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 $O\left(\frac{kn+k^4\log(k/\varepsilon)}{\varepsilon}\log^2\frac{\rho n}{\varepsilon}\right)$ 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 $O\left(\frac{n^4}{\varepsilon}\log\frac{n}{\varepsilon}\log^2\frac{\rho n}{\varepsilon}\right)$ in the worst case. Our algorithm relies on a new triangulation refinement method, which produces a triangulation of size $O(n+k^2)$ such that no triangle has two angles less than $\min\{\pi/(2k),\pi/12\}$.

1 Introduction

Let $\mathcal{T}$ be a planar subdivision with n vertices. We assume that $\mathcal{T}$ forms a simplicial complex: each face of $\mathcal{T}$, except the unbounded face, is a triangle, and the intersection between any two faces is either empty, or is a common edge or vertex. 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 of 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 loss 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 [8]. They also give an example showing that an optimal path could have $\Omega(n^2)$ links. There has been extensive research on lowering the dependency on n by discretizing the geometric environment [1, 2, 6, 7, 9]. 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. [2] and Sun and Reif [9]. Aleksandrov et al. [2] obtained a running time of $O(\frac{n}{\sqrt{\varepsilon}}\log\frac{n}{\varepsilon}\log\frac{1}{\varepsilon})$, the hidden constant being $O(\Gamma\log(\rho/\theta_{\min}))$, where $\theta_{\min}$ is the smallest angle in $\mathcal{T}$, Γ is the average of the reciprocals of the sines of the angles in $\mathcal{T}$, and ρ is the ratio between the largest and the smallest weight. Sun and Reif [9] 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 [2, 9]. Cheng et al. [3] use a different discretization scheme and obtained a running time of $O\left(\frac{\rho\log\rho}{\varepsilon}n^3\log\frac{\rho n}{\varepsilon}\right)$, which does not depend on any geometric parameter such as $\theta_{\min}$ or Γ.

In this paper, we present an algorithm for finding a $(1+\varepsilon)$-approximate shortest path in weighted regions that runs in $O\left(\frac{kn+k^4\log(k/\varepsilon)}{\varepsilon}\log^2\frac{\rho n}{\varepsilon}\right)$ 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

*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

a polynomial in n, $1/\varepsilon$ and $\log \rho$ in the worst case. The exponent of the polynomial in n is also much smaller than the result by Mitchell and Papadimitriou [8].

The improvement comes from a few innovations. We observe that the discretization schemes in [1, 9] 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 their 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 predetermined. 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 [9] 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 maximal 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 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 node is *transversal* if its two incident 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 links (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. ([8]) *There exists a shortest path P such that it is a refraction path, and for every pair of critical links that appear consecutively 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. Let $\theta = \min\{\pi/(2k), \pi/12\}$, so the k-th smallest angle is at least $\pi/k \geqslant 2\theta$. In this section, we show how to refine the original subdivision $\mathcal{T}$ into another subdivision $\mathcal{T}^*$ 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 $\frac{11}{15}\pi$ [10]. We present a simpler algorithm with worse angle bound (while still good enough for our purposes), but with a number of triangles 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_ip_{i+1} is parallel to the angle bisector of the largest angle in f_i.[1] By construction, a propagation polyline does not intersect itself or another propagation 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. For sake of contradiction, assume that e_i and e_j bound the same face, with $i < j-1$, and $j-i$ is minimum. By our assumption that, for any r, the edges e_r and e_{r+2} do not 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. $\sum_{r=i}^{j-1} \angle(e_r, e_{r+1}) \geqslant \pi$) in order for e_i and e_j to bound the same face. By minimality of $j-i$, the pairs $\{e_r, e_{r+1}\}$ in this sum must be distinct. As all angles in this sum are less than 2θ, and the k-th smallest angle is at least 2θ, we must have $j - i < k$. So $\sum_{r=i}^{j-1} \angle(e_r, e_{r+1}) < 2k\theta \leqslant \pi$, a contradiction.

We have just proved that no two edges in the sequence $e_0, \ldots, e_m$ bound the same face. As the k-th smallest angle is at least 2θ, it follows that $m < k$. □

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

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

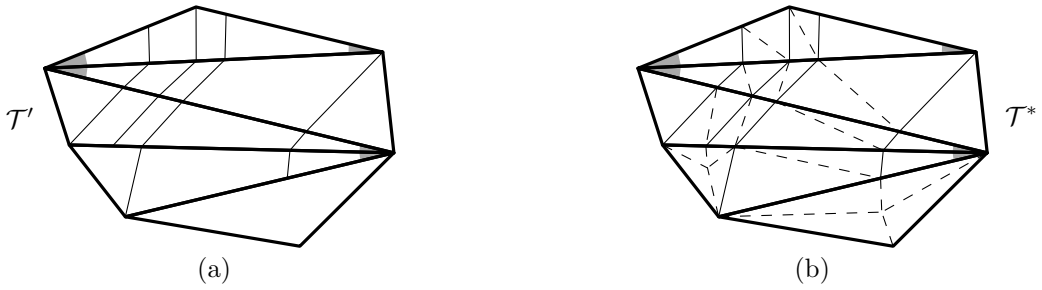

Figure 1: (a) The original subdivision is shown in bold, and its angles that are less than 2θ are shaded. The light polylines are propagation polylines. (b) Obtain the final triangulation by adding diagonals to quadrilaterals and triangulating triangular faces using incenters.

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. □

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 triangular 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 repetition.*

Proof. The subdivision $\mathcal{T}^*$ can be constructed in time linear in its size, 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 less than k^2 vertices of the former type, because, by Lemma 2.2, less 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 ending 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 a diagonal into two triangles each of which has at most one angle less than $\theta \leqslant \pi/12$. 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 is output 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.

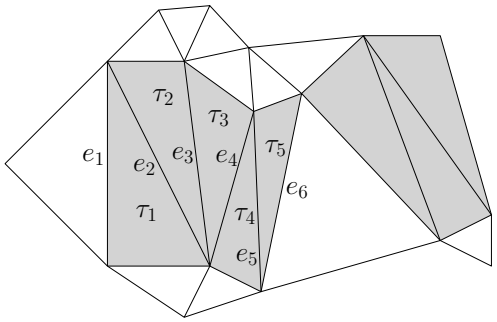

Figure 2: A subdivision comprising two strips (shaded).

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 polyline. 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 possible.

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. □

Strips. We say that a triangle τ of $\mathcal{T}^*$ is *thin* if it has exactly one angle less than θ. The two edges that make an angle less than θ are called *long edges*, and the other edge, is *short*. Consider the subgraph of the dual graph of $\mathcal{T}^*$, where each node represents a thin triangle, and two nodes are adjacent if the two corresponding triangles share a long edge. By Theorem 2.1 (ii), this graph has degree at most two, and by (iii), it contains no cycle. It follows that any connected component is a path, and by (iii), it has $O(k)$ edges. The set of thin triangles corresponding to each such path will be called a *strip*. (See Figure 2.) In other words, a strip is a maximal sequence of thin triangles that share a long edge. To summarize:

LEMMA 2.3. *The set of thin triangles of $\mathcal{T}^*$ is partitioned into strips of $O(k)$ triangles each.*

Consider a strip $\tau_1, \dots, \tau_{m-1}$. Let e_i be the common edge of τ_{i-1} and τ_i, and let e_1 and e_m be the other long edge of τ_1 and τ_{m-1}, respectively. We say that $(e_1, \dots, e_m)$ is the *edge sequence* of this strip. The edges $e_2, \dots, e_{m-1}$ are *interior edges* of the strip, and the other edges of the strip are *boundary edges*. Hence, the boundary edges are e_1, e_m, and the short edges of the strip.

3 Approximation graph $\mathcal{G}$

In this section, we show how to obtain a graph $\mathcal{G}$ that contains a $(1+\varepsilon)$-approximate shortest path for our original weighted region problem. Section 4 will present an efficient algorithm for computing an approximate shortest path in $\mathcal{G}$. In order to compute $\mathcal{G}$, we first compute a refinement $\mathcal{T}^*$ of $\mathcal{T}$ using Theorem 2.1, and then we discretize $\mathcal{T}^*$ as follows.

We follow the same approach used by [1, 9] to discretize boundary edges of strips and edges outside strips. Interior edges of a strip are not discretized. Take an edge vu that is not an interior edge of any strip. Let $\theta = \min\{\pi/(2k), \pi/12\}$ be the same value as in Theorem 2.1. Since vu is not an interior edge, at least one of the two angles at u with side vu is larger than θ. If both angles are larger than θ, let α_{vu} be the smallest of the two. Otherwise, define α_{vu} to be the larger one. So we have $\alpha_{vu} > \theta = \Omega(1/k)$. Let L be the length of the Euclidean shortest path from s to t, which can be computed in $O(n \log n)$ time [4]. 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. We place p_0 at distance $c_0 \varepsilon L/n^2$ from v, unless the length $|vu|$ is smaller. Then for $i > 0$, we place p_i at distance $(1 + c_1 \varepsilon \sin\alpha(v))^i |vp_0|$ from v, until this length exceeds $|vu|$. Hence, we have $|p_{i-1}p_i| = |vp_{i-1}| c_1 \varepsilon \sin\alpha_{vu}$. Similarly, we create Steiner points $q_0, q_1, \dots$ such that $|q_0 u| = c_0 \varepsilon L/n^2$, and for $i > 0$, $|q_{i-1}q_i| = |uq_{i-1}| c_1 \varepsilon \sin\alpha_{uv}$.

Let $C(s, 2\rho L)$ and $D(s, 2\rho L)$ denote the circle and the disk centered at s with radius $2\rho L$, respectively. For

any edge vu that intersects the circle $C(s, 2\rho L)$, we add one Steiner point in the interior of $vu \cap D(s, 2\rho L)$. (It will be needed in the proof of Lemma 3.1, to ensure that we always have a Steiner point to snap to.)

Finally, all Steiner points outside the disk $D(s, 2\rho L)$ are removed. Observe that, for $\varepsilon < 1$, any $(1+\varepsilon)$-approximate shortest path has length at most $2\rho L$, and hence lies in $D(s, 2\rho L)$. Removing those Steiner points does not affect the correctness of the algorithm but, as we shall see, it allows us to obtain a bound on the number of Steiner points that is independent of the geometric 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 each pair of graph vertices in the same face, and each pair of graph vertices (which may not be in the same face) in the boundary of the same strip. Let (p, q) be a pair of graph vertices between which there is a graph edge. The weight of the edge is denoted by $\mu(p, q)$. If p and q are in the same face of $\mathcal{T}^*$, then $\mu(p, q)$ is simply the cost of the straight segment connecting them. If p and q are in the boundary of some strip, then $\mu(p, q)$ is the cost of the shortest path inside the strip from p to q.

We set $c_0 = 1/128$ and $c_1 = 1/16$. The Lemma below shows that $\mathcal{G}$ gives an accurate discretization of our problem.

LEMMA 3.1. *Each edge of $\mathcal{T}^*$ has $O(\frac{k}{\varepsilon} \log \frac{\rho n}{\varepsilon})$ Steiner points, and the shortest path in $\mathcal{G}$ is a $(1+\varepsilon/2)$-approximate shortest path in $\mathcal{T}^*$.*

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

Let P be a shortest path from s to t in $\mathcal{T}^*$, so P lies inside $D(s, 2\rho L)$. We convert P to a path in $\mathcal{G}$ by applying the rule (a) below, then (b) and (c). After applying rule (a), any node of P that is not a vertex of $\mathcal{T}^*$ is at distance more than $c_0\varepsilon L/n^2$ from the endpoints of the edge that contains it, and hence this edge must contain Steiner points.

(a) For each vertex v of $\mathcal{T}^*$, let N_v denote the intersection of the union of the faces incident to v with the radius-$c_0\varepsilon L/n^2$ disk centered at v. Let p_v and q_v be the first and last nodes of P that lie in the interior of N_v. Snap both p_v and q_v to v and remove the subpath between them.

(b) For each maximal subpath P_S of P contained in a strip S, if the first node p_S of P_S is not a vertex of $\mathcal{T}^*$, snap it to the nearest Steiner point along the edge of $\mathcal{T}^*$ containing p_S. Do the same with the last node q_S of P_S.

(c) 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 $\text{cost}(P') \leqslant (1 + \varepsilon/2)\,\text{cost}(P)$.

In Case (a), 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 $k \leqslant 4n-8$. By Theorem 2.1(i), $\mathcal{T}^*$ has no more than $n + k^2 + k < 16n^2$ vertices. The value we chose for c_0 implies that the total error due to snapping to vertices of $\mathcal{T}^*$ is at most $\varepsilon L/4 \leqslant (\varepsilon/4)\ \text{cost}(P)$.

Now consider the error due to snapping nodes to Steiner points. Take any such node p. Let e be the edge containing p. In Case (b), p does not belong to any strip, thus all face angles adjacent to e are at least θ. In Case (c), assume that $p = p_S$, without loss of generality. Then p is on the boundary of the strip S, so we are in one of these two cases: Either e is a short edge of a triangle τ in S, and therefore, the two angles of τ adjacent to e are at least θ, or e is one of the two long edges bounding S, and thus the two angles of the triangle τ' bounding e from outside S 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 $\phi \geqslant \theta$. Note that v is an endpoint of e. Then, $|pq| \geqslant |vp| \sin\phi$. The distance between p and its nearest Steiner point on e is no more than $2(c_1\varepsilon \sin\phi) \cdot |vp| \leqslant 2c_1\varepsilon|pq| \leqslant \frac{\varepsilon}{8}|pq|$. 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 $(\varepsilon/4)\ \text{cost}(P)$. □

4 Computing an approximate shortest path in $\mathcal{G}$

Our approximation graph $\mathcal{G}$ 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. So computing the shortest path using Dijkstra's algorithm directly is too slow. Sun and Reif proposed a BUSHWHACK algorithm [9], 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.

During the course of the algorithm, any vertex or Steiner point will be marked as *visited* or *unvisited*. For any visited point v, we record a cost $d(v)$ which is the (approximate) cost of the current best path from s to v. Initially, only s is marked as visited and we set $d(s) = 0$.

We also maintain a set of *intervals*, that are subsegments of edges of $\mathcal{T}^*$. 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 is created when r_I is visited, and thus $d(r_I)$ is known. This interval 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 $d(I)$ of I is the minimum of the costs of points in I. The intervals are stored in a priority queue, and we will repeatedly extract the interval with lowest cost.

There are five types of intervals. See Figure 3. 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 $d(r_I) + \text{cost}(r_I x)$. 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 refraction path inside the strip with 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. No efficient algorithm is known for computing 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 with weight w_f is simply the 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 $\text{cost}_{\hexagon}$ to denote the cost of a path under this polygonal metric. One can verify that for every segment ℓ, $\cos(\pi/h)\,\text{cost}_{\hexagon}(\ell) \leqslant \text{cost}(\ell) \leqslant \text{cost}_{\hexagon}(\ell)$. Setting $h = O(1/\sqrt{\varepsilon})$ with an appropriate constant, we can obtain $\text{cost}_{\hexagon}(\ell) \leqslant (1 + \varepsilon/3)\,\text{cost}(\ell)$. For a point x in a type-V interval, we define the cost of x to be $d(r_I)+\text{cost}_{\hexagon}(T(r_I, x))$, where $T(r_I, x)$ is the transversal path inside the strip from r_I to x with the minimum $\text{cost}_{\hexagon}$.

Our algorithm consists of two main functions ProcessPoint or ProcessInterval described below. ProcessPoint(v) creates intervals rooted at v and enqueue them. Initially, we call ProcessPoint(s). Then we repeatedly extract the interval I with smallest cost from the priority queue, and call ProcessInterval(I) until $d(t)$ is determined. When creating an interval in ProcessPoint or 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. 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. The intervals in each group of an edge e are stored in a balanced binary search tree, sorted according to the positions of their roots along the boundary of the face or the strip that is being considered. It will allow us to prune and trim intervals efficiently. An interval that is dequeued is kept in this data structure, as it may be used for trimming at a later stage.

ProcessPoint(a Steiner point or vertex v)

1. For every edge e such that e is not an interior edge of any strip and e is the edge opposite v in the same face, create and enqueue type-I intervals on e with root v.
2. For every edge e_v that contains v and is not an interior edge of any strip, create and enqueue type-II intervals on e_v with v as their common root.
3. For every strip S that contains v on its boundary and for every interior edge e of S, create and enqueue type-III intervals on e with v as their common root.
4. For every strip S that contains v on its boundary and for every boundary edge e 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

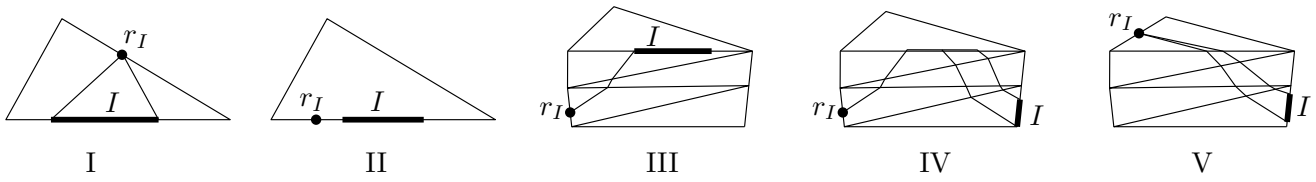

Figure 3: Five types of intervals.

of S that I propagates to, with r_I as their common root.

2. Otherwise, let p be the Steiner point or vertex in I that has the smallest cost.
   (a) If p has not been visited yet, then mark p as visited, set $d(p) = d(I)$, and call ProcessPoint(p).
   (b) Split I into two intervals I_1 and I_2. Interval I_1 only contains p and has cost $d(p)$. Interval I_2 is the smallest interval that contains the the remaining vertices or Steiner points of I, and its cost is the minimum cost of its two endpoints. We insert I_2 into the priority queue, but not I_1, since p has already been visited.

In Step 2b of ProcessInterval, we can partition I into I_1 and I_2, because 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. We keep an interval I_1 containing only p, without enqueueing it. The reason for doing so is that we will trim the intervals created later so that they do not overlap I_1. As these intervals have higher costs, p will not be inserted again into an interval of the same group as I_1. (Interval groups are explained below.) Even though I_1 is not enqueued, it is recorded in the balanced binary search tree storing its group.

We elaborate below on the creation of intervals. For this purpose, let S denote a strip and let $e_1, e_2, \ldots$ be the edge sequence of S. Orient these edges from left to right in a consistent manner, as follows. We denote by a_i and b_i the left and right endpoints of e_i, respectively. The left and right endpoints of e_1 are defined arbitrarily. The other edges are oriented inductively. So for $i > 1$, if e_{i-1} and e_i meet at a_{i-1}, then we set $a_i = a_{i-1}$, and b_i is the other endpoint of e_i. Otherwise, e_{i-1} and e_i meet at b_{i-1}, and we set $b_i = b_{i-1}$.

4.1 Type-I intervals. Refer to step 1 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 cover 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 [9] 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.

4.2 Type-II intervals. Refer to step 2 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.

4.3 Type-III intervals. Refer to Step 3 of ProcessPoint(v). For any edge $e_i = a_ib_i$ 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 4.1 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.

The proof of the Lemma below will appear in the full version of this paper.

LEMMA 4.1. *Let S be a strip. For every point x on 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 determine whether $R_{uv}(x,u)$ (resp. $R_{uv}(x,v)$) exists in $O(1)$ time and if so, report the destination and cost of $R_{uv}(x,u)$ (resp. $R_{uv}(x,v)$) in $O(1)$ 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.

4.4 Type-IV intervals. Refer to Step 1 of ProcessInterval(I). Suppose that I is a type-III interval on some interior edge of a strip S. If I extends all the way to an endpoint of the edge it is on, 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 4.3 below shows that we do not lose optimal paths in the latter case. By Lemma 4.1, each interval can be created in $O(1)$ time.

We first need to the following technical lemma, whose proof will appear in the full version of this paper.

LEMMA 4.2. *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 $1 \leqslant j < \ell < m$ 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 transversal nodes in order before reaching e_m. (Refer to Figure 4a.) There exists a path Q from the source of P to the destination of P such that Q intersects $e_1, \ldots, e_m$ in order, $\text{cost}(Q) \leqslant \text{cost}(P)$, 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 $e_j = e_1$.*

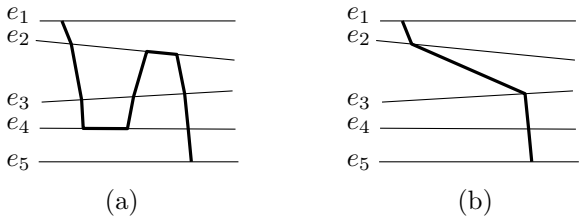

Figure 4: (a) A path P with two critical links. (b) A shorter transversal path Q.

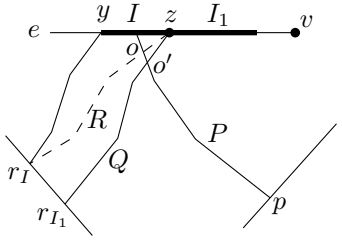

Figure 5: Interval I_1 causes interval I to be trimmed at z. P is the refraction path from r_I to p, which uses part of I as a critical link. Q is the shortest transversal path from r_{I_1} to z. R is the shortest transversal path from r_I to z.

LEMMA 4.3. *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 on 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, there exists a path P^* in the strip from a Steiner point or vertex r on the same boundary edge as r_I to p such that $d(r) + \text{cost}(P^*) < d(r_I) + \text{cost}(P)$.*

Proof. Recall that when I was created, I contained an endpoint of e, say v. As I does not contain v any longer, it must have been trimmed by some interval $I_1 \subset e$. By the working of the algorithm, I must contain the

endpoint of I_1 that is farthest from v, say z, and I is trimmed at z. See Figure 5.

Let R and Q be the shortest transversal paths from r_I and r_{I_1}, respectively, to z. Let P' denote the subpath of P after the critical link. Since R is a transversal path, by Snell's law, R cannot cross $P[r_I, y]$ which means that $P[r_I, y]$, yz 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 $\text{cost}(R[r_I, o]) < \text{cost}(P[r_I, o])$ or $d(r_{I_1}) + \text{cost}(Q[r_{I_1}, o']) < d(r_I) + \text{cost}(P[r_I, o'])$. 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, $d(r_{I_1}) + \text{cost}(Q[r_{I_1}, o') \cdot P[o', p]) < d(r_I) + \text{cost}(P)$.

If $\text{cost}(R[r_I, o]) < \text{cost}(P[r_I, o])$, then we are done. So from now on, we assume it is not the case, which means that $\text{cost}(P[r_I, o]) \leqslant \text{cost}(R[r_I, o])$. Apply Lemma 4.2 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, $\text{cost}(X) \geqslant \text{cost}(R)$, while in the latter case, $\text{cost}(X) \geqslant \text{cost}(P[r_I, y] \cdot yz)$. So $\min\{\text{cost}(P[r_I, y] \cdot yz), \text{cost}(R)\} < \text{cost}(P[r_I, o] \cdot R[o, z])$. By the assumption that $\text{cost}(P[r_I, o]) \leqslant \text{cost}(R[r_I, o])$, we have $\min\{\text{cost}(P[r_I, y] \cdot yz), \text{cost}(R)\} < \text{cost}(R)$, or equivalently,

$$\text{cost}(P[r_I, y] \cdot yz) < \text{cost}(R). \tag{4.1}$$

For sake of contradiction, assume that $d(r_I) + \text{cost}(P[r_I, o']) \leqslant d(r_{I_1}) + \text{cost}(Q[r_{I_1}, o'])$. One can use the same argument to show that $d(r_I) + \min\{\text{cost}(P[r_I, y] \cdot yz), \text{cost}(R)\} < d(r_{I_1}) + \text{cost}(Q)$. Then, it follows from (4.1) that $d(r_I) + \text{cost}(P[r_I, y] \cdot yz) < d(r_{I_1}) + \text{cost}(Q)$. But then I should have caused the algorithm to prune away I_1 completely, a contradiction. □

Type-IV intervals on a boundary edge e of a strip S are divided into four groups as follows. A 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 $\mathsf{pred\_idx}(I) \leqslant i$.

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 $\mathsf{pred\_idx}(I_1) < \mathsf{pred\_idx}(I_2) \leqslant i$ 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 Lemma 4.5. We first need a technical lemma, its proof will appear in the full version of the paper.

LEMMA 4.4. *Let $P_{1,x}$, $P_{2,x}$, and $P_{2,y}$ be three paths from r_1 to x, 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 $c_1 + \text{cost}(P_{1,x}) \leqslant c_2 + \text{cost}(P_{2,x})$ for some real numbers c_1 and c_2, and $\text{cost}(P_{2,x}) \leqslant \text{cost}(P_{2,y}[r_2, o]) + \text{cost}(P_{1,x}[o, x])$, then $c_1 + \text{cost}(P_{1,x}[r_1, o] \cdot P_{2,y}[o, y]) \leqslant c_2 + \text{cost}(P_{2,y})$.*

LEMMA 4.5. *Suppose that $\mathsf{pred}(I_1)$ and $\mathsf{pred}(I_2)$ are oriented from left to right and $\mathsf{pred\_idx}(I_1) < \mathsf{pred\_idx}(I_2) \leqslant i$. 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 $d(r_{I_1}) + \text{cost}(P_{1,x}) \leqslant d(r_{I_2}) + \text{cost}(P_{2,x})$ 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, $d(r_{I_1}) + \text{cost}(Q_{1,y}) \leqslant d(r_{I_2}) + \text{cost}(P_{2,y})$.*

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

Proof. Consider (i). See Figures 6(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 $\mathsf{pred\_idx}(I_1) < \mathsf{pred\_idx}(I_2) \leqslant i$ 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 6(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,

$$\text{cost}(P_{2,x}) \leqslant \text{cost}(P_{2,y}[r_{I_2}, o]) + \text{cost}(P_{1,x}[o, x]). \tag{4.2}$$

By Lemma 4.4, $d(r_{I_1}) + \text{cost}(P_{1,x}[r_{I_1}, o] \cdot P_{2,y}[o, y]) \leqslant d(r_{I_2}) + \text{cost}(P_{2,y})$. 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 (4.2) holds for the other two cases left.

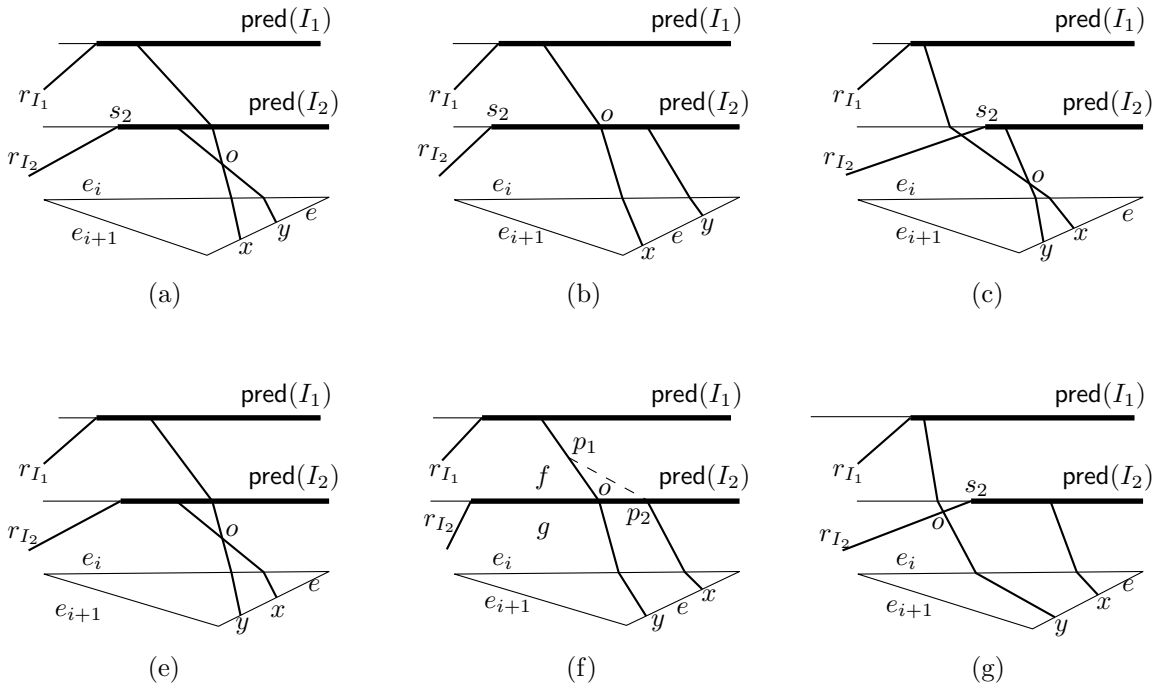

Figure 6: Illustrations for the proof of Lemma 4.5. The top row is for (i) and the bottom row is for (ii).

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 6(b): The second case is that s_2 is to the left of $P_{1,x}$. Refer to Figure 6(c).

Consider (ii). Lemma 4.4 is applicable if the following triangle inequality holds:

$$\text{(4.3)}\quad \text{cost}(P_{1,x}) \leqslant \text{cost}(P_{1,y}[r_{I_1}, o]) + \text{cost}(P_{2,x}[o, x]).$$

If so, Lemma 4.4 implies that $d(r_{I_2}) + \text{cost}(P_{2,x}[r_{I_2}, o] \cdot P_{1,y}[o, y]) \leqslant d(r_{I_1}) + \text{cost}(P_{1,y})$. Since $Q_{2,y}$ is the shortest path in the strip from r_{I_2} to y, we thus obtain $d(r_{I_2}) + \text{cost}(Q_{2,y}) \leqslant d(r_{I_1}) + \text{cost}(P_{1,y})$ as stated in (ii). So it remains to prove (4.3).

There are three cases depending on how $P_{1,y}$ crosses $P_{2,x}$ as shown in Figures 6(e–g). If the crossing happens after the critical links of both $P_{1,y}$ and $P_{2,x}$ (Figure 6(e)), then (4.3) 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 6(f). We cannot immediately conclude that (4.3) 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 since $w_f < w_g$, that critical link is redundant and can be eliminated without increasing the path cost as shown by the dashed edge in Figure 6(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 $\text{cost}(P_{1,x}) \leqslant \text{cost}(X) \leqslant \text{cost}(P_{1,y}[r_{I_1}, o]) + \text{cost}(P_{2,x}[o, x])$. Therefore, (4.3) still holds.

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 6(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 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 6(f) and conclude that (4.3) 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 4.2 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, $\text{cost}(P_{1,x}) \leqslant \text{cost}(X' \cup Y) \leqslant \text{cost}(P_{1,y}[r_{I_1}, o]) + \text{cost}(P_{2,x}[o, x])$, i.e. (4.3) holds. □

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 4.5(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 4.5. 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 4.5(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 4.5(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 a type-IV interval are linear, so trimming can be done in $O(1)$ time.

4.5 Type-V intervals. Type-V intervals are created at Step 4 of ProcessPoint(v).

LEMMA 4.6. *After preprocessing a boundary edge e' of a strip S in $O(\frac{k^2}{\sqrt{\varepsilon}} \log \frac{k}{\varepsilon})$ time, given any source on e' and any edge e'' in the interior or on the boundary of S, one can determine in $O(\log \frac{k}{\varepsilon})$ time the interval I on e'' that contains the destinations of shortest* $\text{cost}_{\pentagon}$ *transversal paths from the given source whose interiors do not contain vertices. In addition, the* $\text{cost}_{\pentagon}$ *from any source point in e' to any destination in I can be found in $O(\log \frac{k}{\varepsilon})$ time.*

Proof. Let $e_2, e_3, \ldots$ be the interior edges of S. Let e_1 be a boundary edge of S in the same face as e_2. Assume $e' = e_1$. The algorithm is exactly the same if e' is a different boundary edge, since the edge sequence for a transversal path from e' to any edge in S is predetermined.

Parameterize edges from left to right uniformly by parameters in $[0, 1]$. Use $e_i(\lambda_i)$ to denote the point on e_i with parameter λ_i. So $e_i(0)$ is the left endpoint of e_i and $e_i(1)$ is the right endpoint. Let $F_i(\lambda_1, \lambda_i)$ be the function $[0, 1]^2 \to \mathbb{R}$ that represents the $\text{cost}_{\pentagon}$ of shortest $\text{cost}_{\pentagon}$ transversal paths from points on e_1 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 $\text{cost}_{\pentagon}$ of segments from a point on e_{i-1} to a point on e_i. We have $F_2 = L_2$ and $F_{i+1}(\lambda_1, \lambda_i) = \min_{(\lambda_2,\ldots,\lambda_i)\in[0,1]^{i-1}} \sum_{1 \leqslant j \leqslant i} L_{i+1}(\lambda_j, \lambda_{j+1})$ for $i \geqslant 2$. We are only interested in the values of F_{i+1} that can be realized by $\lambda_2, \ldots, \lambda_i$ each of which is in $(0, 1)$, because otherwise the path bends at a vertex of some edge e_j, and such paths will not be considered by the algorithm. Denote this restriction of F_{i+1} by F^*_{i+1}. That is, the domain of F^*_{i+1} contains points $(\lambda_1, \lambda_{i+1})$ such that there exists $\lambda_i \in (0, 1)$ satisfying $F_{i+1}(\lambda_1, \lambda_{i+1}) = F^*_i(\lambda_1, \lambda_i) + L_{i+1}(\lambda_i, \lambda_{i+1})$. Let $\mathcal{D}(F_i)$ and $\mathcal{D}(F^*_i)$ be the projections of the graphs of F_i and F^*_i, respectively, in the $\lambda_1\lambda_i$-plane, and let $\mathcal{D}(L_{i+1})$ be the projection of the graph of L_i in the $\lambda_i\lambda_{i+1}$-plane.

It is clear that L_i is convex and piecewise linear. Moreover, L_i has $O(1/\sqrt{\varepsilon})$ linear pieces, and those linear pieces meet at either $(0, 0)$ or $(1, 1)$ depending on the endpoint e_{i-1} and e_i share. (See Figure 7a.) One can also show by induction that F^*_i is a convex piecewise linear function.

Let $\partial F^*_i(\lambda_1, \lambda_i)/\partial\lambda_i^-$ and $\partial F^*_i(\lambda_1, \lambda_i)/\partial\lambda_i^+$ denote the left and right derivatives of $F^*_i(\lambda_1, \lambda_i)$ with respect to λ_i, respectively. Similarly, $\partial L_{i+1}(\lambda_i, \lambda_{i+1})/\partial\lambda_i^-$ and $\partial L_{i+1}(\lambda_i, \lambda_{i+1})/\partial\lambda_i^+$ denote the left and right derivatives of $L_{i+1}(\lambda_i, \lambda_{i+1})$ with respect to λ_i. By convexity of F^*_i and L_{i+1}, a $\lambda_i \in (0, 1)$ satisfies the equation $F^*_{i+1}(\lambda_1, \lambda_i) = F^*_i(\lambda_1, \lambda_i) + L_{i+1}(\lambda_i, \lambda_{i+1})$, if and only if $\partial F^*_i(\lambda_1, \lambda_i)/\partial\lambda_i^+ + \partial L_{i+1}(\lambda_i, \lambda_{i+1})/\partial\lambda_i^+ \geqslant 0$, and $\partial F^*_i(\lambda_1, \lambda_i)/\partial\lambda_i^- + \partial L_{i+1}(\lambda_i, \lambda_{i+1})/\partial\lambda_i^- \leqslant 0$.

We now prove by induction that $\mathcal{D}(F^*_i)$ is a polygon of size $O(\frac{i}{\sqrt{\varepsilon}})$ that does not have any vertex in its interior, whose interior edges have positive slopes, and whose boundary consists of two monotonically decreasing chains. (See Figure 7b.) Let $\Gamma_1, \Gamma_2, \ldots$ be the faces in $\mathcal{D}(F^*_i)$ from left to right. Since the interior edges of $\mathcal{D}(F^*_i)$ have positive slopes, $\Gamma_1, \Gamma_2, \ldots$ are also sorted from top to bottom. In the same face of $\mathcal{D}(F^*_i)$, $\partial F^*_i(\lambda_1, \lambda_i)/\partial\lambda_i^+ = \partial F^*_i(\lambda_1, \lambda_i)/\partial\lambda_i^-$ and remains constant. Let γ_j be the partial derivative in the interior of the face Γ_j. Then, on edge $\Gamma_j \cap \Gamma_{j+1}$, we have $\partial F^*_i(\lambda_1, \lambda_i)/\partial\lambda_i^+ = \gamma_j$ and $\partial F^*_i(\lambda_1, \lambda_i)/\partial\lambda_i^- = \gamma_{j+1}$. By convexity of F^*_i, we also have $\gamma_1 > \gamma_2 > \ldots$

Let $Z_1, Z_2, \ldots$ be the faces of $\mathcal{D}(L_{i+1})$ from left to right, or equivalently, from top to bottom. Let ζ_j be the derivative $\partial L_{i+1}(\lambda_i, \lambda_{i+1})/\partial\lambda_i^+ = \partial L_{i+1}(\lambda_i, \lambda_{i+1})/\partial\lambda_i^-$ in the interior of Z_j. We have $\zeta_1 < \zeta_2 < \ldots$, and on edge $Z_j \cap Z_{j+1}$, $\partial L_{i+1}(\lambda_i, \lambda_{i+1})/\partial\lambda_i^+ = \zeta_{j+1}$ and $\partial L_{i+1}(\lambda_i, \lambda_{i+1})/\partial\lambda_i^- = \zeta_j$.

The domain of $F^*_i(\lambda_1, \lambda_i) + L_{i+1}(\lambda_i, \lambda_{i+1})$ is the intersection of the extrusion of $\mathcal{D}(F^*_i)$ along λ_{i+1} axis and the extrusion of $\mathcal{D}_{L_i}$ along λ_1 axis. The surface with equation $F^*_{i+1}(\lambda_1, \lambda_{i+1}) = F^*_i(\lambda_1, \lambda_i) + L_{i+1}(\lambda_i, \lambda_{i+1})$ in the $\lambda_1\lambda_i\lambda_{i+1}$ space consists of linear patches that are intersections of the extrusions of Γ_ℓ and $Z_j \cap Z_{j+1}$ such that $\zeta_j \leqslant -\gamma_\ell \leqslant \zeta_{j+1}$, or intersections of extrusions of $\Gamma_\ell \cap \Gamma_{\ell+1}$ and Z_j such that $-\gamma_\ell \leqslant \zeta_j \leqslant -\gamma_{\ell+1}$. So we can construct the faces of $\mathcal{D}(F^*_{i+1})$ by scanning through

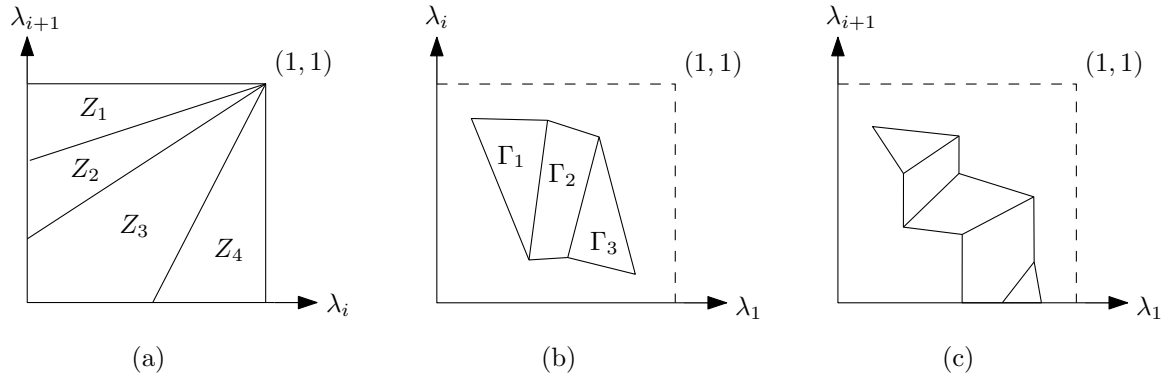

Figure 7: (a) The projection $\mathcal{D}(L_{i+1})$ of the graph of L_{i+1}. Assume that e_i and e_{i+1} meet at $e_i(1)$ and $e_{i+1}(1)$, then the edges in the $\mathcal{D}(L_{i+1})$ all pass through $(1,1)$. (b) The projection $\mathcal{D}(F_i^*)$ of the graph of F_i^*. The slopes of all interior edges are positive, and the boundary of $\mathcal{D}(F_i^*)$ consists of two monotonically decreasing chains. (c) The projection $\mathcal{D}(F_{i+1}^*)$ of the graph of F_{i+1}^*.

the union of the two sequences $-\gamma_1 < -\gamma_2 < \dots$ and $\zeta_1 < \zeta_2 < \dots$ from left to right. More precisely, our algorithm works as follows. We omit the termination condition: The algorithm terminates whenever ℓ or j exceed the number of faces in $\mathcal{D}(F_i^*)$ or $\mathcal{D}(L_{i+1})$, respectively.

1. Set $\ell := 1$ and set j to be the smallest integer such that $-\gamma_\ell \leqslant \zeta_{j+1}$.
2. (Precondition: $\zeta_j < -\gamma_\ell \leqslant \zeta_{j+1}$.) While $\zeta_j \leqslant -\gamma_\ell \leqslant \zeta_{j+1}$, output the projection of the intersection of extrusions of Γ_ℓ and $Z_j \cap Z_{j+1}$, increment ℓ and repeat.
3. (Precondition: $-\gamma_{\ell-1} \leqslant \zeta_{j+1} < -\gamma_\ell$.) Decrement ℓ and increment j.
4. (Precondition: $-\gamma_\ell \leqslant \zeta_j < -\gamma_{\ell+1}$.) While $-\gamma_\ell \leqslant \zeta_j \leqslant -\gamma_{\ell+1}$, output the projection of the intersection of extrusions of $\Gamma_\ell \cap \Gamma_{\ell+1}$ and Z_j, increment j and repeat.
5. (Precondition: $\zeta_{j-1} \leqslant -\gamma_{\ell+1} < \zeta_j$.) Increment ℓ and decrement j, and go to Step 2.

A polygon that is output at Step 4 has at most $O(1)$ vertices, because it is the extrusion of the segment $\Gamma_\ell \cap \Gamma_{\ell+1}$ clipped by $O(1)$ planes: two planes parallel to λ_1 and through two sides of Z_j and the faces of the unit cube. Since j can only be incremented $O(1/\sqrt{\varepsilon})$ times at step 4, the total size of the polygons constructed at step 4 is $O(1/\sqrt{\varepsilon})$.

Consider a polygon being output at step 2. Notice that if $Z_j \cap Z_{j+1}$ spans the whole λ_i interval from 0 to 1, its extrusion completely cuts through the extrusion of Γ_ℓ, in which case the output polygon has the same number of vertices as Γ_ℓ does. If $Z_j \cap Z_{j+1}$ only covers part of the λ_i interval from 0 to 1 (e.g., $Z_3 \cap Z_4$ in Figure 7a), it may cause an output polygon to have more vertices in place $\lambda_{i+1} = 0$ if the edges of $\mathcal{D}(L_{i+1})$ meet at $(1,1)$ or $\lambda_{i+1} = 1$ if the edges of $\mathcal{D}(L_{i+1})$ meet at $(0,0)$. Those additional vertices are intersections of the boundary of $\mathcal{D}(F_i^*)$ (or its copy in plane $\lambda_{i+1} = 1$) with the extrusion of $Z_j \cap Z_j$. The boundary of $\mathcal{D}(F_i^*)$ consists of two monotonically decreasing chains, so it can be crossed by a line parallel to λ_1 axis at most twice. Therefore, we have $O(1/\sqrt{\varepsilon})$ additional vertices.

In total, $\mathcal{D}(F_{i+1}^*)$ has $O(1/\sqrt{\varepsilon})$ more vertices than $\mathcal{D}(F_i^*)$, and thus it has size $O((i+1)/\sqrt{\varepsilon})$. Because the interior edges of both $\mathcal{D}(L_{i+1})$ and $\mathcal{D}(F_i^*)$ have positive slopes and the boundary of $\mathcal{D}(F_i^*)$ consists of two monotonically decreasing chains, the interior edges of $\mathcal{D}(F_{i+1}^*)$ have positive slopes and the boundary of $\mathcal{D}(F_{i+1}^*)$ consists of two monotonically decreasing chains.

Therefore, given F_i^*, one can compute F_{i+1}^* in $O(i/\sqrt{\varepsilon})$ time. Let $\bar{e}_{i+1}$ be the third edge in the same face as e_i and e_{i+1}, and let $\bar{F}_{i+1}^*$ be the cost function from e_1 to $\bar{e}_{i+1}$. We can use the same algorithm for computing F_{i+1}^* to compute $\bar{F}_{i+1}^*$ in $O(i/\sqrt{\varepsilon})$ time. After obtaining F_{i+1}^*, we can build a point location data structure in $O(\frac{k}{\sqrt{\varepsilon}} \log \frac{k}{\varepsilon})$ time and evaluate the function value at any query point in $O(\log \frac{k}{\varepsilon})$ time [5]. Given a source point on e_1, we can also compute the range of $\bar{e}_{i+1}$ in the domain of $\mathcal{D}(\bar{F}_{i+1}^*)$ in $O(\log \frac{k}{\varepsilon})$ time by binary searching along the two monotonic chains of the boundary. ◻

For every pair of boundary edges e' and e'' of S, we apply Lemma 4.6 to build a data structure so that for every point $x \in e'$ and every point $y \in e''$, the

minimum $\text{cost}_{\pentagon}$ 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, apply Lemma 4.6 to compute a type-V interval containing points to which the shortest $\text{cost}_{\pentagon}$ traversal paths from r do not pass through vertices in the interior of the paths. So creating a type-V interval takes $O(\log \frac{k}{\varepsilon})$ time. 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 $\text{cost}_{\pentagon}$ from r_{I_i} to x. Lemma 4.7 below shows that there is at most one tie point or a contiguous tie segment. First, do binary search to find two adjacent Steiner points $p, q \in I_1 \cap I_2$ that sandwich a tie point, i.e., $d(r_1) + \text{cost}_{\pentagon}(P_{1,p}) \leqslant d(r_2) + \text{cost}_{\pentagon}(P_{2,p})$ and $d(r_1) + \text{cost}_{\pentagon}(P_{1,q}) \geqslant d(r_2) + \text{cost}_{\pentagon}(P_{2,q})$. In each iteration of the binary search, the cost of a point can be computed in $O(\log \frac{k}{\varepsilon})$ time using Lemma 4.6. Then apply Lemma 4.7 to trim I_1 and I_2 to two disjoint intervals at p and q, and update the costs of trimmed intervals in the priority queue.

If a tie point in $I_1 \cap I_2$ does not exist, by Lemma 4.7, the inferior interval can be pruned altogether. A pruned interval is deleted from both priority queue and the interval group.

LEMMA 4.7. *Let I_1 and I_2 be two type-V intervals on a boundary edge e of a strip S. If $d(r_{I_2}) + \text{cost}_{\pentagon}(P_{2,x}) \leqslant d(r_{I_1}) + \text{cost}_{\pentagon}(P_{1,x})$ 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 a type-V interval on e that overlaps with I_1 and $d(r) + \text{cost}(Q_y) \leqslant d(r_{I_1}) + \text{cost}_{\pentagon}(P_{1,y})$.

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

$$\text{cost}_{\pentagon}(P_{1,y}[r_{I_1}, o]) + \text{cost}_{\pentagon}(P_{2,x}[o, x]) \geqslant \text{cost}_{\pentagon}(P_{1,x}). \tag{4.4}$$

Let Q be the path with the minimum $\text{cost}_{\pentagon}$ 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

$$\text{cost}_{\pentagon}(P_{2,x}[r_{I_2}, 0]) + \text{cost}_{\pentagon}(P_{1,y}[o, y]) \geqslant \text{cost}_{\pentagon}(P_{2,y}). \tag{4.5}$$

Adding (4.4) and (4.5) yields $\text{cost}_{\pentagon}(P_{1,y}) + \text{cost}_{\pentagon}(P_{2,x}) \geqslant \text{cost}_{\pentagon}(P_{1,x}) + \text{cost}_{\pentagon}(P_{2,y})$. Therefore, $d(r_{I_2}) + \text{cost}_{\pentagon}(P_{2,y}) + \text{cost}_{\pentagon}(P_{1,x}) \leqslant d(r_{I_2}) + \text{cost}_{\pentagon}(P_{1,y}) + \text{cost}_{\pentagon}(P_{2,x})$ which is at most $d(r_{I_1}) + \text{cost}_{\pentagon}(P_{1,y}) + \text{cost}_{\pentagon}(P_{1,x})$ by the assumption of the lemma. It follows that $d(r_{I_2}) + \text{cost}_{\pentagon}(P_{2,y}) \leqslant d(r_{I_1}) + \text{cost}_{\pentagon}(P_{1,y})$.

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

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

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, $d(r) + \text{cost}_{\pentagon}(Q_y) \leqslant d(r_{I_2}) + \text{cost}_{\pentagon}(Q)$. It follows that $d(r) + \text{cost}_{\pentagon}(Q) \leqslant d(r_{I_1}) + \text{cost}_{\pentagon}(P_{1,y})$. □

4.6 Analysis and proof of correctness. Based on the trimming approach given above, we can now analyze our algorithm.

LEMMA 4.8. *The algorithm runs in $O\left(\frac{kn+k^4 \log(k/\varepsilon)}{\varepsilon} \log^2 \frac{\rho n}{\varepsilon}\right)$ 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, 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 neighboring Steiner points on the incident edges. In total, there are $O(mn)$ type-I and type-II intervals, and $O(k^3 m)$ type-III and type-V intervals. Consider type-IV intervals. Only two type-III intervals on the same interior edge of some strip can propagate to $O(k)$ type-IV intervals. Any other type-III interval can only propagate to one type-IV interval. Therefore, we have at most $O(k^3)$ more type-IV intervals than type-III intervals.

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, while trimming two type-V intervals takes $O\left((\log m)(\log \frac{k}{\varepsilon})\right)$

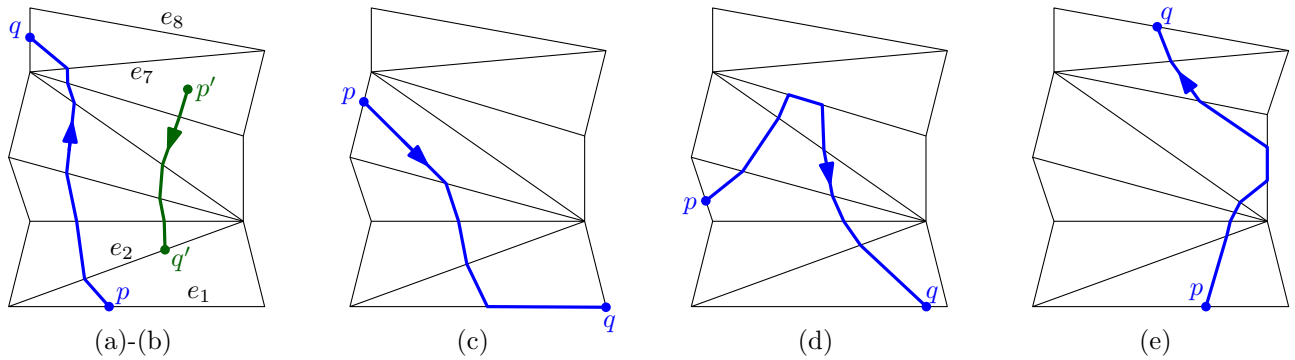

Figure 8: Shortest paths in a strip. In the first figure, the path p-q is of type (a) and the path p'-q' is of type (b).

time. Therefore, creating all types of intervals takes $O(mn \log m)$, $O(mn \log m)$, $O(k^3 m \log m)$, $O(k^3 m \log m)$, and $O\left(k^3 m (\log m)(\log \frac{k}{\varepsilon})\right)$ 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 dequeuing 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 4.1 and 4.6. So the total running time is $O\Big(mn \log(mn) + k^3 m (\log m)(\log \frac{k}{\varepsilon}) + \frac{k^4}{\sqrt{\varepsilon}} \log \frac{k}{\varepsilon}\Big) = O\left(\frac{kn+k^4 \log(k/\varepsilon)}{\varepsilon} \log^2 \frac{\rho n}{\varepsilon}\right)$. □

The following lemma gives the structure of the shortest paths within a strip. It will be needed to prove that our algorithm is correct. The proof will appear in the full version of this paper.

LEMMA 4.9. *Consider a strip S with edge sequence $e_1, \ldots, e_m$, and two points p, q in S. Then within S, there is a shortest path P from p to q of one of the following types:*

(a) *P is a transversal path whose interior crosses a subsequence $e_i, e_{i+1}, \ldots, e_j$ of the interior edges of S.*

(b) *P is a transversal path whose interior crosses a subsequence $e_j, e_{j-1}, \ldots, e_i$ of the interior edges of S.*

(c) *P consists of a transversal path of type (a) or (b) and a critical link.*

(d) *P consists of a type (a) and a path of type (b), connected by critical link on a long edge.*

(e) *P consists of two paths of type (a) or (b), connected by a critical link on a short edge.*

(f) *P consists of several paths of type (a), (b), (c), or (d) meeting at vertices of S.*

We can now prove the correctness of our algorithm.

LEMMA 4.10. *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. The links that do not lie in any strip are handled exactly, and in the same way as the BUSHWHACK algorithm: We are just simulating Dijkstra's algorithm while trimming edges that do not contribute to the solution. We now consider the edges of $\mathcal{G}$ connecting two points within a strip S.

So consider a maximal subpath within S, and split it at each vertex of $\mathcal{T}$ into subpaths $\{P_i\}$ of type (a)–(e) as in Lemma 4.9. Let P_i be one of these subpaths, and let p and q denote its starting and ending point, respectively. If P_i is of type (c) and if the critical link is at p, then this case is handled exactly by propagating a type-III interval. If the critical link is at q, then our algorithm will first propagate along the edge e containing the critical link, and it reduces to the case of a type (a) or (b) path. If P_i is of type (d) or (e), then it is handled exactly by our algorithm by propagating type-III and IV intervals within S.

If P_i is of type (a) or (b), our algorithm handles it by propagating a type (V) interval. It does not produce an exact result, as we use the distance function $\text{cost}_{\pentagon}$. However, the cost of P_i satisfies $\text{cost}(P_i) \leqslant \text{cost}_{\pentagon}(P_i) \leqslant (1+\varepsilon/3)\,\text{cost}(P_i)$. If P_i^* is the optimal p-q path according to the distance function cost, we also have $\text{cost}(P_i^*) \leqslant \text{cost}_{\pentagon}(P_i^*) \leqslant (1+\varepsilon/3)\,\text{cost}(P_i^*)$. As P_i is optimal according to $\text{cost}_{\pentagon}$, we have $\text{cost}_{\pentagon}(P_i) \leqslant \text{cost}_{\pentagon}(P_i^*)$, and hence $\text{cost}(P_i) \leqslant \text{cost}_{\pentagon}(P_i) \leqslant \text{cost}_{\pentagon}(P_i^*) \leqslant (1+\varepsilon/3)\,\text{cost}(P_i^*)$.

So our algorithm overestimates the weight of the edges of $\mathcal{G}$ by a factor at most $(1+\varepsilon/3)$. Thus it produces a $(1+\varepsilon/3)$-approximate shortest path in $\mathcal{G}$. □

Our main result follows from Lemmas 3.1, 4.8 and 4.10.

THEOREM 4.1. *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 $O\left(\frac{kn+k^4\log(k/\varepsilon)}{\varepsilon}\log^2\frac{\rho n}{\varepsilon}\right)$ time, where ρ is the ratio of the maximum weight to the minimum weight.*

References

[1] 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.

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

[3] 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.

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

[5] D. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12 (1983), 28–35.

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

[7] 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.

[8] 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.

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

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