Approximate Shortest Descending Paths*

Siu-Wing Cheng† Jiongxin Jin‡

Abstract

We present an approximation algorithm for the shortest descending path problem. Given a source s and a destination t on a terrain, a shortest descending path from s to t is a path of minimum Euclidean length on the terrain subject to the constraint that the height decreases monotonically as we traverse that path from s to t. Given any $\varepsilon \in (0,1)$, our algorithm returns in $O(n^4 \log(n/\varepsilon))$ time a descending path of length at most $1+\varepsilon$ times the optimum. This is the first algorithm whose running time is polynomial in n and $\log(1/\varepsilon)$ and independent of the terrain geometry.

1 Introduction

Euclidean shortest paths in the plane or on polygonal surfaces have applications in robotics and geographic information systems. It is also a classic optimization problem in computational geometry, so many results on exact and approximation algorithms are known. Hershberger and Suri [8] presented an $O(n \log n)$-time algorithm to find a shortest path in the plane among polygonal obstacles with n vertices. This algorithm was later extended by Schreiber and Sharir [13] to find a shortest path on a convex polyhedron with n vertices in $O(n \log n)$ time. For a non-convex polygonal surface with n vertices, Mitchell et al. [10] presented a shortest path algorithm that runs in $O(n^2 \log n)$ time, and Chen and Han [6] subsequently improved the running time to $O(n^2)$. There are also fast approximation algorithms. Agarwal et al. [1] proposed an algorithm to find a $(1+\varepsilon)$-approximate shortest path on a convex polyhedron in $O(n \log(1/\varepsilon) + 1/\varepsilon^3)$ time. Varadarajan and Agarwal [14] proposed two algorithms to find an approximate shortest path on a non-convex polyhedron. One returns a $7(1+\varepsilon)$ approximation in $O(n^{5/3} \log^{5/3} n)$ time and the other returns a $15(1+\varepsilon)$ approximation in $O(n^{8/5} \log^{8/3} n)$ time. More information can be found in Mitchell's survey [9].

The utility of a path is enhanced if appropriate features of the environment can be modeled. When hiking along a path on a rugged terrain[1], the common experience is that how much the path goes up and down is also an important concern in addition to the path length. A path is *descending* (resp. *ascending*) if the height is monotonically decreasing (resp. increasing) from source to destination. De Berg and van Kreveld [7] pioneered the study of some height constrained path query problems on terrains. They presented an algorithm to preprocess a terrain with n vertices in $O(n \log n)$ time into a data structure of $O(n)$ size so that for any two query points s and t, several queries can be answered in $O(\log n)$ time, such as whether there is a descending or ascending path from s to t, and whether there exists a path from s to t that

*Research supported by the Research Grant Council, Hong Kong, China (project no. 612109). The work was done while Jin was at the Department of Computer Science and Engineering, HKUST. A preliminary version appeared in Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, 2013, 144–155.

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

‡Google Inc., Seattle, USA. Email: jamesjjx@google.com

[1] A polygonal surface such that any vertical line intersects the surface in at most once point.

stays below a given height. De Berg and van Kreveld posed the combinations of path length optimization and height constraints as open problems.

There has been recent progress on the problem of finding a shortest descending path (SDP) from a source to a destination on a terrain. Roy et al. [12] presented an algorithm to compute an SDP on a convex or concave terrain in $O(n^2 \log n)$ time, and another algorithm to compute the SDP that passes through a given sequence of k parallel edges on a general terrain in $O(k \log k)$ time. Later, Ahmed and Lubiw [4] showed that a $(1+\varepsilon)$-approximate SDP that passes through a given sequence of k edges on a general terrain can be computed in $O(k^{3.5} \log(1/\varepsilon))$ time. Eventually, Ahmed et al. [3] discovered two algorithms to construct a $(1+\varepsilon)$-approximate SDP on a general terrain. The time complexities of these algorithms are $O\left(\frac{n^2 L}{\varepsilon h \cos\theta} \log \frac{nL}{\varepsilon h \cos\theta}\right)$ and $O\left(\frac{n^2 L}{\varepsilon h} \log^2 \frac{nL}{\varepsilon h}\right)$, where L is the largest edge length in the terrain, h is the smallest distance from a vertex to a non-incident edge in the same terrain face, and θ is the largest acute angle between a non-horizontal edge and a vertical line. Ahmed and Lubiw [5] recently showed that if there exists an algorithm to compute the SDP that passes through a given sequence of k terrain edges in $R(k)$ time, then one can use Chen and Han's approach [6] to compute an SDP in $O(n^2 R(n))$ time. Unfortunately, it is still unclear how to compute the SDP that passes through a given edge sequence. Moreover, replacing the exact algorithm by a $(1+\varepsilon)$-approximate SDP algorithm for a given edge sequence does not give a good approximation—it leads to a $\left(1 + c^{O(n)}\varepsilon\right)^{O(n)}$-approximate SDP, where c is a constant bigger than one [2, Section 3.4.3].

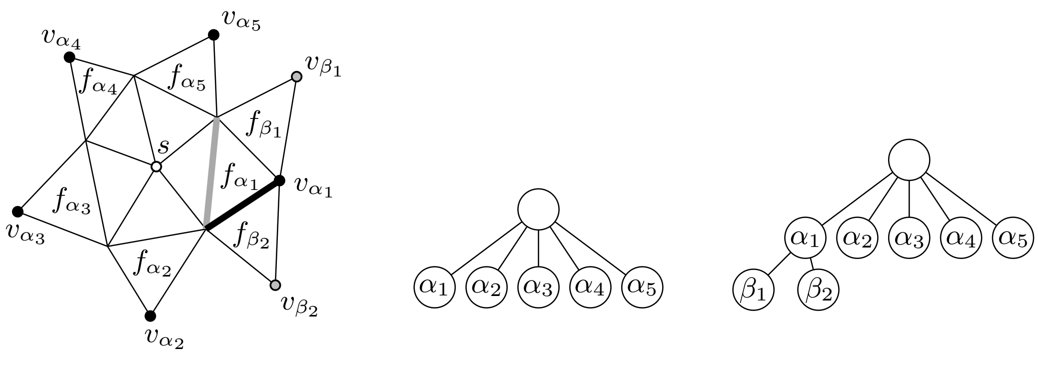
Our main result is that for any given $\varepsilon \in (0,1)$, a $(1+\varepsilon)$-approximate SDP can be computed in $O(n^4 \log(n/\varepsilon))$ time.[2] This is the first algorithm whose running time is polynomial in n and $\log(1/\varepsilon)$ and independent of the terrain geometry.

We observe that a constant factor approximation of the SDP length helps us to formulate an additive error bound for the SDP approximation algorithm. To this end, we compute an SDP in the L_∞ metric using Chen and Han's sequence tree [6]. We present two solutions, one based on linear programming and another faster combinatorial algorithm. In all, we can compute in $O(n^3 \log n)$ time the L_∞ SDP length, which is at most the SDP length and at least the SDP length divided by $\sqrt{3}$.

Our main SDP approximation algorithm is also based on Chen and Han's sequence tree. A key task is to compute an approximate SDP from s to a vertex v through a given sequence of edges. Our idea is to impose an *additive error* instead of requiring a $(1+\varepsilon)$-approximate SDP. The analysis is tightly coupled with the pruning of the sequence tree, which is essential for guaranteeing a small tree size. Since we cannot compute exact SDPs, we inevitably make mistakes in pruning the sequence tree, and we may not generate the sequence of edges passed through by an SDP from s to t. Hence, we cannot hope to prove the correctness of our algorithm by arguing that some perturbation of an SDP from s to t has been computed as a candidate. We introduce the *quasi-length* of a path and prune the sequence tree using the quasi-length instead of the true path length. This ensures that a short path remains after each pruning, which is a key step in the analysis.

Quasi-length is a new concept and it may be useful for other path problems on polygonal surfaces. The framework of our SDP approximation algorithm may also be applicable in solving other shortest path problems on terrains with constraints on height or other features.

[2]The existence of a descending path from the source to the destination can first be verified in $O(n \log n)$ time using the result of de Berg and van Kreveld [7].



(a) (b) (c)

Figure 1: In (b), the initial sequence tree consists of only the root and its five children corresponding to the face corners $(f_{\alpha_i}, v_{\alpha_i})$ for $i \in [1, 5]$ in (a). In (c), when the tree grows from the leaf α_1, α_1 acquires two children corresponding to the face corners $(f_{\beta_i}, v_{\beta_i})$ for $i \in [1, 2]$ in (a). The edge sequence σ_{β_2} is $(e_{\alpha_1}, e_{\beta_2})$, where e_{α_1} is the grey edge and e_{β_2} is the bold edge in (a).

2 Preliminaries

Let $\mathcal{T}$ be the input terrain, which is a polygonal surface in $\mathbb{R}^3$ that projects injectively onto the xy-plane. Let n be the number of vertices of $\mathcal{T}$. Without loss of generality, assume that every face of $\mathcal{T}$ is a triangle, and that the given source s and destination t are vertices of $\mathcal{T}$. For every point p in $\mathcal{T}$, let p_x, p_y and p_z denote the x-, y- and z-coordinates of p, respectively. When clockwise and anticlockwise orders are used subsequently, we refer to the view of $\mathcal{T}$ from above.

A polygonal path P in $\mathcal{T}$ is oriented from its source to destination, consisting of directed segments called *links*. P *conforms* to $\mathcal{T}$ if every link is contained in an edge or a face of $\mathcal{T}$. We assume that all polygonal paths in this paper conform to $\mathcal{T}$, which can be enforced by splitting the links at their crossings with the edges of $\mathcal{T}$. The endpoints of the links are called *nodes*. We use $\|P\|$ and $\|P\|_\infty$ to denote the Euclidean length and L_∞ length of P, respectively. P is *descending* if for every pair of points a, b that appear in order along P, $a_z \geq b_z$. For every pair of points $a, b \in P$ appearing in order along P, $P[a, b]$ denotes the subpath from a to b. If P and Q are two descending paths such that P's destination coincides with Q's source, their concatenation $P \cdot Q$ is also a descending path.

The *edge sequence* of P, denoted $\mathrm{seq}(P)$, is the list of all edges $(e_1, e_2, \ldots, e_k)$ that intersect the interior of P ordered by the intersections along P. The edges containing the source or destination of P do not belong to $\mathrm{seq}(P)$. We assume that P does not reflect at an edge as it does not happen for an SDP. Thus, for $i \in [1, k-1]$, e_i and e_{i+1} are distinct edges of the same face, and $e_{i-1} \neq e_{i+1}$. If the interior of P passes though a vertex, we have some freedom in setting $\mathrm{seq}(P)$. For example, let $u_1, u_2, \ldots, u_h$ be the vertices adjacent to v in anticlockwise order; a path entering v from the face $u_1 v u_h$ and leaving v through the face $u_i v u_{i+1}$ can be viewed as crossing $vu_1, vu_2, \ldots, vu_i$ or $vu_h, vu_{h-1}, \ldots, vu_{i+1}$. We use $|\mathrm{seq}(P)|$ to denote the number of edges in $\mathrm{seq}(P)$.

A descending path is a *locally shortest descending path* (LSDP) if it is the shortest one among all descending paths with the same source, destination, and edge sequence.[3] There is a unique LSDP for a given edge sequence [5]. An SDP is a shortest LSDP over all possible edge sequences.

We use a variant of Chen and Han's *sequence tree* [6] to capture the possible edge sequences

[3] An LSDP is allowed to go through a vertex by our definition, while this is forbidden in [5].

of LSDPs from s to other vertices of $\mathcal{T}$. In Chen and Han's version, the root represents s and the other internal nodes represent intervals on edges of $\mathcal{T}$ crossed by geodesic paths that originate from s. We retain their main ideas, but we use a different tree node definition in order to blend with the other parts of our algorithm. Our sequence tree models paths from s to t with non-empty edge sequences. The edge sequence can be empty only if s and t are vertices of the same face, in which case the SDP from s to t is trivially the segment st. We assume that s and t are not vertices of the same face for the rest of the paper.

Each non-root node α of the sequence tree corresponds to a face corner (f_α, v_α)—the corner of the face f_α at its vertex v_α—and the edge e_α of f_α opposite v_α. Let $\deg(s)$ denote the degree of s. For $i \in [1, \deg(s)]$, let e_{α_i} denote a distinct edge opposite s and let f_{α_i} denote the face incident to e_{α_i} but not s. The initial sequence tree consists of only the root and its children $\alpha_1, \ldots, \alpha_{\deg(s)}$. Each child node α_i stores e_{α_i} and the face corner $(f_{\alpha_i}, v_{\alpha_i})$. Figures 1(a,b) show an example.

In general, each tree node α stores an edge sequence σ_α in addition to the face corner (f_α, v_α). If α is the root, then $\sigma_\alpha = \emptyset$; otherwise, α has a parent β and $\sigma_\alpha = \sigma_\beta \cdot (e_\alpha)$. Intuitively, the tree path from the root to a node α represents the paths in $\mathcal{T}$ from s to points in f_α with edge sequence σ_α.

The sequence tree is grown by attaching child nodes to a leaf node α. Let e_{β_1}, e_α, and e_{β_2} be the edges of f_α in anticlockwise order around the boundary of f_α. For $i \in \{1, 2\}$, let f_{β_i} be the face that shares e_{β_i} with f_α, and let v_{β_i} be the vertex of f_{β_i} opposite e_{β_i}. The node α may acquire a left child β_1 and/or a right child β_2 corresponding to the face corners $(f_{\beta_1}, v_{\beta_1})$ and $(f_{\beta_2}, v_{\beta_2})$, respectively. Figures 1(a,c) show an example. Thus, every non-root node has at most two children. As the tree grows, it is possible that multiple tree nodes correspond to the same face corner.

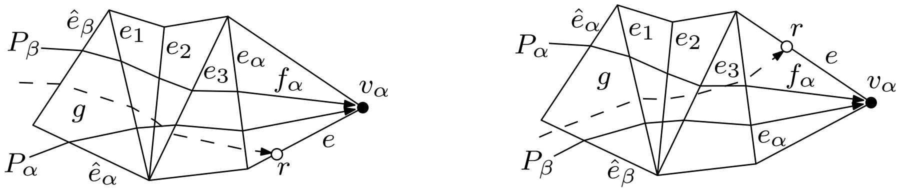
The sequence tree grows in a breadth-first manner until the number of tree levels equals the number of faces in $\mathcal{T}$. It is unnecessary to have more levels because an SDP does not visit a face more than once. If the sequence tree grows without any pruning, the tree size would become exponential in n eventually. To control the tree size, we impose the *one-corner one-split property: while a face corner may correspond to multiple nodes in the current sequence tree, at most one such tree node has two children in the current sequence tree.* If growing from a leaf violates the one-corner one-split property, the sequence tree is pruned to restore this property. Given the one-corner one-split property, Chen and Han proved that $O(n^2)$ nodes are ever created in constructing the sequence tree, including those that are pruned in some intermediate steps.

3 Exact algorithm for L_∞ SDP

We show how to compute the minimum L_∞ length of a descending path from s to t, denoted by opt_∞, which is a constant factor approximation of the SDP length opt from s to t because $\mathsf{opt}_\infty \leq \mathsf{opt} \leq \sqrt{3}\mathsf{opt}_\infty$. Knowing a constant factor approximation will allow our approximate SDP algorithm to focus on a smaller search space, and this is essential for making the time complexity independent of the terrain geometry.

We grow Chen and Han's sequence tree as described in Section 2. Recall that each tree node α in the sequence tree stores the edge sequence σ_α. To compute an L_∞ SDP from s to t, whenever we insert a new tree node α into the sequence tree, we compute an L_∞ LSDP from s to v_α with edge sequence σ_α. Denote this L_∞ LSDP stored at α by P_α. We will present two solutions for computing P_α, one based on linear programming (Section 3.1) and another faster combinatorial algorithm (Section 3.2). We ignore the computation of P_α for now.

To enforce the one-corner one-split property, we need a notion of *dominance* among the sequence tree nodes. Let α and β be two tree nodes corresponding to the same face corner



(a) (b)

Figure 2: Assume that α dominates β. The longest common suffix of seq(P_α) and seq(P_β) is $(e_1, e_2, e_3, e_\alpha)$. In (a), α dominates β on the right, and Lemma 3.1(ii) says that any L_∞ LSDP with edge sequence σ_β from s to a point $r \in e$ (shown as a dashed path) cannot be shorter than the L_∞ LSDP with edge sequence σ_α from s to r. In (b), α dominates β on the left, and Lemma 3.1(ii) gives a symmetric conclusion.

(f_α, v_α). The node α *dominates* β if $\|P_\alpha\|_\infty < \|P_\beta\|_\infty$, or $\|P_\alpha\|_\infty = \|P_\beta\|_\infty$ and $|\sigma_\alpha| < |\sigma_\beta|$. That is, the shorter of P_α and P_α is preferred, and if P_α and P_β have the same L_∞ length, the one with a shorter edge sequence is favored. Ties are broken arbitrarily when $\|P_\alpha\|_\infty = \|P_\beta\|_\infty$ and $|\sigma_\alpha| = |\sigma_\beta|$.

Once we know that α dominates β, we need to test whether α dominates β on the left or right. We prune the left child of β if β is dominated on the left; otherwise, we prune the right child of β. The test works as follows. The general case is that σ_α and σ_β are not suffixes of each other. In this case there exist edges $\hat{e}_\alpha$ and $\hat{e}_\beta$ in σ_α and σ_β, respectively, that immediately precede the longest common suffix of σ_α and σ_β.[4] Let e_1 be the first edge in the longest common suffix of σ_α and σ_β. Note that $\hat{e}_\alpha$, $\hat{e}_\beta$ and e_1 are contained in the same face g. Refer to Figure 2. The node α dominates β *on the right* (resp. *left*) if $(e_1, \hat{e}_\beta, \hat{e}_\alpha)$ are in anticlockwise (resp. clockwise) order around the boundary of g. In the special case that σ_α is a proper suffix of σ_β, replace $\hat{e}_\alpha$ by s in the test above. Similarly, if σ_β is a proper suffix of σ_α, replace $\hat{e}_\beta$ by s.

Lemma 3.1 below is the basis for the subsequent correctness proof of our pruning procedure. The shortcut argument in proving the first part of Lemma 3.1(ii) is essentially the same as that in [6]. But there is the subtlety that shortcutting a path by a line segment may not strictly decrease its L_∞ length. Therefore, we need to show that a node does not cause itself to be pruned and we need to handle paths with the same L_∞ length.

Lemma 3.1 *Let α and β be two tree nodes that correspond to the same face corner (f_α, v_α) such that α dominates β on the right (resp. left). Let e be the edge that follows e_α immediately in anticlockwise (resp. clockwise) order around the boundary of f_α.*

(i) *α is not a descendant of β.*

(ii) *For each point $r \in e$ and each L_∞ LSDP Q_β with edge sequence σ_β from s to r, every L_∞ LSDP Q_α with edge sequence σ_α from s to r satisfies $\|Q_\alpha\|_\infty \leq \|Q_\beta\|_\infty$ and if $\|Q_\alpha\|_\infty = \|Q_\beta\|_\infty$, then $|\sigma_\alpha| \leq |\sigma_\beta|$.*

Proof. Recall that P_α and P_β are L_∞ LSDPs from s to $v_\alpha = v_\beta$ with edge sequences σ_α and σ_β, respectively.

Consider (i). For the sake of contradiction, suppose that α is a descendant of β. So σ_β is a proper prefix of σ_α and, therefore, $|\sigma_\alpha| > |\sigma_\beta|$. It follows that $\|P_\alpha\|_\infty < \|P_\beta\|_\infty$ as α dominates

[4]Since α and β correspond to the same face corner (f_α, v_α), e_α is the last edge in both σ_α and σ_β. Hence, σ_α and σ_β have a non-empty common suffix.

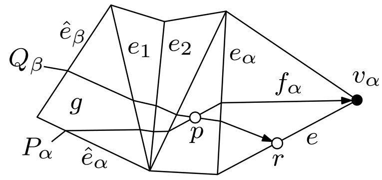


Figure 3: P_α and Q_β intersect at a point p such that $Q_\beta[p,r]$ and $P_\alpha[p,v_\alpha]$ have the same edge sequence.

β. P_α traverses the same edges and faces as P_β until f_β. Afterwards, P_β stops at v_β, and P_α leaves f_β through some point p on an edge of f_β other than e_β. Shortcut P_α by connecting p to v_β directly and removing the rest of P_α. This gives a descending path Q from s to v_β such that $\mathrm{seq}(Q) = \sigma_\beta$ and $\|Q\|_\infty \leq \|P_\alpha\|_\infty < \|P_\beta\|_\infty$. But then Q is shorter than the L_∞ LSDP P_β from s to v_β with edge sequence σ_β, a contradiction.

Consider (ii). Without loss of generality, assume that α dominates β on the right. Let $(e_1, e_2, \ldots, e_k)$ be the longest common suffix of σ_α and σ_β. Note that $e_k = e_\alpha = e_\beta$. Assume that σ_α and σ_β are not suffixes of each other. (So $\hat{e}_\alpha$ and $\hat{e}_\beta$ are well defined.) The two special cases of σ_α being a suffix of σ_β and σ_β being a suffix of σ_α are handled similarly.

We claim that the subpath of P_α from $\hat{e}_\alpha$ to v_α must intersect the subpath of Q_β from $\hat{e}_\beta$ to r. See Figure 3. Let g be the face of $\mathcal{T}$ incident to $\hat{e}_\alpha$, e_1 and $\hat{e}_\beta$. As α dominates β on the right by assumption, $(e_1, \hat{e}_\beta, \hat{e}_\alpha)$ are in anticlockwise order around the boundary of g. If P_α and Q_β intersect in g, our claim holds. If not, then $(Q_\beta \cap \hat{e}_\beta,\ P_\alpha \cap \hat{e}_\alpha,\ P_\alpha \cap e_1,\ Q_\beta \cap e_1)$ are in anticlockwise order around the boundary of g. Inductively, if P_α and Q_β do not intersect in the face incident to e_{i-1} and e_i for any $i \in [2,k]$, then for every $i \in [2,k]$, $(Q_\beta \cap e_{i-1},\ P_\alpha \cap e_{i-1},\ P_\alpha \cap e_i,\ Q_\beta \cap e_i)$ are in anticlockwise order around the boundary of the face incident to e_{i-1} and e_i. Consequently, if P_α and Q_β do not intersect before e_α, then within the face f_α the segment connecting $Q_\beta \cap e_k$ and r must intersect the segment connecting $P_\alpha \cap e_k$ and v_α. This proves our claim.

Let p be the intersection between P_α and Q_β furthest away from s along Q_β. It lies in a face incident to e_i for some $i \in [1,k]$. Note that $P_\alpha[p, v_\alpha]$ and $Q_\beta[p,r]$ have the same edge sequence. Since Q_α is an L_∞ LSDP from s to r with edge sequence σ_α by the assumption of the lemma, we obtain $\|Q_\alpha\|_\infty \leq \|P_\alpha[s,p]\|_\infty + \|Q_\beta[p,r]\|_\infty$ because σ_α is also the edge sequence of the descending path $P_\alpha[s,p] \cdot Q_\beta[p,r]$. Similarly, $\|P_\beta\|_\infty \leq \|Q_\beta[s,p]\|_\infty + \|P_\alpha[p,v_\alpha]\|_\infty$. Combining the two inequalities gives $\|Q_\alpha\|_\infty + \|P_\beta\|_\infty \leq \|Q_\beta\|_\infty + \|P_\alpha\|_\infty$. Since α dominates β, either $\|P_\alpha\|_\infty < \|P_\beta\|_\infty$, or $\|P_\alpha\|_\infty = \|P_\beta\|_\infty$ and $|\sigma_\alpha| \leq |\sigma_\beta|$. In the first case, we get $\|Q_\alpha\|_\infty < \|Q_\beta\|_\infty$. In the second case, we get $\|Q_\alpha\|_\infty \leq \|Q_\beta\|_\infty$ and $|\sigma_\alpha| \leq |\sigma_\beta|$. ◻

The L_∞ SDP algorithm works as follows. We first construct the initial sequence tree as described earlier. It consists of the root corresponding to s and $\deg(s)$ children, which are the leaves of the initial tree. Then we compute L_∞ LSDPs from s to the face corners represented by these $\deg(s)$ leaves. Throughout the algorithm, each face corner (f,v) stores the current tree node, if any, that corresponds to (f,v) and is not dominated by another tree node.

We grow the sequence tree in rounds. In each round, we grow from all the leaves at the bottommost level, and compute the L_∞ LSDPs to the face corners represented by the new leaves. We do not grow from leaves above the bottommost level. To grow from a leaf β, we create a right child (resp. left child) γ of β if β is not dominated on the right (resp. left) by any other tree node. After creating a new leaf γ, we compute an L_∞ LSDP from s to v_γ with edge sequence σ_γ. If no tree node is stored at the face corner (f_γ, v_γ), store γ there. Otherwise, suppose that α is the tree node stored at (f_γ, v_γ). If α dominates γ, then γ can gain only one child in the next round of expansion. If γ dominates α, then store γ at (f_γ, v_γ) instead

(γ cannot be dominated by any other tree node); furthermore, if γ dominates α on the right (resp. left), then delete α's right subtree (resp. left subtree). The one-corner one-split property is thus guaranteed.

The sequence tree expands in a breadth-first manner until the number of tree levels equals the number of faces in $\mathcal{T}$. After the tree stops growing, take the minimum L_∞ LSDP length from s to v_α among all tree nodes α such that $v_\alpha = t$. Return this minimum as opt_∞.

Lemma 3.2 *Let $T(k)$ be the time to compute an L_∞ LSDP from s to any vertex with an edge sequence of length k. An L_∞ SDP from s to t can be computed in $O(n^3+n^2\cdot T(m))$ time, where m and n are the number of faces and vertices in $\mathcal{T}$, respectively.*

Proof. We first analyze the running time. Due to the one-corner one-split property, only $O(n^2)$ nodes are created while constructing the sequence tree. When creating a new leaf γ, we compute an L_∞ LSDP from s to v_γ with edge sequence σ_γ, which takes $T(|\sigma_\gamma|) \le T(m)$ time. Given two tree nodes α and γ that correspond to the same face corner, we check the dominance between them in $O(1)$ time. Then, we trace σ_α and σ_γ backward in $O(n)$ time to decide if it is dominance on the left or right. In total, the running time is $O(n^2(n+T(m))) = O(n^3+n^2\cdot T(m))$.

We establish the correctness as follows. Let P_0 be an L_∞ SDP from s to t with the shortest edge sequence. Then P_0 does not visit a terrain face more than once. (An arbitrary L_∞ SDP may visit a face more than once.) So $|\mathrm{seq}(P_0)|$ is less than the bound on the number of levels in the sequence tree. If the final sequence tree contains a tree node β_0 such that $\sigma_{\beta_0} = \mathrm{seq}(P_0)$, then our algorithm must report a path length no more than $\|P_{\beta_0}\|_\infty = \|P_0\|_\infty$. Suppose that the final sequence tree does not contain such a tree node β_0. There must exist two nodes α_0 and α_1 in some intermediate sequence tree such that σ_{α_0} is a prefix of σ_{β_0} and α_1 dominates α_0. The child of α_0 that would be an ancestor of β_0 is pruned due to the dominance of α_1 over α_0. By Lemma 3.1(i), α_1 remains after this pruning. By Lemma 3.1(ii), we can replace a prefix of P_0 with edge sequence σ_{α_0} by a descending path Q with edge sequence σ_{α_1} so that the resulting descending path P_1 from s to t is not longer than P_0 (in the L_∞ metric). That is, P_1 is also an L_∞ SDP from s to t. The path Q cannot be shorter than the prefix of P_0 replaced (in the L_∞ metric) because P_0 is optimal. Then, Lemma 3.1(ii) further implies that $|\sigma_{\alpha_1}| \le |\sigma_{\alpha_0}|$ and, therefore, $|\mathrm{seq}(P_1)| \le |\sigma_{\beta_0}| = |\mathrm{seq}(P_0)|$. This implies that our algorithm can generate a node β_1 in the final sequence tree such that $\sigma_{\beta_1} = \mathrm{seq}(P_1)$ unless some ancestor of β_1 is dominated, causing β_1 to be pruned. If β_1 is in the final sequence tree, we are done. Otherwise, we can repeat the analysis above. Since the sequence tree is pruned a finite number of times, there must be an L_∞ SDP P_i from s to t with edge sequence σ_{β_i} so that the final sequence tree contains the node β_i. The correctness of the algorithm thus follows. ◻

Theorem 3.1 *Let s and t be two vertices in a polygonal terrain of n vertices. An L_∞ shortest descending path from s to t can be computed in $O(n^3 \log n)$ time.*

Proof. By Lemma 3.2, the correctness is guaranteed and the running time is $O(n^3+n^2\cdot T(m))$. Section 3.1 shows that $T(k)$ is polynomial in the input size using linear programming. Section 3.2 describes a combinatorial algorithm that reduces $T(k)$ to $O(k \log k)$. ◻

3.1 L_∞ LSDP by linear programming

We present a linear program that finds an L_∞ LSDP from s to a vertex w with edge sequence $(e_1, e_2, \ldots, e_k)$. For $i \in [1,k]$, let u_i and v_i denote the endpoints of e_i and every point in e_i can be written as $(1-\zeta_i)u_i + \zeta_i v_i$, where $\zeta_i \in [0,1]$. The x–coordinate of a point in e_i can be written as $(1-\zeta_i)u_{i,x} + \zeta_i v_{i,x}$, and we similarly can do the same for the y- and z- coordinates of any point in e_i. Let $u_0 = v_0 = s$. Let $u_{k+1} = v_{k+1} = w$. An L_∞ LSDP from s to w can be found by solving the following linear program.

$$\begin{array}{llcccl}
\min & \sum_{i=0}^{k} \ell_i & & & & \\
\text{s.t.} & -\ell_i & \leq & (1-\zeta_i)u_{i,x} + \zeta_i v_{i,x} - (1-\zeta_{i+1})u_{i+1,x} - \zeta_{i+1}v_{i+1,x} & \leq & \ell_i, \quad \forall\, i \in [0,k] \\
 & -\ell_i & \leq & (1-\zeta_i)u_{i,y} + \zeta_i v_{i,y} - (1-\zeta_{i+1})u_{i+1,y} - \zeta_{i+1}v_{i+1,y} & \leq & \ell_i, \quad \forall\, i \in [0,k] \\
 & 0 & \leq & (1-\zeta_i)u_{i,z} + \zeta_i v_{i,z} - (1-\zeta_{i+1})u_{i+1,z} - \zeta_{i+1}v_{i+1,z} & \leq & \ell_i, \quad \forall\, i \in [0,k] \\
 & 0 & \leq & \zeta_i \ \leq \ 1, \quad \forall\, i \in [0,k+1] & &
\end{array}$$

The inequalities $(1-\zeta_i)u_{i,z} + \zeta_i v_{i,z} - (1-\zeta_{i+1})u_{i+1,z} - \zeta_{i+1}v_{i+1,z} \geq 0$ in the third set of constraints force the path to be descending. The first three sets of constraints make the value of ℓ_i an upper bound on the L_∞ length of the link that goes from $u_i v_i$ to $u_{i+1}v_{i+1}$. Thus, minimizing $\sum_{i=0}^{k} \ell_i$ gives the L_∞ LSDP length from s to w with edge sequence $(e_1, e_2, \ldots, e_k)$.

3.2 A combinatorial L_∞ LSDP algorithm

This section describes a faster algorithm for computing an L_∞ LSDP from s to a vertex v_α with edge sequence σ_α whenever a new node α is created in the sequence tree. The algorithm works by extending the L_∞ LSDPs computed at the parent of α. For simplicity, we will not mention the edge sequences of the L_∞ LSDPs as they are fixed by the sequence tree.

Suppose that we have already computed L_∞ LSDPs from s to all points on e_α. Our strategy is to extend these paths to L_∞ LSDPs to all points on the two edges of f_α incident to v_α. A point in a terrain edge with endpoints a and b can be written as $p_\zeta = (1-\zeta)a + \zeta b$ for some $\zeta \in [0,1]$. We represent the L_∞ LSDP lengths from s to ab by a function $F_{ab} : I_{ab} \to \mathbb{R}$. That is, $F_{ab}(\zeta)$ is the L_∞ LSDP length from s to p_ζ. The domain I_{ab} of F_{ab} is a subinterval of $[0,1]$ and it is possible that $I_{ab} \neq [0,1]$ due to the descending constraint.

Given a function h that maps some subset $D \subset \mathbb{R}^d$ to $\mathbb{R}$, the *graph* of h, denoted $\mathcal{G}(h)$, is the hypersurface $\{(x_1, x_2, \cdots, x_d, x_{d+1}) : (x_1, x_2, \cdots, x_d) \in D \wedge x_{d+1} = h(x_1, x_2, \cdots, x_d)\}$.

Refer to Figure 4. Let u and w be the other two vertices of f_α. Let e be the edge with endpoints u and v_α. We parametrize e_α and e by $\lambda, \tau \in [0,1]$ such that $p_\lambda = (1-\lambda)u + \lambda w$ denotes the parametrized point on e_α and $p_\tau = (1-\tau)u + \tau v_\alpha$ denotes the parametrized point on e. The LSDP to a point $p_\lambda \in e_\alpha$ is extended to a point $p_\tau \in e$ by appending the segment $p_\lambda p_\tau$. The L_∞ length of $p_\lambda p_\tau$ is $\|\tau(v_\alpha - u) - \lambda(w-u)\|_\infty$. We require $p_\lambda p_\tau$ to be descending, so there is the constraint that $(v_{\alpha,z} - u_z)\tau \leq (w_z - u_z)\lambda$.

Consider a three-dimensional space in which the horizontal plane is the $\tau\lambda$-plane. We label the vertical axis by ϱ which takes on real values. Define seven planes as follows.

$$\begin{aligned}
H_1 &: \quad \varrho = (v_{\alpha,x} - u_x)\tau - (w_x - u_x)\lambda \\
H_2 &: \quad \varrho = -(v_{\alpha,x} - u_x)\tau + (w_x - u_x)\lambda \\
H_3 &: \quad \varrho = (v_{\alpha,y} - u_y)\tau - (w_y - u_y)\lambda \\
H_4 &: \quad \varrho = -(v_{\alpha,y} - u_y)\tau + (w_y - u_y)\lambda \\
H_5 &: \quad \varrho = (v_{\alpha,z} - u_z)\tau - (w_z - u_z)\lambda \\
H_6 &: \quad \varrho = -(v_{\alpha,z} - u_z)\tau + (w_z - u_z)\lambda \\
H_7 &: \quad (w_z - u_z)\lambda - (v_{\alpha,z} - u_z)\tau = 0
\end{aligned}$$

The graph of $|(v_{\alpha,x} - u_x)\tau - (w_x - u_x)\lambda|$ is the upper envelope of H_1 and H_2. Similarly, the graphs of $|(v_{\alpha,y} - u_y)\tau - (w_y - u_y)\lambda|$ and $|(v_{\alpha,z} - u_z)\tau - (w_z - u_z)\lambda|$ are the upper envelops of H_3 and H_4, and H_5 and H_6, respectively. Therefore, the graph of $\|p_\lambda p_\tau\|_\infty$ is the upper envelope of the six planes H_i for $i \in [1,6]$.

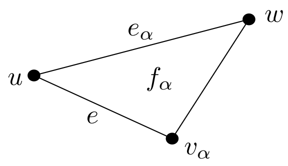
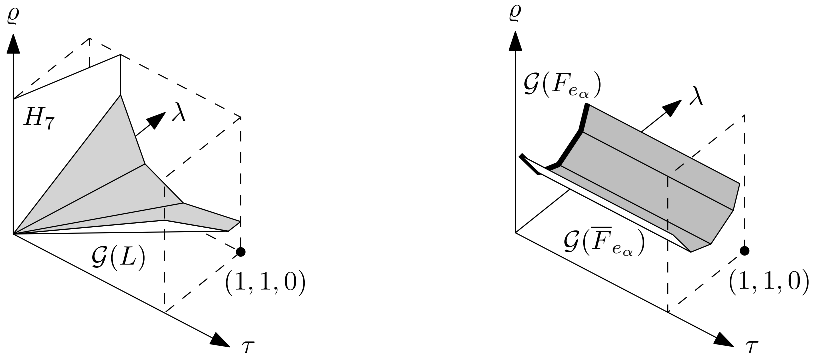


Figure 4: The configuration of f_α.



(a) (b)

Figure 5: In (a), the subset of $\mathcal{G}(L)$ in H_7^+ contains the set of (τ, λ)'s for which $p_\lambda p_\tau$ is descending. In (b), $\mathcal{G}(F_{e_\alpha})$ is shown bold in the $\lambda\varrho$-plane, and each segment on it is extended into a strip. The resulting surface is $\mathcal{G}(\overline{F}_{e_\alpha})$.

Define the function $L(\tau, \lambda)$ to be the L_∞ length of $p_\lambda p_\tau$. The halfspace $H_7^+ : (w_z - u_z)\lambda \geq (v_{\alpha,z} - u_z)\tau$ bounded by H_7 contains the set of (τ, λ)'s for which $p_\lambda p_\tau$ is descending. Therefore, $\mathcal{G}(L) \cap H_7^+$ is the graph of $\|p_\lambda p_\tau\|_\infty$ for which $p_\lambda p_\tau$ is descending. See Figure 5(a) for an illustration. $\mathcal{G}(L) \cap H_7^+$ is convex and it consists of convex polygonal faces that share the origin as a common vertex because H_i contains the origin for $i \in [1, 7]$.

We first show that for every terrain edge e', $F_{e'}$ is a convex piecewise linear function.

Lemma 3.3 *For every terrain edge e', $F_{e'}$ is a convex piecewise linear function.*

Proof. We prove the lemma by induction on the construction of the sequence tree. In the initial sequence tree, the edges opposite s correspond to the children of the root. For every such edge e', if a point q moves linearly along e', then $s_x - q_x$, $s_y - q_y$ and $s_z - q_z$ change linearly. It follows that $F_{e'}$, which gives $\|sq\|_\infty$, is a convex piecewise linear function of constant complexity.

Refer to Figure 4 and consider the induction step in which we want to compute F_e from F_{e_α}. By the induction assumption, F_{e_α} is convex and piecewise linear. Refer to Figure 5(b). We sweep each segment ℓ in $\mathcal{G}(F_{e_\alpha})$ in a direction parallel to the τ-axis to obtain a strip $[0, 1] \times \ell$. This gives a convex surface, which is the graph of a function $\overline{F}_{e_\alpha}$ such that $\overline{F}_{e_\alpha}(\tau, \lambda) = F_{e_\alpha}(\lambda)$ for every $\tau \in [0, 1]$ and every $\lambda \in I_{e_\alpha}$. The domain of $\overline{F}_{e_\alpha} + L$ is the convex set $[0, 1] \times I_{e_\alpha}$. Therefore, $\overline{F}_{e_\alpha} + L$ is convex and piecewise linear because it is the addition of two convex piecewise linear functions.

Let D be the convex set $([0, 1] \times I_{e_\alpha}) \cap H_7^+$, which consists of all (τ, λ)'s for which $p_\lambda p_\tau$ is descending. Thus, the domain of F_e is the interval $I_e = \{\tau \in [0, 1] : (\tau, \lambda) \in D \text{ for some } \lambda\}$. F_e is convex and piecewise linear because $F_e(\tau) = \min_{\lambda:(\tau,\lambda)\in D} \left\{\overline{F}_{e_\alpha}(\tau, \lambda) + L(\tau, \lambda)\right\}$. ◻

Next, we show that F_e can be constructed quickly when F_{e_α} is given.

Lemma 3.4 *If $\mathcal{G}(F_{e_\alpha})$ has k segments, then F_e can be constructed from F_{e_α} in $O(k \log k)$ time.*

Proof. Let $\overline{F}_{e_\alpha}$ be defined as in the proof of Lemma 3.3. Let G denote the subset of $\mathcal{G}(\overline{F}_{e_\alpha} + L)$ clipped within H_7 and the unbounded box $[0,1] \times I_{e_\alpha} \times [0,\infty)$. The projection of G onto the $\tau\lambda$- plane can be computed in $O(k)$ time, because it is the overlay of the projection of $\mathcal{G}(L)$ onto the $\tau\lambda$-plane, which has $O(1)$ size, and the projection of $\mathcal{G}(F_{e_\alpha})$ onto the $\tau\lambda$-plane, which has k parallel segments. The function value of $\overline{F}_{e_\alpha} + L$ at every vertex can be evaluated in $O(1)$ time. Therefore, G can be computed in $O(k)$ time.

We obtain the domain I_e of F_e by traversing G in $O(k)$ time. Let $[a, a']$ denote I_e. We extract in $O(k)$ time the cross-section of G at $\tau = a$, which is a single vertex. $F_e(a)$ is the ϱ-value of this vertex. Then, we sweep a plane orthogonal to the τ-axis from a to a', stopping at the τ-value of every vertex of G. Between two consecutive stops b and b', the sweep plane moves over a strip of G. We track the lowest segment on this strip, or in the degenerate case the triangle or trapezoid on this strip that is perpendicular to the $\tau\varrho$-plane. The projection of this segment, triangle, or trapezoid onto the $\tau\varrho$-plane is the portion of $\mathcal{G}(F_e)$ over the τ-range $[b, b']$. The running time of the plane sweep is $O(k \log k)$. ◻

It may be possible to speed up the plane sweep algorithm in the proof of Lemma 3.4 by exploring more properties of the graph of $\overline{F}_{e_\alpha} + L$, but the time complexity in Lemma 3.4 is not the bottleneck of our approximate SDP algorithm.

The critical issue is how the complexity of F_e depends on the complexity k of F_{e_α} because it affects the time to construct the children of v_α in the sequence tree. The number of segments in $\mathcal{G}(F_e)$ is no more than the number of faces on $\mathcal{G}(\overline{F}_{e_\alpha} + L)$, which is at most $6k$ as it is the upper envelope of $6k$ planes clipped by another 5 planes as explained in the proof of Lemma 3.4. This bound of $6k$ is not useful though because the bound becomes $2^{O(|\sigma_\alpha|)}$ as the construction proceeds down the sequence tree. The rest of this section is devoted to showing that the complexity of F_e is at most the complexity of F_{e_α} plus a fixed constant, which implies that the complexity of F_e is $O(|\sigma_\alpha|)$.

Refer to Figure 4. For every $\tau \in I_e$, define $\lambda_\tau = \min\{\lambda : F_e(\tau) = F_{e_\alpha}(\lambda) + L(\tau, \lambda)\}$. The next lemma shows that λ_τ increases monotonically in τ.

Lemma 3.5 *For every pair of values τ_0 and τ_1 in I_e, if $\tau_0 < \tau_1$, then $\lambda_{\tau_0} \le \lambda_{\tau_1}$.*

Proof. For the sake of contradiction, suppose that there exist $\tau, \tau' \in I_e$ such that $\tau > \tau'$ but $\lambda_\tau < \lambda_{\tau'}$. In this case, the segment connecting $p_{\lambda_\tau} \in e_\alpha$ and $p_\tau \in e$ must cross the segment connecting $p_{\lambda_{\tau'}} \in e_\alpha$ and $p_{\tau'} \in e$. In a convex quadrilateral, the sum of the L_∞ lengths of its two diagonals is at least the sum of the L_∞ lengths of any two opposite sides. Therefore,

$$\begin{aligned}
& F_{e_\alpha}(\lambda_\tau) + L(\tau', \lambda_\tau) + F_{e_\alpha}(\lambda_{\tau'}) + L(\tau, \lambda_{\tau'}) \\
\le\; & F_{e_\alpha}(\lambda_\tau) + L(\tau, \lambda_\tau) + F_{e_\alpha}(\lambda_{\tau'}) + L(\tau', \lambda_{\tau'}) \\
=\; & F_e(\tau) + F_e(\tau'). \qquad (1)
\end{aligned}$$

We claim that the segments $p_{\lambda_\tau} p_{\tau'}$ and $p_{\lambda_{\tau'}} p_\tau$ are descending. Since $p_{\lambda_\tau} p_\tau$ and $p_{\lambda_{\tau'}} p_{\tau'}$ are diagonals of the convex quadrilateral $p_{\lambda_\tau} p_{\lambda_{\tau'}} p_\tau p_{\tau'}$, they cross at some point z. The segments $p_{\lambda_\tau} z$, $p_{\lambda_{\tau'}} z$, $z p_\tau$ and $z p_{\tau'}$ are descending because $p_{\lambda_\tau} p_\tau$ and $p_{\lambda_{\tau'}} p_{\tau'}$ are descending. Therefore, $p_{\lambda_\tau} z p_{\tau'}$ and $p_{\lambda_{\tau'}} z p_\tau$ are two descending paths, implying that the segments $p_{\lambda_\tau} p_{\tau'}$ and $p_{\lambda_{\tau'}} p_\tau$ are descending.

It follows from our claim and the definition of F_e that $F_{e_\alpha}(\lambda_\tau) + L(\tau', \lambda_\tau) \ge F_e(\tau')$ and $F_{e_\alpha}(\lambda_{\tau'}) + L(\tau, \lambda_{\tau'}) \ge F_e(\tau)$. By (1), we conclude that $F_{e_\alpha}(\lambda_{\tau'}) + L(\tau, \lambda_{\tau'}) = F_e(\tau)$ and $F_{e_\alpha}(\lambda_\tau) + L(\tau', \lambda_\tau) = F_e(\tau')$. But the equality $F_{e_\alpha}(\lambda_{\tau'}) + L(\tau, \lambda_{\tau'}) = F_e(\tau)$ contradicts the definition of λ_τ as $\lambda_{\tau'} < \lambda_\tau$ by assumption. ◻

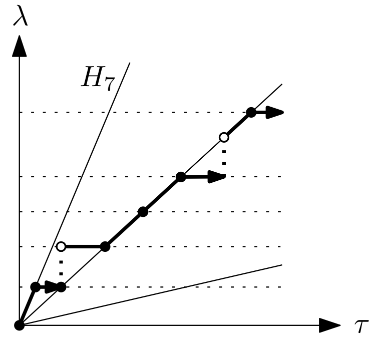


Figure 6: The λ-values of the dotted horizontal lines are the breakpoints of F_{e_α}. The non-bold solid lines are the edges of $\mathcal{L}$. The locus η is shown in bold and it is monotone with respect to both the τ-axis and the λ-axis. We divide η into maximal segments so that no segment crosses a dotted horizontal line or an edge of $\mathcal{L}$.

The next lemma characterizes the value λ_τ for every $\tau \in I_e$. We call a value $\lambda_0 \in I_{e_\alpha}$ a *breakpoint* of F_{e_α} if λ_0 is the λ-value of a vertex of $\mathcal{G}(F_{e_\alpha})$, including the endpoints of I_{e_α}. The breakpoints of F_e are similarly defined.

Lemma 3.6 *For every value τ_0 in I_e, at least one of the following three possibilities hold:*

(i) *λ_{τ_0} is a breakpoint of F_{e_α},*

(ii) *$(\tau_0, \lambda_{\tau_0})$ are the first two coordinates of some point on an edge of $\mathcal{G}(L)$, or*

(iii) *τ_0 and λ_{τ_0} satisfy the equation of the plane H_7.*

Proof. Suppose that λ_{τ_0} is not a breakpoint of F_{e_α}. Then, λ_{τ_0} lies between two successive breakpoints $\lambda_i < \lambda_{i+1}$. Let $\overline{F}_{e_\alpha}$ be defined as in the proof of Lemma 3.3. Let $\varrho = a_i\lambda + b_i$ be the support plane of the strip in $\mathcal{G}(\overline{F}_{e_\alpha})$ that lies above $[\lambda_i, \lambda_{i+1}]$. Add $\varrho = a_i\lambda + b_i$ to the equations of H_j for $j \in [1, 6]$ to obtain the equations of six new planes H'_j for $j \in [1, 6]$. Let $\mathcal{U}$ be the subset of the upper envelope of these H'_j's that lies above $I_e \times [\lambda_i, \lambda_{i+1}]$. Let $\mathcal{U}'$ be the subset of $\mathcal{U}$ that lies in H_7^+. Observe that $\mathcal{U}' \subseteq \mathcal{G}(\overline{F}_{e_\alpha} + L)$.

By our definition of λ_{τ_0}, $(\tau_0, \lambda_{\tau_0})$ must be the first two coordinates of some point on an edge of $\mathcal{U}'$. If τ_0 and λ_{τ_0} do not satisfy the equation of H_7, then $(\tau_0, \lambda_{\tau_0})$ must be the first two coordinates of some point on an edge of $\mathcal{U}$. Although $\mathcal{U}$ is geometrically different from $\mathcal{G}(L)$, their projections onto the $\tau\lambda$-plane are identical because each H'_j is obtained by adding the same linear equation $\varrho = a_i\lambda + b_i$ to H_j. Thus, $(\tau_0, \lambda_{\tau_0})$ must be the first two coordinates of some point on an edge of $\mathcal{G}(L)$. ◻

We use Lemmas 3.5 and 3.6 to show that the complexity of F_e is equal to the complexity of F_{e_α} plus a constant.

Lemma 3.7 *If $\mathcal{G}(F_{e_\alpha})$ consists of k segments, there are at most $k + 3k' + 1$ segments in $\mathcal{G}(F_e)$, where k' is the number of faces in $\mathcal{G}(L)$.*

Proof. Consider increasing the value of τ from the smaller endpoint of I_e to the larger endpoint. As τ increases, the point (τ, λ_τ) traces a locus η in the $\tau\lambda$-plane. Project the subset of $\mathcal{G}(L)$ that lies in H_7^+ onto the $\tau\lambda$-plane. Let $\mathcal{L}$ denote the projected image.

By Lemmas 3.5 and 3.6, η is monotone with respect to both the τ-axis and the λ-axis. Moreover, for each segment in η, either it lies on an edge of $\mathcal{L}$, or it is parallel to the τ-axis

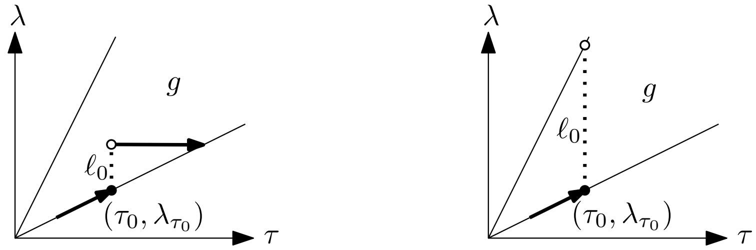


Figure 7: The black dot is the point $(\tau_0, \lambda_{\tau_0})$. On the left, ℓ_0 stops in the interior of g and then turns right in the direction of the positive τ-axis. On the right, ℓ_0 extends all the way across g.

and has the λ-value of a breakpoint of F_{e_α}. Notice that η could be disconnected. While tracing η, we may jump from a point $(\tau_0, \lambda_{\tau_0})$ in the direction of positive λ-axis to a point (τ_0, λ'). It happens when $F_{e_\alpha}(\lambda) + L(\tau_0, \lambda) = F_{e_\alpha}(\lambda_{\tau_0}) + L(\tau_0, \lambda_{\tau_0})$ for every $\lambda \in [\lambda_{\tau_0}, \lambda']$, because the definition of λ_τ prefers λ_{τ_0} to all such larger λ. In summary, η follows an edge of $\mathcal{L}$, or moves in the direction of positive τ-axis, or makes a jump in the direction of positive λ-axis. Figure 6 shows an example. We break η into segments, each being a maximal connected subset that lies on an edge of $\mathcal{L}$ and between two successive breakpoints of F_{e_α}, or that is parallel to the direction of the positive τ-axis and lies inside a face of $\mathcal{L}$. Each segment of η projects to a subset of I_e that lies between two successive breakpoints of F_e. Therefore, the number of segments in η is an upper bound on the complexity of $\mathcal{G}(F_e)$.

We first bound the number of jumps η makes. Suppose that η jumps from a point $(\tau_0, \lambda_{\tau_0})$ in the direction of the positive λ-axis in a face g of $\mathcal{L}$. Let ℓ_0 be the segment in g covered by this jump, i.e., $(\tau_0, \lambda_{\tau_0})$ is the lower endpoint of ℓ_0, and ℓ_0 is parallel to the λ-axis. Refer to Figure 7. As explained earlier, $F_{e_\alpha}(\lambda) + L(\tau_0, \lambda)$ remains the same for all $(\tau_0, \lambda) \in \ell_0$. In other words, F_{e_α} must cancel the component of $L(\tau_0, \lambda)$ that involves λ. The face g corresponds to some plane H_j, so within g the value of L is determined by the equation of H_j. The term involving λ in the equation of H_j is fixed. This implies that ℓ_0 can continue to extend in g, while keeping $F_{e_\alpha}(\lambda) + L(\tau_0, \lambda)$ independent of λ, until the upper endpoint of ℓ_0 reaches a λ-value that is a breakpoint of F_{e_α}, or ℓ_0 hits an edge of g. In the first case, ℓ_0 cannot extend any further because we would switch to the next segment in $\mathcal{G}(F_{e_\alpha})$, which must have a slope different from the current segment in $\mathcal{G}(F_{e_\alpha})$ as F_{e_α} is convex. Thus, the equation of H_j can no longer cancel $F_{e_\alpha}(\lambda)$ for any λ-value larger than the upper endpoint of ℓ_0. On the other hand, there is no edge of $\mathcal{L}$ to follow in the interior of g. Therefore, if the upper endpoint of ℓ_0 is in interior of g, then by Lemma 3.6, η will continue from the upper endpoint of ℓ_0 in the direction of positive τ-axis until η reaches an edge of g. From this point onward, η cannot make any further jump into g in the direction of positive λ-axis because F_{e_α} is convex and therefore a different segment of F_{e_α} cannot cancel the term in the equation of H_j that involves λ. For the second case that ℓ_0 extends all the way across g, η will have to traverse a segment in the direction of the positive τ-axis to enter g again. Such a segment corresponds to a breakpoint of F_{e_α}. Therefore, by the same argument as before, η cannot jump in g again after encountering a breakpoint of F_{e_α}. We conclude that η makes at most one jump in the direction of the positive λ-axis in each face of $\mathcal{L}$.

Let k be the number of segments in $\mathcal{G}(F_{e_\alpha})$. In other words, F_{e_α} has $k+1$ breakpoints. Let k' be the number of faces in $\mathcal{L}$. The above analysis shows that η makes at most k' jumps. Since η is monotone, segments in η that lie on edges of $\mathcal{L}$ have different λ-values at their upper endpoints. Such an upper endpoint is either at some τ-value that η makes a jump, or at some λ-value that is a breakpoint of F_{e_α}. So there are at most $k + k' + 1$ such upper endpoints, which implies that η has at most $k + k' + 1$ segments that lie on edges of $\mathcal{L}$.

It remains to bound the number of segments in η that are parallel to the τ-axis. Consider two such segment h_1 and h_2 in this order along η that lie in the same face g of $\mathcal{L}$. We claim that η must make a jump in g between h_1 and h_2. Suppose not. The right endpoint of h_1 must then be on the lower boundary edge of g. After h_1, η remains on or below the lower boundary edge of g until h_2 because η does not jump into g before that. But this is impossible because the left endpoint of h_2 must be above the lower boundary edge of g in order that h_2 lies in g. Therefore, if η has two segments in a face of $\mathcal{L}$ that are parallel to the τ-axis, then η must make a jump between them. Since η makes at most one jump in a face, we conclude that η has at most two segments in each face that are parallel to the τ-axis in each face. Hence, η has at most $2k'$ such segments in total that are parallel to the τ-axis. ◻

Lemma 3.8 *During the construction of the sequence tree, for each node α created, it takes $O(|\sigma_\alpha| \log |\sigma_\alpha|)$ time to compute an L_∞ LSDP from s to v_α with edge sequence σ_α.*

Proof. Lemma 3.7 implies that for each node α created, $\mathcal{G}(F_{e_\alpha})$ consists of $O(|\sigma_\alpha|)$ segments. By Lemma 3.4, the L_∞ LSDP computation at α takes $O(|\sigma_\alpha| \log |\sigma_\alpha|)$ time. ◻

Plugging the result of Lemma 3.8 into Lemma 3.2 gives Theorem 3.1.

4 Approximation algorithm for SDP

Our approximate SDP algorithm consists of three steps. First, we apply Theorem 3.1 to compute the L_∞ SDP length opt_∞ from s to t. Recall that opt is the SDP length from s to t and $\mathsf{opt}_\infty \leq \mathsf{opt} \leq \sqrt{3}\,\mathsf{opt}_\infty$. Second, we clip the terrain $\mathcal{T}$ within the box centered at s and of side length $4\sqrt{3}\mathsf{opt}_\infty$. Triangulate all non-triangular faces produced by the clipping. Let $\mathcal{T}^*$ denote the clipped terrain. For all $\varepsilon \in (0, 1)$, every $(1+\varepsilon)$-approximate SDP has length at most $2\mathsf{opt} \leq 2\sqrt{3}\mathsf{opt}_\infty$, so it lies completely inside $\mathcal{T}^*$. It thus suffices to work with $\mathcal{T}^*$. Every edge in $\mathcal{T}^*$ has length at most $12\mathsf{opt}_\infty$. This upper bound on the edge lengths allows us to remove the dependence on the spread of $\mathcal{T}$ from the running time. Third, we define the *quasi-length* of a path in $\mathcal{T}^*$, which is based on both the path length and the path's edge sequence length. We combine the sequence tree approach with an approximate LSDP algorithm of Ahmed [2, Section 3.3.7] to compute an approximate SDP from s to t. Our innovation is to prune the sequence tree based on the quasi-lengths of the approximate LSDPs instead of their true lengths.

In the rest of this section, we explain the third step in detail and analyze the whole algorithm. For simplicity, we also use n to denote the number of vertices in $\mathcal{T}^*$, which is asymptotically no more than the number of vertices in $\mathcal{T}$.

4.1 Computing an approximate LSDP

Given a destination v, an edge sequence σ, and an error parameter $\mu \in (0, 1)$, Ahmed proposed an algorithm [2, Section 3.3.7] to compute in $O(|\sigma|^2 \log(L/\mu))$ time an approximate LSDP from s to v with edge sequence σ, where L is the length of the longest edge in σ, and proved that the relative error of the returned path is $O(\mu|\sigma|/h)$, where h is the smallest distance between a vertex and a non-incident edge. Ahmed's algorithm is based on the characterization of the bend angles of an LSDP [5], which allows the LSDP to be traced in time linear in the number of links once the direction of the first link is fixed. Therefore, one can guess the direction of the first link to trace an LSDP, and then binary search to find a good enough initial direction. We describe Ahmed's algorithm below for the sake of completeness. We modify the original analysis [2, Section 3.3.7] to change the error bound from a relative error to an additive error of $|\sigma|\mu$. The additive error bound will be useful in the analysis of our approximate SDP algorithm.

Suppose that $\sigma = (e_1, \ldots, e_k)$. Before the binary search starts, the algorithm first constructs two "boundary paths" Q and R that sandwich the approximate LSDP to be computed. Let $q'_0 = s$, let $r'_{k+1} = v$, and set q'_i to be the highest point in e_i that is not higher than q'_{i-1}. For $i \in [1, k]$, let r'_i be the lowest point in e_i that is not lower than r'_{i+1}. If q'_i or r'_i for some i does not exist, then the algorithm aborts, since the LSDP from s to v with edge sequence σ does not exist. Afterwards, for $i \in [1, k]$, let q_i and r_i be the left and right endpoints of $q'_i r'_i$ with respect to the traversal of the sequence $(e_1, e_2, \ldots, e_k)$. The paths Q and R are $(q_0 = s, q_1, q_2, \ldots, q_k, q_{k+1} = v)$ and $(r_0 = s, r_1, r_2, \ldots, r_k, r_{k+1} = v)$, respectively. Intuitively, the LSDP cannot "cross" Q and "bend to the left", and it cannot "cross" R and "bend to the right".

Let $c_0 = s$. The goal is to find an interval $a_i b_i \subseteq q_i r_i \subseteq e_i$ and a point $c_i \in a_i b_i$ for $i \in [1, k]$ that fulfills the following conditions:

- Both a_i and b_i can be reached by descending segments from c_{i-1}.
- $\|a_i b_i\| \leq \mu/2$ and c_i is equal to a_i or b_i, whichever is higher.
- The LSDP from c_{i-1} to v with edge sequence $(e_i, \cdots, e_k)$ exists and it crosses $a_i b_i$.

The computation of $a_i b_i$ and c_i proceeds in stages from $i = 1$ to k. The ith stage works as follows. Initialize a_i and b_i to be q_i and r_i, respectively. If a_i is higher than c_{i-1}, then set a_i to be the point in $q_i r_i$ that is at the same height as c_{i-1}. Similarly, if b_i is higher than c_{i-1}, then set b_i to be the point in $q_i r_i$ that is at the same height as c_{i-1}. Let c_i be the midpoint of $a_i b_i$. Trace an LSDP from c_{i-1} along the initial direction $c_{i-1} c_i$ until the path hits Q or R. If Q is hit, set a_i to be c_i; otherwise, R is hit and set b_i to be c_i. Set c_i to be the midpoint of $a_i b_i$. Repeat the tracing of an LSDP along $c_{i-1} c_i$ and the update of a_i, b_i and c_i until $\|a_i b_i\| \leq \mu/2$. After the binary search terminates, set c_i to be the higher endpoint of the final interval $a_i b_i$ and proceed to the next stage. The LSDP tracing maintains the property that the LSDP from c_{i-1} to v exists and passes through $a_i b_i$. After completing all k stages, return the path $P = (s = c_0, c_1, c_2, \ldots, c_k, v)$.

Lemma 4.1 *Given a source s, a vertex v, an edge sequence σ, and any $\mu \in (0, 1)$, an approximate shortest descending path from s to v with edge sequence σ and additive error at most $|\sigma|\mu$ can be computed in $O(|\sigma|^2 \log(L/\mu))$ time, where L is the length of the longest edge in σ.*

Proof. It was proved in [2, Section 3.3.7] that the algorithm runs in $O(|\sigma|^2 \log(L/\mu))$ time and returns a descending path D from s to v with edge sequence σ. We analyze the additive error of D. For $i \in [0, k]$, let P_i be the LSDP from c_i to v with edge sequence $(e_{i+1}, \ldots, e_k)$. By the invariants of the algorithm, we can assume that P_i crosses $a_{i+1} b_{i+1}$ for $i \in [0, k-1]$. Note that $P_k = c_k v = D[c_k, v]$. We show by backward induction from $i = k$ down to 0 that $\|D[c_i, v]\| \leq \|P_i\| + (k - i)\mu$.

The base case of $i = k$ is trivial as $D[c_k, v] = P_k$. Let c be the crossing between P_i and $a_{i+1} b_{i+1}$. So $\|c_{i+1} c\| \leq \mu/2$. Since c_{i+1} is the higher of a_{i+1} and b_{i+1}, the concatenation $c_{i+1} c \cdot P_i[c, v]$ is a descending path from c_{i+1} to v with edge sequence $(e_{i+2}, \ldots, e_k)$. Thus, $c_{i+1} c \cdot P_i[c, v]$ is at least as long as P_{i+1}. On the other hand, $\|P_{i+1}\| \geq \|D[c_{i+1}, v]\| - (k - i - 1)\mu$ by induction assumption. Therefore,

$$\|P_i[c, v]\| + \|c_{i+1} c\| \geq \|P_{i+1}\| \geq \|D[c_{i+1}, v]\| - (k - i - 1)\mu.$$

Then,

$$\begin{aligned}
\|P_i\| &= \|P_i[c,v]\| + \|c_ic\| \\
&\geq \|D[c_{i+1},v]\| - (k-i-1)\mu + \|c_ic\| - \|c_{i+1}c\| \\
&\geq \|D[c_{i+1},v]\| - (k-i-1)\mu + (\|c_ic_{i+1}\| - \|c_{i+1}c\|) - \|c_{i+1}c\| \\
&= \|D[c_i,v]\| - (k-i-1)\mu - 2\,\|c_{i+1}c\| \\
&\geq \|D[c_i,v]\| - (k-i)\mu.
\end{aligned}$$

This completes the induction. When $i = 0$, we get $\|D\| = \|D[c_0, v]\| \leq \|P_0\| + k\mu$. □

Due to the one-corner one-split property, there exists a constant $\kappa \geq 1$ such that the construction of the sequence tree creates fewer than κn^2 tree nodes. Set $\mu = \varepsilon \mathsf{opt}_\infty/(\kappa n^2 m(m+1))$, where n is the number of vertices and m is the number of faces in $\mathcal{T}^*$. The fact that every edge of $\mathcal{T}^*$ has length at most $12\mathsf{opt}_\infty$ immediately yields the following result.

Corollary 4.1 *Let $\mu = \varepsilon \mathsf{opt}_\infty/(\kappa n^2 m(m+1))$. Given a destination v and an edge sequence σ in $\mathcal{T}^*$, we can compute in $O(|\sigma|^2 \log(n/\varepsilon))$ time an approximate* LSDP *in $\mathcal{T}^*$ from s to v with edge sequence σ and additive error at most $|\sigma|\mu$.*

4.2 Quasi-length and quasi-dominance

In the construction of the sequence tree, whenever a new tree node α is created, we apply Corollary 4.1 to compute an approximate LSDP from s to v_α with edge sequence σ_α. We denote this path by P_α and store it at α. The remaining task is to define a tree pruning rule to enforce the one-corner one-split property.

Let $\mu = \varepsilon \mathsf{opt}_\infty/(\kappa n^2 m(m+1))$ as defined in Corollary 4.1. Define $\tilde{\mu} = \kappa n^2 m\mu = \varepsilon \mathsf{opt}_\infty/(m+1)$. For every path Q in $\mathcal{T}^*$, the *quasi-length* of Q is equal to $\|Q\| + |\mathrm{seq}(Q)| \cdot \tilde{\mu}$. We define a notion of dominance among the tree nodes in order to impose the one-corner one-split property. Let α and β be two tree nodes corresponding to the same face corner (f_α, v_α). Therefore, P_α and P_β are approximate LSDPs from s to $v_\alpha = v_\beta$ with edge sequences σ_α and σ_β, respectively. Then, α *quasi-dominates* β if the quasi-length of P_α is less than or equal to the quasi-length of P_β. (Break ties arbitrarily.) Assume that α quasi-dominates β. We further decide whether α quasi-dominates β *on the left or right.* The left/right quasi-dominance is determined by the same test used in the definition of left/right dominance for the L_∞ case in Section 3. Refer to Figure 2 for the possible configurations of P_α and P_β.

The next result is the analog of Lemma 3.1 in the L_∞ case.

Lemma 4.2 *Let α and β be two tree nodes that correspond to the same face corner (f_α, v_α) such that α quasi-dominates β on the right (resp. left). Let e be the edge that follows e_α immediately in anticlockwise (resp. clockwise) order around the boundary of f_α.*

(i) *α is not a descendant of β.*

(ii) *For every point $r \in e$ and every* LSDP *Q_β with edge sequence σ_β from s to r, the* LSDP *Q_α with edge sequence σ_α from s to r satisfies $\|Q_\alpha\| \leq \|Q_\beta\| + (|\sigma_\beta| - |\sigma_\alpha|)\tilde{\mu} + |\sigma_\beta|\mu$.*

Proof. Refer to Figure 3 for an illustration of the configuration.

Consider (i). Suppose for the sake of contradiction that α is a descendant of β. So σ_β is a proper prefix of σ_α. P_α traverses the same edges and faces as P_β until f_β. Afterwards, P_β stops at v_β, and P_α leaves f_β through some point p on an edge of f_β other than e_β. Shortcut

P_α by connecting p to v_β directly and removing the rest of P_α. This gives a path Q such that $\mathrm{seq}(Q) = \sigma_\beta$ and $\|Q\| < \|P_\alpha\|$. Thus, $\|Q\| + |\sigma_\beta|\tilde{\mu} < \|P_\alpha\| + (|\sigma_\alpha| - 1)\tilde{\mu}$ because $|\sigma_\beta| < |\sigma_\alpha|$. Since α quasi-dominates β, we get $\|Q\| + |\sigma_\beta|\tilde{\mu} < \|P_\beta\| + (|\sigma_\beta| - 1)\tilde{\mu}$. Therefore, $\|Q\| < \|P_\beta\| - \tilde{\mu} \leq \|P_\beta\| - |\sigma_\beta|\mu$ because $\tilde{\mu} \geq m\mu \geq |\sigma_\beta|\mu$. But the additive error of P_β is at most $|\sigma_\beta|\mu$ by Corollary 4.1, implying that Q is shorter than the LSDP from s to v_β with edge sequence σ_β, a contradiction.

Consider (ii). Without loss of generality, assume that α quasi-dominates β on the right. The same analysis in the proof of Lemma 3.1(ii) shows that P_α intersects Q_β at a point p such that $P_\alpha[p, v_\alpha]$ and $Q_\beta[p, r]$ have the same edge sequence as illustrated in Figure 3. Therefore, $P_\alpha[s, p] \cdot Q_\beta[p, r]$ is a descending path from s to r with edge sequence v_α. Since Q_α is the LSDP from s to r with edge sequence v_α by assumption, we obtain

$$\begin{aligned}
\|Q_\alpha\| &\leq \|P_\alpha[s,p]\| + \|Q_\beta[p,r]\| \\
&= \|P_\alpha\| + \|Q_\beta\| - \|Q_\beta[s,p] \cdot P_\alpha[p,v_\alpha]\| \\
&\leq \|P_\beta\| + (|\sigma_\beta| - |\sigma_\alpha|)\tilde{\mu} + \|Q_\beta\| - \|Q_\beta[s,p] \cdot P_\alpha[p,v_\alpha]\| \qquad (2)
\end{aligned}$$

The last inequality follows from the quasi-dominance of α over β. Note that $Q_\beta[s,p] \cdot P_\alpha[p, v_\alpha]$ is a descending path from s to v_α with edge sequence σ_β. Therefore, if it is shorter than P_β, Corollary 4.1 implies that it is shorter by no more than $|\sigma_\beta|\mu$. That is, $\|P_\beta\| - \|Q_\beta[s,p] \cdot P_\alpha[p, v_\alpha]\| \leq |\sigma_\beta|\mu$. Plugging this inequality into (2) gives (ii). ◻

4.3 Algorithm and analysis

Our approximate SDP algorithm also grows from the leaves at the bottommost level of the sequence tree in rounds until the number of tree levels equals the number of faces in $\mathcal{T}^*$. There are two differences though. First, dominance among the tree nodes is replaced by quasi-dominance. Thus, every dominance check is replaced by a quasi-dominance check. Second, after creating a new leaf α, we use Corollary 4.1 to compute an approximate LSDP P_α from s to v_α with edge sequence σ_α. The one-corner one-split property is enforced by pruning the sequence tree as described in the L_∞ SDP algorithm in Section 3. After the tree stops growing, we return the minimum $\|P_\alpha\|$ among all tree nodes α such that $v_\alpha = t$. This is the approximate SDP length from s to t. The next result shows the correctness of the algorithm.

Lemma 4.3 *For any $\varepsilon \in (0,1)$, our algorithm returns a $(1+\varepsilon)$-approximate* SDP *from s to t.*

Proof. Recall that $\mu = \varepsilon\mathsf{opt}_\infty/(\kappa n^2 m(m+1))$, $\tilde{\mu} = \kappa n^2 m\mu = \varepsilon\mathsf{opt}_\infty/(m+1)$, and m denotes the number of faces in $\mathcal{T}^*$ (which bounds the number of levels in the sequence tree). Let opt be the SDP length from s to t. Let P_0 be an SDP from s to t. So $\|P_0\| = \mathsf{opt}$ and $|\mathrm{seq}(P_0)| < m$ as P_0 does not visit any face more than once.

Suppose that the final sequence tree contains a node γ_0 such that $v_{\gamma_0} = t$ and $\sigma_{\gamma_0} = \mathrm{seq}(P_0)$. Corollary 4.1 gives an approximate LSDP P_{γ_0} such that $\|P_{\gamma_0}\| \leq \|P_0\| + |\sigma_{\gamma_0}|\mu < \mathsf{opt} + m\mu < \mathsf{opt} + \varepsilon\mathsf{opt}_\infty \leq (1+\varepsilon)\mathsf{opt}$. Our algorithm returns a path of length at most $\|P_{\gamma_0}\|$.

Suppose that there is no such node γ_0 in the final sequence tree. There must exist two nodes β_0 and α_1 in some intermediate sequence tree such that σ_{β_0} is a prefix of σ_{γ_0}, and the child of β_0 that would be an ancestor of γ_0 is pruned due to the quasi-dominance of α_1 over β_0. By Lemma 4.2(i), α_1 remains after pruning. We will show that there is an almost equally good descending path from s to t whose edge sequence has α_1 as a prefix.

Without loss of generality, assume that α_1 quasi-dominates β_0 on the right, so the right child of β_0 is pruned. Refer to Figure 8. Let e be the edge that immediately follows e_{β_0} in anticlockwise order around the boundary of f_{β_0}. Since the pruned right child of β_0 would be

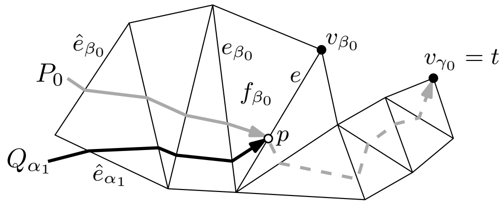


Figure 8: P_0 consists of the grey solid and grey dashed links. The grey dashed path is not represented in the final sequence tree because α_1 quasi-dominates β_0 on the right. The grey solid path has edge sequence σ_{β_0}. Lemma 4.2(ii) guarantees a bold solid path Q_{α_1} from s to p with edge sequence σ_{α_1}. Concatenating Q_{α_1} with the grey dashed path gives a descending path from s to t with edge sequence σ_{γ_1}.

an ancestor of γ_0, P_0 must cross the edge e at some point p. By Lemma 4.2(ii), there is a descending path Q_{α_1} from s to p with edge sequence σ_{α_1} such that

$$\begin{aligned} \|Q_{\alpha_1}\| &\leq \|P_0[s,p]\| + (|\sigma_{\beta_0}| - |\sigma_{\alpha_1}|)\tilde{\mu} + |\sigma_{\beta_0}|\mu \\ &< \|P_0[s,p]\| + (|\sigma_{\beta_0}| - |\sigma_{\alpha_1}|)\tilde{\mu} + m\mu. \end{aligned} \tag{3}$$

Let σ_{γ_1} be the concatenation of σ_{α_1} and the suffix of σ_{γ_0} after σ_{β_0}. So $|\sigma_{\gamma_0}| - |\sigma_{\gamma_1}| = |\sigma_{\beta_0}| - |\sigma_{\alpha_1}|$. The edge sequence of the descending path $Q_{\alpha_1} \cdot P_0[p,t]$ is σ_{γ_1}. This implies that the LSDP from s to t with edge sequence σ_{γ_1} exists. Denote it by P_1. It satisfies the following relation.

$$\begin{aligned} \|P_1\| &\leq \|Q_{\alpha_1}\| + \|P_0[p,t]\| \\ &< \|P_0\| + (|\sigma_{\beta_0}| - |\sigma_{\alpha_1}|)\tilde{\mu} + m\mu \qquad (\because (3)) \\ &= \|P_0\| + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_1}|)\tilde{\mu} + m\mu \\ &= \mathsf{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_1}|)\tilde{\mu} + m\mu. \end{aligned} \tag{4}$$

If the final sequence tree contains a tree node γ_1 such that $v_{\gamma_1} = t$ and $\sigma_{\gamma_1} = \mathrm{seq}(P_1)$, then we can proceed to bound $\|P_{\gamma_1}\|$ and show that our algorithm returns a $(1+\varepsilon)$-approximation. However, it is possible that the final sequence tree does not contain such a node γ_1. We deal with this case first. Our idea is to repeat the previous argument to define another LSDP P_2 from s to t which is only slightly longer. We prove by induction a more general claim.

Claim 4.1 *For $i \in [0, \kappa n^2 - 1]$, if the final sequence tree does not contain a tree node γ_i such that $v_{\gamma_i} = t$ and $\sigma_{\gamma_i} = \mathrm{seq}(P_i)$, then there exists a descending path P_{i+1} from s to t such that $\|P_{i+1}\| \leq \mathsf{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_{i+1}}|)\tilde{\mu} + (i+1)m\mu$.*

Proof. We have already shown the base case when $i = 0$. Assume inductively that

$$\|P_i\| \leq \mathsf{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_i}|)\tilde{\mu} + im\mu \tag{5}$$

for some $i \in [1, \kappa n^2 - 1]$. Suppose that the final sequence tree does not contain a tree node γ_i such that $v_{\gamma_i} = t$ and $\sigma_{\gamma_i} = \mathrm{seq}(P_i)$. Observe that $\|P_i\| \geq \mathsf{opt}$ and for $i < \kappa n^2$, $im\mu < \kappa n^2 m\mu = \tilde{\mu}$. Plugging these two inequalities into (5) gives $(|\sigma_{\gamma_i}| - |\sigma_{\gamma_0}|)\,\tilde{\mu} \leq im\mu < \tilde{\mu}$. It follows that $|\sigma_{\gamma_i}| \leq |\sigma_{\gamma_0}| < m$ because edge sequence lengths are integers. Since our algorithm grows the sequence tree until its height reaches m, it means that the tree node γ_i would have been generated if the sequence tree had not been pruned during its construction. Therefore, there must exist tree nodes β_i and α_{i+1} in some intermediate sequence tree such that γ_i is a descendant of β_i, α_{i+1} and β_i correspond to the same face corner, and the child of β_i that would be an ancestor of γ_i is pruned due to the quasi-dominance of α_{i+1} over β_i. Note that σ_{β_i} is a prefix of σ_{γ_i} as γ_i is a descendant of β_i. Let $\sigma_{\gamma_{i+1}}$ be the edge sequence obtained by replacing the prefix σ_{β_i} of σ_{γ_i} by $\sigma_{\alpha_{i+1}}$. We can prove that the LSDP from s to t with edge sequence $\sigma_{\gamma_{i+1}}$ exists using the same analysis as we used for showing the existence of P_1. Denote it by P_{i+1}. We can also adapt the derivation of (4) to show that

$$\|P_{i+1}\| \leq \|P_i\| + (|\sigma_{\gamma_i}| - |\sigma_{\gamma_{i+1}}|)\tilde{\mu} + m\mu.$$

Plugging (5) into the inequality above proves the claim. ◻

In the proof of Claim 4.1, the path P_i is defined after pruning some intermediate sequence tree, which implies that $P_1, P_2, \ldots$ are defined in this chronological order. Pruning can only happen after the creation of a new leaf. Therefore, there are fewer than κn^2 prunings because fewer than κn^2 tree nodes are ever created. It means that there exists $j \in [1, \kappa^2 n - 1]$ such that $P_1, \ldots, P_j$ are defined but P_{j+1} is not. As a result, the final sequence tree contains a tree node γ_j such that $v_{\gamma_j} = t$ and $\sigma_{\gamma_j} = \text{seq}(P_j)$. Moreover,

$$\begin{aligned}
\left\|P_{\gamma_j}\right\| &\leq \|P_j\| + |\sigma_{\gamma_j}|\mu && (\because \text{ Corollary 4.1}) \\
&\leq \mathsf{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_j}|)\tilde{\mu} + jm\mu + |\sigma_{\gamma_j}|\mu && (\because \text{ Claim 4.1}) \\
&< \mathsf{opt} + |\sigma_{\gamma_0}|\tilde{\mu} + \kappa n^2 m\mu && (\because \tilde{\mu} > \mu \text{ and } j < \kappa n^2) \\
&< \mathsf{opt} + (m+1)\tilde{\mu} && (\because |\sigma_{\gamma_0}| < m \text{ and } \kappa n^2 m\mu = \tilde{\mu}) \\
&= \mathsf{opt} + \varepsilon\mathsf{opt}_\infty && (\because \tilde{\mu} = \varepsilon\mathsf{opt}_\infty/(m+1)) \\
&\leq (1+\varepsilon)\mathsf{opt}.
\end{aligned}$$

The path returned by our algorithm has length at most $\left\|P_{\gamma_j}\right\|$, so it is a $(1+\varepsilon)$ approximate SDP from s to t. ◻

Theorem 4.1 *Given any $\varepsilon \in (0,1)$ and two vertices s and t in a polygonal terrain of n vertices, one can compute a $(1+\varepsilon)$-approximate shortest descending path from s to t in $O(n^4 \log(n/\varepsilon))$ time.*

Proof. We first check the existence of a descending from s to t in $O(n \log n)$ time using de Berg and van Kreveld's result [7]. By Theorem 3.1, it takes $O(n^3 \log n)$ time to compute opt_∞. By Corollary 4.1, computing the LSDP takes $O(n^2 \log(n/\varepsilon))$ time at each sequence tree node. Since $O(n^2)$ tree nodes are ever created, the total time is $O(n^4 \log(n/\varepsilon))$. ◻

5 Conclusion

We present the first $(1+\varepsilon)$-approximate SDP algorithm whose running time is polynomial in n and $\log(1/\varepsilon)$ and independent of the terrain geometry. This is achieved via computing a constant-factor lower bound on the optimal path length and introducing the quasi-length of a path, which is critical in the design of the algorithm as well as the analysis of the approximation ratio. Quasi-length is a new concept and it may be useful for other path problems on polygonal surfaces. The framework of our SDP approximation algorithm may also be applicable in solving other shortest path problems on terrains with other constraints.

References

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

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

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

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

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

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

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

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

[9] J.S.B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633?V701. Elsevier, 2000.

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

[11] 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.

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

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

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