ of a subset of $\mathcal{T}$. We will run multiple rounds of Dijkstra's algorithm on a subgraph H_{alg} of H_ε to generate the $(1+\varepsilon)$-approximate shortest homotopic path. A dense enough discretization is sufficient for this purpose. We will use another subgraph H_{fen} of H_ε to compute the anchor tree for encoding the homotopy of P. This requires H_{fen} to have some extra properties as we explain later in this section. Although H_{alg} and H_{fen} serve different purposes, the $(1+\varepsilon)$-approximate shortest homotopic path (in H_{alg}) has to interact with the anchor tree (in H_{fen}), i.e., cross it. The relations among H_ε, H_{alg}, and H_{fen} facilitate the analysis.

Let L_{st} denote the length of a minimum-length path homotopic to P. Let B denote an axis-parallel box centered at s with width $4\rho L_{st}$. The cost of the shortest path homotopic to P is between L_{st} and ρL_{st}. So for any $\varepsilon \in (0,1)$, the box B contains any $(1+\varepsilon)$-approximate shortest path homotopic to P, which means that only the obstacles inside B are relevant. The restriction to B controls the costs of the paths in the anchor tree, which helps to bound the length of the canonical crossing sequence of P.

For each obstacle inside B, we pick one of its vertices to be an *anchor*. We compute the *anchor triangulation*, a triangulation of the anchors as well as the four corners of B. We superimpose the anchor triangulation on $B \cap \mathcal{T}$ to obtain a subdivision $\mathcal{T}'$. Notice that an

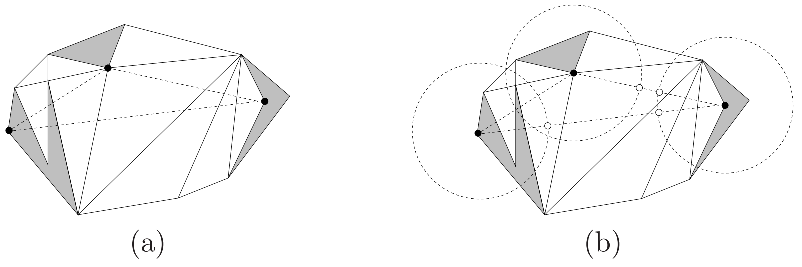


Figure 2: The obstacles are shaded. We ignore the box B for simplicity. In (a), the black dots denote the anchors and the dashed segments form the anchor triangulation. In (b), the circles have radii δ_{fen} and the white dots are the extra vertices inserted.

anchor triangulation edge may be split into several edges in $\mathcal{T}'$ by the obstacles and the edges of $B \cap \mathcal{T}$. Fig. 2(a) gives an illustration. The anchor triangulation edges provide shortcuts in $\mathcal{T}'$ that one can take in building the anchor tree. This controls the length of the canonical crossing sequence of P.

We also need to prevent any path in the anchor tree from spiraling around the obstacles in order to keep the canonical crossing sequence of P short. For this purpose, for each edge uv in the anchor triangulation, the subset of uv within a distance $\delta_{\text{fen}} = \varepsilon L_{st}/\Theta(\rho k n)^{O(1)}$ from u or v plays a special role in building the anchor tree. (We do not allow the tree to cross it.) Either this subset consists of two segments ux and vy or it is the whole edge uv. In the former case, we insert x and y as extra vertices into $\mathcal{T}'$ if they do not fall inside obstacles. Fig. 2(b) shows an example. The exact value of δ_{fen} will be specified in the proof of our main result Theorem 1.

Finally, the subdivision $\mathcal{S}$ is the refinement of $\mathcal{T}'$ so that all regions become triangles. Without loss of generality, we assume that $\mathcal{S}$ is connected. It has $O(hn)$ vertices and $O(hn)$ edges. We construct the graph H_ε as $G_\varepsilon(\mathcal{S})$, which has $O(\frac{h}{\varepsilon} n \log \frac{1}{\varepsilon})$ vertices and $O(\frac{h}{\varepsilon^2} n \log^2 \frac{1}{\varepsilon})$ edges.

5 Anchor tree

We introduce an *anchor tree* $\mathcal{A}$ to connect the anchors. The crossings between $\mathcal{A}$ and P will be used to encode the homotopy of P. Let u_s be a highest vertex in $\mathcal{S}$. The anchor tree $\mathcal{A}$ consists of two parts, a non-self-intersecting subtree in $\mathcal{S}$ that is rooted at u_s and spans all anchors, and a ray that shoots upward from u_s to infinity. So $\mathcal{A}$ is a rooted tree with the root at vertical infinity.

Let $a_1, a_2, \ldots, a_h$ be the enumeration of anchors in $\mathcal{A}$ in post-order. Let α_i denote the directed tree path in $\mathcal{A}$ from a_i to vertical infinity. Although the paths $\alpha_1, \alpha_2, \ldots$ may overlap, we view them as non-crossing and side by side. Fig. 3(a) shows an example. The *crossing sequence* of P is built by traversing P from s to t, appending a symbol $\overleftarrow{a_i}$ or $\overrightarrow{a_i}$ whenever P crosses α_i. We append $\overrightarrow{a_i}$ if α_i is crossed from left to right with respect to its direction. We append $\overleftarrow{a_i}$ otherwise. Fig. 3(b) shows an example. If $\overleftarrow{a_i}$ and $\overrightarrow{a_i}$ are adjacent in the crossing sequence, we can cancel them. It corresponds to a path deformation that does not pass over any obstacle. Repeating until no other symbol can be deleted gives the unique *canonical crossing sequence* as implied by Lemma 5.1 below. Cabello et al. [3] used vertical lines though obstacles to define the crossing sequence when the path cost is its length. The anchor tree generalizes this idea. The same idea of using a tree to encode homotopy was also used by Kaufmann and Mehlhorn [13]. For completeness, we include a proof of Lemma 5.1 in the appendix.

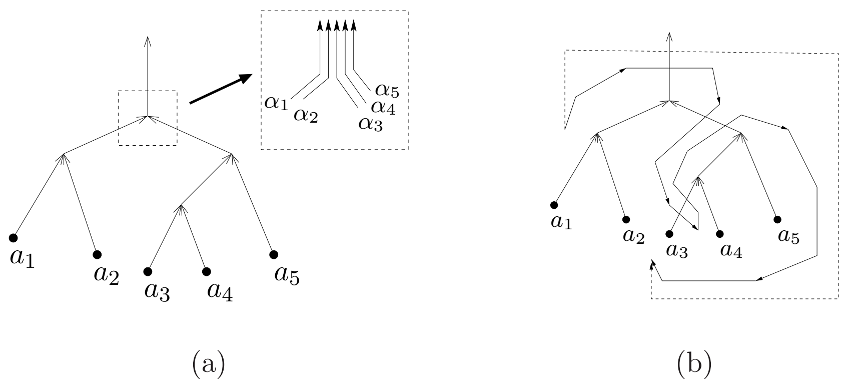


Figure 3: In (b), the crossing sequence of the solid path is $\overrightarrow{a_1}\overrightarrow{a_2}\overrightarrow{a_3}\overrightarrow{a_4}\overrightarrow{a_5}\overleftarrow{a_5}\overleftarrow{a_4}\overleftarrow{a_3}\overrightarrow{a_3}\overleftarrow{a_3}\overrightarrow{a_3}\overrightarrow{a_4}\overrightarrow{a_5}$. It can be reduced to the crossing sequence $\overrightarrow{a_1}\overrightarrow{a_2}\overrightarrow{a_3}\overrightarrow{a_4}\overrightarrow{a_5}$ of the dashed path.

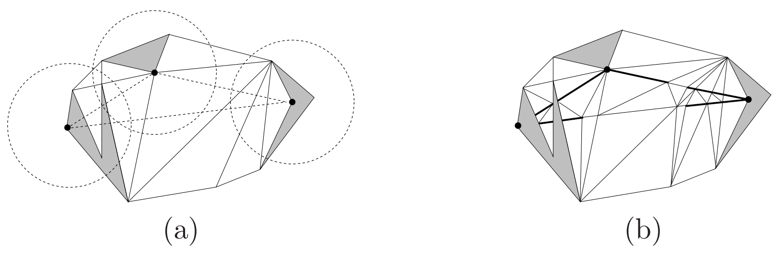


Figure 4: The shaded regions are obstacles. We ignore the box B for simplicity. In (a), the black dots denote the anchors, the dashed segments form the anchor triangulation, and the dashed circles have radii δ_{fen}. In (b), the fences are shown as bold segments and the refined subdivision is $\mathcal{S}$. Notice that a fence may consist of several edges of $\mathcal{S}$.

Lemma 5.1 *Let H denote $\mathbb{R}^2$ minus the obstacles with anchors. Two paths in H with the same endpoints are homotopic if and only if their canonical crossing sequences are identical.*

We construct the subtree of $\mathcal{A}$ rooted at u_s as a shortest path tree in some subgraph of H_ε as follows. For edge uv of the anchor triangulation, the subset of uv within a distance δ_{fen} from u or v consists of collinear edges in $\mathcal{S}$. Due to obstacles, these collinear edges may form several connected components and we call each connected component a *fence*. Fig. 4 shows an example. To keep the canonical crossing sequence of P short, we should prevent any path in $\mathcal{A}$ from spiraling around the obstacles and hence anchors. We achieve this by making the interior of fences impenetrable. This is easily done by splitting some vertices of H_ε as follows. We split every vertex v of $\mathcal{S}$ in the interior of a fence into two copies, one on each side of the fence, and these two copies are not connected. Any edge incident to v is made incident to the copy of v on the same side of the fence. Notice that one can still pass through a fence at its endpoints. We use H_{fen} to denote the resulting graph, which is like a subgraph of H_ε in the sense that every edge in H_{fen} is contained in H_ε. We compute the subtree of $\mathcal{A}$ rooted at u_s as the shortest path tree in H_{fen} from u_s to all anchors. The next result states several properties of $\mathcal{A}$.

Lemma 5.2 *$\mathcal{A}$ does not intersect itself, has $O(\frac{h}{\varepsilon}n\log\frac{1}{\varepsilon})$ size, and can be computed in $O(\frac{h}{\varepsilon^2}n\log\frac{n}{\varepsilon}\log\frac{1}{\varepsilon})$ time. Let γ_i, $i \in [1,h]$, denote the paths in $\mathcal{A}$ between u_s and the anchors.*

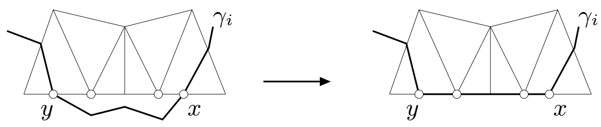


Figure 5: Rerouting along xy to shorten γ_i.

(i) $\mathrm{cost}_{\mathcal{T}}(\gamma_i) = O(\rho^2 n L_{st})$.

(ii) *The subpath of γ_i between any two nodes p and q has cost at most $d_{pq} + O(\rho h \delta_{\mathrm{fen}})$, where d_{pq} is the shortest path cost in H_ε between p and q.*

(iii) *Let y be a crossing point between γ_i and an edge vw of the anchor triangulation. If $|vy| < \delta_{\mathrm{fen}}$, then y lies on an obstacle.*

(iv) *Suppose that γ_i intersects an edge of the anchor triangulation at two points x and y. If xy does not intersect any obstacle, the subpath of γ_i between x and y has cost at most* $\mathrm{cost}_{\mathcal{T}}(xy)$.

Proof. Since $\mathcal{A}$ is a shortest path tree in H_{fen}, if two edges pq and $p'q'$ of $\mathcal{A}$ indeed cross each other, they must do so inside a region of $\mathcal{S}$. Thus, pq and $p'q'$ are the diagonals of a convex quadrilateral inside this region. But then replacing pq and $p'q'$ by pp' and qq' would shorten some paths in $\mathcal{A}$, a contradiction. The size of $\mathcal{A}$ follow from the previous discussion.

Consider (i). A geodesic path in $\mathcal{S}$ from u_s to any anchor has $O(n)$ segments, each with length $O(\rho L_{st})$. So the geodesic path has cost $O(\rho^2 n L_{st})$ and so does γ_i. Consider (ii). The shortest path in H_ε between p and q may cross a fence ℓ several times and it is easy to reroute around ℓ with detour length $O(|\ell|)$ and cost $O(\rho|\ell|)$. Rerouting around all fences gives a path in H_{fen} between p and q, which cannot be shorter than the subpath of γ_i between p and q. The fences form $O(h)$ collinear groups, each has a total length no more than $2\delta_{\mathrm{fen}}$. So the total rerouting error is $O(\rho h \delta_{\mathrm{fen}})$. Consider (iii). If $|vy| < \delta_{\mathrm{fen}}$, then y lies on a fence. So y is a fence endpoint as γ_i can only cross a fence at its endpoints. By construction, if an endpoint of a fence on vw is at distance less than δ_{fen} from v, this endpoint lies on an obstacle. Consider (iv). Observe that xy consists of a linear sequence of edges in H_{fen}. If (iv) is false, we can shorten γ_i as shown in Fig. 5, a contradiction. $\square$

We prove a bound on the length of a canonical crossing sequence that has a low dependence on n and ε. This is the key to achieving a running time nearly linear in kn.

Lemma 5.3 *The canonical crossing sequence S_P of P has length $O(\rho h^2 k \log \frac{\rho k n}{\varepsilon})$.*

Proof. We break the k segments in P at their crossings with the vertical ray in $\mathcal{A}$. There are at most k such crossings, so P is partitioned into at most $2k$ subsegments such that each subsegment may cross the subtree of $\mathcal{A}$ rooted at u_s but not the vertical ray. Our strategy is to deform each subsegment and show an $O(\rho h \log \frac{\rho k n}{\varepsilon})$ bound on the number of crossings between the deformed subsegment and any path from u_s to an anchor in $\mathcal{A}$.

Take a subsegment ℓ and a path γ in $\mathcal{A}$ from u_s to an anchor. Let x and x' be two crossings between ℓ and γ that appear consecutively along γ. The subpath of γ between x and x' forms

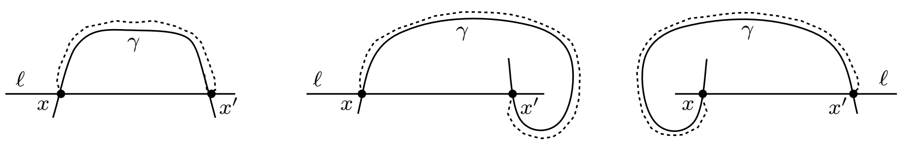


Figure 6: Morph xx' to follow the dashed curve. This eliminates the crossings x and x' on the left, x in the middle, and x' on the right.

a simple cycle with xx'. If no obstacle lies inside this cycle, we deform ℓ by morphing xx' to a curve next to the subpath of γ between x and x' as shown in Fig. 6. This eliminates the crossing x, x', or both. The deformation does not pass over any obstacle as no obstacle lies inside the cycle. So the deformed ℓ is homotopic to ℓ. The deformed ℓ has no new crossing with $\mathcal{A}$ because xx' is replaced by a curve next to a subpath in $\mathcal{A}$. The choices of x and x' imply that γ does not cross ℓ between x and x'. So the deformed ℓ does not cross itself. We repeat until no more crossings with $\mathcal{A}$ can be eliminated. In general, as the current deformed ℓ is not straight, we need to morph its subpath between the crossings x and x' instead of the segment xx'. But the morphings are similar to those in Fig. 6. The next proposition follows by induction.

Proposition 1 *Any subsegment ℓ can be deformed to a homotopic curve σ such that σ does not cross itself, and if a path γ in $\mathcal{A}$ from u_s to an anchor crosses σ at x and x' consecutively along γ, then some obstacle lies inside the cycle formed by the subpaths of γ and σ between x and x'.*

Let σ be the curve homotopic to ℓ in Proposition 1. Take a path γ in $\mathcal{A}$ from u_s to an anchor. We define $\gamma(p, q)$ to be the subcurve of γ between two points p and q on it. The subcurve $\sigma(p, q)$ is similarly defined. Let $x_1, x_2, \ldots$ denote the crossings between γ and σ. All these crossings lie on ℓ. Consider the set of cycles $\{C_{ij} = \sigma(x_i, x_j) \cup \gamma(x_i, x_j) : x_i \text{ and } x_j \text{ are consecutive along } \gamma\}$. We order the subscripts of C_{ij} such that u_s is closer to x_i than x_j along γ. Each cycle C_{ij} is simple because $\sigma(x_i, x_j)$ does not cross itself by Proposition 1.

Proposition 1 allows us to cluster cycles that enclose the same anchors, and cycles in the same cluster are nested. Rotate the plane so that the subsegment ℓ is horizontal. We divide a cluster into a *left-group* and a *right-group*, depending on whether x_i lies to the left or right of x_j on ℓ. There are at most $2h$ left- and right-groups. We show that a left-group has $O(\rho \log \frac{\rho k n}{\varepsilon})$ cycles as follows. The size of a right-group can be analyzed similarly.

Refer to Fig. 7(a) which illustrates a left-group $\{C_{i_1 j_1}, C_{i_2 j_2}, \ldots, C_{i_m j_m}\}$. There exists an edge e of the anchor triangulation that cuts through all cycles in the left-group and ends at some anchor a inside the innermost cycle. (The existence of e is ensured because we include the corners of the box B in the anchor triangulation.) Walk along e away from a. Identify the first crossing between e and each cycle in the left-group. Label these crossings as $y_1, y_2, \ldots, y_m$ at increasing distances from a. Label the cycles so that y_k lies on $C_{i_k j_k}$ for $k \in [1, m]$. It follows that $C_{i_k j_k}$ is nested in $C_{i_{k+1} j_{k+1}}$ for $k \in [1, m-1]$.

Proposition 2 *For $k \in [2, m]$, we have $|ay_k| \geq (1 + 1/\rho)^{k-2} |ay_2|$.*

Proof. By construction, y_{k+1} is the first crossing between e and $C_{i_{k+1} j_{k+1}}$ when we walk along e from the anchor a. Let z_k be the last crossing between e and $C_{i_k j_k}$ before we hit y_{k+1}. So $z_k y_{k+1} \subseteq y_k y_{k+1}$.

We prove the proposition by induction. The base case of $k = 2$ is trivial. Assume that the proposition is true for some $k \in [2, m-1]$. We show that if the proposition

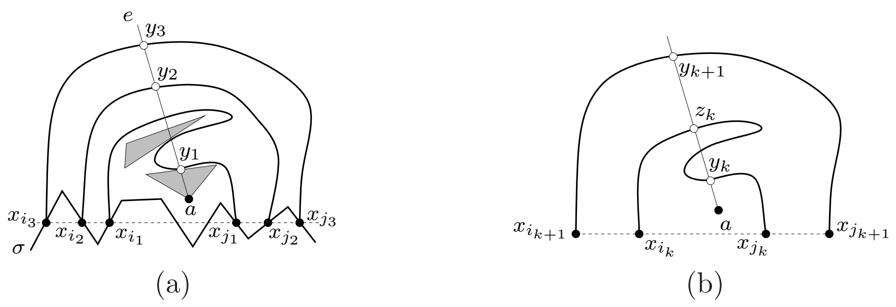


Figure 7: (a) The shaded triangles are obstacles. The dashed line denotes ℓ. The polygonal curve denotes σ. The bold curves denote $\gamma(x_{i_1}, x_{j_1})$, $\gamma(x_{i_2}, x_{j_2})$, and $\gamma(x_{i_3}, x_{j_3})$. (b) The segment $z_k y_{k+1}$ does not intersect any obstacle.

is false for $k+1$, it is possible to shortcut the path by connecting y_{k+1} and z_k. Suppose that x_{i_k}, x_{j_k}, $x_{i_{k+1}}$ and $x_{j_{k+1}}$ appear in this order along γ from u_s.

Refer to Fig. 7(b). The curve $\gamma(x_{i_{k+1}}, y_{k+1}) \cup y_{k+1}z_k \cup \gamma(z_k, x_{j_k})$ forms a loop with the horizontal segment $x_{i_{k+1}}x_{j_k}$ that encloses the anchor a. Thus, $|\gamma(x_{i_{k+1}}, y_{k+1})| + |z_k y_{k+1}| + |\gamma(z_k, x_{j_k})| \geq |ay_{k+1}|$, which implies that

$$|\gamma(x_{i_{k+1}}, y_{k+1})| + |\gamma(z_k, x_{j_k})| \geq |ay_k|. \tag{1}$$

Recall the assumption that x_{i_k}, x_{j_k}, $x_{i_{k+1}}$ and $x_{j_{k+1}}$ appear in this order along γ from u_s. Thus,

$$\gamma(z_k, x_{j_k}) \cup \gamma(x_{i_{k+1}}, y_{k+1}) \ \subset\ \gamma(z_k, y_{k+1}). \tag{2}$$

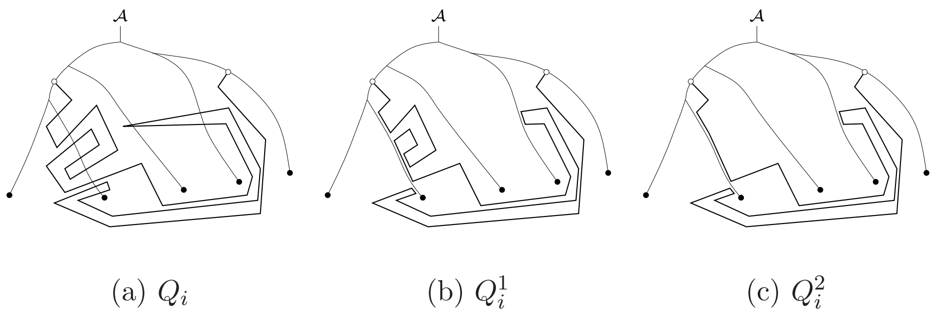
The segment $y_{k+1}z_k$ is sandwiched between $C_{i_{k+1}j_{k+1}}$ and $C_{i_k j_k}$, which enclose the same set of anchors. Thus, $y_{k+1}z_k$ does not intersect any obstacle and Lemma 5.2(iv) is applicable. It implies that the cost of $\gamma(z_k, y_{k+1})$ is at most $\mathrm{cost}_{\mathcal{T}}(z_k y_{k+1}) \leq \rho|z_k y_{k+1}|$. Combining this inequality with (1) and (2), we obtain $|z_k y_{k+1}| \geq |ay_k|/\rho$. By a similar argument, we obtain the same inequality when $x_{i_{k+1}}$, $x_{j_{k+1}}$, x_{i_k}, and x_{j_k} appear in this order along γ from u_s. Hence, $|ay_{k+1}| = |ay_k| + |y_k y_{k+1}| \geq |ay_k| + |z_k y_{k+1}| \geq (1+1/\rho)|ay_k|$, which is at least $(1+1/\rho)^{k-1}|ay_2|$ by induction. □

We claim that $|ay_2| \geq \delta_{\mathrm{fen}}$. If not, Lemma 5.2(iii) implies that y_2 lies on an obstacle, which must be sandwiched between $C_{i_1 j_1}$ and $C_{i_2 j_2}$ or between $C_{i_2 j_2}$ and $C_{i_3 j_3}$. This is a contradiction because the cycles in the left-group enclose the same set of anchors. By Proposition 2, $(1+1/\rho)^{m-2}\delta_{\mathrm{fen}} \leq (1+1/\rho)^{m-2}|ay_2| \leq |ay_m|$. As ay_m lies inside the box $\mathcal{B}$ enclosing the subdivision $\mathcal{S}$ and the graph H_{fen}, $|ay_m|$ is at most the length of the diagonal of $\mathcal{B}$, which is $O(\rho L_{st})$. Thus, $m = O(\frac{1}{\log(1+1/\rho)} \log \frac{\rho L_{st}}{\delta_{\mathrm{fen}}}) = O(\rho \log \frac{\rho L_{st}}{\delta_{\mathrm{fen}}}) = O(\rho \log \frac{\rho k n}{\varepsilon})$ as $\delta_{\mathrm{fen}} = \varepsilon L_{st}/\Theta(\rho k n)^{O(1)}$.

Since there are at most $2h$ groups of cycles and h paths in $\mathcal{A}$, the number of canonical crossings between $\mathcal{A}$ and ℓ is $O(\rho h^2 \log \frac{\rho k n}{\varepsilon})$. As P has k segments, the canonical crossing sequence length becomes $O(\rho h^2 k \log \frac{\rho k n}{\varepsilon})$. □

6 Rerouting along $\mathcal{A}$

Our algorithm will run $|S_P|+1$ rounds of shortest path computation starting from the source s in a subgraph of H_ε. In each round, $\mathcal{A}$ is treated as an obstacle. At the end of each round, we



(a) Q_i (b) Q_i^1 (c) Q_i^2

Figure 8: $Q_i \to Q_i^1 \to Q_i^2$.

cross $\mathcal{A}$ in a way compatible with the remaining symbols in S_P. We reroute the optimum along $\mathcal{A}$ in the analysis so that the structure of the rerouted optimum is similar to ours. So our path is as short as the rerouted optimum. It is thus important to to bound the rerouting error. In this section, we explain the rerouting for a path Q in H_ε with canonical crossing sequence S_Q.

Split Q into a concatenation of subpaths and edges $Q_1 \cdot u_1v_1 \cdot Q_2 \cdot u_2v_2 \cdots$ such that each subpath Q_i has no canonical crossing and each u_iv_i crosses $\mathcal{A}$ at one or more canonical crossings in S_Q. In the following, we describe successive conversions of Q_i, $Q_i \to Q_i^1 \to Q_i^2 \to Q_i^3$, such that Q_i^3 and Q_i are homotopic. All crossings between Q_i and $\mathcal{A}$ are cancellable and canceling two adjacent symbols can be implemented by rerouting Q_i along $\mathcal{A}$ as illustrated by the conversion from Fig. 8(a) to Fig. 8(b). After doing all cancellations, we get a path Q_i^1 that does not cross $\mathcal{A}$. For each path γ in $\mathcal{A}$ from u_s to some anchor, we shortcut Q_i^1 along the right side of γ between the first and last contact points of Q_i^1 (in order along Q_i^1) on the right side of γ, and we shortcut analogously along the left side of γ. The resulting path is Q_i^2. This step is illustrated by the conversion from Fig. 8(b) to Fig. 8(c).

Finally, we convert Q_i^2 to a homotopic path Q_i^3 in H_ε as follows. The path Q_i^2 consists of several disjoint maximal subpaths that are delimited by some vertices of H_ε. (The remaining nodes of Q_i^2 are turns that Q_i^2 make at the edges of $\mathcal{A}$.) Each such maximal subpath lies in a region of $\mathcal{S}$ with the subpath endpoints on the boundary of this region. We straighten Q_i^2 by replacing each such subpath by the edge between the subpath endpoints. This produces Q_i^3.

Lemma 6.1 *Let Q be a path in H_ε with canonical crossing sequence S_Q. We can convert Q to a homotopic path Q^3 in H_ε such that:*

(i) *Q^3 is the concatenation $Q_1^3 \cdot u_1v_1 \cdot Q_2^3 \cdot u_2v_2 \cdots$ such that Q_i^3 does not cross $\mathcal{A}$ and the edge u_iv_i crosses $\mathcal{A}$ at one or more canonical crossings in S_Q.*

(ii) *$\mathrm{cost}_{\mathcal{S}}(Q^3) \leq \mathrm{cost}_{\mathcal{S}}(Q) + O(\rho h^2 \delta_{\mathrm{fen}} |S_Q|)$.*

Proof. Consider (i). Since we preserve the edges u_iv_i that cross $\mathcal{A}$ at one or more canonical crossings in S_Q, it suffices to show that Q_i^3 does not cross $\mathcal{A}$. By construction, Q_i^2 does not cross $\mathcal{A}$. When we convert Q_i^2 to Q_i^3, we shortcut maximal subpaths delimited by some vertices of H_ε. Let $(w, \ldots, w')$ be one such subpath. Note that $(w, \ldots, w')$ lies in some region f of $\mathcal{S}$, the intermediate nodes of $(w, \ldots, w')$ lie in the interior of f, and w and w' are vertices of H_ε on the boundary of f. If an edge e of $\mathcal{A}$ crosses the segment ww', then e is a chord of f and e must cross $(w, \ldots, w')$ too. This is impossible as Q_i^2 does not cross $\mathcal{A}$. Hence, the correctness of (i) follows.

Consider the example in Fig. 9, where γ_j is a path in $\mathcal{A}$ from u_s to an anchor a_j. To bound the rerouting error, it is instructive to view the conversion from Q_i^2 to Q_i^3 as swapping some

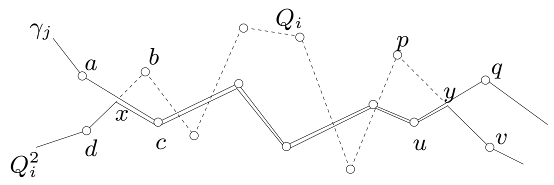


Figure 9: The dashed polyline denotes $Q_i \setminus Q_i^2$.

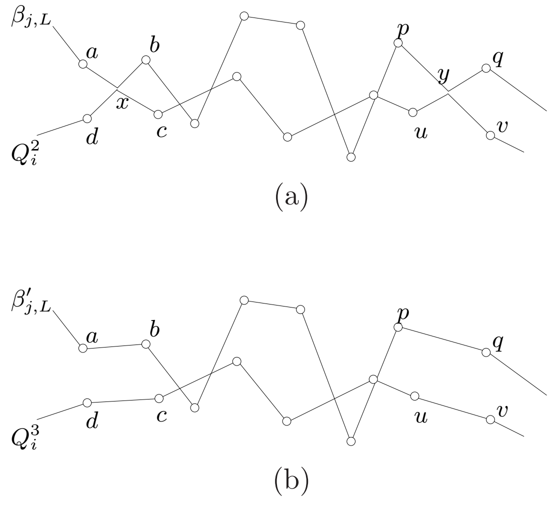


Figure 10: In (a), subpaths between Q_i and γ_j for $j \in [1,h]$ are swapped to yield Q_i^2 and $\beta_{j,L}$ for $j \in [1,h]$. In (b), Q_i^2 and $\beta_{j,L}$ for $j \in [1,h]$ are straightened to yield Q_i^3 and $\beta'_{j,L}$ for $j \in [1,h]$.

subpaths of Q_i and γ_j for $j \in [1,h]$ from Fig. 9 to Fig. 10(a) to yield Q_i^2 and $\beta_{j,L}$ for $j \in [1,h]$, followed by path straightening in Fig. 10(b) to yield Q_i^3 and $\beta'_{j,L}$ for $j \in [1,h]$. The straightening of Q_i^2 is the conversion from Q_i^2 to Q_i^3 as explained previously. We apply the same straightening to obtain $\beta'_{j,L}$ from $\beta_{j,L}$. We label $\beta_{j,L}$ and $\beta'_{j,L}$ with a subscript L to signify the swapping done for the overlap between Q_i^2 and γ_j on the left side of γ_j. In general, Q_i^2 may also overlap with γ_j on its right side and the swapping and straightening would produce another $\beta_{j,R}$ and $\beta'_{j,R}$. Therefore,

$$\begin{aligned}
&\sum_{j=1}^{h}(\mathrm{cost}_{\mathcal{S}}(\beta'_{j,L}) + \mathrm{cost}_{\mathcal{S}}(\beta'_{j,R})) + \mathrm{cost}_{\mathcal{S}}(Q_i^3) \\
\le\ &\sum_{j=1}^{h}(\mathrm{cost}_{\mathcal{S}}(\beta_{j,L}) + \mathrm{cost}_{\mathcal{S}}(\beta_{j,R})) + \mathrm{cost}_{\mathcal{S}}(Q_i^2) \\
=\ &\sum_{j=1}^{h} 2\,\mathrm{cost}_{\mathcal{S}}(\gamma_j) + \mathrm{cost}_{\mathcal{S}}(Q_i).
\end{aligned}$$

By Lemma 5.2(ii), we have

$$\sum_{j=1}^{h} 2\,\mathrm{cost}_{\mathcal{S}}(\gamma_j) \ \le\ \sum_{j=1}^{h}(\mathrm{cost}_{\mathcal{S}}(\beta'_{j,L}) + \mathrm{cost}_{\mathcal{S}}(\beta'_{j,R})) + O(\rho h^2 \delta_{\mathrm{fen}}).$$

It follows that $\mathrm{cost}_{\mathcal{S}}(Q_i^3) \le \mathrm{cost}_{\mathcal{S}}\, Q_i + O(\rho h^2 \delta_{\mathrm{fen}}) = \mathrm{cost}_{\mathcal{T}}\, Q_i + O(\rho h^2 \delta_{\mathrm{fen}})$ and hence

$$\mathrm{cost}_{\mathcal{S}}(Q^3) \ \le\ \mathrm{cost}_{\mathcal{T}}\, Q + O(\rho h^2 \delta_{\mathrm{fen}} |S_Q|).$$

□

7 Main algorithm

First, we construct $\mathcal{A}$ using Lemma 5.2 and superimpose it on $\mathcal{S}$ in time linear in the size of $\mathcal{A}$. Since $\mathcal{A}$ bends only at vertices of H_ε on the edges of $\mathcal{S}$, no new nodes are generated, so the overlay has size $O(\frac{h}{\varepsilon} n \log \frac{1}{\varepsilon})$ by Lemma 5.2.

Next, we obtain H_{alg} from H_ε as follows. The neighborhood of each vertex in $\mathcal{A}$ is cut into 2 to h connected components by $\mathcal{A}$. We split each vertex in $\mathcal{A}$ and place one copy in each connected component. The copies of the same vertex have a natural circular order around the vertex. We connect two copies of a vertex by a *dummy edge* if they are adjacent in the circular order. In total, we add $O(|\mathcal{A}|)$ copies and $O(|\mathcal{A}|)$ dummy edges. Then we obtain H_{alg} deleting any edge of H_ε that intersecting $\mathcal{A}$ and all the dummy edges. (We do not need the dummy edges in H_{alg}, but we need to use them to cross $\mathcal{A}$ after each round of the shortest path computation.)

Take a (triangular) region f of $\mathcal{S}$. Each edge of $\mathcal{A}$ in f is a chord. Since $\mathcal{A}$ does not intersect itself, its edges divide f into several zones. We partition the vertices of H_ε on the boundary of f according to the zones that they belong to. Vertices in the same zone are given the same zone id. This can be done in $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ time by walking around the boundary of f once. Then, an edge pq of H_ε in f intersects an edge of $\mathcal{A}$ iff the zone ids of p and q are different. Hence, it takes $O(1)$ time to check one edge and hence H_{alg} can be constructed in $O(\frac{h}{\varepsilon^2} n \log^2 \frac{1}{\varepsilon})$ time.

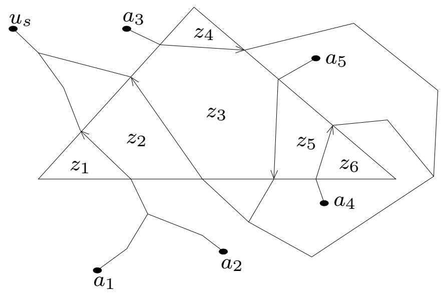


Figure 11: The division of a region into zones.

We intersect $\mathcal{A}$ with P by brute force to find its canonical crossing sequence S_P in $O(\frac{h}{\varepsilon}kn\log\frac{1}{\varepsilon})$ time. We run $|S_P|+1$ rounds of shortest path computation in H_{alg}. In the initialization, for each vertex p of H_{alg}, we set a vector $p[i]=\infty$ for $i\in[0,|S_P|]$. The entry $p[i]$ is supposed to store the shortest path cost in H_{alg} from s to p subject to the constraint that the canonical crossing sequence of the path consists of the first i symbols in S_P.

In the first round, we set $s[0]=0$ and compute shortest paths in H_{alg} from s to all other vertices. The shortest path cost of a vertex p is stored at $p[0]$ during this round. For each edge of H_ε and each dummy edge pq, let σ_{pq} denote the canonical crossing sequence of pq. At the end of the first round, for any edge of H_ε and any dummy edge pq such that σ_{pq} is a prefix of S_P, we update $q[|\sigma_{pq}|]$ to be $\min\{q[|\sigma_{pq}|], p[0]+\text{cost}_{\mathcal{S}}(pq)\}$. The cost of pq is 0 if it is a dummy edge.

In general, for $j\geq 1$, the $(j+1)$th round begins with selecting vertices v of H_{alg} such that $v[j]\neq\infty$ and run Dijkstra's algorithm in H_{alg} from these vertices as multiple sources. This is akin to the computation of a weighted Voronoi diagram. The shortest path cost of a vertex p is stored at $p[j]$ during this round. Similarly, at the end of the $(j+1)$th round, we find all the edges of H_ε and dummy edges pq such that σ_{pq} matches S_P from the $(j+1)$th to the $(j+|\sigma_{pq}|)$th symbols, and update $q[j+|\sigma_{pq}|]$. That is, $q[j+|\sigma_{pq}|]=\min\{q[j+|\sigma_{pq}|],\ p[j]+\text{cost}_{\mathcal{S}}(pq)\}$. The final shortest path cost from s to t is stored at $t[|S_P|]$.

At the end of each round, we have to find all eligible edges to update the entries $q[\cdot]$'s. A dummy edge has a canonical crossing sequence with length $O(h)$. For an edge pq in H_ε, it lies inside a region. Such an edge pq may cross $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ segments in $\mathcal{A}$ and crossing one such segment corresponds to gaining up to $O(h)$ symbols. It means that $|\sigma_{pq}|=O(\frac{h}{\varepsilon}\log\frac{1}{\varepsilon})$. It is time-consuming to check every edge in H_ε and every dummy edge to identify the eligible ones. Fortunately, we can do it more efficiently by preprocessing.

Lemma 7.1 *We can build a data structure in $O(|S_P|\frac{h}{\varepsilon^2}n\log^2\frac{1}{\varepsilon})$ time so as to report the eligible edges in time proportional to their number at the end of each round.*

Proof. We first consider the regular edges in H_ε. Each region f of $\mathcal{S}$ is split by $\mathcal{A}$ into disjoint *zones*, each being a simple polygon. There are $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ zones in f because each zone contains some vertex of H_ε in f. We can build a dual tree T_f to model the adjacency of the zones in f. Each node of T_f represents a zone and two zones are connected in T_f if they are adjacent. Building T_f takes $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ time. Fig. 11 shows the zones in a region f.

For each zone z in f, root T_f at z and attach z to a dummy parent. Then we expand each edge between a zone z' and its child zone z'' into $O(h)$ edges, each containing one symbol that is gained by going from zone z' to zone z''. Denote by $T_{f,z}$ the resulting rooted tree. It has $O(\frac{h}{\varepsilon}\log\frac{1}{\varepsilon})$ size. Fig.12 shows T_{f,z_1} for the example in Fig. 11. In $T_{f,z}$, we can read off the

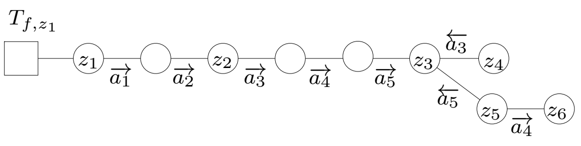


Figure 12: T_{f,z_1}.

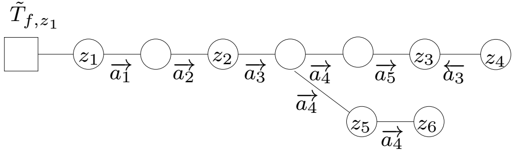


Figure 13: $\tilde{T}_{f,z_1}$.

symbol sequence from any vertex p in zone z to any vertex q in another zone. But this sequence may not be canonical. To obtain canonical sequences, we perform a BFS of $T_{f,z}$ while modifying $T_{f,z}$ on the fly. Suppose that we visit a node x from its parent x' and let ϕ be the symbol on the edge xx'. The path from z to x' gives a sequence of symbols $\phi_1, \phi_2, \cdots, \phi_{i-1}, \phi_i$. If ϕ does not cancel ϕ_i, we just continue with the BFS. If ϕ cancels ϕ_i, we detach x from x', make x a child of the grandparent x^* of x', and set ϕ_{i-1} to be the symbol on the edge xx^*. Then, we continue with the BFS. Basically, we are reducing the crossing sequences while generating them. Let $\tilde{T}_{f,z}$ denote the final rooted tree converted from $T_{f,z}$, which is a prefix tree of canonical crossing sequences from z to all other zones in f. Fig. 13 shows $\tilde{T}_{f,z_1}$ obtained from T_{f,z_1} in Fig. 12.

Then, we find in S_P the occurrences of all canonical crossing sequences in $\tilde{T}_{f,z}$ as follows. We construct a suffix tree for S_P in $O(|S_P|)$ time [6]. Next, we traverse $\tilde{T}_{f,z}$ in a depth-first manner while navigating up and down the suffix tree correspondingly. It takes $O(|\tilde{T}_{f,z}| + |S_P|)$ time to find for each sequence σ starting from z in $\tilde{T}_{f,z}$ the subtree of the suffix tree that stores exactly the suffixes of S_P beginning with σ, which can then be traversed to output all occurrences of σ. There are $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ sequences in $\tilde{T}_{f,z}$ and each appears at most $|S_P|$ times in S_P. Therefore, the total time to find all the occurrences of the sequences in $\tilde{T}_{f,z}$ in S_P is $O(|\tilde{T}_{f,z}| + |S_P| + |S_P|\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}) = O(|S_P|\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$. Repeating for all zones in all regions gives a running time of $O(|S_P|\frac{h}{\varepsilon^2}n\log^2\frac{1}{\varepsilon})$. We use $|S_P|$ lists to store the results. The jth list contains all zone pairs (z, z') such that the canonical crossing sequence from z to z' matches S_P at the jth position. We also build a two-dimensional array $E[\cdot,\cdot]$ indexed by zone pairs such that $E[z, z']$ stores the vertex pair (p, q) where $p \in z$ and $q \in z'$. This takes $O(\frac{h}{\varepsilon^2}n\log^2\frac{1}{\varepsilon})$ time. At the end of the jth round, for each zone pair (z, z') in the jth list, we report all vertex pairs in $E[z, z']$.

Consider the dummy edges. For each dummy edge pq, we search for σ_{pq} in the suffix tree of S_P. We can find all the occurrences of σ_{pq} in S_P in $O(|S_P| + h)$ time, since $|\sigma_{pq}| = O(h)$. In total, it takes $O(|S_P|\frac{h}{\varepsilon}n\log\frac{1}{\varepsilon})$ time. We again use $|S_P|$ lists to store the results. The jth list contains all the dummy edges such that σ_{pq} matches S_P at position j. Then, at the end of round j, we can report all the eligible dummy edges by just outputting the jth list. $\square$

Theorem 1 *Let P be a polygonal path of k segments in a weighted subdivision $\mathcal{T}$ with h obstacles and n vertices. For any $\varepsilon \in (0, 1)$, we can compute a $(1+\varepsilon)$-approximate shortest path homotopic to P in $O(\frac{h^3}{\varepsilon^2}kn\,\mathrm{polylog}(k, n, \frac{1}{\varepsilon}))$ time, where the hidden constant depends on ρ and some geometric parameters.*

Proof. Let O be the shortest path in $\mathcal{T}$ homotopic to P. Using the analysis of Sun and Reif [17], the path O can be snapped to a $1+\varepsilon$ homotopic approximation O' in H_ε. Then, O' can be converted to a path O'' that satisfies Lemma 6.1. Our algorithm returns a path cost at most $\mathrm{cost}_{\mathcal{S}}(O'') \leq \mathrm{cost}_{\mathcal{S}}(O') + O(\rho h^2 \delta_{\mathrm{fen}} |S_P|) \leq (1+\varepsilon)\,\mathrm{cost}_{\mathcal{S}}(O) + O(\rho h^2 \delta_{\mathrm{fen}} |S_P|)$. If we set $\delta_{\mathrm{fen}} = \varepsilon L_{st}/(\rho h^2 |S_P|)$, the additive term becomes $O(\varepsilon L_{st}) = O(\varepsilon\, \mathrm{cost}_{\mathcal{S}}(O))$. Hence, our path cost is $(1+O(\varepsilon))\,\mathrm{cost}_{\mathcal{S}}(O)$. The factor $1+O(\varepsilon)$ can be made $1+\varepsilon$ by manipulating the constants.

By Lemma 7.1, the preprocessing takes $O(|S_P|\frac{h}{\varepsilon^2}\log^2\frac{1}{\varepsilon})$ time. Consider the shortest path computation. Since H_{alg} has $O(\frac{h}{\varepsilon}n\log\frac{1}{\varepsilon})$ vertices and $O(\frac{h}{\varepsilon^2}n\log^2\frac{1}{\varepsilon})$ edges, one round of Dijkstra takes $O(\frac{h}{\varepsilon^2}n\,\mathrm{polylog}(k,n,\frac{1}{\varepsilon}))$ time. We use eligible edges pq to update the entries $q[i]$'s at the end of each round, which takes $O(\frac{h}{\varepsilon^2}n\log^2\frac{1}{\varepsilon})$ time. Hence, the total running time of all rounds is $O(|S_P|\frac{h}{\varepsilon^2}n\,\mathrm{polylog}(k,n,\frac{1}{\varepsilon})) = O(\frac{h^3}{\varepsilon^2}kn\,\mathrm{polylog}(k,n,\frac{1}{\varepsilon}))$, where the hidden constant depends on ρ and some geometric parameters. □

Acknowledgment

Research of Cheng and Jin are partly supported by the Research Grant Council, Hong Kong, China (project no. 612107).

References

[1] L. Aleksandrov, A. Maheshwari, and J.-R. Sack. Determining approximate shortest paths on weighted polyhedral surfaces. *Journal of the ACM*, 52(1):25–53, 2005.

[2] S. Bespamyatnikh. Computing homotopic shortest paths in the plane. *Journal of Algorithms*, 49 (2003), 284–303.

[3] S. Cabello, Y. Liu, A. Mantler and J. Snoeyink Testing Homotopy for Paths in the Plane. *Discrete and Computational Geometry*, 31 (2004), 61–81.

[4] R. Cole and A. Siegel. River routing every which way, but loose. *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*, 1982, 65–73.

[5] A. Efrat, S.G. Kobourov and A. Lubiw. Computing homotopic shortest paths efficiently. *Computational Geometry: Theory and Applications*, 35 (2006), 162–172.

[6] M. Farach. Optimal suffix tree construction with large alphabets. *Proc. 38th Annu. Sympos. Found. Comput. Sci.*, 1997, 137–143.

[7] K.D. Forbus, J. Uhser and V. Chapman. Qualitative spatial reasoning about sketch maps. *AI Magazine*, 24 (2004), 61–72.

[8] M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34 (1987), 596–615.

[9] S. Gao, M. Jerrum, M. Kaufmann, K. Kehlhorn, W. Rülling, and C. Storb. On continuous homotopic one layer routing. *Proceedings of the 4th Annual Symposium on Computational Geometry*, 1998, 392–402.

[10] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4 (1994), 63–98.

[11] J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28 (1999), 2215–2256.

[12] J.R. Munkres. *Topology: a first course.* Prentice Hall, 1975.

[13] M. Kaufmann and K. Mehlhorn. On local routing of two-terminal nets. *J. Comb. Theory*, Ser. B, 55(1992), 33–72.

[14] M. Lanthier, A. Maheshwari, and J.-R. Sack. Shortest anisotropic paths on terrains. *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, pages 524–533, 1999.

[15] C.E. Leiserson and F.M. Maley. Algorithms for routing and testing routability of planar VLSI layouts. *Proceedings of the 17th Annual Symposium on Theory of Computing*, 1985, 69–78.

[16] J. Mitchell and C. Papadimitriou. The weighted region problem: Finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38(1):18–73, 1991.

[17] Z. Sun and J. Reif. On finding approximate optimal paths in weighted regions. *Journal of Algorithms*, 58(1):1–32, 2006.

A Proof of Lemma 5.1

Let $f, g : [0,1] \to |\mathcal{T}|$ be two paths from s to t. We omit the analysis for the backward direction because it is easy to deform f to g if they have the same canonical crossing sequence. Assume that f and g are homotopic. Without loss of generality, we can assume that the crossing sequences of f and g are canonical. We show that their crossing sequences are the same using covering space and covering map in topology [12].

We perturb the interior of every α_i so that the perturbed paths are physically interior-disjoint. Specifically, let β_i denote the slightly displaced α_i (a_i is still the endpoint of β_i), we ensure that $\text{int}(\beta_i) \cap \text{int}(\beta_j) = \emptyset$, a_i is the only contact point between β_i and the obstacles, and the left-to-right orders of the α_i's and the β_i's along the vertical ray in $\mathcal{A}$ are the same. Hence, the path f crosses the β_i's in the same order as the α_i's. So does g.

Recall that H is $\mathbb{R}^2$ minus the obstacles with anchors. So H is an open set and, in particular, it does not contain any anchor. We cut H open by duplicating the interiors of $\beta_1, \beta_2, \ldots$. Denote the cut open H by S. There are two copies of the interior of β_i in S, and we denote the left copy by $\beta_{i,0}$ and the right copy by $\beta_{i,1}$ (with respect to the direction of α_i). We create infinitely many labelled copies of S, denoted by $S(\sigma_k, k)$, where k is any non-negative integer and σ_k is a sequence of k symbols in $\{\beta_{1,0}, \beta_{1,1}, \beta_{2,0}, \beta_{2,1}, \ldots\}$, repetitions allowed. We glue $S(\sigma_k, k)$ and $S(\sigma_{k+1}, k+1)$ together, whenever σ_{k+1} is equal to σ_k appended with some $\beta_{i,\delta}$, by identifying $\beta_{i,\delta}$ in $S(\sigma_k, k)$ with $\beta_{i,1-\delta}$ in $S(\sigma_{k+1}, k+1)$. Denote the resulting set by $\widetilde{H}$, which is like a stack of copies of S glued together.

Define $\pi : \widetilde{H} \to H$ to be the "vertical projection" of points in $\widetilde{H}$ onto H. We claim that $\widetilde{H}$ is a covering space and π is a covering map. It requires showing that every point $x \in H$ has an open neighborhood B_x such that $\pi^{-1}(B_x)$ consists of disjoint sets homeomorphic to B_x. If x does not lie on any β_i, we can choose a small enough B_x that avoids all β_i's. Then,

$\pi^{-1}(B_x)$ consists of disjoint copies of B_x, one from each $S(\sigma_k, k)$ in $\tilde{H}$. If x lies on some β_i, then $x \in \mathrm{int}(\beta_i)$ because H contains no anchor and we can choose a small enough B_x that avoids other β_j's. Then $\pi^{-1}(B_x)$ consists of disjoint copies of B_x, each straddling some $S(\sigma_k, k)$ and $S(\sigma_{k+1}, k+1)$ in $\tilde{H}$ such that σ_k is a prefix of σ_{k+1}. This shows that $\tilde{H}$ is a covering space and π is a covering map.

We also claim that $\widetilde{H}$ contains no cycle of copies of S. If not, take the copy $S(\sigma_{k+1}, k+1)$ in a cycle such that σ_{k+1} is the longest. Then, $S(\sigma_{k+1}, k+1)$ is connected to two different copies $S(\sigma_k, k)$ and $S(\sigma'_k, k)$ in this cycle. So $\sigma_k \neq \sigma'_k$ but both σ_k and σ'_k are prefixes of σ_{k+1} with length k, an impossibility.

Let x_0 be the point in $S(\texttt{null}, 0)$ such that $\pi(x_0) = s$. By standard results in topology [12], since f and g are homotopic, there are unique paths $\tilde{f}, \tilde{g} : [0,1] \to \widetilde{H}$ such that $\tilde{f}(0) = \tilde{g}(0) = x_0$, $\pi \circ \tilde{f} = f$, $\pi \circ \tilde{g} = g$, and $\tilde{f}$ and $\tilde{g}$ are homotopic. So $\tilde{f}$ and $\tilde{g}$ have the same endpoints.

Trace the path $\tilde{f}$ in $\widetilde{H}$. We claim that if we leave some $S(\sigma_k, k)$ by crossing some $\beta_{i,\delta}$, we cannot reenter $S(\sigma_k, k)$. Because there is no cycle of copies of S in $\widetilde{H}$, the only way to reenter $S(\sigma_k, k)$ is to cross the same $\beta_{i,\delta}$ in reverse direction. If this can happen, then after leaving $S(\sigma_k, k)$ and before reentering $S(\sigma_k, k)$, we must enter and leave some $S(\sigma_{k'}, k')$, both by crossing the same $\beta_{j,\delta'}$. But then an occurrence of $\overleftarrow{a_j}$ is adjacent to an occurrence of $\overrightarrow{a_j}$ in the crossing sequence of f, contradicting its canonicity. This establishes our claim for $\tilde{f}$. A similar claim holds for $\tilde{g}$.

If the crossing sequences of f and g are different, then $\tilde{f}$ and $\tilde{g}$ must enter different copies of S at some point. Neither $\tilde{f}$ nor $\tilde{g}$ can visit the same copy of S twice as proved above. So $\tilde{f}$ and $\tilde{g}$ cannot converge later because $\widetilde{H}$ contains no cycle of copies of S. That is, $\tilde{f}$ and $\tilde{g}$ end in different copies of S. This is impossible because $\tilde{f}$ and $\tilde{g}$ are homotopic and they have the same endpoints.