Shortest Paths on Polyhedral Surfaces and Terrains*

Siu-Wing Cheng
HKUST
Hong Kong
scheng@cse.ust.hk

Jiongxin Jin
Google Inc.
651 N 34th St, Seattle, WA, 98103
jamesjjx@google.com

ABSTRACT

We present an algorithm for computing shortest paths on polyhedral surfaces under convex distance functions. Let n be the total number of vertices, edges and faces of the surface. Our algorithm can be used to compute L_1 and L_∞ shortest paths on a polyhedral surface in $O(n^2 \log^4 n)$ time. Given an $\varepsilon \in (0,1)$, our algorithm can find $(1+\varepsilon)$-approximate shortest paths on a terrain with gradient constraints and under cost functions that are linear combinations of path length and total ascent. The running time is $O\left(\frac{1}{\sqrt{\varepsilon}} n^2 \log n + n^2 \log^4 n\right)$. This is the first efficient PTAS for such a general setting of terrain navigation.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms

Algorithms, Theory

Keywords

shortest path, convex distance function, polyhedral surface, terrain

1. INTRODUCTION

Finding shortest paths is a classical geometric optimization problem. In recent years, the spatial database and geographical information system communities show interest in the shortest path problem on terrains under anisotropic cost models, i.e., the path cost at any point on the terrain depends on the travel direction [10, 12, 15, 17]. The motivations are two-fold. First, when planning a roadway or hiking on a terrain, it is impossible to ascend or descend along slopes that are too steep. Second, the cost of a subpath may depend on the its slope. Anisotropic cost models on polyhedral surfaces also relate to or generalize previous results in the algorithm community: the shortest path problem on polyhedral surfaces [6, 13, 16], the weighted region problem [4, 14], the anisotropic shortest path problem in the plane [8, 9], and the consideration of total ascent or descent of paths on a terrain [5]. The shortest descending path problem [1, 2, 7] and the shortest gently descending path problem [3] are special cases obtained by enforcing particular gradient constraints.

For Euclidean shortest paths on a polyhedral surface of n vertices, edges and faces, Mitchell et al. [13] presented an algorithm that runs in $O(n^2 \log n)$ time, which was subsequently improved by Chen and Han [6] to $O(n^2)$. Varadarajan and Agarwal [16] proposed two approximation algorithms that run in subquadratic time: $7(1+\varepsilon)$- and $15(1+\varepsilon)$-approximate shortest paths can be found in $O(n^{5/3} \log^{5/3} n)$ and $O(n^{8/5} \log^{8/3} n)$ time, respectively.

In the weighted region problem, each face f has a weight w_f and the subpath cost within f is w_f times the subpath length. A shortest path may bend when crossing edges (which also happens under anisotropic cost models). Mitchell and Papadimitriou [14] presented an algorithm for planar weighted regions that runs in $O(n^8 \log(NW/\delta))$ time, where N is the largest integer coordinate, W is the ratio of the maximum weight to the minimum weight, and δ is a precision parameter. Aleksandrov et al. [4] developed an algorithm for polyhedral surfaces that has a running time linear in n and dependent on some geometric parameters.

Cheng et al. [8] proposed a $(1+\varepsilon)$-approximation algorithm for the anisotropic shortest path problem in a planar subdivision in which every face has a convex distance function. Later, a data structure was developed to answer $(1+\varepsilon)$-approximate anisotropic shortest path queries [9].

De Berg and van Kreveld [5] studied some path query problems on terrains with height constraints, and they posed the optimization of path length and total ascent as an open problem. There are $(1+\varepsilon)$-approximate algorithms for the shortest descending path problem [1] and the shortest gently descending path problem [3] that have running times dependent on some geometric parameters. Recently, we developed a $(1+\varepsilon)$-approximate shortest descending path algorithm that runs in $O(n^4 \log(n/\varepsilon))$ time [7].

This paper presents an algorithm for a shortest path problem on a polyhedral surface, which we call the PolyPath problem. Each face f is associated with a convex polygon

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

STOC'14, May 31–June 03 2014, New York, NY, USA.
 ACM 978-1-4503-2710-7/14/05 ...$15.00
http://dx.doi.org/10.1145/2591796.2591821.

H_f that induces a convex distance function d_f. The length of a subpath in f is measured using d_f. Given two points s and t on the polyhedral surface and an integer m, the goal is to find a shortest one among all paths that have at most m links and no critical refraction at any surface edge.[1] The latter constraint can be removed if $d_f(p,q) = d_g(p,q)$ for every two adjacent faces f and g and every two points $p, q \in f \cap g$. Our algorithm runs in $O(hmn \log mn + mn \log^2 m \log^2 hm)$ time, where h is the maximum size of the convex polygons associated with the faces. It follows that an L_1 or L_∞ shortest path on a polyhedral surface can be computed in $O(n^2 \log^4 n)$ time.

On terrains, for every constant $c_1 > 0$ and every constant $c_2 \geq 0$, we can optimize $c_1 \cdot$ Euclidean path length plus $c_2 \cdot$ total ascent with a relative error ε under gradient constraints. The total ascent is the total increase in heights of all ascending subpaths, which measures the energy spent in increasing the potential energy. The weighted sum of the path length and its total ascent gives rise to a convex distance function, which can be approximately induced by a convex polygon of size $O(1/\sqrt{\varepsilon})$. This allows us to reduce the problem to an instance of PolyPath such that $m = O(n)$ and $h = O(1/\sqrt{\varepsilon})$. Gradient constraints are specified by the maximum ascent and descent gradients allowed in $\mathcal{T}$.[2] This only changes the convex distance function slightly. Section 4 describes these reductions. In all, our algorithm can return a $(1+\varepsilon)$-approximate shortest path in $O\left(\frac{1}{\sqrt{\varepsilon}} n^2 \log n + n^2 \log^4 n\right)$ time, which makes it the first PTAS for such a general setting of terrain navigation. A $(1+\varepsilon)$-approximate shortest descending path can thus be computed in $O\left(\frac{1}{\sqrt{\varepsilon}} n^2 \log n + n^2 \log^4 n\right)$ time.

Our results address the problems in the applications [10, 12, 15, 17] mentioned earlier. A shortest path that satisfies gradient constraints is sought on a terrain in [12]. So our terrain algorithm is directly applicable. A main problem treated in [10, 15, 17] is to optimize path length and penalize large slopes. As illustrated in Figure 5 in [10], one may model the cost function as a convex function in slope. Such a convex function translates to a convex distance function in a face. Therefore, if an upper bound m on the number of links can be specified, our PolyPath algorithm can be used to obtain a $(1+\varepsilon)$-approximation after approximating the convex distance function as in our terrain algorithm.

There are several difficulties in solving the PolyPath problem. A *locally shortest path* (*LSP*) for a sequence σ of edges is a shortest path that crosses the edges in σ. An algorithm needs to extend an LSP from one face to the next. In the Euclidean and weighted region cases, the extension is determined locally by unfolding to a straight line and following Snell's law, respectively. In our case, we first discover how an LSP bends at a surface edge. In fact, an LSP may bend in various ways, and we focus on a special LSP in order to characterize the bending. However, the extension is not determined locally. In the Euclidean and weighted region cases, the local extension allows to construct a function to describe the costs of the LSPs that start from an interval on an edge, cross the edges in σ, and end at an interval on the last edge in σ. This is important as an algorithm cannot extend an infinite number of LSPs. We show that such a function can be constructed in our case by proving that LSPs are preserved under *sliding*, i.e. translating each segment of the path while keeping it parallel with the original one. Thus, after constructing one LSP for some edge sequence, the cost of another LSP with the same edge sequence is a function of the amount of sliding. Our third contribution is to compose a shortest path by combining shorter LSPs in a hierarchical fashion using Chen and Han's sequence tree [6], which yields the claimed running time.

[1] A path makes a critical refraction at an edge e if there are two non-collinear links such that one lies on e and these two links meet at a node in the interior of e.

[2] The ascent and descent gradient bounds may be different, but they are the same for all faces.

2. PRELIMINARIES

Let $\mathcal{T}$ denote the input polyhedral surface with n vertices, edges and faces. Without loss of generality, assume that each face of $\mathcal{T}$ is a triangle, and the source s and the destination t are vertices of $\mathcal{T}$. Each face f of $\mathcal{T}$ is associated with a convex polygon H_f, which contains the origin, lies in a plane parallel to f, and induces the distance function d_f. We allow the origin to be on the boundary of H_f. The cost of a directed segment $pq \subset f$ is $\text{cost}(pq) = d_f(p,q) = \inf\left\{\lambda > 0 : \frac{1}{\lambda}(q-p) \in H_f\right\}$, which can be computed in $O(\log |H_f|)$ time by binary search.

We use $\vec{u}$ to denote a vector and $\hat{u}$ to denote the unit vector in the same direction as $\vec{u}$. Given $\vec{u}$ and $\vec{v}$, $\theta(\vec{u}, \vec{v})$ denotes the angle measured from $\vec{u}$ to $\vec{v}$ in counter-clockwise direction, which takes value in $[0, 2\pi)$. The inner product of $\vec{u}$ and $\vec{v}$ is denoted by $\langle \vec{u}, \vec{v} \rangle$.

All polygonal paths in this paper are oriented from their sources to their destinations. A *link* of a polygonal path is a maximal segment in a face or on an edge of $\mathcal{T}$, and its endpoints are called *nodes*. We assume that every node is either a vertex or a point in the interior of an edge because a node in the interior of a face can be removed by shortcutting without increasing the path cost. By the requirement of the PolyPath problem, we can further assume that every node in the interior of an edge is a *transversal* node, that is, its two incident links lie in the interiors of two distinct faces.

Let p_i, $i \in [0,k]$, be the nodes in order along a path P. Let $\vec{v}_i = p_i - p_{i-1}$ for $i \in [1,k]$. The *direction vector* of P is $(\hat{v}_1, \ldots, \hat{v}_k)$. We can specify P as $(p_0, p_1, \ldots, p_k)$ or as $(p_0, (\hat{v}_1, \ldots, \hat{v}_k))$. The subpath of P from a point x to another point y is denoted by $P[x,y]$. Define $\text{cost}(P) = \sum_{\text{face } f} \text{cost}(P \cap f)$ and $\|P\|$ to be the length of P.

The edges that P crosses in order is its *edge sequence*. It includes the edge containing P's destination but not the edge containing P's source. A path may have multiple edge sequences if its interior passes though a vertex. Suppose that the edges $e_1, e_2, \ldots, e_k$ are incident to a vertex ν in circular order. If a path moves from the face bounded by e_1 and e_k to ν onward to the face bounded by e_i and e_{i+1}, then one edge sequence contains the substring $e_1 e_2 \ldots e_i$, and another edge sequence contains the substring $e_k e_{k-1} \ldots e_{i+1}$. A shortest path from s to t is a shortest LSP over all edge sequences.

3. SOLVING PolyPath

We first characterize the LSPs by their direction vectors in Section 3.1. Then we propose an algorithm in Section 3.2 to solve the PolyPath problem.

3.1 Properties of LSPs

Let $\sigma = (e_1, e_2, \ldots, e_k)$ be the edge sequence of some LSP

that starts from a point p_0 on some face boundary and ends at a point p_k on some other face boundary. Thus, e_i and e_{i+1} are distinct edges of the same face, and e_i and e_{i+2} do not bound the same face. Let e_0 denote an edge adjacent to e_1 that contains the source of the LSP. For $i \in [1,k]$, let f_i denote the face bounded by e_{i-1} and e_i.

For $i \in [1,k]$, define the positive and negative sides of a point on e_i as follows. Orient e_i to obtain a directed segment a_ib_i so that f_i and f_{i+1} are on the left and right of a_ib_i, respectively. Let $\vec{e}_i$ denote the vector $b_i - a_i$. Given two points $p, q \in e_i$, we say that q lies on the *positive* or *negative* side of p if $\langle q-p, \vec{e}_i \rangle > 0$ or $\langle q-p, \vec{e}_i \rangle < 0$, respectively. The head and tail of the oriented e_i are the *positive and negative endpoints* of e_i, respectively.

There may be multiple LSPs that start from p_0, end at p_k, and share an edge sequence σ. Let $P = (p_0, (\hat{v}_1, \ldots, \hat{v}_k))$ and let $Q = (p_0, (\hat{w}_1, \ldots, \hat{w}_k))$ be two such LSPs. We say that $\hat{v}_i$ is *smaller* than $\hat{w}_i$ if $\theta(\vec{e}_i, \hat{v}_i) < \theta(\vec{e}_i, \hat{w}_i)$. The *canonical LSP* from p_0 to p_k with edge sequence σ is the LSP that has the lexicographically smallest direction vector. Intuitively, the canonical LSP hits every oriented e_i at a point closest to its negative endpoint.

LEMMA 3.1. *Let $P = (p_0, p_1, \ldots, p_k)$ and $Q = (q_0 = p_0, q_1, \ldots, q_k = p_k)$ be two LSPs from p_0 to p_k with the same edge sequence. If P is a canonical LSP, then for $i \in [1, k-1]$, q_i does not lie on the negative side of p_i.*

PROOF. Let j be the smallest integer such that $p_j \neq q_j$. Since P is the canonical LSP and $P[p_0, p_{j-1}] = Q[q_0, q_{j-1}]$, p_j must be on the negative side of q_j. If for all $i > j$, $p_i = q_i$ or p_i is on the negative side of q_i, then we are done. Otherwise, let i be the smallest integer such that p_i is on the positive side of q_i. Then $p_{i-1}p_i$ must cross $q_{i-1}q_i$, say at x. Since both P and Q are LSPs, $P[x, p_k]$ and $Q[x, p_k]$ are both LSPs and have the same cost, implying that

$$\begin{aligned} & \text{cost}(P[p_0, p_{i-1}]) + d_{f_i}(p_{i-1}q_i) + \text{cost}(Q[q_i, p_k]) \\ \leq\ & \text{cost}(P[p_0, x]) + \text{cost}(Q[x, p_k]) = \text{cost}(P). \end{aligned}$$

We obtain a new LSP $R = (p_0, p_1, \ldots, p_{i-1}, q_i, q_{i+1}, \ldots, q_k)$, where $\theta(\vec{e}_i, q_i - p_{i-1}) < \theta(\vec{e}_i, p_i - p_{i-1})$. But then the direction vector of R is lexicographically smaller than that of P, contradicting the assumption that P is a canonical LSP. □

We will characterize a canonical LSP via the derivative of its cost, which may not change smoothly as its destination moves. Thus, we define the derivative using limit and it depends on how the limit is approached. Recall that $\sigma = (e_1, \ldots, e_k)$ and e_0 is an edge adjacent to e_1 containing the source p_0 of P. Let $\sigma_{ij} = (e_{i+1}, \ldots, e_j)$. For every point $p \in e_i$, define a function $C_{p,\sigma_{ij}}(x)$ to be the cost of an LSP with edge sequence σ_{ij} from p to a point $x \in e_j$. For every point $q \in e_j$, define the function $D_{q,\sigma_{ij}}(x)$ be the cost of an LSP with edge sequence σ_{ij} from a point $x \in e_i$ to q. We use $x' \to x^+$ and $x' \to x^-$ to denote x' approaching x from the positive and negative sides of x, respectively. Define:

$$\begin{aligned} \partial C^+_{p,\sigma_{ij}}(x) &= \lim_{x' \to x^+} \tfrac{C_{p,\sigma_{ij}}(x') - C_{p,\sigma_{ij}}(x)}{\|xx'\|}, \\ \partial C^-_{p,\sigma_{ij}}(x) &= \lim_{x' \to x^-} \tfrac{C_{\sigma_{ij}}(p,x) - C_{p,\sigma_{ij}}(x')}{\|xx'\|}, \\ \partial D^+_{q,\sigma_{ij}}(x) &= \lim_{x' \to x^+} \tfrac{D_{q,\sigma_{ij}}(x') - D_{q,\sigma_{ij}}(x)}{\|xx'\|}, \text{ and} \\ \partial D^-_{q,\sigma_{ij}}(x) &= \lim_{x' \to x^-} \tfrac{D_{q,\sigma_{ij}}(x) - D_{q,\sigma_{ij}}(x')}{\|xx'\|}. \end{aligned}$$

LEMMA 3.2. *$C_{p,\sigma_{ij}}(x)$ and $D_{q,\sigma_{ij}}(x)$ are convex piecewise linear functions in x. If y is on the positive side of x in e_j, then $\partial C^+_{p,\sigma_{ij}}(y) \geq \partial C^-_{p,\sigma_{ij}}(y) \geq \partial C^+_{p,\sigma_{ij}}(x) \geq \partial C^-_{p,\sigma_{ij}}(x)$. If y is on the positive side of x in e_i, then $\partial D^+_{q,\sigma_{ij}}(y) \geq \partial D^-_{q,\sigma_{ij}}(y) \geq \partial D^+_{q,\sigma_{ij}}(x) \geq \partial D^-_{q,\sigma_{ij}}(x)$.*

PROOF. $C_{p,\sigma_{ij}}(x)$ is the minimum of $\sum_{\ell=i+1}^{j} \text{cost}(x_{\ell-1}x_\ell)$, where $x_i = p$, $x_j = x$, $x_\ell \in e_\ell$ for $\ell \in (i,j)$. The function $\text{cost}(x_{\ell-1}x_\ell)$ is convex and piecewise linear in $x_{\ell-1}$ and x_ℓ, so $C_{p,\sigma_{ij}}(x)$ is the minimization of the cross-section of a convex piecewise linear function. This implies the properties of $C_{p,\sigma_{ij}}$, $\partial C^+_{p,\sigma_{ij}}$ and $\partial C^-_{p,\sigma_{ij}}$ stated in the lemma. The same argument works for $D_{q,\sigma_{ij}}(x)$. □

Our algorithm will form new LSP by concatenating shorter ones. It is clear that if we split a canonical LSP $(p_0, \ldots, p_k)$ at p_i, we obtain two shorter canonical LSPs. Lemma 3.3 below shows that the converse is true under some conditions.

LEMMA 3.3. *If a path $P = (p_0, p_1, \ldots, p_k)$ is a canonical LSP with edge sequence σ, where $p_i \in e_i$, then the following conditions hold for every $i \in [1, k-1]$.*

(i) *$P[p_0, p_i]$ and $P[p_i, p_k]$ are canonical LSPs with edge sequences σ_{0i} and σ_{ik}, respectively.*

(ii) *p_i is the positive endpoint of e_i or $\partial C^+_{p_0,\sigma_{0i}}(p_i) + \partial D^+_{p_k,\sigma_{ik}}(p_i) \geq 0$.*

(iii) *p_i is the negative endpoint of e_i or $\partial C^-_{p_0,\sigma_{0i}}(p_i) + \partial D^-_{p_k,\sigma_{ik}}(p_i) < 0$.*

Conversely, if the conditions above hold for some $i \in [1, k-1]$, then P is a canonical LSP from p_0 to p_k with edge sequence σ.

PROOF. Suppose that P is a canonical LSP. Then $P[p_0, p_i]$ and $P[p_i, p_k]$ are canonical LSPs as well. If p_i is not the positive endpoint of e_i, pick a point p'_i on the positive side of p_i and arbitrarily close to p_i. By the definition of the functions $\partial C^+_{p_0,\sigma_{0i}}$ and $\partial D^+_{p_k,\sigma_{ik}}$, we obtain

$$\begin{aligned} & C_{p_0,\sigma_{0i}}(p'_i) + D_{p_k,\sigma_{ik}}(p'_i) - C_{p_0,\sigma_{0i}}(p_i) - D_{p_k,\sigma_{ik}}(p_i) \\ =\ & \left(\partial C^+_{p_0,\sigma_{0i}}(p_i) + \partial D^+_{p_k,\sigma_{ik}}(p_i)\right) \cdot \|p'_ip_i\|. \end{aligned}$$

Since P is an LSP, $C_{p_0,\sigma_{0i}}(p'_i) + D_{p_k,\sigma_{ik}}(p'_i) \geq C_{p_0,\sigma_{0i}}(p_i) + D_{p_k,\sigma_{ik}}(p_i)$, which implies that $\partial C^+_{p_0,\sigma_{0i}}(p_i) + \partial D^+_{p_k,\sigma_{ik}}(p_i) \geq 0$. If p_i is not the negative endpoint of e_i, we pick $p'_i \in e_i$ on the negative side of p_i and sufficiently close to p_i. Then,

$$\begin{aligned} & C_{p_0,\sigma_{0i}}(p_i) + D_{p_k,\sigma_{ik}}(p_i) - C_{p_0,\sigma_{0i}}(p'_i) + D_{p_k,\sigma_{ik}}(p'_i) \\ =\ & \left(\partial C^-_{p_0,\sigma_{0i}}(p_i) + \partial D^-_{p_k,\sigma_{ik}}(p_i)\right) \cdot \|p'_ip_i\|. \end{aligned}$$

By Lemma 3.1, $C_{p_0,\sigma_{0i}}(p'_i) + D_{p_k,\sigma_{ik}}(p'_i) > C_{p_0,\sigma_{0i}}(p_i) + D_{p_k,\sigma_{ik}}(p_i)$, and therefore, $\partial C^-_{p_0,\sigma_{0i}}(p_i) + \partial D^-_{p_k,\sigma_{ik}}(p_i) < 0$.

Conversely, suppose that the three conditions are satisfied for some $i \in [1, k-1]$. Let p'_i be the intersection point between e_i and the canonical LSP from p_0 to p_k with edge sequence σ. If $p'_i = p_i$, we are done. Suppose that $p'_i \neq p_i$.

Consider the case of p'_i lying on the positive side of p_i. By Lemma 3.2, $C_{p_0,\sigma_{0i}}$ and $D_{p_k,\sigma_{ik}}$ are convex functions. Therefore,

$$\begin{aligned} C_{p_0,\sigma_{0i}}(p'_i) &\geq C_{p_0,\sigma_{0i}}(p_i) + \partial C^+_{p_0,\sigma_{0,i}}(p_i) \cdot \|p_ip'_i\| \\ D_{p_k,\sigma_{ik}}(p'_i) &\geq D_{p_k,\sigma_{ik}}(p_i) + \partial D^+_{p_k,\sigma_{ik}}(p_i) \cdot \|p_ip'_i\| \end{aligned}$$

Combining these two inequalities and condition (ii) in the lemma gives

$$C_{p_0,\sigma_{0i}}(p'_i) + D_{p_0,\sigma_{0i}}(p'_i) \geq C_{p_0,\sigma_{0i}}(p_i) + D_{p_0,\sigma_{0i}}(p_i),$$

which shows that P is also an LSP. However, p_i is on the negative side of p'_i, which is a contradiction to Lemma 3.1.

Consider the case of p'_i lying on the negative side of p_i. By the convexity argument again, we obtain

$$\begin{aligned} C_{p_0,\sigma_{0i}}(p'_i) &\geq C_{p_0,\sigma_{0i}}(p_i) + \partial C^-_{p_0,\sigma_{0,i}}(p_i) \cdot \|p_i p'_i\| \\ D_{p_k,\sigma_{ik}}(p'_i) &\geq D_{p_k,\sigma_{ik}}(p_i) + \partial D^-_{p_k,\sigma_{ik}}(p_i) \cdot \|p_i p'_i\| \end{aligned}$$

But then these two inequalities and condition (iii) in the lemma imply that

$$C_{p_0,\sigma_{0i}}(p'_i) + D_{p_0,\sigma_{0i}}(p'_i) > C_{p_0,\sigma_{0i}}(p_i) + D_{p_0,\sigma_{0i}}(p_i).$$

But P cannot be shorter than an LSP, a contradiction. □

Lemma 3.4 below shows that when we slide an LSP, the path cost changes linearly.

LEMMA 3.4. *Let $P = (p_0, \ldots, p_k)$ be an LSP with edge sequence σ, where p_i lies in the interior of e_i for $i \in [1,k]$. Let $Q = (q_0, \ldots, q_k)$ be another path such that $q_i \in e_i$ for $i \in [0,k]$ and $q_{i-1}q_i$ is parallel to $p_{i-1}p_i$ for $i \in [1,k]$. For $i,j \in [0,k]$ such that $i < j$, define δ_{ij} and γ_{ij} by the relations $\|p_iq_i\| = \delta_{ij} \cdot \|p_jq_j\|$ and $\text{cost}(Q[q_i,q_j]) = \text{cost}(P[p_i,p_j]) + \gamma_{ij} \cdot \langle q_j - p_j, \hat{e_j} \rangle$. Then, δ_{ij} and γ_{ij} depend on the direction vector of $P[p_i,p_j]$ only, $\delta_{i-1,i}$ and $\gamma_{i-1,i}$ can be computed in $O(1)$ time, and for all $\ell \in [i+1, j-1]$, $\delta_{ij} = \delta_{i\ell}\delta_{\ell j}$ and $\gamma_{ij} = \delta_{\ell j}\gamma_{i\ell} + \gamma_{\ell j}$.*

PROOF. Let $\hat{v}_i$ be the direction of $p_{i-1}p_i$. By the sine law, $\delta_{i-1,i} = \sin(\theta(\hat{v}_i, \hat{e}_{i-1}))/\sin(\theta(\hat{v}_i, \hat{e}_i))$. The edges e_{i-1} and e_i share a negative endpoint a or a positive endpoint b, and $\|q_{i-1}q_i\| = \sin(\theta(\hat{e}_i, \hat{e}_{i-1})) \cdot \|aq_i\|/\sin(\theta(\hat{v}_i, \hat{e}_{i-1}))$ and $\|q_{i-1}q_i\| = \sin(\theta(\hat{e}_{i-1}, \hat{e}_i)) \cdot \|q_ib\|/\sin(\theta(\hat{v}_i, \hat{e}_{i-1}))$, respectively. Similar identities hold for $\|p_{i-1}p_i\|$. Thus, $\gamma_{i-1,i} = c_i \cdot \sin(\theta(\hat{e}_i, \hat{e}_{i-1}))/\sin(\theta(\hat{v}_i, \hat{e}_{i-1}))$, where c_i is the cost of a unit segment with direction $\hat{v}_i$ in the face bounded by e_{i-1} and e_i. So $\delta_{i-1,i}$ and $\gamma_{i-1,i}$ depend on $\hat{v}_i$ only. Assume that $i < j-1$. For all $\ell \in (i,j)$, $\|p_iq_i\| = \delta_{i\ell} \cdot \|p_\ell q_\ell\| = \delta_{i\ell}\delta_{\ell j} \cdot \|p_jq_j\|$, and

$$\begin{aligned} & \text{cost}(Q[q_i,q_j]) \\ = \ & \text{cost}(Q[q_i,q_\ell]) + \text{cost}(Q[q_\ell,q_j]) \\ = \ & \text{cost}(P[p_i,p_\ell]) + \gamma_{i\ell} \cdot \langle q_\ell - p_\ell, \hat{e}_\ell \rangle \\ & + \text{cost}(P[p_\ell,p_j]) + \gamma_{\ell j} \cdot \langle q_j - p_j, \hat{e}_j \rangle \\ = \ & \text{cost}(P[p_i,p_j]) + (\delta_{\ell j}\gamma_{i\ell} + \gamma_{\ell j}) \cdot \langle q_j - p_j, \hat{e}_j \rangle. \end{aligned}$$

So $\delta_{ij} = \delta_{i\ell}\delta_{\ell j}$ and $\gamma_{ij} = \delta_{\ell j}\gamma_{i\ell} + \gamma_{\ell j}$. Inductively, δ_{ij} and γ_{ij} depend on the direction vector of $P[i,j]$ only. □

We want to show that $\partial C^+_{p_0,\sigma_{0k}}$, $\partial C^-_{p_0,\sigma_{0k}}$, $\partial D^+_{p_k,\sigma_{0k}}$, and $\partial D^-_{p_k,\sigma_{0k}}$ depend on the direction vector only, i.e., not on the location of p_0 and p_k. Then, Lemmas 3.3 and 3.4 allow us to form canonical LSPs by sliding and concatenating shorter ones. The first step is a conditional version of this result.

LEMMA 3.5. *Let $P = (p_0, \ldots, p_k)$ be an LSP with edge sequence σ, where p_i lies in the interior of e_i for $i \in [1,k]$. Define δ_{ij} and γ_{ij} as in Lemma 3.4. If there exists $i \in [1, k-1]$ such that $\partial C^+_{p_0,\sigma_{0i}}$ and $\partial C^-_{p_0,\sigma_{0i}}$ depend only on the director vector of $P[p_0,p_i]$, and $\partial D^+_{p_k,\sigma_{ik}}$ and $\partial D^-_{p_k,\sigma_{ik}}$ depend only on the direction vector of $P[p_i,p_k]$, then:*

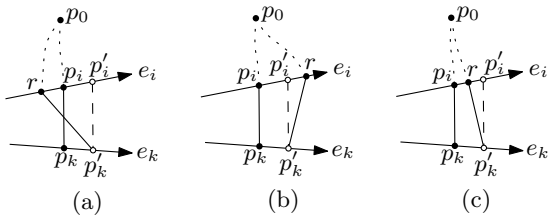

Figure 1: Three cases depending on the position of r relative to p_i and p'_i.

(i) $\partial C^+_{p_0,\sigma_{0k}}(p_k) = \min\{\partial C^+_{p_i,\sigma_{ik}}(p_k),\ \delta_{ik} \cdot \partial C^+_{p_0,\sigma_{0i}}(p_i) + \gamma_{ik}\}$ *and*
$\partial C^-_{p_0,\sigma_{0k}}(p_k) = \min\{\partial C^-_{p_i,\sigma_{ik}}(p_k),\ \delta_{ik} \cdot \partial C^-_{p_0,\sigma_{0i}}(p_i) + \gamma_{ik}\}$.

(ii) $\partial D^+_{p_k,\sigma_{0k}}(p_0) = \min\{\partial D^+_{p_i,\sigma_{0i}}(p_0),\ \frac{1}{\delta_{0i}}\partial D^+_{p_k,\sigma_{ik}}(p_i) + \frac{\gamma_{0i}}{\delta_{0i}}\}$ *and*
$\partial D^-_{p_k,\sigma_{0k}}(p_0) = \min\{\partial D^-_{p_i,\sigma_{0i}}(p_0),\ \frac{1}{\delta_{0i}}\partial D^-_{p_0,\sigma_{0i}}(p_i) + \frac{\gamma_{0i}}{\delta_{0i}}\}$.

PROOF. Consider the derivation of $\partial C^+_{p_0,\sigma_{0k}}(p_k)$ in (i). The derivation of $\partial C^-_{p_0,\sigma_{0k}}(p_k)$ is symmetric. Take a point $p'_k \in e_k$ on the positive side of p_k and arbitrarily close to p_k. For $j \in [i,k]$, let p'_j be the point in e_j such that $p'_jp'_{j+1}$ is parallel to p_jp_{j+1}. Since p'_k is arbitrarily close to p_k, p'_i is also arbitrarily close to p_i. Therefore,

$$\begin{aligned} & C_{p_0,\sigma_{0i}}(p_i) + C_{p_i,\sigma_{ik}}(p'_k) \\ = \ & C_{p_0,\sigma_{0i}}(p_i) + C_{p_i,\sigma_{ik}}(p_k) + \partial C^+_{p_i,\sigma_{ik}}(p_k) \cdot \|p_kp'_k\| \\ = \ & C_{p_0,\sigma_{0k}}(p_k) + \partial C^+_{p_i,\sigma_{ik}}(p_k) \cdot \|p_kp'_k\|, \text{ and} \end{aligned}$$

$$\begin{aligned} & C_{p_0,\sigma_{0i}}(p'_i) + C_{p'_i,\sigma_{ik}}(p'_k) \\ = \ & C_{p_0,\sigma_{0i}}(p_i) + \partial C^+_{p_0,\sigma_{0i}}(p_i) \cdot \|p_ip'_i\| + \\ & C_{p_i,\sigma_{ik}}(p_k) + \gamma_{ik} \cdot \|p_kp'_k\| \\ = \ & C_{p_0,\sigma_{0k}}(p_k) + \big(\delta_{ik} \cdot \partial C^+_{p_0,\sigma_{0k}}(p_k) + \gamma_{ik}\big) \cdot \|p_kp'_k\|. \end{aligned}$$

The correctness of (i) follows if we can show that $C_{p_0,\sigma_{0k}}(p'_k)$ equals $C_{p_0,\sigma_{0i}}(p_i) + C_{p_i,\sigma_{ik}}(p'_k)$ or $C_{p_0,\sigma_{0i}}(p'_i) + C_{p'_i,\sigma_{ik}}(p'_k)$.

Let Q be an LSP from p_0 to p'_k with edge sequence $\sigma_{0k} = \sigma$. Let r be the node of Q on e_i. There are three cases as shown in Figure 1 depending on the position of r.

Suppose that r is on the negative side of p_i. See Figure 1(a). $Q[r,p'_k]$ and $P[p_i,p_k]$ cross in this case, say at point x. Since P and Q are LSPs, their subpaths are also LSPs. Thus, $\text{cost}(P[p_0,x]) = \text{cost}(Q[p_0,x])$, and so $C_{p_0,\sigma_{0k}}(p'_k) = \text{cost}(Q) = \text{cost}(P[p_0,x]) + \text{cost}(Q[x,p'_k]) \geq C_{p_0,\sigma_{0i}}(p_i) + C_{p_i,\sigma_{ik}}(p'_k)$. An LSP to p'_k cannot cost more than any path to p'_k via p_i. Thus, $C_{p_0,\sigma_{0k}}(p'_k) = C_{p_0,\sigma_{0i}}(p_i) + C_{p_i,\sigma_{ik}}(p'_k)$.

Suppose that r is on the positive side of p'_i. See Figure 1(b). Since $C_{p_0,\sigma_{0i}}$ is a convex function by Lemma 3.2, $C_{p_0,\sigma_{0i}}(r) \geq C_{p_0,\sigma_{0i}}(p_i) + \partial C^+_{p_0,\sigma_{0i}}(p_i) \cdot \|p_ir\| = C_{p_0,\sigma_{0i}}(p'_i) + \partial C^+_{p_0,\sigma_{0i}}(p_i) \cdot \|p'_ir\|$, where the last equality follows from the fact that p'_i is arbitrarily close to p_i. Because $p'_jp'_{j+1}$ is parallel to p_jp_{j+1} for all $j \in [i,k-1]$, we obtain $\partial D^+_{p'_k,\sigma_{ik}}(p'_i) = \partial D^+_{p_k,\sigma_{ik}}(p_i)$ by the assumption that $\partial D^+_{p_k,\sigma_{ik}}$ depends on the direction vector of $P[i,j]$ only. Thus, $D_{p'_k,\sigma_{ik}}(r) \geq D_{p'_k,\sigma_{ik}}(p'_i) + \partial D^+_{p'_k,\sigma_{ik}}(p'_i) \cdot \|p'_ir\| = C_{p'_i,\sigma_{ik}}(p'_k) + \partial D^+_{p_k,\sigma_{ik}}(p_i) \cdot \|p'_ir\|$. By combining the two inequalities above, we obtain $C_{p_0,\sigma_{0k}}(p'_k) = C_{p_0,\sigma_{0i}}(r) + D_{p'_k,\sigma_{ik}}(r) \geq C_{p_0,\sigma_{0i}}(p'_i) +$

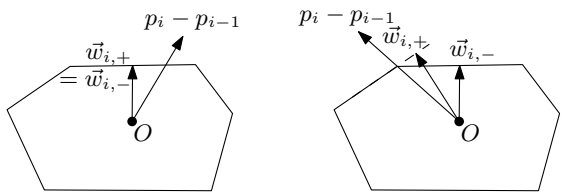

Figure 2: Left: The ray in the direction of $p_i - p_{i-1}$ crosses the boundary of H_{f_i} at a point that is not a vertex. $\vec{w}_{i,-} = \vec{w}_{i,+}$. Right: The ray in the direction of $p_i - p_{i-1}$ crosses the boundary of H_{f_i} at a vertex, so $\vec{w}_{i,-}$ and $\vec{w}_{i,+}$ are defined by the edges of H_{f_i} incident to that vertex.

$C_{p'_i\sigma_{ik}}(p'_k) + \left(\partial C^+_{p_0,\sigma_{ik}}(p_i) + \partial D^+_{p_k,\sigma_{ik}}(p_i)\right) \|p'_i r\|$, which is at least $C_{p_0,\sigma_{0i}}(p'_i) + C_{p'_i\sigma_{ik}}(p'_k)$ by Lemma 3.3.

Suppose that $r \in p_i p'_i$. See Figure 1(c). Since $\|p_i r\|$ and $\|r p'_i\|$ are arbitrarily small, $C_{p_0,\sigma_{0k}}(p'_k) = C_{p_0,\sigma_{0i}}(r) + D_{p'_k,\sigma_{ik}}(r) = C_{p_0,\sigma_{0i}}(p_i) + \partial C^+_{p_0,\sigma_{0i}}(p_i) \cdot \|p_i r\| + D_{p'_k,\sigma_{ik}}(p'_i) - \partial D^+_{p'_k,\sigma_{ik}}(p'_i) \cdot \|p'_i r\|$, which is linear in $\|p_i r\|$ by our assumption that $\partial C^+_{p_0,\sigma_{0i}}$ and $\partial D^+_{p_k,\sigma_{ik}}$ depend only on the direction vectors of $P[p_0, p_i]$ and $P[i, k]$ (hence $\partial D^+_{p'_k,\sigma_{ik}}(p'_i) = \partial D^+_{p_k,\sigma_{ik}}(p_i)$). Thus, $C_{p_0,\sigma_{0k}}(p'_k)$ is minimized when $r = p_i$ or $r = p'_i$. which means $C_{p_0,\sigma_{0k}}(p'_k) = C_{p_0,\sigma_{0i}}(p_i) + D_{p'_k,\sigma_{ik}}(p_i) = C_{p_0,\sigma_{0i}}(p_i) + C_{p_i,\sigma_{ik}}(p'_k)$ or $C_{p_0,\sigma_{0k}}(p'_k) = C_{p_0,\sigma_{0i}}(p'_i) + D_{p'_k,\sigma_{ik}}(p'_i) = C_{p_0,\sigma_{0i}}(p'_i) + C_{p'_i,\sigma_{ik}}(p'_k)$.

The correctness of (ii) can be proved in a similar way. □

Lemma 3.5 lends itself to an inductive proof to establish the same result unconditionally, as stated in Lemma 3.6.

LEMMA 3.6. *Let $P = (p_0, \ldots, p_k)$ be an LSP with edge sequence σ, where p_i lies in the interior of e_i for $i \in [1, k]$. Let δ_{ij} and γ_{ij} be defined as in Lemma 3.4. Then $\partial C^+_{p_0,\sigma_{0k}}$, $\partial C^-_{p_0,\sigma_{0k}}$, $\partial D^+_{p_k,\sigma_{0k}}$, and $\partial D^-_{p_k,\sigma_{0k}}$ depend only on the direction vector of P. Moreover, the formulae in Lemma 3.5 hold for all $i \in [1, k-1]$.*

PROOF. We first show that $\partial C^+_{p_{i-1},(e_i)}(p_i)$ depends only on the direction of $p_i - p_{i-1}$. Divide all directions into *cones*, each being the set of directions from the origin to all points in one edge of the polygon H_{f_i} defining the distance function for the face f bound by e_{i-1} and e_i.

If $p_i - p_{i-1}$ points to a vertex of H_{f_i}, there are two cones that contain $p_i - p_{i-1}$. We use ℓ_- to denote the support line of the edge of H_{f_i} defining the cone that comes first in anticlockwise order among these two cones, and ℓ_+ denotes the support line of the edge of H_{f_i} that defines the other cone. If $p_i - p_{i-1}$ points to the interior of an edge of H_{f_i}, then both ℓ_+ and ℓ_- denote the support line of this edge. Let $\vec{w}_{i,+}$ and $\vec{w}_{i,-}$ be the vectors that are orthogonal to ℓ_+ and ℓ_-, respectively. See Figure 2. It follows that $\text{cost}(p_{i-1}p'_i) = \frac{\langle p'_i - p_{i-1}, \vec{w}_{i,+} \rangle}{\|\vec{w}_{i,+}\|^2}$ and $\text{cost}(p_{i-1}p_i) = \frac{\langle p_i - p_{i-1}, \vec{w}_{i,+} \rangle}{\|\vec{w}_{i,+}\|^2}$. So $\partial C^+_{p_{i-1},(e_i)}(p_i) = \frac{\langle \hat{e}_i, \vec{w}_{i,+} \rangle}{\|\vec{w}_{i,+}\|^2}$, which only depends on the direction of $p_i - p_{i-1}$. Similarly, one can verify that $\partial C^-_{p_{i-1},(e_i)}(p_i) = \frac{\langle \hat{e}_i, \vec{w}_{i,-} \rangle}{\|\vec{w}_{i,-}\|^2}$, $\partial D^+_{p_i,(e_i)}(p_{i-1}) = -\frac{\langle \hat{e}_{i-1}, \vec{w}_{i,+} \rangle}{\|\vec{w}_{i,+}\|^2}$, $\partial D^-_{p_i,(e_i)}(p_{i-1}) = -\frac{\langle \hat{e}_{i-1}, \vec{w}_{i,-} \rangle}{\|\vec{w}_{i,-}\|^2}$. They all depend on the direction of $p_i - p_{i-1}$ only.

$\partial C^+_{p_0,\sigma_{01}}(p_1)$ and $\partial C^-_{p_0,\sigma_{01}}(p_1)$ depend only on the direction of $p_1 - p_0$ as discussed above. Applying Lemma 3.5(i) with $i = 1$ and $k = 2$ shows that $\partial C^+_{p_0,\sigma_{02}}(p_2)$ and $\partial C^-_{p_0,\sigma_{02}}(p_2)$ depend only on the directions of $p_1 - p_0$ and $p_2 - p_1$. By repeatedly applying Lemma 3.5(i) with $k = i + 1$, one can show that $\partial C^+_{p_0,\sigma_{0i}}(p_i)$ and $\partial C^-_{p_0,\sigma_{0i}}(p_i)$ depend only on the direction vector of $P[p_0, p_i]$ for all $i \in [1, k-1]$.

By Lemma 3.5(ii), one can similarly show that $\partial D^+_{p_k,\sigma_{ik}}(p_i)$ and $\partial D^-_{p_k,\sigma_{ik}}(p_i)$ depend only on the direction vector of $P[p_i, p_k]$. Thus, the conditions on Lemma 3.5(i) and (ii) can be removed. □

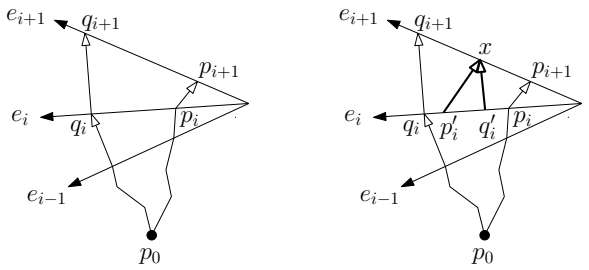

Figure 3: Illustration for the proof of Lemma 3.7.

Lemma 3.7 below follows from Lemmas 3.2, 3.3, and 3.6. It implies that once two canonical LSPs diverge, they cannot cross afterward.

LEMMA 3.7. *Let $P = (p_0, \ldots, p_k)$ and $Q = (q_0, \ldots, q_k)$ be two canonical LSPs with edge sequence σ such that p_i and q_i lie in the interior of e_i for $i \in [1, k-1]$. If $\theta(p_i - p_{i-1}, \vec{e}_i) > \theta(q_i - q_{i-1}, \vec{e}_i)$ for some $i \in [1, k-1]$, then $\theta(p_j - p_{j-1}, \vec{e}_j) \geq \theta(q_j - q_{j-1}, \vec{e}_j)$ for all $j > i$.*

PROOF. It suffices to show that if $\theta(p_\ell - p_{\ell-1}, \vec{e}_\ell) \geq \theta(q_\ell - q_{\ell-1}, \vec{e}_\ell)$ for all $\ell \in [1, i]$ and $\theta(p_\ell - p_{\ell-1}, \vec{e}_\ell) > \theta(q_\ell - q_{\ell-1}, \vec{e}_\ell)$ for some $\ell \in [1, i]$, then $\theta(p_{i+1} - p_i, \vec{e}_{i+1}) \geq \theta(q_{i+1} - q_i, \vec{e}_{i+1})$. By Lemma 3.6, we can assume that $p_0 = q_0$. So q_i is on the positive side of p_i. Assume that $p_i p_{i+1}$ and $q_i q_{i+1}$ are not parallel because we are done otherwise.

Translate the segments $p_i p_{i+1}$ and $q_i q_{i+1}$ to obtain parallel segments $p'_i x$ and $q'_i x$, respectively, that meet at some point $x \in e_{i+1}$. Refer to Figure 3. The choices of p'_i, q'_i and x are quite arbitrary as long as $p_i p_{i+1}$ and $q_i q_{i+1}$ are parallel to $p'_i x$ and $q'_i x$, respectively.

We claim that $\partial D^+_{x,(e_{i+1})}(q'_i) < \partial D^+_{x,(e_{i+1})}(p'_i)$. Suppose not. Then, $\partial D^+_{x,(e_{i+1})}(q'_i) \geq \partial D^-_{x,(e_{i+1})}(q'_i) \geq \partial D^+_{x,(e_{i+1})}(p'_i)$ as $D_{x,(e_{i+1})}$ is convex and $p'_i \neq q'_i$. Lemma 3.6 implies that $\partial D^-_{q_{i+1},(e_{i+1})}(q_i) = \partial D^-_{x,(e_{i+1})}(q'_i) \geq \partial D^+_{x,(e_{i+1})}(p'_i) = \partial D^+_{p_{i+1},(e_{i+1})}(p_i)$. Since q_i is on the positive side of p_i, by Lemma 3.2, $\partial C^-_{p_0,\sigma_{0i}}(q_i) \geq \partial C^+_{p_0,\sigma_{0i}}(p_i)$. Then $\partial C^-_{p_0,\sigma_{0i}}(q_i) + \partial D^-_{q_{i+1},(e_{i+1})}(q_i) \geq \partial C^+_{p_0,\sigma_{0i}}(p_i) + \partial D^+_{p_{i+1},(e_{i+1})}(p_i)$, which is non-negative by Lemma 3.3(ii). This is a contradiction because $\partial C^-_{p_0,\sigma_{0i}}(q_i) + \partial D^-_{q_{i+1},(e_{i+1})}(q_i)$ should be negative by Lemma 3.3(iii).

By our claim, q'_i lies on the negative side of p'_i. Since q_i is on the positive side of p_i, the relation $\theta(p_{i+1} - p_i, \vec{e}_{i+1}) \geq \theta(q_{i+1} - q_i, \vec{e}_{i+1})$ must hold in order that the sliding switches the order of p_i and q_i to align p_{i+1} and q_{i+1}. □

3.2 Algorithm

Chen and Han introduced the *sequence tree* to capture the edge sequences of LSPs in the L_2 case [6]. The tree is grown

until the number of tree levels meets the input upper bound on the number of links allowed in the solution path. The best path discovered from s to t is the shortest path desired. Constructing a new tree node involves finding a new shortest path with a particular edge sequence. The key is to use the structural properties in the last subsection to carry out this step and do it fast.

A sequence tree node α is a *vertex-node* or an *edge-node* which represents a vertex, denoted ν_α, or an edge of $\mathcal{T}$, denoted e_α. A *face corner* (f, ν) is the corner at a vertex ν of a face f. An edge-node α *annexes* a face corner (f, ν) if e_α is the edge of f opposite ν and the parent of α does not correspond to another vertex or edge of f. (Since e_α is opposite two face corners, the second condition ensures that α annexes the face corner just included by the growing tree.)

The root corresponds to the source s. The nodes on the tree path from the root to α correspond to an edge sequence, denoted σ_α. Let α_0 be the nearest ancestor vertex-node of α. The edge-nodes on the tree path from α_0 to α correspond to a suffix of σ_α, denoted $\tilde{\sigma}_\alpha$. The edge sequences σ_α and $\tilde{\sigma}_\alpha$ are used in the analysis, but they are not stored at α. If α is a vertex-node, P_α denotes the canonical LSP from s to ν_α that passes through the edges in σ_α. We compute $\text{cost}(P_\alpha)$ and store it at α, but P_α is used in the analysis only.

The sequence tree is grown in a breadth-first manner until the number of tree levels meets the input upper bound m. When an edge-node annexing a face corner (f, ν) is expanded, it gains at most one vertex-node corresponding to ν and two edge-nodes corresponding to the edges of f incident to ν. When a vertex-node α is expanded, it gains at most one vertex-node for each vertex adjacent to ν_α and one edge-node for each edge opposite ν_α. Multiple nodes may correspond to the same edge or vertex. To control the tree size, Chan and Han introduced the *one-corner one-split property*: *at any time, at most one vertex-node corresponding to the same vertex is allowed to have any child node; at most one edge-node annexing the same face corner is allowed to have two child edge-nodes.* This property ensures that at most $O(n)$ tree nodes are ever created at each level [6, Theorem 8]. This is the reason why we forbid critical refractions. If they are allowed, the one-corner one-split property cannot be enforced and the sequence tree may be much larger.

A notion of *dominance* is needed to maintain the one-corner one-split property. Let α and β be two vertex-nodes corresponding to the same vertex ν or two edge-nodes annexing the same face corner (f, ν). Let α_0 and β_0 be the nearest ancestor vertex-nodes of α and β, respectively. Let P and Q be the canonical LSPs from ν_{α_0} and ν_{β_0} to ν that pass through the edges in $\tilde{\sigma}_\alpha$ and $\tilde{\sigma}_\beta$, respectively. We say that α *dominates* β if $\text{cost}(P_{\alpha_0}) + \text{cost}(P) < \text{cost}(P_{\beta_0}) + \text{cost}(Q)$, or $\text{cost}(P_{\alpha_0}) + \text{cost}(P) = \text{cost}(P_{\beta_0}) + \text{cost}(Q)$ but α is expanded before β in growing the tree. Assume that α dominates β. Suppose they are vertex-nodes. If β has been expanded, we remove all tree nodes descending from it; otherwise, we will not expand β. Suppose that α and β are edge-nodes. There is an edge e incident to ν such that every LSP from ν_{β_0} to e through the edges in $\tilde{\sigma}_\beta$ crosses P. If β has been expanded, we prune the child node of β corresponding to e; otherwise, when we expand β, we will not generate a child node corresponding to e.

After we construct a new leaf α of the sequence tree, it takes $O(\log mn)$ amortized time to test the dominance and prune the tree, modulo the time to compute the costs of LSPs: $\text{cost}(P_\alpha)$ if α is a vertex-node, or the costs of LSPs from ν_{α_0} to e_α with edge sequence $\tilde{\sigma}_\alpha$ if α is an edge-node. In the rest of this subsection, we describe the dominance testing, the pruning, and the computation of the costs of LSPs when constructing a new leaf.

3.2.1 Dominance checking and tree pruning

The vertex-node case is easy. For each vertex of $\mathcal{T}$, we record the current corresponding vertex-node β that dominates all other vertex-nodes corresponding to ν_β. When a new vertex-node α corresponding to ν_β is created, we compare α and β to see which of the two dominates the other. If β is dominated, we delete all descendants of β. A node can only be deleted at most once. We charge the pruning work to the creation of the pruned nodes. Thus, it takes only $O(1)$ amortized time modulo the time for computing the cost of the LSP from s to ν_α with edge sequence σ_α.

It takes more time to handle edge-nodes. For every face corner (f, ν), we record the edge-node β that annexes (f, ν) and dominates all other edge-nodes annexing (f, ν). We say that β *occupies* (f, ν). Suppose that a new edge-node α annexing (f, ν) is generated. Let α' and β' be the nearest proper ancestor vertex-nodes of α and β, respectively. Let P and Q be the LSPs from $\nu_{\alpha'}$ and $\nu_{\beta'}$ to ν through the edges in $\tilde{\sigma}_\sigma$ and $\tilde{\sigma}_\beta$, respectively. We must have computed and recorded $\text{cost}(P_{\beta'}) + \text{cost}(Q)$ beforehand as β occupies (f, ν). Therefore, modulo the time to compute $\text{cost}(P)$, we can compare $\text{cost}(P_{\alpha'}) + \text{cost}(P)$ with $\text{cost}(P_{\beta'}) + \text{cost}(Q)$ to decide the dominance in $O(1)$ time. Without loss of generality, assume that α dominates β. Then, α replaces β as the edge-node that occupies (f, ν).

To decide which child edge-node of β to prune, we need to refine the notion of dominance. Consider the two edge sequences σ_α and σ_β. Let e denote the first edge in the longest common suffix of σ_α and σ_β.

- If σ_α is not a suffix of σ_β, let e_α be the edge in σ_α before e. Then α dominates β *on the positive side* (resp. *negative side*) if e_α and e share the positive (resp. negative) endpoint. Refer to Figure 4(top).

- If σ_α is a suffix of σ_β, let e_β be the edge in σ_β before e, and α dominates β on the positive side (resp. negative side) if e_β and e share the negative (resp. positive) endpoint. Refer to Figure 4(bottom).

We use e^+ and e^- to denote the two edges of f incident to ν such that ν is the negative and positive endpoints of e^+ and e^-, respectively. Suppose that α dominates β on the positive side. If β has been expanded, we delete the child node of β corresponding to e^+ as well as its descendants. Again, this pruning takes $O(1)$ amortized time. If β has not yet been expanded, we will not let β gain a child node corresponding to e^+. The pruning is symmetric if α dominates β on the negative side.

Tracing σ_α and σ_β to decide whether α dominates β on the positive or negative side would take $\Theta(\min\{|\sigma_\alpha|, |\sigma_\beta|\})$ time. Instead, we use some data structures for making this decision. For every face corner (f, ν), we maintain an ordered list of edge-nodes annexing it. These edge-nodes correspond to the same edge e of f. Let u^+ and u^- be the positive and negative endpoints of e, respectively. Let $g = wu^+u^-$ be the face of $\mathcal{T}$ that shares e with f. The ordering of two edge-nodes α and β in the ordered list for (f, ν) is determined

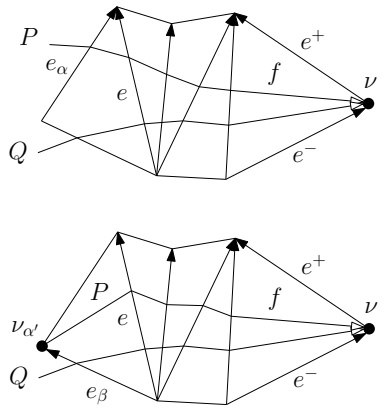

Figure 4: α dominates β on the positive side. The child-node of β corresponding to e^+ will have no descendant in the sequence tree.

as follows. Let α' and β' be the parent nodes of α and β, respectively.

- Suppose that α' and β' are edge-nodes annexing different corners of g. If $e_{\alpha'}$ and e share the common positive endpoint u^+, then α precedes β in the ordered list for (f, ν); otherwise, β precedes α.
- If α' and β' are edge-nodes annexing the same corner of g, and α' precedes β' in the ordered list for that face corner, then α precedes β in the ordered list for (f, ν).
- If β' is an edge-node annexing (g, u^+) and α' is a vertex-node corresponding to w, then α precedes β in the ordered list for (f, ν).
- If β' is a vertex-node corresponding to w and α' is an edge-node annexing (g, u^-), then α precedes β in the ordered list for (f, ν).

Assume that α dominates β. If α precedes β in the ordered list for (f, ν), then α dominates β on the positive side; otherwise, α dominates β on the negative side. The rules above are based on the information at the parents of α and β in such a way that the decision process is equivalent to tracing σ_α and σ_β. This explains the correctness. Since an edge-node annexing (f, ν) can change, we need to represent the sorted list for (f, ν) with a balanced binary search tree. The total size of such sorted lists is at most the sequence tree size which is $O(mn)$. Therefore, the dominance testing can be done in $O(\log mn)$ time.

The following lemma was originally proved for L_∞ metric. Since the proof only uses the triangle inequality, the result can be generalized to our case.

LEMMA 3.8 ([7, LEMMA 3.1]). *Let α and β be two edge-nodes annexing the same face corner (f, ν) such that α dominates β on the positive side (resp. negative side). Let e be the edge in f whose negative (resp. positive) endpoint is ν.*

(i) *α is not a descendant of β.*

(ii) *Let α_0 and β_0 be the nearest proper ancestor vertex-nodes of α and β, respectively. For every point $x \in e$ and every LSP Q with edge sequence $\tilde{\sigma}_\beta \cdot (e)$ from ν_{β_0} to ν, the LSP P with edge sequence $\tilde{\sigma}_\alpha \cdot (e)$ from ν_{α_0} to ν satisfies $\text{cost}(P_{\alpha_0}) + \text{cost}(P) \le \text{cost}(P_{\beta_0}) + \text{cost}(Q)$, and if they are equal, then α is expanded before β.*

3.2.2 Edge-node creation

Let α be a new edge-node created at tree level ℓ. Let α_0 be the nearest ancestor vertex-node of α at tree level $\ell_0 < \ell$. For $j \in (\ell_0, \ell]$, let e_j be the edge corresponding to the edge-node at tree level j on the tree path from α_0 to α. Let e_{ℓ_0} denote the edge incident to ν_{α_0} and adjacent to e_{ℓ_0+1} in σ_α. We do some processing at α to aid the future growth of the subtree rooted at α. For all $i \ge 0$ such that 2^i divides ℓ, we compute a data structure $\mathcal{L}^i_\alpha$ to represent the canonical LSPs from any point in $e_{\ell-2^i}$ to some point in e_ℓ, which can be represented by their direction vectors by Lemmas 3.3 and 3.6. The insight is that only some critical direction vectors matter, and the rest can be linearly interpolated from them.

Let $I_{\alpha,\mathbf{v}} \subseteq e_{\ell-2^i}$ be the interval of origins of canonical LSPs that reach e_ℓ with direction vector $\mathbf{v}$ and edge sequence $(e_{\ell-2^i+1}, \ldots, e_\ell)$.[3] Let $A_{\alpha,\mathbf{v}} : I_{\alpha,\mathbf{v}} \to \mathbb{R}$ and $a_{\alpha,\mathbf{v}} : I_{\alpha,\mathbf{v}} \to e_\ell$ be functions such that $A_{\alpha,\mathbf{v}}(p)$ is the cost of the canonical LSP from p to e_ℓ with direction vector $\mathbf{v}$ and edge sequence $(e_{\ell-2^i+1}, \ldots, e_\ell)$, and $a_{\alpha,\mathbf{v}}(p)$ is the destination of this LSP. Let $B_{\alpha,\mathbf{v}} : a_{\alpha,\mathbf{v}}[I_{\alpha,\mathbf{v}}] \to \mathbb{R}$ and $b_{\alpha,\mathbf{v}} : a_{\alpha,\mathbf{v}}[I_{\alpha,\mathbf{v}}] \to e_{\ell-2^i}$ be functions such that $B_{\alpha,\mathbf{v}}(q)$ is the cost of the canonical LSP from $e_{\ell-2^i}$ to q with direction vector $\mathbf{v}$ and edge sequence $(e_{\ell-2^i+1}, \ldots, e_\ell)$, and $b_{\alpha,\mathbf{v}}(q)$ is the source of this LSP. These four functions are affine and they can be stored in $O(1)$ space and evaluated in $O(1)$ time. The direction vectors in $\mathcal{L}^i_\alpha$ are stored in lexicographic order: two directions $\hat{v}_i$ and $\hat{w}_i$ for the links hitting e_i are ordered by comparing $\theta(\hat{e}_i, \hat{v}_i)$ and $\theta(\hat{e}_i, \hat{w}_i)$. The following properties are enforced on $\mathcal{L}^i_\alpha$.

P1: Each direction vector in $\mathcal{L}^i_\alpha$ is the direction vector of some canonical LSP from $e_{\ell-2^i}$ to e_ℓ.

P2: Any two adjacent direction vectors differ in exactly one entry. These two different directions point to the same edge of the convex polygon defining the distance function for the corresponding face.

P3: Let $\mathbf{v}$ and $\mathbf{w}$ be two adjacent direction vectors. For every $p \in I_{\alpha,\mathbf{v}} \cap I_{\alpha,\mathbf{w}}$ and every $t \in [0, 1]$, the cost of an LSP from p to $t\, a_{\alpha,\mathbf{v}}(p) + (1-t) a_{\alpha,\mathbf{w}}(p)$ is $tA_{\alpha,\mathbf{v}}(p) + (1-t)A_{\alpha,\mathbf{w}}(p)$.[4]

P4: Let $\mathbf{v}$ and $\mathbf{w}$ be two adjacent direction vectors. For every $q \in a_{\alpha,\mathbf{v}}[I_{\alpha,\mathbf{v}}] \cap a_{\alpha,\mathbf{w}}[I_{\alpha,\mathbf{w}}]$ and every $t \in [0, 1]$, the cost of an LSP from $t\, b_{\alpha,\mathbf{v}}(q) + (1-t) b_{\alpha,\mathbf{w}}(q)$ to q is $tB_{\alpha,\mathbf{v}}(q) + (1-t)B_{\alpha,\mathbf{w}}(q)$.

The first direction vector in $\mathcal{L}^i_\alpha$ is stored in its full form. For any other direction vector, we only store the directions of the first and last links and the difference from its predecessor in $\mathcal{L}^i_\alpha$. By P2, the storage required by $\mathcal{L}^i_\alpha$ is $O(2^i)$ plus the number of direction vectors in the list.

[3] Let P be the canonical LSP from $I_{\alpha,\mathbf{v}}$ to e_ℓ with direction vector $\mathbf{v}$ and edge sequence $(e_{\ell-2^i+1}, \ldots, e_\ell)$. By Lemma 3.6, we can slide P until it is stuck, and the path remains a canonical LSP during the sliding. Thus, $I_{\alpha,\mathbf{v}}$ is an interval, and so is $a_{\alpha,\mathbf{v}}[I_{\alpha,\mathbf{v}}]$.

[4] By Lemma 3.3, given two canonical LSPs from p to e_ℓ with adjacent direction vectors $\mathbf{v}$ and $\mathbf{w}$, any linearly interpolation of the two different directions yield another direction vector for which there is a canonical LSP from p to e_ℓ. The same holds for two canonical LSPs with adjacent direction vectors $\mathbf{v}$ and $\mathbf{w}$ from $e_{\ell-2^i}$ to the same point in e_ℓ.

The construction of $\mathcal{L}^i_\alpha$ proceeds in increasing i. The base case is $\mathcal{L}^0_\alpha$. Let H_ℓ denote the convex polygon that induces the distance function for the face bounded by $e_{\ell-1}$ and e_ℓ. $\mathcal{L}^0_\alpha$ consists of the direction vector $(-\hat{e}_{\ell-1}, \hat{e}_\ell)$ or $(\hat{e}_{\ell-1}, -\hat{e}_\ell)$ depending on whether $e_{\ell-1}$ and e_ℓ share a negative or positive endpoint, respectively, and every vector consisting of a single direction that points to a vertex of H_ℓ and can be used to go from $e_{\ell-1}$ to e_ℓ. For $i > 0$, let β be the ancestor edge-node of α at level $\ell - 2^{i-1}$, and let $(\mathbf{u}_1, \ldots, \mathbf{u}_r)$ and $(\mathbf{v}_1, \ldots, \mathbf{v}_{r'})$ be the sequences of direction vectors in $\mathcal{L}^{i-1}_\beta$ and $\mathcal{L}^{i-1}_\alpha$, respectively. $\mathcal{L}^{i-1}_\beta$ has been computed as β is at tree level $\ell - 2^{i-1}$ and 2^i divides ℓ. Choose an arbitrary point $p \in I_{\beta,\mathbf{u}_k} \cap I_{\beta,\mathbf{u}_{k+1}}$.

For all $k \in [1, r-1]$, take an arbitrary $p \in I_{\beta,\mathbf{u}_k} \cap I_{\beta,\mathbf{u}_{k+1}}$ and compute $\lambda_{\beta,k} = \frac{A_{\beta,\mathbf{u}_{k+1}}(p) - A_{\beta,\mathbf{u}_k}(p)}{\left\| a_{\beta,\mathbf{u}_k}(p) \;\; a_{\beta,\mathbf{u}_{k+1}}(p) \right\|}$. By P3, $\lambda_{\beta,k}$ equals $\partial C^+_{p,\sigma}$ at $a_{\beta,\mathbf{u}_k}(p)$, where $\sigma = (e_{\ell-2^i+1}, \ldots, e_{\ell-2^{i-1}})$, which is consistent with Lemma 3.6: $\partial C^+_{p,\sigma}$ is independent of the source p and the destination. Similarly, for $k \in [1, r'-1]$, take an arbitrary $q \in a_{\alpha,\mathbf{v}_k}[I_{\alpha,\mathbf{v}_k}] \cap a_{\alpha,\mathbf{v}_{k+1}}[I_{\alpha,\mathbf{v}_{k+1}}]$ and compute $\pi_{\alpha,k} = \frac{B_{\alpha,\mathbf{v}_{k+1}}(q) - B_{\alpha,\mathbf{v}_k}(q)}{\left\| b_{\alpha,\mathbf{v}_k}(q) \;\; b_{\alpha,\mathbf{v}_{k+1}}(q) \right\|}$, which equals $\partial D^+_{q,\sigma}$ at $b_{\alpha,\mathbf{v}_k}(q)$. By Lemma 3.3, $\mathcal{L}^i_\alpha$ consists of every concatenation $\mathbf{u}_j\mathbf{v}_k$ such that $\lambda_{\beta,j} + \pi_{\alpha,k} \geq 0$ and $\lambda_{\beta,j-1} + \pi_{\alpha,k-1} < 0$. By Lemma 3.2, $\lambda_{\beta,j} \leq \lambda_{\beta,j+1}$ and $\pi_{\alpha,k} \leq \pi_{\alpha,k+1}$, so we can scan $\lambda_{\beta,j}$ in increasing j and $\pi_{\alpha,k}$ in decreasing k to identify the good concatenations. We first find $k_0 \in [1, r']$ such that $\lambda_{\beta,1} + \pi_{\alpha,k_0} \geq 0$ and $\lambda_{\beta,1} + \pi_{\alpha,k_0-1} < 0$, and so $\mathbf{u}_1\mathbf{v}_{k_0}$ is a good concatenation. Note that $\lambda_{\beta,1} + \pi_{\alpha,k} < 0$ for all $k < k_0$. Next, we find $k_1 \leq k_0$ such that $\lambda_{\beta,2} + \pi_{\alpha,k_1} \geq 0$ and $\lambda_{\beta,2} + \pi_{\alpha,k_1-1} < 0$. Thus, $\lambda_{\beta,2} + \pi_{\alpha,k} \geq 0$ and $\lambda_{\beta,1} + \pi_{\alpha,k-1} < 0$ for all $k \in [k_1, k_0]$, which makes $\mathbf{u}_2\mathbf{v}_k$ a good concatenation for all $k \in [k_1, k_0]$. Repeating the above gives $\mathcal{L}^i_\alpha$. When adding a concatenation $\mathbf{uv}$, we compute in $O(1)$ time $I_{\alpha,\mathbf{uv}} = b_{\beta,\mathbf{u}}[a_{\beta,\mathbf{u}}[I_{\beta,\mathbf{u}}] \cap I_{\alpha,\mathbf{v}}]$, $A_{\alpha,\mathbf{uv}} = A_{\beta,\mathbf{u}} + A_{\alpha,\mathbf{v}} \circ a_{\beta,\mathbf{u}}$, $a_{\alpha,\mathbf{uv}} = a_{\alpha,\mathbf{v}} \circ a_{\beta,\mathbf{u}}$, $B_{\alpha,\mathbf{uv}} = B_{\alpha,\mathbf{v}} + B_{\beta,\mathbf{u}} \circ b_{\alpha,\mathbf{v}}$, and $b_{\alpha,\mathbf{uv}} = b_{\beta,\mathbf{u}} \circ b_{\alpha,\mathbf{v}}$, where the operator $\circ$ composes two functions.

LEMMA 3.9. *For every $i \geq 0$, $\mathcal{L}^i_\alpha$ satisfies* P1–P4.

PROOF. Consider the base case of $i = 0$. P1 holds because any direction vector added has only one link. P2 holds by the choices of directions picked by the algorithm. Suppose that $\mathbf{v}_j = (\hat{v}_j)$ and $\mathbf{v}_{j+1} = (\hat{v}_{j+1})$ are two adjacent direction vectors. The direction of the oriented segment from p to any point between $a_{\alpha,\mathbf{v}_j}(p)$ and $a_{\alpha,\mathbf{v}_{j+1}}(p)$ lies between $\hat{v}_j$ and $\hat{v}_{j+1}$. Since $\hat{v}_j$ and $\hat{v}_{j+1}$ point to the same edge of the convex polygon that defines the distance function, the cost of the segment from p to a point between $a_{\alpha,\mathbf{v}_j}(p)$ and $a_{\alpha,\mathbf{v}_{j+1}}(p)$ is a linear interpolation of $A_{\alpha,\mathbf{v}_j}(p)$ and $A_{\alpha,\mathbf{v}_{j+1}}(p)$. Thus, P3 holds. P4 can be proved similarly.

Consider the case of $i > 0$. Let ℓ be the level of α. Let β be the ancestor edge-node of α at level $\ell - 2^{i-1}$. Assume that P1–P4 hold for both $\mathcal{L}^{i-1}_\alpha$ and $\mathcal{L}^{i-1}_\beta$. P1 holds for $\mathcal{L}^i_\alpha$ by our method to identify good concatenations.

Consider P2. Any two successive concatenations added to $\mathcal{L}^i_\alpha$ share either a prefix, i.e. $\mathbf{uv}$ and $\mathbf{uv}'$, or a suffix, i.e. $\mathbf{uv}$ and $\mathbf{u}'\mathbf{v}$. By P2, $\mathbf{u}$ and $\mathbf{u}'$ differ in exactly one entry, and so do $\mathbf{v}$ and $\mathbf{v}'$. It follows that $\mathcal{L}^i_\alpha$ satisfies P2.

Consider P3. Take any two adjacent direction vectors in $\mathcal{L}^{i-1}_\alpha$ and $\mathcal{L}^{i-1}_\beta$. They differ in one entry by P2 and we can write them as $\mathbf{u} = \mathbf{w}(\hat{w}_0)\mathbf{w}'$ and $\mathbf{v} = \mathbf{w}(\hat{w}_1)\mathbf{w}'$. Let P and Q be the canonical LSPs from p to $a_{\alpha,\mathbf{u}}(p)$ and $a_{\alpha,\mathbf{v}}(p)$ respectively. Consider the canonical LSP R from p to a point $q = (1-t)a_{\alpha,\mathbf{u}}(p) + t\, a_{\alpha,\mathbf{v}}(p)$ for some $t \in [0, 1]$. By Lemma 3.3, the direction vector of R is $\mathbf{w}(\hat{w})\mathbf{w}'$, where $\hat{w}$ lies between $\hat{w}_0$ and $\hat{w}_1$. Let rx, ry and rz be the segments of P, R and Q, respectively, that have directions $\hat{w}_0$, $\hat{w}$ and $\hat{w}_1$, respectively. Because $R[y, q]$, $P[x, a_{\alpha,\mathbf{u}}(p)]$ and $Q[z, a_{\alpha,\mathbf{v}}(p)]$ have the same direction vector, we get $y = (1-t)x + tz$, and $\text{cost}(R[y, q]) = (1-t)\,\text{cost}(P[x, a_{\alpha,\mathbf{u}}(p)]) + t\,\text{cost}(Q[z, a_{\alpha,\mathbf{v}}(p)])$. By P2, $\text{cost}(ry) = (1-t)\,\text{cost}(rx) + t\,\text{cost}(rz)$. Therefore, $\text{cost}(R) = (1-t)\,\text{cost}(P) + t\,\text{cost}(Q)$.

P4 can be proved similarly. □

LEMMA 3.10. *An edge-node at level ℓ takes $O(2^i h)$ time to create, where 2^i is the largest power of 2 that divides ℓ, and h is the maximum size of the convex polygons associated with the faces.*

PROOF. Let α be a new edge-node at level ℓ. $\mathcal{L}^0_\alpha$ stores $O(h)$ direction vectors. For $i > 0$, let β be the edge-node at level $\ell - 2^{i-1}$, the size of $\mathcal{L}^i_\alpha$ is at most the total size of $\mathcal{L}^{i-1}_\alpha$ and $\mathcal{L}^{i-1}_\beta$. Inductively, we obtain a time bound of $\sum_{j=0}^{i-1} O(2^j h) = O(2^i h)$. □

3.2.3 Compute an LSP to a vertex

Suppose that we expand an edge-node α at tree level ℓ that annexes a face corner (f, ν). Let α_0 be the nearest ancestor vertex-node of α at tree level $\ell_0 < \ell$. Let $(e_{\ell_0+1}, \ldots, e_\ell)$ be the edge sequence corresponding to the edge-nodes on the tree path from α_0 to α. We are to create a vertex-node β for ν and compute the cost of the canonical LSP P_β from s to ν through the edges $(e_{\ell_0+1}, \ldots, e_\ell)$.

For $i = 1, 2, \ldots$, find the largest ℓ_i such that $\ell_i \leq \ell$ and $\ell_i - \ell_{i-1}$ is a power of 2 that divides both ℓ_i and ℓ_{i-1}. This gives a sequence $\ell_0 < \ell_1 < \ldots < \ell_r = \ell$, where $r = O(\log \ell)$. For $i \geq 1$, let $k_i = \ell_i - \ell_{i-1}$ and let α_i be the ancestor edge-node of α at level ℓ_i. We also use α_r to denote α. Let σ_i denote the edge sequence $(e_{\ell_0+1}, \ldots, e_{\ell_i})$ for $i \in [1, r]$. P_β is the concatenation of P_{α_0} and the canonical LSP P from ν_{α_0} to ν through the edges $(e_{\ell_0+1}, \ldots, e_\ell)$. We already know $\text{cost}(P_{\alpha_0})$. We compute $\text{cost}(P)$ by combining the $\mathcal{L}^{k_i}_{\alpha_i}$'s in at most $r+1$ stages. At the end of the i-th stage, $i \in [1, r]$, we fix the prefix Q_i of P up to e_{ℓ_i}.[5]

Assume that Q_{i-1} is fixed. Let x_{i-1} denote its destination. Assume that we have computed $\partial C^+_{\nu_{\alpha_0},\sigma_{i-1}}(x_{i-1})$ and $\partial C^-_{\nu_{\alpha_0},\sigma_{i-1}}(x_{i-1})$. Consider the canonical LSPs from ν_{α_0} through $(e_{\ell_{i-1}+1}, \ldots, e_\ell)$ with Q_{i-1} as a common prefix. By Lemmas 3.2 and 3.7, these LSPs spread out from x_{i-1} to e_ℓ and form a fan that contains ν_α. We want to construct a path R from x_{i-1} to a point $y \in e_{\ell_i}$ such that $Q_i = Q_{i-1}R$ and the canonical LSPs that spread out from y to e_ℓ form a fan that contains ν_α. Let $\mathbf{u}_{i-1}$ be the direction vector of Q_{i-1}. The idea is to find the direction vector $\mathbf{v}$ in $\mathcal{L}^{k_i}_{\alpha_i}$ by binary search such that $\mathbf{uv}$ is the direction vector of Q_i.

The binary search works as follows. Let $\mathbf{v}$ be the "median" direction vector in the sublist of $\mathcal{L}^{k_i}_{\alpha_i}$ that we are working on. If $x_{i-1} \notin I_{\alpha_i,\mathbf{v}}$, we remove half of the sublist of $\mathcal{L}^{k_i}_{\alpha_i}$ and recurse. Suppose that $x_{i-1} \in I_{\alpha_i,\mathbf{v}}$. We find the smallest direction vector $\mathbf{w}$ and the largest direction vector $\mathbf{w}'$ such that $\mathbf{u}_{i-1}\mathbf{vw}$ and $\mathbf{u}_{i-1}\mathbf{vw}'$ extend Q_{i-1} to two canonical LSPs through $(e_{\ell_{i-1}+1}, \ldots, e_\ell)$ and the face f. (We will

[5] We define the prefix Q_r of P only up to $e_{\ell_r} = e_\ell$ instead of an edge of f incident to ν. It is because we will apply Lemmas 3.4–3.6, which require the nodes the path other than its source to be in the interior of edges.

describe how to find $\mathbf{w}$ and $\mathbf{w}'$ shortly.) If these two LSPs lie on the same side of ν_α, by Lemma 3.7, we can remove half of the sublist of $\mathcal{L}_{\alpha_i}^{k_i}$ and recurse. If these two LSPs sandwich ν_α, then $\mathbf{u}_{i-1}\mathbf{v}$ extends Q_{i-1} to Q_i. The destination of Q_i is $y = a_{\alpha_i,\mathbf{v}}(x_{i-1})$ and $\text{cost}(Q_i) = \text{cost}(Q_{i-1}) + A_{\alpha_i,\mathbf{v}}(x_{i-1})$. By Lemmas 3.4–3.6, $\partial C^+_{\nu_{\alpha_0},\sigma_i}(y)$ and $\partial C^-_{\nu_{\alpha_0},\sigma_i}(y)$ can be computed in $O(1)$ time. Then we fix the next prefix Q_{i+1}. If we proceed all the way to stage r and fix Q_r, then $\text{cost}(P) = \text{cost}(Q_r) + \text{cost}(x_r\nu)$, where x_r is the destination of Q_r. The binary search may also finish with two adjacent direction vectors in $\mathcal{L}_{\alpha_i}^{k_i}$ without fixing Q_i, a terminating case that we discuss after the next paragraph.

How do we find the smallest and largest direction vectors $\mathbf{w}$ and $\mathbf{w}'$? By Lemma 3.3, we find the smallest direction vector $\mathbf{w}_{i+1}$ in $\mathcal{L}_{\alpha_{i+1}}^{k_{i+1}}$ by binary search such that $\mathbf{v}\mathbf{w}_{i+1}$ is the direction vector of some canonical LSP from x_{i-1} through $(e_{\ell_{i-1}+1}, \ldots, e_{\ell_{i+1}})$. Let $y = a_{\alpha_i,\mathbf{v}}(x_{i-1})$ and let $z = a_{\alpha_{i+1},\mathbf{w}_{i+1}}(y)$. We apply Lemmas 3.4–3.6 to compute $\partial C^+_{\nu_{\alpha_0},\sigma_{i+1}}(z)$ and $\partial C^-_{\nu_{\alpha_0},\sigma_{i+1}}(z)$ in $O(1)$ time. Then, we find the smallest direction vector $\mathbf{w}_{i+2}$ in $\mathcal{L}_{\alpha_{i+2}}^{k_{i+2}}$ by binary search and extend to $\mathbf{v}\mathbf{w}_{i+1}\mathbf{w}_{i+2}$. Repeating the above gives $\mathbf{w}_{i+1}\mathbf{w}_{i+2}\ldots\mathbf{w}_r$. Finally, we pick the smallest direction $\hat{w}_{r+1}$ according to Lemma 3.3 that extends $\mathbf{w}_{i+1}\mathbf{w}_{i+2}\ldots\mathbf{w}_r$ through f, and $\mathbf{w}_{i+1}\mathbf{w}_{i+2}\ldots\mathbf{w}_r(\hat{w}_{r+1})$ is the desired $\mathbf{w}$. The largest direction vector $\mathbf{w}'$ is obtained symmetrically.

Recall the terminating case that Q_i cannot be fixed and the binary search finishes with two adjacent direction vectors $\mathbf{v}$ and $\mathbf{v}'$ in $\mathcal{L}_{\alpha_i}^{k_i}$. We find the largest direction vector $\mathbf{w}$ as before to extend $\mathbf{v}$ through $(e_{\ell_i+1}, \ldots, e_\ell)$ and f. Note that $\mathbf{v}\mathbf{w}$ and $\mathbf{v}'\mathbf{w}$ extend Q_{i-1} to two canonical LSPs that sandwich ν, so $\mathbf{w}$ is the direction vector of the subpath of P from e_{ℓ_i} to ν. The last direction $\hat{w}_{r+1}$ in $\mathbf{w}$ brings us from ν to a point $z \in e_\ell$, and the cost is $\text{cost}(z\nu)$. We continue to $b_{\alpha_r,\mathbf{w}_r}(z) \in e_{\ell_{r-1}}$ and the cost accumulates to $\text{cost}(z\nu) + B_{\alpha_i,\mathbf{w}_r}(z)$, and so on to a point $y \in e_{\ell_i}$ between $a_{\alpha_i,\mathbf{v}}(x_{i-1})$ and $a_{\alpha_i,\mathbf{v}'}(x_{i-1})$, where x_{i-1} is the destination of Q_{i-1}. Let C be the cost of the path that we have retraced from ν to e_{ℓ_i}. Suppose that $y = (1-t)a_{\alpha_i,\mathbf{v}}(x_{i-1}) + t\,a_{\alpha_i,\mathbf{v}'}(x_{i-1})$. By P3 and Lemma 3.3, $\text{cost}(P) = \text{cost}(Q_{i-1}) + (1-t)A_{\alpha_i,\mathbf{v}}(x_{i-1}) + tA_{\alpha_i,\mathbf{v}'}(x_{i-1}) + C$.

THEOREM 3.1. *Let $\mathcal{T}$ be a polyhedral surface with n vertices in an instance of* PolyPath. *Given a source s, a destination t and an integer m, the shortest path from s to t on $\mathcal{T}$ with no more than m links can be found in $O(hmn \log mn + mn \log^2 m \log^2 hm)$ time, where h is the maximize size of the convex polygons that define the distance functions in the faces of $\mathcal{T}$.*

PROOF. Consider the correctness of the algorithm. Let P_0 be the shortest path from s to t with no more than m links. By the requirement of the PolyPath problem, we can assume that every node of P_0 is either a transversal node or a vertex of $\mathcal{T}$. If there are multiple choices for P_0, we pick P_0 to be one that has the fewest nodes.

The sequence tree is grown to contain the prefix of P_0 until the vertex-node corresponding to t is reached or an edge-node α_0 is dominated by some other edge-node β such that the child node of α_0 that would contain a longer prefix of P_0 is pruned. In the former case, the sequence tree captures the edge sequence of P_0, and the algorithm computes the cost of the LSP with respect to that edge sequence, so we are done. Consider the latter case. Let x be the intersection point between P_0 and the edge corresponding to α_0 and β. By Lemma 3.8, there exists a path Q from s to x with edge sequence σ_β that is at least as good as $P[s,x]$. Let $P_1 = Q \cdot P[x,t]$. By our choice of P_0, $\text{cost}(P_1) = \text{cost}(P_0)$, P_1 has the same number of nodes as P_0, and β is at the same depth as α_0 but expanded earlier. Note that β cannot be dominated by any other edge-node. The subtree of β grows to contain P_1, or a descendant α_1 of β is dominated by some other edge-node and the child of α_1 that would contain a longer prefix of P_1 is pruned. We can then repeat the analysis above, which can happen at most m times. The correctness thus follows.

Consider the running time. We spend $O(\log mn)$ amortized time in each invocation of dominance testing and pruning. So we create $O(mn)$ tree nodes in $O(mn \log mn)$ time.

Divide the edge-nodes into $O(\log m)$ groups such that an edge-node is in group i if its level is a multiple of 2^i but not 2^{i+1}. Group i contains $O(mn/2^i)$ edge-nodes. By Lemma 3.10, creating a node in group i takes $O(2^i h)$ time. So it takes $O(hnm \log m)$ time to create all the edge-nodes.

To compute the cost of an LSP for a vertex-node, we fix $O(\log m)$ prefixes Q_i's. To extend Q_{i-1} to Q_i, we binary search in $\mathcal{L}_{\alpha_i}^{k_i}$ in $O(\log 2^i h) = O(\log \ell h) = O(\log hm)$ probes by Lemma 3.10. Each probe requires $O(\log \ell) = O(\log m)$ binary searches among the lists $\mathcal{L}_{\alpha_{i+1}}^{k_{i+1}}$, $\mathcal{L}_{\alpha_{i+2}}^{k_{i+2}}, \ldots$. So it takes $O(\log^2 m \log^2 hm)$ time to compute the cost of an LSP. The total time spent on all vertex-nodes is thus $O(mn \log^2 m \log^2 hm)$. We can reconstruct the direction vector of the shortest path in a similar way as in dealing with the terminating case of not fixing some Q_i. So constructing the path takes only $O(m)$ time. □

4. APPLICATIONS

Under the L_1 and L_∞ metrics, $h = O(1)$. Under the L_p metric for some $p \geq 2$, Dudley's result [11] allows us to approximate the "unit disk" by a polygon of $O(1/\sqrt{\varepsilon})$ vertices such that the polygon diameter is approximated with an ε relative error, i.e., $h = O(1/\sqrt{\varepsilon})$. In the above cases, $m = O(n)$ because there exists a shortest path that visits a face no more than once.

THEOREM 4.1. *Given a polyhedral surface of size n, the L_1 and L_∞ shortest paths between two vertices can be computed in $O(n^2 \log^4 n)$ time, and for every constant $p \geq 2$ and every $\varepsilon \in (0,1)$, a $(1+\varepsilon)$-approximate L_p shortest path can be computed in $O\left(\frac{1}{\sqrt{\varepsilon}} n^2 \log n + n^2 \log^4 n\right)$ time.*

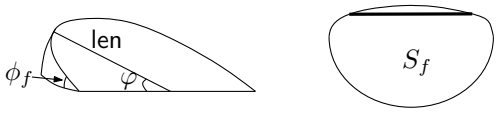

Figure 5: Left: The face f makes an angle ϕ_f with the horizontal, and the ascent is $\mathsf{len} \cdot \sin\varphi \sin\phi_f$. Right: The bold segment represents the clipping.

Consider path planning on a terrain with the cost function $c_1 \cdot$ Euclidean length $+ c_2 \cdot$ total ascent for some constants $c_1 > 0$ and $c_2 \geq 0$. Refer to the left image in Figure 5. The ascent within a face f is $\mathsf{len} \cdot \sin\varphi \sin\phi_f$, where len is the distance travelled in f, ϕ_f is the gradient of a face

f, and φ is the angle between the travel direction and the horizontal. Let S_f denote the "unit disk" induced. On the uphill side, the boundary of S_f satisfies the equation $1 = (c_1 + c_2 \sin\varphi \sin\phi_f)\,\mathsf{len}$; on the downhill side, the boundary of S_f is the half-circle with radius $1/c_1$. S_f is convex with bounded aspect ratio, so we can approximate it by Dudley's result [11] to obtain a PolyPath problem instance with $h = O(1/\sqrt{\varepsilon})$ and $m = O(n)$.

We can incorporate uphill gradient constraints. Let ψ be the input limit on the uphill path gradient. Let pq be an oriented segment in the interior of f that makes an angle larger than ψ with the horizontal. We can traverse a zigzag path within f from p to q in which each segment makes an angle ψ with the horizontal. The path length is equal to the height difference between p and q divided by $\sin\psi$, irrespective of the exact zigzag pattern. Under this constraint, the top part of S_f that makes an angle at least ψ with the horizontal should be clipped. Refer to the right image in Figure 5. We can similarly handle downhill gradient constraints. Note that such a zigzag is treated as a "single" link.

There are some technical issues. Let P be a shortest path from s to t that satisfies the gradient constraints. Consider the intersections between P and a face f. Let p be the first point of entry. Let q be the last point of exit. Suppose that neither p nor q is a vertex of f. Without loss of generality, assume that q is higher than p. Let ψ be the ascent gradient bound. We assume that the gradient of pq exceeds ψ; otherwise, we can connect p and q directly. Let p_0 and q_0 be in the interior of f arbitrarily close to p and q, respectively, such that pp_0 and q_0q satisfy the gradient constraints. We can follow pp_0, then a zigzag path from p_0 to q_0 and then q_0q to reach q from p with constant ascent gradient ψ. The lengths of segments pp_0 and q_0q are negligible, so the length of this path is $H/(\sin\psi)$, where H is the height difference between p and q. The subpath of P from p to q cannot be shorter because the same height difference H is covered with a ascent gradient no greater than ψ. At the same time, the ascent of the zigzag path is the smallest possible because it goes monotonically upward. If q is a vertex of f, it may be impossible to move from any point in f straight to q. After $O(n)$-time preprocessing, for every vertex, we can determine if it can be reached from some point in its close neighborhood under the gradient constraints. If there is a point q'' in a face f'' incident to q such that $q''q$ satisfies the gradient constraints, then we can first move from p to a point $q' \in f$ near q using a zigzag path such that every segment of the zigzag path has uphill gradient ψ, then to q'' and then to q. We can make q' and q'' arbitrarily close to q, so the detour cost can be made negligible. A similar detour may be needed for p if it is also a vertex.

This gives an instance of PolyPath with $h = O(1/\sqrt{\varepsilon})$ and $m = O(n)$. The algorithm need to be slightly modified due to the technical issues we mentioned earlier. If a vertex cannot be reached locally, then we never create the corresponding vertex-node in the sequence tree. If no path can be extended from from a vertex, then we do not expand the corresponding vertex-node.

Theorem 4.2. *Given a source s and a destination t on a polyhedral terrain of size n, we can find a $(1+\varepsilon)$-approximate shortest path under the cost function of c_1 length $+ c_2$ ascent for some constants $c_1 > 0$ and $c_2 \geq 0$, where* length *is the Euclidean path length and* ascent *is the total ascent. Gradient constraints can be imposed. The running time is $O\left(\frac{1}{\sqrt{\varepsilon}} n^2 \log n + n^2 \log^4 n\right)$.*

5. REFERENCES

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

[2] M. Ahmed and A. Lubiw. Shortest descending paths: towards an exact algorithm. *Internat. J. Comput. Geom. Appl.*, 21 (2011), 431–466.

[3] M. Ahmed, A. Lubiw, and A. Maheshwari. Shortest gently descending paths. *WALCOM*, 2009, 59–70.

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

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

[6] J. Chen and Y. Han. Shortest paths on a polyhedron, part I: computing shortest paths. *Internat. J. Comput. Geom. Appl.*, 6 (1996), 127–144.

[7] S.-W. Cheng and J. Jin. Approximate shortest descending paths. *Proc. 24th Annu. ACM-SIAM Sympos. Discrete Alg.*, 2013, 144–155.

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

[9] S.-W. Cheng, H.-S. Na, A. Vigneron, and Y. Wang. Querying approximate shortest paths in anisotropic regions. *SIAM J. Comput.*, 39 (2010), 1888–1918.

[10] W. Collischonn and J.V. Pilar. A direction dependent least-costs path algorithm for roads and canals. *Internat. J. Geo. Info. Sci.*, 14 (2000), 391–406.

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

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

[13] J.S.B. Mitchell, D.M. Mount and C.H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16 (1987), 647–668.

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

[15] J.A. Roles and H. ElAarag. A smoothest path algorithm and its visualization tool. *Proc. IEEE*, 2013, 1–6.

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

[17] C. Yu, J. Lee, and M.J. Munro-Stasiuk. Extensions to least-cost path algorithms for roadway planning. *Internat. J. Geo. Info. Sci.*, 17 (2003), 361–376.