Triangulation Refinement and Approximate Shortest Paths in Weighted Regions*

Siu-Wing Cheng† Jiongxin Jin‡ Antoine Vigneron§

Abstract

Let $\mathcal{T}$ be a planar subdivision with n vertices. Each face of $\mathcal{T}$ has a weight from $[1, \rho] \cup \{\infty\}$. A path inside a face has cost equal to the product of its length and the face weight. In general, the cost of a path is the sum of the subpath costs in the faces intersected by the path. For any $\varepsilon \in (0, 1)$, we present a fully polynomial-time approximation scheme that finds a $(1+\varepsilon)$-approximate shortest path between two given points in $\mathcal{T}$ in $O\left(\frac{kn+k^4}{\varepsilon} \log^2 \frac{\rho n}{\varepsilon}\right)$ time, where k is the smallest integer such that the sum of the k smallest angles in $\mathcal{T}$ is at least π. Therefore, our running time can be as small as $O\left(\frac{n}{\varepsilon} \log^2 \frac{\rho n}{\varepsilon}\right)$ if there are $O(1)$ small angles and it is $O\left(\frac{n^4}{\varepsilon} \log^2 \frac{\rho n}{\varepsilon}\right)$ in the worst case.

*Research supported by Research Grants Council, Hong Kong, China (project no. 611812)

†Hong Kong University of Science and Technology. Email: scheng@cse.ust.hk

‡Google Inc. Email: jamesjjx@google.com

§King Abdullah University of Science and Technology. Email: antoine.vigneron@kaust.edu.sa

1 Introduction

Let $\mathcal{T}$ be a planar subdivision with n vertices. Each face f of $\mathcal{T}$ is associated with a weight $w_f \in [1, \rho] \cup \{\infty\}$. Faces with weight ∞ serve as obstacles. An edge e shared by faces f and g has weight $w_e = \min\{w_f, w_g\}$. An edge e of f not shared with another face (i.e., e is on the boundary of $\mathcal{T}$) has weight $w_e = w_f$. The cost for a path P is $\text{cost}(P) = \sum_{\text{face } f} w_f \cdot |P \cap f| + \sum_{\text{edge } e} w_e \cdot |P \cap e|$, where $|\cdot|$ denotes the length of a subpath or total lengths of subpaths. Without of generality, we assume that all faces in $\mathcal{T}$ are triangles. Given two points in $\mathcal{T}$, the shortest path is the minimum-cost path joining the two points. For $\varepsilon \in (0, 1)$, a path is a $(1 + \varepsilon)$-approximate shortest path if its cost is at most $1 + \varepsilon$ times the optimum.

Finding exact shortest paths in a weighted subdivision seems difficult, and only approximation algorithms are known so far. Mitchell and Papadimitriou first studied the problem and proposed an algorithm based on the continuous Dijkstra paradigm that runs in $O(n^8 \log \frac{N\rho n}{\varepsilon})$ time, where N is the largest vertex coordinate in the input [30]. They also give an example showing that an optimal path could have $\Omega(n^2)$ links. There has been extensive research on lowering the dependence on n by discretizing the geometric environment [28, 26, 6, 7, 37]. The idea is to add Steiner points, form a dense graph on the input vertices and the Steiner points, and then return the shortest path in the graph as a $(1 + \varepsilon)$-approximate solution. The two best results in this category are due to Aleksandrov et al. [7] and Sun and Reif [37]. Aleksandrov et al. [7] obtained a running time of $O(\frac{n}{\sqrt{\varepsilon}} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$, where the hidden constant is $O(\Gamma \log(\rho/\theta_{\min}))$ and Γ is the average of the reciprocals of the sinuses of the angles in $\mathcal{T}$. Sun and Reif [37] developed the BUSHWHACK algorithm which has a running time of $O(\frac{n}{\varepsilon} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$, where the hidden constant is $O((1/\theta_{\min}) \log(1/\theta_{\min}))$. Therefore, a single tiny angle in $\mathcal{T}$ can ruin the running time bounds in [7, 37]. Cheng et al. [15] use a different discretization scheme and obtained a running time of $O\left(\frac{\rho \log \rho}{\varepsilon} n^3 \log \frac{\rho n}{\varepsilon}\right)$.

In this paper, we present an algorithm for finding a $(1 + \varepsilon)$-approximate shortest path in weighted regions that runs in $O(\frac{kn+k^4}{\varepsilon} \log^2 \frac{\rho n}{\varepsilon})$ time, where k is the smallest integer such that the sum of the smallest k angles in $\mathcal{T}$ is at least π. The hidden constant does not depend on any other parameter. In the worst case, k could be $\Theta(n)$, but when there are not too many "small" angles, say $k = O(n^{1/3})$, then the running time is simply $\tilde{O}(kn/\varepsilon)$. Our running time bound is the first that is small for "easy" instances and yet bounded by a polynomial in n and $\log \frac{\rho}{\varepsilon}$ in the worst case. The exponent of the polynomial in n is also much smaller than the result by Michell and Papadimitriou [30].

The improvement comes from a few innovations. We observe that the discretization schemes in [6, 37] work very well in the absence of small angles. Our idea is not to place Steiner points on the two edges that bound a small angle. First, we refine the input triangulation into a new triangulation with $O(n + k^2)$ vertices such that each triangle has at most one angle that is less than $\pi/(2k)$. This allows us to group triangles with angles less than $\pi/(2k)$ into disjoint *strips* that do not contain vertices in the interior. Since the dual graph of a strip is a simple path, for every pair of points on the strip boundary, the shortest transversal path between them that lies inside the strip crosses a unique edge sequence that is predetrmined. Next, we place Steiner points on the strip boundaries. Since those edges bound only large angles, the number of Steiner points is under control. To build the discrete graph, we need to connect two Steiner points on the boundary of the same strip using approximate shortest path inside the strip. For further speed up, we adapt the BUSHWHACK algorithm [37] carefully inside strips to avoid building the entire discrete graph.

We introduce some notation. A path consists of *links* and *nodes*, where each link is a maximum segment that lies inside a triangle or on an edge, and a node is an endpoint of a link. Without loss of generality, we assume that a path does not have bend in the interior of a face as such a bend can be removed to shorten the path. That is, all nodes are on edges. A

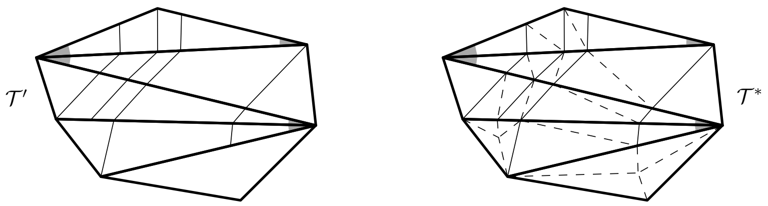


Figure 1: (a) The original subdivision is shown in bold, and its angles that are less than 2θ are shaded. The light polylines are propagation polylines. (b) Obtain the final triangulation by adding diagonals to quadrilaterals and triangulating triangular faces using incenters.

node is *transversal* if its two inciddent links lie in the interiors of two different faces. A path is *transversal* if all nodes, except its source and destination, are transversal nodes. A node is *critical* if it is in the interior of an edge e, one of its incident link (called the *critical link*) is on e, and the other incident link is in the interior of a face.

Consider two consecutive links pq and qr of a path such that pq is in face f, qr is in face g, and q is in the interior of the edge $f \cap g$. Let ℓ be the line through q perpendicular to $f \cap g$. Let θ_f (resp. θ_g) be the non-obtuse angle between ℓ an pq (resp. qr). We say the path obeys Snell's law at q if ℓ separates pq from qr and $w_f \sin\theta_f = w_g \sin\theta_g$. A path that obeys Snell's law at all non-vertex nodes is called a *refraction path.*

Lemma 1.1 ([30])**.** *There exists a shortest path P such that it is a refraction path and for every pair of critical links that appear consecutivey along P, the subpath between them contains a vertex.*

2 Triangulation refinement

Let k be the smallest integer such that the sum of the smallest k angles in $\mathcal{T}$ is at least π. Define $\theta = \min\{\pi/(2k), \pi/12\}$. By definition, the k-th smallest angle is at least $\pi/k \geq 2\theta$. In this section, we show how to refine the original subdivision $\mathcal{T}$ into $\mathcal{T}^*$ in which every triangle has at most one angle less than θ. There is a known refinement with $O(n^2)$ vertices and angles at most $\frac{11}{15}\pi$ [38]. We present a simpler algorithm with worse angle bound (while still good enough for our purposes) but the number of vertices is sensitive to k.

A *propagation polyline* $(p_0, p_1, \ldots, p_m)$ is a polyline such that for every $i \in [0, m-1]$, p_i is on an edge e_i of $\mathcal{T}$, e_i and e_{i+1} bound a face f_i such that the angle of f_i at $e_i \cap e_{i+1}$ is less than 2θ, e_i and e_{i+2} do not bound the same face, and $p_i p_{i+1}$ is parallel to the angle bisector of the largest angle in f_i.[1] By definition, a propagation polyline does not intersect itself or another propation polyline. For any triangle in $\mathcal{T}$ that has an angle less than 2θ, we generate a propagation polyline by starting from its largest angle and extending the polyline while maintaining the above properties until it cannot be extended further. See Figure 1(a).

For two edges e, e' in the same triangle, $\angle(e, e')$ denotes the angle formed by the two edges, which is in $(0, \pi)$.

Lemma 2.1. *Let $(e_0, e_1, \ldots, e_m)$ be any sequence of edges in $\mathcal{T}$ such that for $i \in [0, m-1]$, e_i and e_{i+1} bound a face angle less than 2θ and e_i and e_{i+2} do not bound the same face. Then $m < k$ and for any $i < j-1$, e_i and e_j do not bound the same face in $\mathcal{T}$.*

Proof. We have $\sum_{0 \leq i < k} \angle(e_i, e_{i+1}) < \pi$ as $\angle(e_i, e_{i+1}) < 2\theta$ for any i. In order for two edges e_i, e_j, where $i < j-1$, to bound the same face, the edges $e_i, e_{i+1}, \ldots, e_j$ must turn a total angle of at least π, i.e. $\sum_{r=i}^{j-1} \angle(e_r, e_{r+1}) \geq \pi$. This is impossible. So $m < k$ and for any $i < j-1$, e_i and e_j do not bound the same face. ◻

[1]If there are more than one largest angle in f_i, we break ties by choosing the vertex with the largest index.

Lemma 2.2. *There are fewer than k propagation polylines, and each propagation polyline has fewer than than k segments and crosses any face of $\mathcal{T}$ at most once.*

Proof. Each propagation polyline starts at a vertex of a triangle that has an angle less than 2θ. There are at most $k-1$ such triangles, so there are at most $k-1$ propagation polylines. The second part of the lemma follows directly from Lemma 2.1. $\square$

Let $\mathcal{T}'$ be the overlay of $\mathcal{T}$ and all the propagation polylines. By the definition of propagation polylines, one can verify that $\mathcal{T}'$ has the following properties.

P1. Each face of $\mathcal{T}'$ is either a triangle or a quadrilateral.

P2. Every triangular face of $\mathcal{T}'$ has at most one angle less than 2θ and such an angle is an angle in $\mathcal{T}$.

P3. In every quadrilateral face of $\mathcal{T}'$, there are two parallel sides that lie on propagation polylines, and there are two other sides that lie on two edges in $\mathcal{T}$ which bound a face angle less than 2θ in $\mathcal{T}$.

P4. A propagation polyline ends either at a vertex or on the boundary of a trianguar face in $\mathcal{T}'$ (also a face in $\mathcal{T}$) with all angles at least 2θ.

For every quadrilateral in $\mathcal{T}'$, triangulate it by adding an arbitrary diagonal. For every triangle $f \in \mathcal{T}'$ with all angles at least 2θ, if some propagation polyline ends on the boundary of f, we connect the incenter of f (the common intersection of the angle bisectors) to the vertices of f and the propagation polyline endpoints on the boundary of f. See Figure 1(b). This gives the final triangulation $\mathcal{T}^*$.

Theorem 2.1. *Given a triangulation $\mathcal{T}$ with n vertices such that the sum of the k smallest angles is at least π, one can compute in $O(n+k^2)$ time a refined triangulation $\mathcal{T}^*$ that enjoys the following properties. Let $\theta = \min\{\pi/(2k), \pi/12\}$.*

(i) *$\mathcal{T}^*$ has no more than $n+k^2+k$ vertices.*

(ii) *Every triangle in $\mathcal{T}^*$ has at most one angle less than θ.*

(iii) *Let $(e_1, e_2, \dots)$ be any sequence of edges in $\mathcal{T}^*$ such that for any i, e_i and e_{i+1} bound a face angle less than θ, and e_i and e_{i+2} do not bound the same face. This sequence has $O(k)$ edges and no repetitions.*

Proof. The running time is clearly linear in the size of $\mathcal{T}^*$, which is $O(n+k^2)$, assuming (i) holds. Let $\mathcal{T}'$ be the overlay of $\mathcal{T}$ and the propagation polylines. Consider (i). The vertices of $\mathcal{T}^*$ that are not in $\mathcal{T}$ are either intersections between propagation polylines and edges of $\mathcal{T}$ or incenters of some triangles in $\mathcal{T}'$. There are fewer than k^2 vertices of the former type, because, by Lemma 2.2, fewer than k such vertices are generated by one propagation polyline and there are less than k propagation polylines. The incenter of a triangle is only added as a vertex when there are propagation polylines end at the boundary of that triangle, so at most k such vertices are created. In total, $\mathcal{T}^*$ has no more than $n+k^2+k$ vertices.

Consider (ii). In general, the angle between the angle bisector of the largest angle of a triangle and any edge of that triangle is at least $\pi/6$ because it is at least half of the largest angle. It follows that any angle in a quadrilateral in $\mathcal{T}'$ is in $(\pi/6, 5\pi/6)$. Therefore, a quadrilateral of $\mathcal{T}'$ is divided by an diagonal into two triangles each of which has at most one angle less than $\theta \le \pi/12$. What about triangles in triangular faces of $\mathcal{T}'$? By P2, every triangular face of $\mathcal{T}'$ has at most one angle less than 2θ. If a triangular face τ has one angle less than 2θ, no refinement is needed by P4, so τ will be outputted directly as a triangle in $\mathcal{T}^*$. If all angles of τ are no less than 2θ, we may need to triangulate τ using its incenter. Consider a segment from τ's incenter to an edge of τ. The acute angle between the segment and that edge of τ is at least half of the smallest angle of τ. So a triangle in the refinement of τ has at most one angle that is smaller than θ, which is at τ's incenter.

Consider (iii). Suppose that an edge e_1 in the sequence is incident to an incenter of some triangle in $\mathcal{T}'$. By the above analysis, e_2 must be incident to the same incenter as well, and inductively, all e_i's are incident to the same incenter. Observe that a vertex at an incenter has degree at most $k+3$, implying that at least one angle at the vertex is at least $2\pi/(k+3) > \theta$. So no repetition is possible, and the sequence contains no more than $k+3$ edges.

Suppose that e_1 is on an edge of the original subdivision $\mathcal{T}$. Then for any i, e_i is either part of some edge of $\mathcal{T}$, or a diagonal of a quadrilateral of $\mathcal{T}'$. Remove diagonal edges from the original edge list $(e_1, e_2, \dots)$ and replace others with the corresponding edges in $\mathcal{T}$, and let $(e'_1, e'_2, \dots)$ be the result. Since the original sequence does not contain two consecutive diagonal edges, the length of the new sequence is at least half of the length of the original one, e'_i and e'_{i+1} are in a same face of $\mathcal{T}$, and $\angle(e'_i, e'_{i+1}) < 2\theta$. By Lemma 2.1, the new sequence has length $O(k)$, and no two non-consecutive e'_i and e'_j bound the same face of $\mathcal{T}'$. It follows that the original sequence also has length $O(k)$, and it does not have repeated edges.

Suppose that e_1 is is part of a propagation polylines. Then e_i is a part of a propagation polyline if i is odd, and a diagonal of some quadrilateral of $\mathcal{T}'$ if i is even. Moreover, all these edges are inside a same triangle of $\mathcal{T}$, say τ. By Lemma 2.2, there are $O(k)$ propagation polylines, and each intersects with τ at most once. Therefore, the sequence has length $O(k)$. Clearly, edge repetitions are not possibile.

The remaining case is that e_1 is a diagonal of a quadrilateral of $\mathcal{T}'$. Then e_2 must fall in one of the last two categories above, so we can repeat the same argument. □

3 Algorithm

3.1 Approximation graph $\mathcal{G}$

We first apply Theorem 2.1 to compute a refinement $\mathcal{T}^*$ of $\mathcal{T}$. We then discretize $\mathcal{T}^*$ to obtain a discrete graph $\mathcal{G}$ which contains a $(1+\varepsilon)$-approximate shortest path.

Let $(e_1, e_2, \dots, e_m)$ be a longest sequence of edges that satisfies Theorem 2.1(iii). For $1 \le i < m$, let τ_i be the triangle that contains e_i and e_{i+1}. Then the triangle sequence $\tau_1, \dots, \tau_{m-1}$ is called a *strip*. Edges $e_2, e_3, \dots, e_{m-1}$ are *interior edges* of the strip. Edges of τ_i that are not interior edges are *boundary edges* of the strip. So any triangle τ_i except τ_1 and τ_{m-1} has one edge in the strip boundary, which is opposite the smallest angle of τ_i. τ_1 and τ_{m-1} have two boundary edges of the strip, including e_1 and e_m.

We follow the same approach used by [6, 37] to discretize boundary edges of strips and edges outside strips. Interior edges of a strip are not discretized. Take edge vu that is not an interior edge of any strip. Let $\theta = \min\{1/k, \pi/12\}$ be the same value as in Theorem 2.1. Since vu is not an interior edges, at least one of the two angles at u with a side vu is larger than θ. If both angles are larger than θ, let α_{vu} be the smaller of the two; otherwise, define α_{vu} to be the larger one. So $\alpha_{vu} > \theta = \Omega(1/k)$. Let L be the length of the Euclidean shortest path form s to t, which can be computed in $O(n \log n)$ time [22]. Place Steiner points $p_0, p_1, \dots$ on vu as follows. Let c_0 and c_1 be some constants to be defined later. p_0 is placed at distance $c_0\varepsilon L/n^2$ from v, and for $i > 0$, $|p_{i-1}p_i| = (c_1\varepsilon \sin\alpha_{vu})|vp_{i-1}|$. Similarly, also create Steiner points $q_0, q_1, \dots$ such that $|q_0u| = c_0\varepsilon L/n^2$, and for $i > 0$, $|q_{i-1}q_i| = (c_1\varepsilon \sin\alpha_{uv})|uq_{i-1}|$.

Finally, all Steiner points outside the disk $D(s, 2\rho L)$ centered at s with radius $2\rho L$ are removed. Observe that, for $\varepsilon < 1$, any $(1+\varepsilon)$-approximate shortest path has length no more than $2\rho L$ and hence lies in $D(s, 2\rho L)$. So removing those Steiner points does not affect the correctness of the algorithm, and yet it is necessary for obtaining results that are independent of geometry parameters. The remaining Steiner points and vertices of $\mathcal{T}^*$ form the vertex set of the approximation graph $\mathcal{G}$.

In $\mathcal{G}$, there is an edge between a pair of graph vertices in the same face or a pair of graph vertices (which may not be in the same face) in the boundary of the same strip. Let (u, v) be a pair of graph vertices between which there is a graph edge. The weight of the edge is denoted by $\mu(u, v)$. If u and v are in the same face of $\mathcal{T}^*$, $\mu(u, v)$ is simply the cost of the straight segment connecting them. If u and v are in the boundary of some strip, $\mu(u, v)$ is the cost of the shortest path inside the strip from u to v.

Lemma 3.1. *Each edge of $\mathcal{T}^*$ has $O(\frac{k}{\varepsilon}\log\frac{\rho n}{\varepsilon})$ Steiner points. Set $c_0 = 1/128$ and $c_1 = 1/16$. The shortest path in $\mathcal{G}$ is a $(1+\varepsilon/2)$-approximate shortest path in $\mathcal{T}^*$.*

Proof. Take any edge vu of $\mathcal{T}^*$. Let $p_0, p_1, \ldots$ be the series of Steiner points on vu created when we process v. In the end, we remove those outside the disk $D(s, 2\rho L)$. Let the remaining Steiner points be $p_i, p_{i+1}, \ldots, p_m$. Since $|vp_i|(1 + c_1\varepsilon \sin\alpha(v))^{m-i} = |vp_m| \leq |vp_i| + 4\rho L$, $|vp_i| \geq |vp_0| = \Theta(\varepsilon L/n^2)$, and $\alpha(v) = \Omega(1/k)$, we have $m - i = O(\frac{k}{\varepsilon}\log\frac{\rho n}{\varepsilon})$.

Let P be the shortest path in $\mathcal{T}^*$. So P lies inside $D(s, 2\rho L)$. Convert it to a path in $\mathcal{G}$ as follows. Take a vertex v of $\mathcal{T}^*$. Let p_v and q_v be the first and last nodes of P that are at distance less than $c_0\varepsilon L/n^2$ from v. Snap both p_v and q_v to v and remove the subpath between them. Do the same for all vertices in $\mathcal{T}^*$. Then, for any non-vertex node x of P such that x lies either outside any strip or on the boundary of some strip, snap x to the closest Steiner point on the edge of $\mathcal{T}^*$ that contains x. Let P' be the resulting path. It suffices to show that $\text{cost}(P') \leq (1 + \varepsilon/2)\,\text{cost}(P)$.

Snapping nodes to a vertex of $\mathcal{T}^*$ incurs an error no more than $2c_0\varepsilon L/n^2$. $\mathcal{T}$ has n vertices, so it has no more than $2n - 4$ faces, which implies that $k \leq 4n - 8$. By Theorem 2.1(i), $\mathcal{T}^*$ has no more than $n + k^2 + k < 16n^2$ vertices. By the setting of c_0, the total error due to snapping to vertices of $\mathcal{T}^*$ is no more than $\varepsilon L/4 \leq (\varepsilon/4)\,\text{cost}(P)$.

Consider the error due to snapping nodes to Steiner points. Take any such node p. Let e be the edge containing p. If p is outside any strip, then all face angles adjacent to e are at least θ. If p is on a strip boundary, then e must be the edge opposite the smallest angle of a triangle τ in the strip, and therefore, the two angles of τ adjacent to e are at least θ. We conclude that some link pq incident to p must straddle an angle ϕ of some triangle at a vertex v such that $\phi \geq \theta$. Note that v is an endpoint of e. Then, $|pq| \geq |vp|\sin\phi$. The distance between p and its nearest Steiner point on e is no more than $2(c_1\varepsilon\sin\phi)\cdot|vp| \leq 2c_1\varepsilon|pq| \leq \frac{\varepsilon}{8}|pq|$. The link pq can be charged a snapping error at most twice (once for each endpoint). Therefore, the total snapping error charged to the links is at most $(\varepsilon/4)\,\text{cost}(P)$. ◻

3.2 Compute an approximate shortest path in $\mathcal{G}$.

$\mathcal{G}$ has $O(\frac{kn+k^3}{\varepsilon}\log\frac{\rho n}{\varepsilon})$ vertices and $O(\frac{k^2n+k^5}{\varepsilon^2}\log^2\frac{\rho n}{\varepsilon})$ edges. Computing the shortest path using Dijkstra algorithm directly is too slow. Sun and Reif proposed a BUSHWHACK algorithm [37], which avoids generating all graph edges explicitly. The intuition is that we do not need to compute graph edges that are known not to contribute to optimal paths. We adapt the basic idea. However, one challenge is that we cannot compute shortest paths in weighted regions exactly in general, even when the edge sequence is known. It requires care to control the errors because our algorithm may drop graph edges to which that the optimal path is snapped.

We maintain a priority queue of vertices, Steiner points and *intervals* (to be explained later). Every element a in the priority queue is associated with a cost $d(a)$, which is the (approximate) cost of the current best path from s to a. Initially, the priority queue contains all vertices and Steiner points with $d(s) = 0$ and $d(p) = \infty$ for any other point p. The algorithm repeatedly extracts the element with the minimum cost from the priority queue and uses it to update the costs of other vertices and intervals. Once an element is dequeued, its cost is determined.

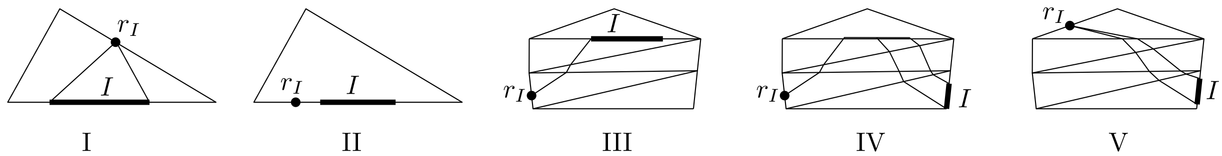


Figure 2: Five types of intervals.

Before presenting the algorithm in detail, we need to elaborate on intervals. An interval I on an edge e has a *root*, denoted by r_I, which is a vertex or a Steiner point. The interval I will be defined after $d(r_I)$ is computed, and I consists of the destinations of (approximate) shortest paths from r_I to e that cross the same sequence of edges (without passing through any vertex). The cost of a point $x \in I$ is $d(r_I)$ plus the cost of the path from r_I to x. The cost of I is the minimum of the costs of points in I.

There are five types of intervals. See Figure 2. A type-I interval and its root are in the same face but not on the same edge, while a type-II interval and its root are on the same edge. The cost of any point x on a type-I or type-II interval I is $d(r_I) + \mathrm{cost}(r_I x)$. A type-III interval I lies on an edge e in the interior of a strip and its root r_I lies on the strip boundary. Any point on I is reached from r_I by a path inside the strip that obeys Snell's law and whose last link is a critical link on e. A type-IV interval I lies on an edge e on the boundary of a strip, and its root r_I lies on the strip boundary. It can be viewed as the result of propagating a type-III interval. Every point in I is reached from r_I by a path inside the strip that obeys Snell's law and has a critical link on e. The cost of a point x in a type-III or type-IV interval is $d(r_I)$ plus the cost of the refraction path inside the strip between r_I and x. A type-V interval I and its root r_I lie on the boundary of the same strip, and I contains the destinations of paths whose subpaths after r_I are transversal paths. There is no known algorithm that can compute shortest transversal paths exactly. So we have to define the cost of a point x on a type-V interval to be $d(r_I)$ plus the cost of some approximate shortest transversal path from r_I to x as explained below.

One way to define a metric is via defining its unit disk: the distance between two points p and q is $\min_\lambda \{t \in C : p + \lambda t = q\}$, where C is the unit disk that defines the metric. Then the unit disk for the cost function for a face f is simply a Euclidean disk centered at the origin with radius $1/w_f$. For every face f, define a new metric whose unit disk is a regular polygon with h vertices inscribed to the Euclidean disk centered at the origin with radius $1/w_f$. We use $\mathrm{cost}_{\pentagon}$ to denote the cost of a path under this polygonal metric. One can verify that for every segment ℓ, $\cos(\pi/h)\,\mathrm{cost}_{\pentagon}(\ell) \le \mathrm{cost}(\ell) \le \mathrm{cost}_{\pentagon}(\ell)$. Setting $h = O(1/\sqrt{\varepsilon})$ with an appropriate constant, we can obtain $\mathrm{cost}_{\pentagon}(\ell) \le (1+c\varepsilon)\ \mathrm{cost}(\ell)$ for any constant c. For a point x in a type-V interval, we define the cost of x to be $d(r_I) + \mathrm{cost}_{\pentagon}(T(r_I, x))$, where $T(r_I, x)$ is the transversal path inside the strip from r_I to x with the minimum $\mathrm{cost}_{\pentagon}$.

As mentioned before, the algorithm repeatedly extracts elements with the smallest cost. Depending on the type of the dequeued element, we invoke either $\mathsf{ProcessPoint}$ or $\mathsf{ProcessInterval}$ below. When creating an interval in $\mathsf{ProcessPoint}$ or $\mathsf{ProcessInterval}$, we may also trim or prune existing intervals. Intervals on each edge are put in groups, which we will explain when we elaborate the interval creation below. When an interval is dequeued from the priority queue, it is not removed from the group. Intervals in the same group are kept disjoint, but intervals from different groups are allowed to overlap. Therefore, a Steiner point or vertex p may be shared between two intervals I and I', the cost of p in I can be smaller than the cost of p in I', and $d(p)$ will be determined by the minimum cost of p in the intervals that contain p.

$\mathsf{ProcessPoint}$(a Steiner point or vertex v)

1. For every edge e_v that contains v and does not lie inside any strip, create and enqueue type-II intervals on e_v with v as their common root.

2. (a) For every edge e such that e does not lie inside any strip and e is the edge of a face opposite v, create and enqueue type-I intervals on e with v as their common root.
   (b) For every strip S that contains v on the boundary and for every edge e in the interior of S, create and enqueue type-III intervals on e with v as their common root.
   (c) For every strip S that contains v on the boundary and for every edge e in the boundary of S, create and enqueue type-V intervals on e with v as their common root.

ProcessInterval(an interval I)

1. If I is a type-III interval in some strip S, create type-IV intervals on the boundary of S that I propagates to. Note that r_I is the common root of these type-IV intervals.
2. Otherwise, let p be the Steiner point or vertex in I that has the smallest cost and do the following.
   (a) If the cost of p in I is smaller than $d(p)$, update $d(p)$. (If p is still in the priority queue, this step will make p the next element to be dequeued.)
   (b) We maintain the invariant that the costs for points in I are monotonic. So all other Steiner points and vertices in I are on one side of p. Since $d(p)$ has already been determined, we shrink I to the shortest interval that covers the remaining Steiner points and vertices in I. If the new I is non-empty, its cost is the minimum cost of its two endpoints and we insert the new I into the priority queue.

We elaborate below on the creation of intervals. For this purpopse, let S denote a strip and Let $e_1, e_2, \ldots$ be the sequence of the edges in the interior of S. Orient edges from left to right as follows. The left and right endpoints of e_1 are defined arbitrarily. For $i \geq 1$, orient e_i so that e_i and e_{i+1} share a common left or right endpoint. Let a_i and b_i denote be the left and right endpoints of e_i, respectively.

Type-I intervals Refer to step 2(a) of ProcessPoint(v). Suppose that e is not an edge in the interior of some strip. The distances from v to points on e is a convex function. Let p be v's nearest point in e, which is either an endpoint of e or such that vp is perpendicular to e. Create two type-I intervals I_1, I_2 that covers the Steiner points and vertices on both sides of p respectively. Note that the points in each interval have monotonic costs, and the endpoint closer to p has the smallest cost. The edge e is incident to two triangles, so the roots of type-I intervals on e can lie on the four other edges of these two triangles. We divide the type-I intervals on e into at most four different groups depending on the location of their roots. We may already have intervals on e in the same group as I_1 and I_2. If they overlap with I_1 or I_2, trimming is needed to make them disjoint. When two intervals overlap, either one is strictly inferior in the sense that its cost evaluated at every point is greater than or equal to the other interval's cost evaluated at the same point, in which case the inferior interval can be removed from the group altogether, or there is a tie point in their intersection such that one interval is better on one side of the tie point and the other interval is better on the other side, in which case both intervals are trimmed to that tie point. Refer to [37] for a full proof. If an interval does not contain any Steiner point or vertex after trimming, remove it from the group as well as the priority queue. Otherwise, the cost of the trimmed interval is equal to the minimum cost of the Steiner points or vertices at its two ends (by the cost monotonicity), and we update the interval cost accordingly.

Type-II intervals Refer to step 1 of ProcessPoint(v). Let a and b be the two endpoints of e_v. Consider the case that v is a Steiner point. Let p and p' be the Steiner points in av and vb, respectively, that are closest to v. Note that pa and $p'b$ cover the Steiner points and vertices on e_v except for v ($d(v)$ has already been determined). Create two intervals pa and $p'b$ with costs $d(v) + \text{cost}(vp)$ and $d(v) + \text{cost}(vp')$, respectively. In the case that v is a or b, only one type-II interval is created as in the above. There are two groups of type-II intervals on e_v depending on which endpoint of e_v is contained by them when they were created. Suppose that two intervals I_1 and I_2 in the group for a overlap. Assume that I_1 contains the endpoint x of I_2 that is farther from a. If the cost of x in I_1 is at most the cost of x in I_2, then delete I_2 from the group and the priority queue. Otherwise, trim I_1 so that I_1 and I_2 meet at x and their interiors are disjoint. The case of I_1 and I_2 in the group for b is handled symmetrically.

Type-III intervals Refer to Step 2(b) of ProcessPoint(v). For any edge $e_i = a_i b_i$ in the interior of the strip, if there is a refraction path inside the strip from v to a point $x \in e_i$ onward to a_i such that xa_i is a critical link, then create a type-III interval xa_i. We symmetrically create another type-III interval with endpoint b_i. These two intervals can be created in $O(1)$ time by Lemma 3.2 below. We divide type-III intervals on e_i into groups. Two intervals are in the same group if their roots lie on the same strip boundary edge and when these intervals were created, they contained the same endpoint of e_i. The trimmings of intervals in the same group are done in the same way as for type-II intervals.

Lemma 3.2. *Let S be a strip. For every point x in the boundary of S and for every edge uv in the interior of S, let $R_{uv}(x, u)$ denote the transversal path from x to uv that enters uv at a critical angle and will then follow uv towards u. The path $R_{uv}(x, v)$ is symmetrically defined. Note that $R_{uv}(x, u)$ or $R_{uv}(x, v)$ may not exist. We can preprocess S in $O(k^3)$ time so that for any x and uv, we can report in $O(1)$ time whether $R_{uv}(x, u)$ (resp. $R_{uv}(x, v)$) exists and if so, the destination and cost of $R_{uv}(x, u)$ (resp. $R_{uv}(x, v)$).*

Proof. Since $R_{uv}(x, u)$ enteres uv at a critical angle, the direction of each link in $R_{uv}(x, u)$ is completely determined by Snell's law. Let e be the boundary edge of S that contains x. The position of the destination of $R_{uv}(x, u)$ in uv is a linear function in the position of x in e. Similarly, the cost of $R_{uv}(x, u)$ is also a linear function of the position of x in e. These two linear functions can be constructed in $O(k)$ time by tracing from uv towards e using Snell's law. Repeating over all interior edges and all boundary edges of S takes $O(k^3)$ time. ◻

Type-III intervals are intermediate intervals that lead to type-IV intervals. The advantage of generating type-III intervals is that it becomes possible to do some pruning which allows us to generate fewer type-IV intervals in the end.

Type-IV intervals Type-IV intervals are generated when a type-III interval I on some edge e_i in the interior of a strip is dequeued. If I extends all the way to an endpoint of e_i, create a type-IV interval for every boundary edge of the strip. Otherwise, only create one type-IV interval on the boundary edge that r_I is on. Lemma 3.3 below shows that we do not do worse in the latter case. By Lemma 3.2, each interval can be created in $O(1)$ time.

Lemma 3.3. *Let I be a type-III interval on an interior edge e of a strip S such that I does not contain any endpoint of e. Let P be a refraction path in S from r_I to a point p in the boundary of S such that P contains exactly one critical link, which is a subset of I, and p and r_I lie on distinct boundary edges of S. Then, P is not the shortest path in S from r_I to p, or there exists a Steiner point or vertex r on the same boundary edge as r_I such that $d(r) + \text{cost}(P^*) < d(r_I) + \text{cost}(P)$, where P^* is the shortest path in S from r to p.*

Type-IV intervals on a boundary edge e of a strip S are divided into four groups as follows. An type-IV interval I on e is generated from a type-III interval on some interior edge e_ℓ. We use $\mathsf{pred}(I)$ to denote this type-III interval and $\mathsf{pred\_idx}(I)$ to denote the index ℓ. Note that I and $\mathsf{pred}(I)$ share the common root r_I. The interval $\mathsf{pred}(I)$ has one of two possible orientations depending on which endpoint of $\mathsf{pred}(I)$ the path from r_I to $\mathsf{pred}(I)$ enters. We call this the *starting endpoint* of $\mathsf{pred}(I)$. Suppose that e is incident to the face in S that is bounded by e_i and e_{i+1}. Then, the group that I belongs to is determined by the orientation of $\mathsf{pred}(I)$ and whether $\mathsf{pred\_idx}(I) \leq i$.

We need to update the type-IV intervals in the same group so that they do not overlap. Let I_1 and I_2 be two overlapping type-IV intervals on e in the same group. Note that $\mathsf{pred}(I_1)$ and $\mathsf{pred}(I_2)$ cannot be on the same edge because otherwise I_1 and I_2 do not overlap by the trimming of of $\mathsf{pred}(I_1)$ and $\mathsf{pred}(I_2)$. Assume that $\mathsf{pred\_idx}(I_1) < \mathsf{pred\_idx}(I_2) \leq i$ and the two intervals are from left to right. Let s_2 be the starting endpoint of $\mathsf{pred}(I_2)$. The trimming is based on the following lemma.

Lemma 3.4. *Suppose that* $\mathsf{pred}(I_1)$ *and* $\mathsf{pred}(I_2)$ *are oriented from left to right and* $\mathsf{pred\_idx}(I_1) < \mathsf{pred\_idx}(I_2) \leq i$. *Let s_i be the starting endpoint of* $\mathsf{pred}(I_i)$. *For $i \in [1, 2]$ and for all $y \in I_i$, let $P_{i,x}$ denote the refraction path from r_{I_i} through* $\mathsf{pred}(I_i)$ *to x that defines I_i. For $i \in [1, 2]$ and for all $x \in I_1 \cap I_2$, let $Q_{i,x}$ be the shortest path inside the strip from r_{I_i} to x.*

(i) *Suppose that $d(r_{I_1}) + \mathrm{cost}(P_{1,x}) \leq d(r_{I_2}) + \mathrm{cost}(P_{2,x})$ for some point $x \in I_1 \cap I_2$. If s_2 is to the right (resp. left) of $P_{1,x}$ with respect to its orientation from r_{I_1} to x, then for every point $y \in I_2$ to the right (resp. left) of x, $d(r_{I_1}) + \mathrm{cost}(Q_{1,y}) \leq d(r_{I_2}) + \mathrm{cost}(P_{2,y})$.*

(ii) *If $d(r_{I_2}) + \mathrm{cost}(P_{2,x}) \leq d(r_{I_1}) + \mathrm{cost}(P_{1,x})$ for some point $x \in I_1 \cap I_2$, then for every point $y \in I_1$ to the left of x, $d(r_{I_2}) + \mathrm{cost}(Q_{2,y}) \leq d(r_{I_1}) + \mathrm{cost}(P_{1,y})$.*

Suppose that I_1 has smaller costs than I_2 for all points in $I_1 \cap I_2$. If $I_2 \setminus I_1$ is connected or empty, trim I_2 to $I_2 \setminus I_1$. If $I_2 \setminus I_1$ consists of two disconnected intervals, we prune away one or more components in $I_2 \setminus I_1$ using Lemma 3.4(i) as follows. Let x and y be the endpoints of $I_1 \cap I_2$. Let $P_{1,x}, P_{1,y}$ be the refraction paths from r_{I_1} to x and y as defined in Lemma 3.4. If s_2 is to the right of both $P_{1,x}$ and $P_{1,y}$, we trim I_2 by taking the left interval in $I_2 \setminus I_1$. If s_2 is to the left of both $P_{1,x}$ and $P_{1,y}$, take the right interval in $I_2 \setminus I_1$. If s_2 is sandwiched between $P_{1,x}$ and $P_{1,y}$, I_2 is pruned altogether.

Suppose that I_2 has smaller costs than I_1 for all points in $I_1 \cap I_2$. If $I_1 \setminus I_2$ is connected, trim I_1 to $I_1 \setminus I_2$; otherwise, trim I_1 by taking the right interval in $I_1 \setminus I_2$ according to Lemma 3.4(ii).

The last case is that there is some tie point $x \in I_1 \cap I_2$ such that the two intervals have the same cost at x. By Lemma 3.4(ii), the part of I_1 to x's left is suboptimal and can be trimmed, which also implies that I_1 has a smaller cost at any point in $I_1 \cap I_2$ to the right of x. So we trim the part of I_1 to the left of x and trim the part of I_2 to the right of x.

The costs of points in an type-IV interval are linear, so trimming can be done in $O(1)$ time.

Type-V intervals

Lemma 3.5. *Let e be any fixed edge on a strip boundary. After $O(\frac{k^2}{\sqrt{\varepsilon}} \log \frac{k}{\varepsilon})$ preprocessing time, given any source on e and any destination on the boundary of the same strip, one can determine whether a shortest pcost transversal path between them passes through vertices in its interior, and return its pcost if it does not in $O(\log \frac{k}{\varepsilon})$ time.*

For every pairs of boundary edges e and e' of S, we apply Lemma 3.5 (proof in appendix) to build a data structure so that for every point $x \in e$ and every point $y \in e'$, the minimum cost of a transversal path from x to y in S be answered in $O(\log \frac{k}{\varepsilon})$ time. The total preprocessing time is $O(\frac{k^4}{\sqrt{\varepsilon}} \log \frac{k}{\varepsilon})$. When a Steiner point or vertex r in the boundary of S is dequeued, for every boundary edge e of S that does not contain r, we binary search with the data structure built to find the two extreme traversal paths from r to the Steiner points or vertices on e that do not pass through vertices in the interior of the paths. The destinations of these two paths are the endpoints of the type-V interval on e with root r. All type-V intervals on e with roots in the boundary of S are put into one group. If e is shared between S and another strip S', there may be another group of type-V intervals on e for S'.

Suppose two intervals I_1 and I_2 overlap. For any $x \in I_i$, let $P_{i,x}$ denote the transversal path with the minimum $\mathrm{cost}_{\pentagon}$ from r_{I_i} to x. First, apply Lemma 3.5 to find the tie point $x \in I_1 \cap I_2$ such that $d(r_1) + \mathrm{cost}_{\pentagon}(P_{1,x}) = d(r_2) + \mathrm{cost}_{\pentagon}(P_{2,x})$. Lemma 3.6 below allows us to trim I_1 and I_2 to two disjoint intervals meeting at x without losing optimal paths. Then, update the costs of trimmed intervals in the priority queue. If such a tie point in $I_1 \cap I_2$ does not exist, by Lemma 3.6, the inferior interval can be pruned altogether. A pruned interval is deleted from both priority queue and the interval group.

Lemma 3.6. *Let I_1 and I_2 be two type-V intervals on the boundary edge e of a strip S. If $d(r_{I_2}) + \mathrm{cost}_{\pentagon}(P_{2,x}) \leq d(r_{I_1}) + \mathrm{cost}_{\pentagon}(P_{1,x})$ for some point $x \in I_1 \cap I_2$, then $I_1 \cap \gamma$ can be trimmed, where γ is the part of the boundary of S delimited by r_{I_2} and x that excludes r_{I_1}.*

Lemma 3.7. *The algorithm runs in $O(\frac{kn+k^4}{\varepsilon}\log^2\frac{\rho n}{\varepsilon})$ time.*

Proof. Let $m = O(\frac{k}{\varepsilon}\log\frac{\rho n}{\varepsilon})$ be an upper bound on the number of Steiner points placed on every edge.

Each Steiner point propagates to $O(1)$ type-I and type-II intervals, $O(k)$ type-III intervals, $O(k^2)$ type-IV, and $O(k)$ type-V intervals. The generation of intervals from a vertex depends on its vertex degree, but we can charge these interval generations to the neighbordng Steiner points on the incident edges.

Consider interval trimmings. When a new interval is created, we search for overlapping intervals in the same group. Since intervals in the same group are kept disjoint, we can put them in a sorted list so finding overlapping intervals takes $O(\log m)$ time. The number trimming done for a new interval is at most 2 plus the number of pruned intervals. An interval can only be pruned once, so on average, a new interval is trimmed $O(1)$ times. Trimming two intervals of type I-IV takes $O(1)$ time. The costs of points on a type-V interval is a convex piecewise linear function. To find the tie point, we first do binary searches to find the linear pieces in two functions that cross, and then compute the crossing point of the two segments. So trimming type-V intervals takes $O(\log\frac{k}{\varepsilon})$ time. Creating all types of intervals takes $O(mn\log m)$, $O(mn\log m)$, $O(k^2 m\log m)$, $O(k^3 m\log m)$, and $O(k^2 m\log\frac{km}{\varepsilon})$ times, respectively.

The number of groups of intervals on an edge is $O(k)$ for type-III and a constant for type-I, II, IV, and V. Therefore, a Steiner point is contained in $O(1)$ intervals. (A type-III interval is on an interior edge of strip and it contains no Steiner point.) The dequening of a Steiner point may trigger the trimming of intervals and the associated priority queue updates. Hence, there are $O(mn)$ such trimmings and priority queue updates. Preprocessing takes $O(\frac{k^4}{\sqrt{\varepsilon}}\log\frac{k}{\varepsilon})$ time by Lemmas 3.2 and 3.5. So the total running time is $O(mn\log(mn) + k^2 m\log m + k^3 m\log m + k^2 m\log\frac{km}{\varepsilon} + \frac{k^4}{\sqrt{\varepsilon}}\log\frac{k}{\varepsilon}) = O(\frac{kn+k^4}{\varepsilon}\log^2\frac{\rho n}{\varepsilon})$.

$\square$

Lemma 3.8. *The algorithm returns a path whose cost is no more than $(1+\varepsilon/3)$ times the cost of the shortest path in the approximation graph $\mathcal{G}$.*

Proof. $\square$

The theorem follows from Lemmas 3.7, 3.1 and 3.8.

Theorem 3.1. *Let $\mathcal{T}$ be a planar triangulation with n vertices such that the sum of the smallest k angles is at least π. Given two points on $\mathcal{T}$, one can compute a $(1+\varepsilon)$-approximate shortest path in $O(\frac{kn+k^4}{\varepsilon}\log^2\frac{\rho n}{\varepsilon})$ time, where ρ is the ratio of the maximum weight to the minimum weight.*

References

[1] P.K. Agarwal, S. Har-Peled, M. Sharir, and K.R. Varadarajan. Approximating shortest paths on a convex polytope in three dimensions. *Journal of ACM*, 44 (1997), 567–584.

[2] M. Ahmed. Constrained Shortest Paths in Terrains and Graphs. PhD thesis, University of Waterloo, Canada, 2009.

[3] M. Ahmed, S. Das, S. Lodha, A. Lubiw, A. Maheshwari and S. Roy. Approximation algorithms for shortest descending paths in terrains. *Journal of Discrete Algorithms*, 8 (2010), 214–230.

[4] M. Ahmed and A. Lubiw. Shortest descending paths through given faces. *Computational Geometry: Theory and Applications*, 42 (2009), 464–470.

[5] M. Ahmed and A. Lubiw. Shortest descending paths: towards an exact algorithm. *International Journal of Computational Geometry and Applications*, 21 (2011), 431–466.

[6] L. Aleksandrov, A. Maheshwari, and J.-R. Sack. Approximation algorithms for geometric shortest path problems. *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000, 286–295.

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

[8] M. de Berg and M. van Kreveld. Trekking in the Alps without freezing or getting tired. *Algorithmica*, 18 (1997), 306–323.

[9] J. Canny and J.H. Reif. New lower bound techniques for robot motion planning problems. *Proceedings of the 28th Annual IEEE Symposium on Foundation of Computer Science*, 1987, 49–60.

[10] T. Chan. Optimal output-sensitive convex hull algorithm in two and three dimensions. *Discrete and Computational Geometry*, 16 (1996), 361–368.

[11] J. Chen and Y. Han. Shortest paths on a polyhedron, part I: computing shortest paths. *International Journal of Computational Geometry and Applications*, 6 (1996), 127–144.

[12] S.-W. Cheng, J. Jin, A. Vigneron, and Y. Wang. Approximate Homotopic Shortest Paths in Weighted Regions. *International Journal of Computational Geometry and Applications*, 22 (2012), 83–102.

[13] S.-W. Cheng and J. Jin. Approximate shortest descending paths. *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2013.

[14] S.-W. Cheng and J. Jin. Shortest Paths on Polyhedral Surfaces and Terrains. *Proceedings of the 46th Annual Symposium on the Theory of Computing*, 2014.

[15] S.-W. Cheng, H.-S. Na, A. Vigneron, and Y. Wang. Approximate shortest paths in anisotropic regions. *SIAM Journal on Computing*, 38 (2008), 802–824.

[16] J. Choi, J. Sellen, and C.-K. Yap. Approximate Euclidean shortest path in 3-space. *Proceedings of the 10th Annual Symposium on Computational Geometry*, 1994, 41–48.

[17] S. Choi, J. Park, E. Lim, and W. Yu. Global path planning on uneven elevation maps. *Proceedings of the 9th International Conference on Ubiquitous Robots and Ambient Intelligence*, 2012, 49–54.

[18] K.L. Clarkson. Approximation algorithms for shortest path motion planning. *Proceedings of the 19th Annual ACM Symposium on Theory of Computation*, 1987, 56–65.

[19] K. Deng, X. Zhou, H.T. Shen, Q. Liu, K. Xu, and X. Lin. A multi-resolution surface distance model for k-nn query processing. *VLDB Journal*, 17 (2008), 1101–1119.

[20] R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Approximation Theory*, 10(3):227–236, 1974.

[21] E. Galin, A. Peytavie, N. Maréchal, and E. Guérin. Procedural generation of roads. *Eurographics*, 2010.

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

[23] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4 (1984), 373–395.

[24] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *SIGGRAPH*, 2003, 954–961.

[25] M. van Kreveld. On quality paths on polyhedral terrains. *IGIS'94*, LNCS 884, 1994, 113–122.

[26] M. Lanthier, A. Maheshwari, and J.-R. Sack. Approximating weighted shortest paths on polyhedral surfaces. *Algorithmica*, 30 (2001), 527–562.

[27] L. Liu and R. C.-W. Wong. Finding shortest path on land surface. *SIGMOD'11*, 433–444.

[28] C. Mata and J.S.B. Mitchell. A new algorithm for computing shortest paths in weighted planar subdivisions. *Proceedings of the 13th Annual Symposium on Computational Geometry*, 1997, 264–273.

[29] J.S.B. Mitchell, D.M. Mount and C.H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16 (19897), 647–668.

[30] J.S.B. Mitchell and C.H. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of ACM*, 38 (1991), 18–73.

[31] C.H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Information Processing Letters*, 20 (1985), 259–263.

[32] N.C. Rowe and R.S. Ross. Optimal grid-free path planning across arbitrarily contoured terrain with anisotropic friction and gravity effects. *IEEE Transactions on Robotics and Automation*, 6 (1990), 540–553.

[33] S. Roy, S. Das, and S.C. Nandy. Shortest monotone descent path problem in polyhedral terrain. *Computational Geometry: Theory and Applications*, 27 (2007), 115–133.

[34] Y. Schreiber and M. Sharir. An optimal-time algorithm for shortest paths on a convex polytope in three dimensions. *Proceedings of the 22nd annual symposium on Computational geometry*, 2006, 30–39.

[35] C. Shahabi, L.-A. Tang, and S. Xing. Indexing land surface for efficient knn query. *PVLDB*, 1 (2008), 1020–1031.

[36] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM Journal on Computing*, 15 (1996), 193–215.

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

[38] T.S. Tan. An Optimal Bound for High-Quality Conforming Triangulations. *Discrete and Computational Geometry*, 15 (1996), 169–193.

[39] K.R. Varadarajan and P.K. Agarwal. Approximating shortest paths on a non-convex polyhedron. *SIAM Journal on Computing*, 30 (2001), 1321–1340.

[40] C. Yu, J. Lee, and M.J. Munro-Stasiuk. Extensions to least-cost path algorithms for roadway planning. *International Journal of Geographical Information Science*, 17 (2003), 361–376.

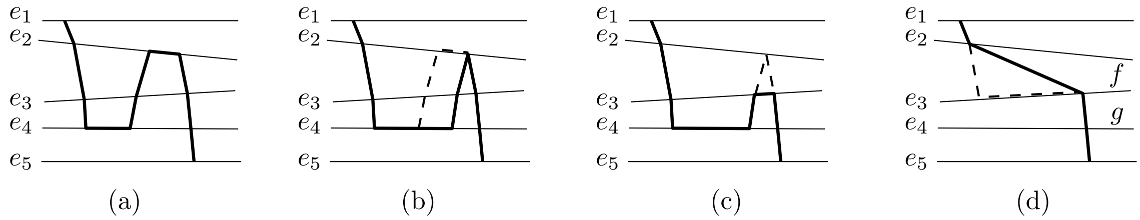


Figure 3: (a) A path with two critical links. (b) Slide the subpath between the two critical links to one side to eliminate one critical link. (c) Shortcutting the reflective bending point reduces the path cost. (d) Assume $w_f \leq w_g$. Eliminating the critical link as shown in the figure decreases the path cost. If $w_g \leq w_f$, the shortcut can be made inside g.

A Missing proofs in Section 3.2

Lemma A.1. *Let $e_1, e_2, \ldots, e_m$ be a sequence of edges such that for $i \in [1, m-1]$, e_i and e_{i+1} bound the same face, but e_i and e_{i+2} do not. Let P be a refraction path from e_1 to e_m with exactly two (possibly degenerate) critical links on e_j and e_ℓ for some $1 \leq j < \ell < m$ such that P crosses $e_1, \ldots, e_{\ell-1}$ at transversal nodes in order, rebounds at the critical link on e_ℓ, crosses $e_{\ell-1}, \ldots, e_{j+1}$ at transversal nodes in order, rebounds at the critical link at e_j, and finally crosses $e_{j+1}, \ldots, e_{m-1}$ at transveral nodes in order before reaching e_m. (Refer to Figure 3.) There exists a path Q from the source of P to the detination of P such that Q intersects $e_1, \ldots, e_m$ in order, $\mathrm{cost}(Q) \leq \mathrm{cost}(P)$, and either Q is a transversal path or Q contains exactly one critical link on e_1. Moreover, if Q has a critical link, then $e_j = e_1$.*

Proof. Sliding P's subpath between the two critical links while maintaining the directions of all links changes the path cost linearly. Slide the subpath to the side that does not increase the path cost until one of the critical links shrinks to a single point. Shortcut the reflective point. See Figure 3. The result is another path with two critical links but with a smaller cost, and the critical links are on $e_{\ell-1}$ and e_j, or e_ℓ and e_{j+1}. Repeat the above alternative sliding and shortcutting until the path has only one critical link. Inductively, it can be shown that the critical link must be on some edge e_i for $i \in [j, \ell]$. If the critical link is not on e_1, it can be eliminated as shown in Figure 3(d). ◻

Lemma A.2. *Let $P_{1,x}$, $P_{1,y}$, $P_{2,x}$, $P_{2,y}$ be four paths from r_1 to x, r_1 to y, r_2 to x, and r_2 to y, respectively, where r_1, r_2, x, y are four arbitrary points in the subdivision. Suppose that $P_{1,x}$ and $P_{2,y}$ cross at a point o. If $c_1 + \mathrm{cost}(P_{1,x}) \leq c_2 + \mathrm{cost}(P_{2,x})$ for some real numbers c_1 and c_2, and $\mathrm{cost}(P_{2,x}) \leq \mathrm{cost}(P_{2,y}[r_2, o]) + \mathrm{cost}(P_{1,x}[o, x])$, then $c_1 + \mathrm{cost}(P_{1,x}[r_1, o] \cdot P_{2,y}[o, y]) \leq c_2 + \mathrm{cost}(P_{2,y})$.*

Proof. Combining the two given inequalities gives $c_1 + \mathrm{cost}(P_{1,x}) \leq c_2 + \mathrm{cost}(P_{2,y}[r_2, o]) + \mathrm{cost}(P_{1,x}[o, x])$. The lemma follows because $P_{1,x} = P_{1,x}[r_1, o] \cdot P_{1,x}[o, x]$ and $P_{2,y} = P_{2,y}[r_2, o] \cdot P_{2,y}[o, y]$. ◻

A.1 Proof of Lemma 3.3

Lemma 3.3. *Let I be a type-III interval on an interior edge e of a strip S such that I does not contain any endpoint of e. Let P be a refraction path in S from r_I to a point p in the boundary of S such that P contains exactly one critical link, which is a subset of I, and p and r_I lie on distinct boundary edges of S. Then, P is not the shortest path in S from r_I to p, or there exists a Steiner point or vertex r on the same boundary edge as r_I such that $d(r) + \mathrm{cost}(P^*) < d(r_I) + \mathrm{cost}(P)$, where P^* is the shortest path in S from r to p.*

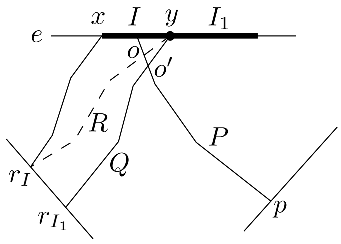


Figure 4: Interval I_1 causes interval I to be trimmed at y. P is the refraction path from r_I to p, which uses part of I as a critical link. Q is the shortest transversal path from r_{I_1} to y. R is the shortest transversal path from r_I to y.

Proof. Recall that when I was created, I contained an endpoint of e, say v. Therefore, I must have been trimmed by the algorithm in order that I does not contain v currently. It means that some interval $I_1 \subset e$ caused I to be trimmed. By the working of the algorithm, I must contain the endpoint of I_1 further from v, say y, and I is trimmed at y. See Figure 4.

Let R and Q be the shortest transversal paths from r_I and r_{I_1}, respectively, to p. Let P' denote the subpath of P after the critical link. Since R is a transversal path, by Snell's law, R cannot cross $P[r_I, x]$, which means that $P[r_I, x]$, xy and R bound a closed region. It follows that P' must cross R at some point o in order to reach p. Similarly, P' must cross Q at some point o'.

It is sufficient to prove that that $\mathrm{cost}(R[r_I, o]) < \mathrm{cost}(P[r_I, o])$ or $d(r_{I_1}) + \mathrm{cost}(Q[r_{I_1}, o']) < d(r_I) + \mathrm{cost}(P[r_I, o'])$. In the first case, $R[r_I, o] \cdot P[o, p]$ is a shorter path than P from r_I to p. In the second case, $d(r_{I_1}) + \mathrm{cost}(Q[r_{I_1}, o') \cdot P[o', p]) < d(r_I) + \mathrm{cost}(P)$.

If $\mathrm{cost}(R[r_I, o]) < \mathrm{cost}(P[r_I, o])$, we are done. Suppose not. That is, $\mathrm{cost}(P[r_I, o]) \le \mathrm{cost}(R[r_I, o])$. Apply Lemma A.1 on the $R[z, o] \cdot P[o, r_I]$ to obtain a shorter path X from z to r_I. So either X is a transversal path, or its first link is a critical link on e. In the former case, $\mathrm{cost}(X) \ge \mathrm{cost}(R)$, while in the latter case, $\mathrm{cost}(X) \ge \mathrm{cost}(P[r_I, y] \cdot yz)$. So $\min\{\mathrm{cost}(P[r_I, y] \cdot yz), \mathrm{cost}(R)\} < \mathrm{cost}(P[r_I, o] \cdot R[o, p])$. By the assumption that $\mathrm{cost}(P[r_I, o]) \le \mathrm{cost}(R[r_I, o])$, we have $\min\{\mathrm{cost}(P[r_I, y] \cdot yz), \mathrm{cost}(R)\} < \mathrm{cost}(R)$, or equivalently,

$$\mathrm{cost}(P[r_I, y] \cdot yz) < \mathrm{cost}(R). \tag{1}$$

For the sake of contradiction, assume that $d(r_I) + \mathrm{cost}(P[r_I, o']) \le d(r_{I_1}) + \mathrm{cost}(Q[r_{I_1}, o'])$. One can use the same argument to show that $d(r_I) + \min\{\mathrm{cost}(P[r_I, y] \cdot yz), \mathrm{cost}(R)\} < d(r_{I_1}) + \mathrm{cost}(Q)$. Then, it follows from (1) that $d(r_I) + \mathrm{cost}(P[r_I, y] \cdot yz) < d(r_{I_1}) + \mathrm{cost}(Q)$. But then I should have caused the algorithm to prune away I_1 completely, a contradiction. □

A.2 Proof of Lemma 3.4

Lemma 3.4. *Suppose that* $\mathsf{pred}(I_1)$ *and* $\mathsf{pred}(I_2)$ *are oriented from left to right and* $\mathsf{pred\_idx}(I_1) < \mathsf{pred\_idx}(I_2) \le i$. *Let* s_i *be the starting endpoint of* $\mathsf{pred}(I_i)$. *For* $i \in [1, 2]$ *and for all* $y \in I_i$, *let* $P_{i,x}$ *denote the refraction path from* r_{I_i} *through* $\mathsf{pred}(I_i)$ *to* x *that defines* I_i. *For* $i \in [1, 2]$ *and for all* $x \in I_1 \cap I_2$, *let* $Q_{i,x}$ *be the shortest path inside the strip from* r_{I_i} *to* x.

(i) *Suppose that* $d(r_{I_1}) + \mathrm{cost}(P_{1,x}) \le d(r_{I_2}) + \mathrm{cost}(P_{2,x})$ *for some point* $x \in I_1 \cap I_2$. *If* s_2 *is to the right (resp. left) of* $P_{1,x}$ *with respect to its orientation from* r_{I_1} *to* x, *then for every point* $y \in I_2$ *to the right (resp. left) of* x, $d(r_{I_1}) + \mathrm{cost}(Q_{1,y}) \le d(r_{I_2}) + \mathrm{cost}(P_{2,y})$.

(ii) *If* $d(r_{I_2}) + \mathrm{cost}(P_{2,x}) \le d(r_{I_1}) + \mathrm{cost}(P_{1,x})$ *for some point* $x \in I_1 \cap I_2$, *then for every point* $y \in I_1$ *to the left of* x, $d(r_{I_2}) + \mathrm{cost}(Q_{2,y}) \le d(r_{I_1}) + \mathrm{cost}(P_{1,y})$.

Proof. Consider (i). See Figures 5(a–c). Suppose that s_2 is to the right of $P_{1,x}$ with respect to its orientation from r_{I_1} to x. Let y be a point in I_2 to the right of x. Since $\mathsf{pred\_idx}(I_1) <$

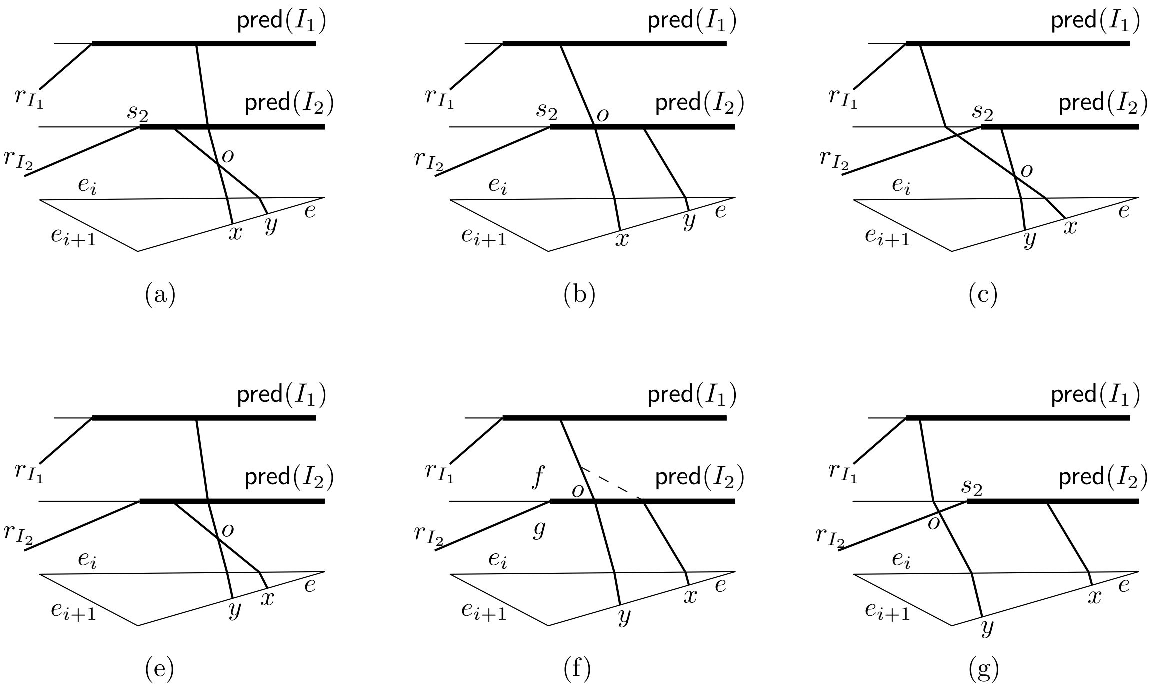


Figure 5: Illustrations for the proof of Lemma 3.4. The top row is for (i) and the bottom row is for (ii).

$\mathsf{pred\_idx}(I_2) \leq i$ and r_{I_2} is on the boundary of the strip, the subpath of $P_{2,y}$ after the critical link must cross $P_{1,x}$ at some point o. Suppose that it crosses the subpath of $P_{2,x}$ after the critical link. Refer to Figure 5(a). Since $P_{1,x}[o,x]$ and $P_{2,y}[o,y]$ cross the same sequence of edges in order and $P_{2,x}$ is the shortest one among paths with the same edge sequence from r_{I_2} to x,

$$\mathrm{cost}(P_{2,x}) \leq \mathrm{cost}(P_{2,y}[r_{I_2}, o]) + \mathrm{cost}(P_{1,x}[o,x]). \tag{2}$$

By Lemma A.2, $d(r_{I_1}) + \mathrm{cost}(P_{1,x}[r_{I_1}, o] \cdot P_{2,y}[o,y]) \leq d(r_{I_2}) + \mathrm{cost}(P_{2,y})$. The lemma follows because $Q_{1,y}$ by definition is no longer than $P_{1,x}[r_{I_1}, o] \cdot P_{2,y}[o,y]$. One can show that (2) holds for the other two cases left. The first case is that s_2 is to the right of $P_{1,x}$ and $P_{1,x}$ crosses the critical link of $P_{2,y}$. Refer to Figure 5(b): The second case is that s_2 is to the left of $P_{1,x}$. Refer to Figure 5(c).

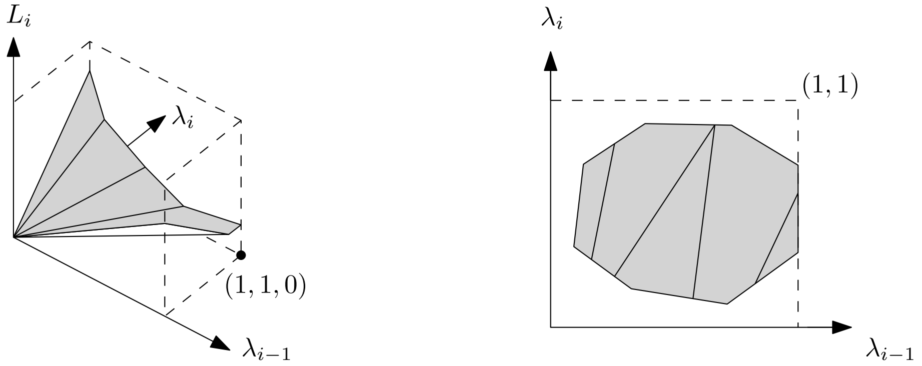
Consider (ii). Lemma A.2 is applicable if the following triangle inequality holds:

$$\mathrm{cost}(P_{1,x}) \leq \mathrm{cost}(P_{1,y}[r_{I_1}, o]) + \mathrm{cost}(P_{2,x}[o,x]). \tag{3}$$

If so, Lemma A.2 implies that $d(r_{I_2}) + \mathrm{cost}(P_{2,x}[r_{I_2}, 0] \cdot P_{1,y}[o,y]) \leq d(r_{I_1}) + \mathrm{cost}(P_{1,y})$. Since $Q_{2,y}$ is the shortest path in the strip from r_{I_2} to y, we thus obtain $d(r_{I_2}) + \mathrm{cost}(Q_{2,y}) \leq d(r_{I_1}) + \mathrm{cost}(P_{1,y})$ as stated in (ii). So it remains to prove (3).

There are three cases depending on how $P_{1,y}$ crosses $P_{2,x}$ as shown in Figures 5(e–g). If the crossing happens after the critical links of both $P_{1,y}$ and $P_{2,x}$ (Figure 5(e)), then (3) holds because $P_{1,y}[r_{I_1}, o] \cdot P_{2,x}[o,x]$ and $P_{1,x}$ cross the same edge sequence and $P_{1,x}$, being a refraction path, is the shortest one among such paths.

Suppose that $P_{1,y}$ crosses the critical link of $P_{2,x}$ as shown in Figure 5(f). We cannot immediately conclude that (3) holds this time because $P_{1,y}[r_{I_1}, o] \cdot P_{2,x}[o,x]$ and $P_{1,x}$ do not cross the same edge sequence—$P_{1,y}[r_{I_1}, o] \cdot P_{2,x}[o,x]$ has a critical link on the same edge as $\mathsf{pred}(I_2)$ while $P_{1,x}$ does not. But that critical link is redundant and can be eliminated without increasing the path cost as shown by the dashed edge in Figure 5(f). Let X be the path from r_{I_1} to x obtained after the shortcut. Now X and $P_{1,x}$ cross the same edge sequence, so $\mathrm{cost}(P_{1,x}) \leq \mathrm{cost}(X) \leq \mathrm{cost}(P_{1,y}[r_{I_1}, o]) + \mathrm{cost}(P_{2,x}[o,x])$. Therefore, (3) still holds.



(a) (b)

Figure 6: (a) If e_{i-1} and e_i meets at $e_{i-1}(0)$ and $e_i(0)$, then the graph of L_i passes through the origin. (b) The domain of of F_i^* is a convex polygon inside the unit square, and the projections of the edges of the graph of F_i^* do not intersect in the interior of the domain polygon.

The last case is that the subpath of $P_{1,y}$ after its critical link crosses the subpath of $P_{2,x}$ before the critical link of $P_{2,x}$. Refer to Figure 5(g). Let p_1 be the node of $P_{1,y}$ before o and p_2 be the node of $P_{2,x}$ after o. Let $X = P_{1,y}[r_{I_1}, p_1]) \cdot p_1p_2 \cdot P_{2,x}[p_2, x]$. The resulting path has at most three critical links: one on the same edge as $\mathsf{pred}(I_1)$, critical link p_1p_2 on an edge e_j for some $j < i$, and one on the same edge as $\mathsf{pred}(I_2)$.

If e_j happens to be the edge containing $\mathsf{pred}(I_2)$, then $p_2 = s_2$ and the p_1p_2 merge with the critical link on $\mathsf{pred}(I_2)$ to form one critical link. Then, we can shortcut this merged critical link as in Figure 5(f) and conclude that (3) holds.

In general, e_j is different from the edge containing $\mathsf{pred}(I_2)$. Let X' be the prefix of X from r_{I_1} to and including the critical link on the same edge as $\mathsf{pred}(I_1)$. Let X'' be the suffix of X after this critical link. Apply Lemma A.1 to X'' to shorten it to a transversal path Y. The union of X' and Y is a path from r_{I_1} to x that has the same edge sequence as $P_{1,x}$. Therefore, $\text{cost}(P_{1,x}) \le \text{cost}(X' \cup Y) \le \text{cost}(P_{1,y}[r_{I_1}, o]) + \text{cost}(P_{2,x}[o, x])$, i.e. (3) holds. $\square$

A.3 Proof of Lemma 3.5

Proof. Let $e_1, e_2, \ldots$ be the interior edges of the same strip such that e_1 and e are in the same face and e_i and e_{i+1} are in the same face. For convenience, let $e_0 = e$. Parametrize edges uniformly by a parameter in $[0, 1]$. Use $e_i(\lambda_i)$ to denote the point on e_i with parameter λ_i. So $e_i(0)$ and $e_i(1)$ are endpoints of e_i. Fix an arbitrary point p in e. Let $F_i(\lambda_0, \lambda_i)$ be the function $[0, 1]^2 \to \mathbb{R}$ that represents the p-costs of the shortest p-cost transversal paths from points on e to points on e_i. Let $L_i(\lambda_{i-1}, \lambda_i)$ be the function $[0, 1]^2 \to \mathbb{R}$ that represents the p-cost of segments from a point on e_{i-1} to a point on e_i. We have $F_1 = L_1$ and $F_{i+1}(\lambda_0, \lambda_i) = \min_{\lambda_i \in [0,1]} F_i(\lambda_0, \lambda_i) + L_{i+1}(\lambda_i, \lambda_{i+1})$. We are only interested in the values of F_{i+1} that can be realized by a $\lambda_i \in (0, 1)$, because otherwise the path bends at a vertex of e_i, and such paths will not be considered by the algorithm. Denote this restriction of F_{i+1} by F_{i+1}^*.

It is clear that L_i is convex and piecewise linear. Moreover, L_i has $O(\frac{1}{\sqrt{\varepsilon}})$ linear pieces, and those linear pieces meet at a single point (See Figure 6). Inductively, one can also show that F_i is a convex piecewise linear function. If a $\lambda_i \in (0, 1)$ satisfies the equation $F_{i+1}(\lambda_0, \lambda_i) = F_i(\lambda_0, \lambda_i) + L_{i+1}(\lambda_i, \lambda_{i+1})$, then it must be the case that $\partial F_i(\lambda_0, \lambda_i)/\partial \lambda_i^+ + \partial L_{i+1}(\lambda_i, \lambda_{i+1})/\partial \lambda_i^+ \ge 0$, and $\partial F_i(\lambda_0, \lambda_i)/\partial \lambda_i^- + \partial L_{i+1}(\lambda_i, \lambda_{i+1})/\partial \lambda_i^- \ge 0$ by the convexity of F_i and L_{i+1}.

Inductively, we claim that the domain of F_i^* is a convex polygon of size $O(\frac{i}{\sqrt{\varepsilon}})$, the projection of the graph of F_i^* does not have a vertex in the interior of the domain, and the projection of any edge of the graph of F_i^* is not parallel to the λ_i axis.

$\square$

A.4 Proof of Lemma 3.6

Lemma 3.6. *Let I_1 and I_2 be two type-V intervals on the boundary edge e of a strip S. If $d(r_{I_2}) + \mathrm{cost}_{\pentagon}(P_{2,x}) \leq d(r_{I_1}) + \mathrm{cost}_{\pentagon}(P_{1,x})$ for some point $x \in I_1 \cap I_2$, then $I_1 \cap \gamma$ can be trimmed, where γ is the part of the boundary of S delimited by r_{I_2} and x that excludes r_{I_1}.*

Proof. Let y be any point in $I_1 \cap \gamma$. It suffices to prove that there exists a Steiner point or vertex r in the boundary of S such that r is the root of an type-V interval on e that overlaps with I_1 and $d(r) + \mathrm{cost}(Q_y) \leq d(r_{I_1}) + \mathrm{cost}_{\pentagon}(P_{1,y})$.

Since $y \in I_1 \cap \gamma$, $P_{1,y}$ must cross $P_{2,x}$, say at o. The concatenation of $P_{1,y}[r_{I_1}, o]$ and $P_{2,x}[o, x]$ is also a transversal path, so

$$\mathrm{cost}_{\pentagon}(P_{1,y}[r_{I_1}, o]) + \mathrm{cost}_{\pentagon}(P_{2,x}[o, x]) \geq \mathrm{cost}_{\pentagon}(P_{1,x}). \tag{4}$$

Let Q be the path with the minimum $\mathrm{cost}_{\pentagon}$ from r_{I_2} to y that crosses the sequence of interior edges between r_{I_2} and y. The concatenated path $P_{2,x}[r_{I_2}, o] \cdot P_{1,y}[o, y]$ is not shorter than Q.

If Q is a transversal path, then $Q = P_{2,y}$ and

$$\mathrm{cost}_{\pentagon}(P_{2,x}[r_{I_2}, 0]) + \mathrm{cost}_{\pentagon}(P_{1,y}[o, y]) \geq \mathrm{cost}_{\pentagon}(P_{2,y}). \tag{5}$$

Adding (4) and (5) yields $\mathrm{cost}_{\pentagon}(P_{1,y}) + \mathrm{cost}_{\pentagon}(P_{2,x}) \geq \mathrm{cost}_{\pentagon}(P_{1,x}) + \mathrm{cost}_{\pentagon}(P_{2,y})$. Therefore, $d(r_{I_2}) + \mathrm{cost}_{\pentagon}(P_{2,y}) + \mathrm{cost}_{\pentagon}(P_{1,x}) \leq d(r_{I_2}) + \mathrm{cost}_{\pentagon}(P_{1,y}) + \mathrm{cost}_{\pentagon}(P_{2,x})$ which is at most $d(r_{I_1}) + \mathrm{cost}_{\pentagon}(P_{1,y}) + \mathrm{cost}_{\pentagon}(P_{1,x})$ by the assumption of the lemma. It follows that $d(r_{I_2}) + \mathrm{cost}_{\pentagon}(P_{2,y}) \leq d(r_{I_1}) + \mathrm{cost}_{\pentagon}(P_{1,y})$.

If Q is not a transversal path, the analysis in the previous paragraph gives

$$d(r_{I_2}) + \mathrm{cost}_{\pentagon}(Q) \leq d(r_{I_1}) + \mathrm{cost}_{\pentagon}(P_{1,y}).$$

Since Q is not a transversal path, it contains some vertices in its interior. Let r be the last vertex in Q before y. The subpath Q_y from r to y is a transversal path. Also, $d(r) + \mathrm{cost}_{\pentagon}(Q_y) \leq d(r_{I_2}) + \mathrm{cost}_{\pentagon}(Q)$. It follows that $d(r) + \mathrm{cost}_{\pentagon}(Q) \leq d(r_{I_1}) + \mathrm{cost}_{\pentagon}(P_{1,y})$. $\square$