Approximate Shortest Homotopic Paths in Weighted Regions*

Siu-Wing Cheng[1], Jiongxin Jin[1], Antoine Vigneron[2], and Yajun Wang[3]

[1] Department of Computer Science and Engineering, HKUST, Hong Kong
[2] INRA, UR 341 Mathématiques et Informatique Appliquées, Jouy-en-Josas, France
[3] Microsoft Research Asia, Beijing, China

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 (e.g. [8,11]). Forbus et al. [6] described a planning system in which a user makes a path sketch for vehicles or people and then the system generates the detailed optimized path homotopic to the sketch. 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.

The *weighted region* model is the first non-Euclidean cost model and there has been much work on it (e.g. [1,12,13]). 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 [12] runs in $O(n^8 \log \frac{nN\rho}{\varepsilon})$ time, where n is the number 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. 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 (e.g. [1,13]). Sun and Reif [13] 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. Aleksandrov et al. [1] achieved the best dependence on n

* The research of Cheng and Jin was supported by the Research Grant Council, Hong Kong, China (project no. 612107).

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. No result is known so far on the shortest homotopic path problem in weighted regions, although several results are known when the cost of a path is its length [2,4,9].

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 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 [13] 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 $|Q| \leq \operatorname{cost}_{\mathcal{T}}(Q) \leq \rho|Q|$.

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 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

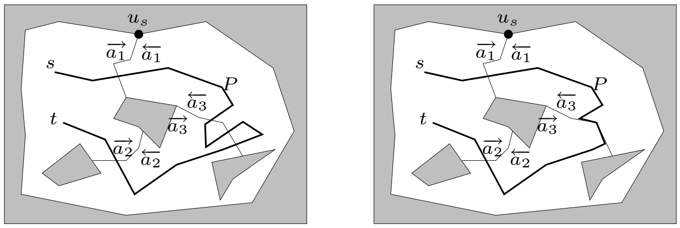

u_s
$\overrightarrow{a_1}$ $\overleftarrow{a_1}$
s
P
$\overleftarrow{a_3}$
$\overrightarrow{a_3}$
t
$\overrightarrow{a_2}$ $\overleftarrow{a_2}$
u_s
$\overrightarrow{a_1}$ $\overleftarrow{a_1}$
s
P
$\overleftarrow{a_3}$
$\overrightarrow{a_3}$
t
$\overrightarrow{a_2}$ $\overleftarrow{a_2}$

Fig. 1. The obstacles are shaded. After canceling one $\overleftarrow{a_3}$ and one $\overrightarrow{a_3}$, the path P becomes a new path P' that crosses the edge a_3 once.

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 $a_1, a_2, \ldots, a_h$. We choose each such edge a_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 a_i's). In Fig. 1, the trace is $\overrightarrow{a_1}\overleftarrow{a_3}\overrightarrow{a_3}\overleftarrow{a_3}\overleftarrow{a_2}$, where $\overleftarrow{a_i}$ means crossing a_i from right to left and $\overrightarrow{a_i}$ means crossing a_i from left to right. If $\overleftarrow{a_i}$ and $\overrightarrow{a_i}$ appears consecutively in the trace, we can cancel the two crossings. This corresponds to making a shortcut along a_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, the reduced trace S_P is a unique encoding of the homotopy of P. Indeed, two paths P and Q with the same endpoints are homotopic if and only if $S_P = S_Q$.

Since the tree edges are shortest paths, a shortcut (canceling two adjacent symbols in the trace) 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 S_P. The path P^* makes no redundant crossing. A natural approach to compute such a shortest path is as follows. Assume that S_P starts with $\overrightarrow{a_1}\overleftarrow{a_3}\overleftarrow{a_2}\ldots$. We know that P^* will first reach a_1 from the left. As we do not know at which point of a_1 it arrives, we can discretize a_1 by placing many vertices along it. For each of these vertices, we compute an approximate shortest path from s, treating the edges a_i of our tree as obstacles. As these paths avoid our spanning tree, they lie in a simply connected region. Thus, we do not need to consider their homotopy class and we can apply known algorithms for approximate shortest paths in weighted regions.

After crossing a_1, we know that P^* will reach a_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 a_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 a_3. We repeat this process for each symbol in S_P, 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. 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 S_P 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 as dictated by S_P. To conform to S_P, we have the 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 issue is that we have to keep S_P 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 S_P, 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 S_P. 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 [13]. 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 [7]. 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. [1] have a related construction with 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 a $(1+\varepsilon)$-approximate shortest homotopic path. A dense enough discretization is sufficient for this purpose. We will use another graph H_{fen} whose edges are contained in H_ε to compute the anchor tree for encoding the homotopy of P. This requires extra properties as we explain below. Although H_{alg} and H_{fen} serve different purposes, a $(1+\varepsilon)$-approximate shortest homotopic path has to interact with the anchor tree, 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 keeps short the canonical crossing sequence of P.

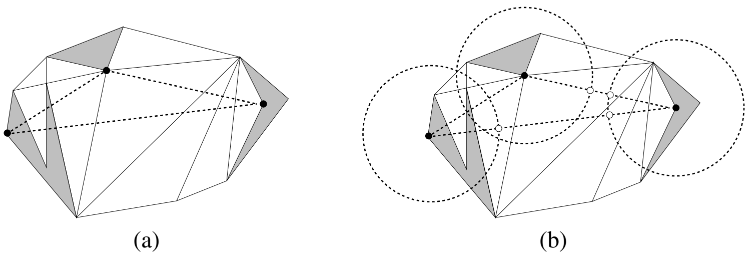


Fig. 2. The obstacles are shaded. We ignore the box B for simplicity. In (a), the black dots denote the anchors and the dashed segments form the anchor triangulation. In (b), the circles have radii δ_{fen} and the white dots are the extra vertices inserted.

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 anchor triangulation edge may be split by the obstacles and the edges of $B \cap \mathcal{T}$ into several edges in $\mathcal{T}'$. 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 need to prevent any path in the anchor tree from spiraling around the obstacles in order to keep short the canonical crossing sequence of P. 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. Either this subset consists of two segments ux and vy or it is the 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 Theorem 1, our main result.

The subdivision $\mathcal{S}$ is the refinement of $\mathcal{T}'$ so that all regions become triangles. W.l.o.g., we assume that $\mathcal{S}$ is connected. It has $O(hn)$ vertices and $O(hn)$ edges. We construct 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 vertex in $\mathcal{S}$ with the largest y-coordinate. 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 $a_1, a_2, \dots, a_h$ be the anchors in $\mathcal{A}$. Let α_i denote the directed tree path in $\mathcal{A}$ from a_i to vertical infinity. Although the paths $\alpha_1, \alpha_2, \dots$ 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

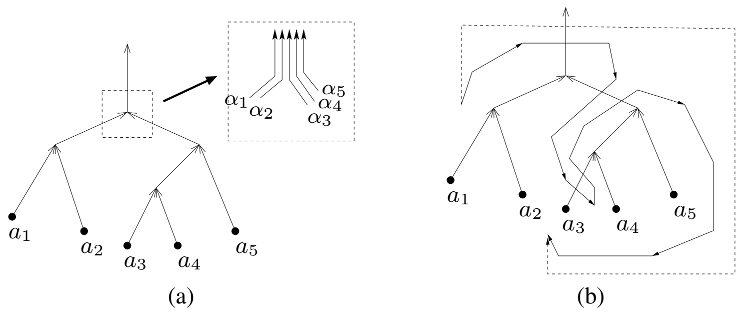


Fig. 3. In (b), the crossing sequence of the solid path is $\overrightarrow{a_1}\overrightarrow{a_2}\overrightarrow{a_3}\overrightarrow{a_4}\overrightarrow{a_5}\overleftarrow{a_5}\overleftarrow{a_4}\overleftarrow{a_3}\overrightarrow{a_3}\overleftarrow{a_3}\overrightarrow{a_3}\overrightarrow{a_4}\overrightarrow{a_5}$. 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.

not pass over any obstacle. Repeating until no other symbol can be deleted gives the unique *canonical crossing sequence* as implied by Lemma 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 [10].

Lemma 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, its subset 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 as that edge. Notice that one can still pass through a fence at its endpoints. We use H_{fen} to denote the resulting graph. Note that each edge in H_{fen} coincides with an edge 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 2. *$\mathcal{A}$ 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.*

(i) $\text{cost}_{\mathcal{T}}(\gamma_i) = O(\rho^2 n L_{st})$.
(ii) *The subpath of γ_i between any two nodes p and q has cost at most $d_{pq} + O(\rho h \delta_{\text{fen}})$, 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 $|vy| < \delta_{\text{fen}}$, then y lies on an obstacle.*

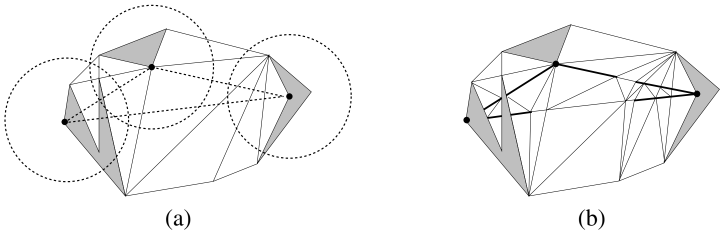


Fig. 4. The shaded regions are obstacles. We ignore the box B for simplicity. In (a), the black dots denote the anchors, the dashed segments form the anchor triangulation, and the dashed circles have radii δ_{fen}. In (b), the fences are shown as bold segments and the refined subdivision is $\mathcal{S}$.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

(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* $\text{cost}_{\mathcal{T}}(xy)$.

A key property of the anchor tree is that it ensures that the crossing sequence of P has low dependence on n and ε.

Lemma 3. *The canonical crossing sequence S_P of P has length $O(\rho h^2 k \log \frac{\rho k n}{\varepsilon})$.*

Proof. (Sketch.) 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 segment ℓ in P 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 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. 5. This eliminates the crossing x, x', or both. The deformed ℓ is homotopic to ℓ because the deformation does not pass over any obstacle (no obstacle lies inside the cycle). The deformed ℓ has no new crossing with $\mathcal{A}$ because xx' is replaced by a curve next to a subpath in $\mathcal{A}$. Also, the deformed ℓ does not cross itself because the choices of x and x' ensure that γ does not cross ℓ between x and x'. We repeat until no more

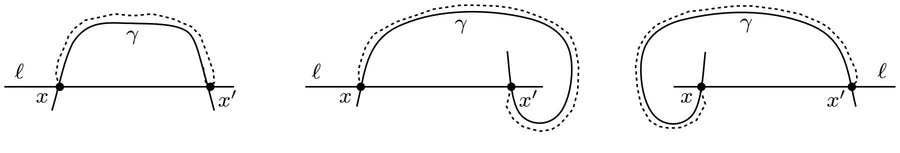


Fig. 5. Morph xx' to follow the dashed curve. This eliminates the crossings x and x' on the left, x in the middle, and x' on the right.

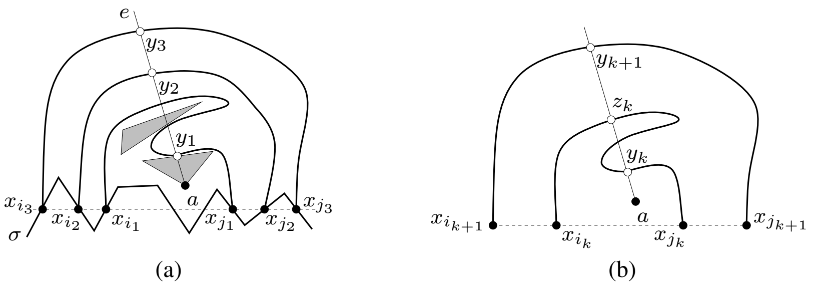


Fig. 6. (a) The shaded triangles denote the obstacles; the dashed line denotes ℓ; the polygonal curve denotes σ; the bold curves denote $\gamma(x_{i_1}, x_{j_1})$, $\gamma(x_{i_2}, x_{j_2})$ and $\gamma(x_{i_3}, x_{j_3})$. (b) z_k is the last crossing along e with $C_{i_k j_k}$ before y_{k+1}. As $z_k y_{k+1}$ avoids the obstacles, by Lemma 2(iv), $|y_k y_{k+1}| \geq |z_k y_{k+1}| \geq \text{cost}_{\mathcal{T}}(\gamma(z_k, y_{k+1}))/\rho \geq |ay_k|/\rho$. So $|ay_{k+1}| \geq (1 + 1/\rho)|ay_k|$.

crossings with $\mathcal{A}$ can be eliminated. Let σ be the final deformed ℓ. By induction, we can show that σ is homotopic to ℓ, and σ does not cross itself.

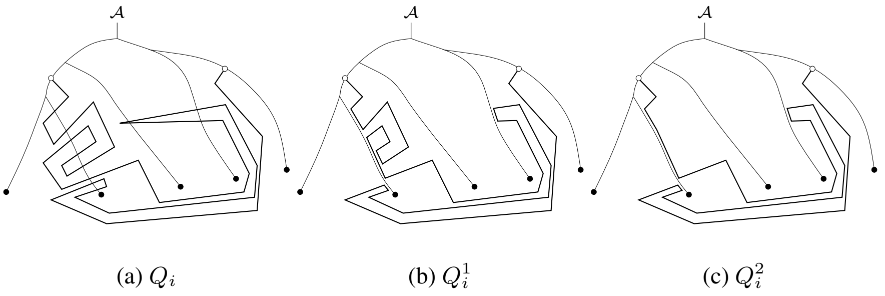
We define $\gamma(p, q)$ to be the subcurve of γ between two points p and q on it. The subcurve $\sigma(p, q)$ is similarly defined. Let $x_1, x_2, \ldots$ denote the crossings between γ and σ. All these crossings lie on ℓ by our deformation. Consider the set of cycles $\{C_{ij} = \sigma(x_i, x_j) \cup \gamma(x_i, x_j) : x_i$ and x_j 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 is simple and it must enclose some anchors. We cluster the cycles that enclose the same anchors. The 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. The two sets of anchors enclosed by two different cycles are either disjoint or one set is a subset of the other set. Therefore, there are at most $2h$ left- and right-groups. We show that a left-group has $O(\rho \log \frac{\rho kn}{\varepsilon})$ cycles as follows. The size of a right-group can be analyzed similarly.

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 $k \in [1, m-1]$. See Figure 6(a) for an example. We can show that $|ay_2| \geq \delta_{\text{fen}}$ and $|ay_k| \geq (1 + 1/\rho)^{k-2}|ay_2|$ for $k \in [2, m]$ by Lemma 2 and the optimality of γ. Figure 6(b) illustrates the idea of the proof. The details are omitted. We have $(1 + 1/\rho)^{m-2}\delta_{\text{fen}} \leq (1 + 1/\rho)^{m-2}|ay_2| \leq |ay_m|$, which is at most $|\gamma(x_{i_m}, x_{j_m})| \leq \text{cost}_{\mathcal{T}}(\gamma)$. Thus, $m = O\big(\frac{1}{\log(1+1/\rho)} \log \frac{\text{cost}_{\mathcal{T}}(\gamma)}{\delta_{\text{fen}}}\big) = O\big(\rho \log \frac{\rho kn}{\varepsilon}\big)$ as $\text{cost}_{\mathcal{T}}(\gamma) = O(\rho^2 n L_{st})$ and $\delta_{\text{fen}} = \varepsilon L_{st}/\Theta(\rho kn)^{O(1)}$. $\square$

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 cross $\mathcal{A}$ in a way compatible with the remaining symbols in S_P. We reroute the optimal path along $\mathcal{A}$ in the analysis so that the structure of the rerouted optimum is similar to ours. Our path is as short as the rerouted optimum by construction. It is thus important 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 edge u_iv_i crosses $\mathcal{A}$ at one or more canonical crossings in S_Q. We describe successive conversions from Q_i below: $Q_i \to Q_i^1 \to Q_i^2 \to Q_i^3$, such that the homotopy is preserved. All crossings between Q_i and $\mathcal{A}$ are cancellable. Canceling two adjacent symbols can be implemented by rerouting Q_i along $\mathcal{A}$. After doing all the cancellations, we get a path Q_i^1 that does not cross $\mathcal{A}$. This step is illustrated by the conversion from Fig. 7(a) to Fig. 7(b). 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 on the right side of γ, and shortcut analogously along the left side of γ. The resulting path is Q_i^2. This step is illustrated by the conversion from Fig. 7(b) to Fig. 7(c).



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

Fig. 7. $Q_i \to Q_i^1 \to Q_i^2$

Finally, we convert Q_i^2 to a homotopic path Q_i^3 in H_ε as follows. Assume for now that γ_j is the only path in $\mathcal{A}$ that overlaps with Q_i^2. We snap Q_i^2 to some nodes in $Q_i^2 \cap \gamma_j$ and we use the example in Fig. 8 to illustrate this step. The white dots denote some vertices of H_ε. The path Q_i^2 starts to follow γ_j at the first contact x until Q_i^2 leaves γ_j at y. To obtain Q_i^3, we replace dx and xc by dc, and we replace uy and yv by uv.

For bounding the rerouting error, it is instructive to view the whole process as a direct conversion from Q_i to Q_i^3 by swapping subpaths between γ_j and Q_i. That is, delete ac and qu from γ_j, delete bd and pv from Q_i, and then insert ab, cd, pq, and uv. The converted Q_i is Q_i^3 and the subpath of γ_j between c and u is replaced by the subpath of Q_i between b and p. Let β_j denote the converted γ_j. Analogous to the fact that given a convex quadrilateral, the total length of its diagonals is at least the total length of any two opposite sides, we can show that $\mathrm{cost}_{\mathcal{S}}(\gamma_j)+\mathrm{cost}_{\mathcal{S}}(Q_i) \geq \mathrm{cost}_{\mathcal{S}}(\beta_j)+\mathrm{cost}_{\mathcal{S}}(Q_i^3)$. By Lemma 2(ii), we have $\mathrm{cost}_{\mathcal{S}}(\gamma_j) \leq \mathrm{cost}_{\mathcal{T}}(\beta_j) + O(\rho h \delta_{\mathrm{fen}})$, which implies that $\mathrm{cost}_{\mathcal{T}}(Q_i^3) \leq \mathrm{cost}_{\mathcal{S}}(Q_i) + O(\rho h \delta_{\mathrm{fen}})$. So far, we have only considered the rerouting of Q_i^2 along one path in $\mathcal{A}$. Rerouting along all h paths gives $\mathrm{cost}_{\mathcal{S}}(Q_i^3) \leq \mathrm{cost}_{\mathcal{S}}(Q_i) + O(\rho h^2 \delta_{\mathrm{fen}})$.

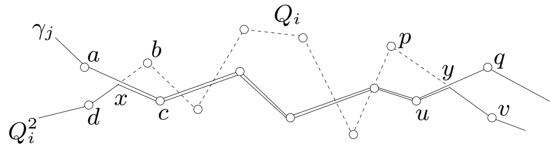


Fig. 8. The dashed polyline denotes $Q_i \setminus Q_i^2$

Lemma 4. *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 u_iv_i crosses $\mathcal{A}$ at one or more canonical crossings in S_Q.*

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

7 Main Algorithm

First, we construct $\mathcal{A}$ using Lemma 2 and superimpose it on $\mathcal{S}$. 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 2 and can be constructed in linear time.

Next, we obtain a subgraph H_{alg} of H_ε by deleting any edge pq that intersects $\mathcal{A}$. 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 $p[i]$ will 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. Let σ_{pq} denote the canonical crossing sequence of the segment pq. At the end of the round, for any edge pq of H_ε such that σ_{pq} is a prefix of S_P, we update $q[|\sigma_{pq}|]$ to be $\min\{q[|\sigma_{pq}|], p[0] + \mathrm{cost}_{\mathcal{S}}(pq)\}$. In general, the jth round begins with selecting vertices v of H_{alg} such that $v[j-1] \neq \infty$ 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 $p[j-1]$ during this round. Similarly, at the end of the jth round, we find all edges pq of H_ε such that σ_{pq} matches S_P from the jth to the $(j + |\sigma_{pq}| - 1)$th symbols, and update $q[j + |\sigma_{pq}| - 1]$. That is, $q[j + |\sigma_{pq}| - 1] = \min\{q[j + |\sigma_{pq}| - 1], p[j-1] + \mathrm{cost}_{\mathcal{S}}(pq)\}$. The final shortest path cost is stored at $t[|S_P|]$.

At the end of each round, we have to find all eligible edges in H_ε to update the entries $q[i]$'s. Each edge pq in H_ε lies inside a region. It 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. Fortunately, we can do it more efficiently by preprocessing.

Lemma 5. *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 H_ε in time proportional to their number at the end of each round.*

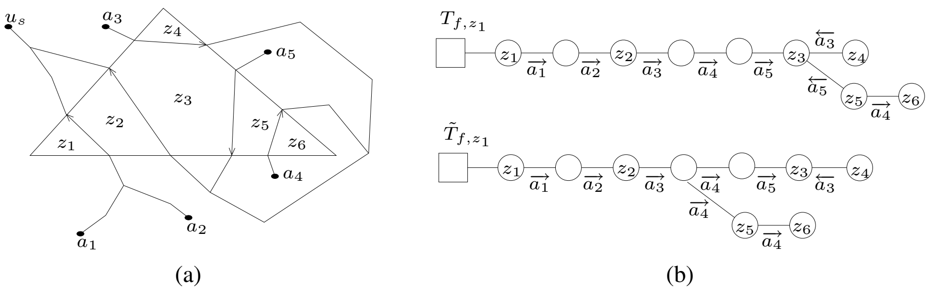


Fig. 9. (a) The division of a region into zones. (b) T_{f,z_1} and $\tilde{T}_{f,z_1}$.

Proof. 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 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. 9(a) 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.9(b) shows T_{f,z_1} for the example in Fig. 9(a). In $T_{f,z}$, we can read off the symbol sequence from any vertex p in zone z to any vertex q in another zone. But this sequence may not be canonical. 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 $x'x$. 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 x^*x. 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 $T_{f,z}$. $\tilde{T}_{f,z}$ is a prefix tree of canonical crossing sequences from z to all other zones in f. The bottom figure in Fig. 9(b) shows an example.

Then, we find in S_P the occurrences of all sequences in $\tilde{T}_{f,z}$ as follows. We construct a suffix tree T_S for S_P in $O(|S_P|)$ time [5]. Then, we traverse $\tilde{T}_{f,z}$ in a depth-first manner while navigating up and down T_S correspondingly. It takes $O(|\tilde{T}_{f,z}| + |S_P|)$ time to find for each sequence σ in $\tilde{T}_{f,z}$ the subtree of T_S 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. At the end of the jth round, for each zone pair (z, z') in the jth list, we report all edges pq of H_ε such that p is in z and q is in z'. $\square$

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 [13], 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 4. 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 5, the preprocessing takes $O(|S_P|\frac{h}{\varepsilon^2}n\log^2\frac{1}{\varepsilon})$ 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 the eligible edges pq of H_ε 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. The total running time 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. $\square$

References

1. Aleksandrov, L., Maheshwari, A., Sack, J.-R.: Determining approximate shortest paths on weighted polyhedral surfaces. J. ACM 52, 25–53 (2005)
2. Bespamyatnikh, S.: Computing homotopic shortest paths in the plane. J. Alg. 49, 284–303 (2003)
3. Cabello, S., Liu, Y., Mantler, A., Snoeyink, J.: Testing Homotopy for Paths in the Plane. Discr. Comput. Geom. 31, 61–81 (2004)
4. Efrat, A., Kobourov, S.G., Lubiw, A.: Computing homotopic shortest paths efficiently. Comput. Geom. Theory and Appl. 35, 162–172 (2006)
5. Farach, M.: Optimal suffix tree construction with large alphabets. In: Proc. 38th Annu. Sympos. Found. Comput. Sci., pp. 137–143 (1997)
6. Forbus, K.D., Uhser, J., Chapman, V.: Qualitative spatial reasoning about sketch maps. AI Magazine 24, 61–72 (2004)
7. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM 34, 596–615 (1987)
8. Gao, S., Jerrum, M., Kaufmann, M., Kehlhorn, K., Rülling, W., Storb, C.: On continuous homotopic one layer routing. In: Proc. 4th Annu. Sympos. Comput. Geom., pp. 392–402 (1998)
9. Hershberger, J., Snoeyink, J.: Computing minimum length paths of a given homotopy class. Comput. Geom. Theory and Appl. 4, 63–98 (1994)
10. Kaufmann, M., Mehlhorn, K.: On local routing of two-terminal nets. J. Comb. Theory, Ser. B 55, 33–72 (1992)
11. Leiserson, C.E., Maley, F.M.: Algorithms for routing and testing routability of planar VLSI layouts. In: Proc. 17th Annu. Sympos. Theory of Comput., pp. 69–78 (1985)
12. Mitchell, J., Papadimitriou, C.: The weighted region problem: Finding shortest paths through a weighted planar subdivision. J. ACM 38, 18–73 (1991)
13. Sun, Z., Reif, J.: On finding approximate optimal paths in weighted regions. J. Alg. 58, 1–32 (2006)