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 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, 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

1A 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 \(O(k^3.5 \log(1/\varepsilon))\) 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 \(O\left(\frac{n^2 L}{\varepsilon h \cos \theta} \log \frac{n L}{\varepsilon h \cos \theta}\right)\) and \(O\left(\frac{n^2 L}{\varepsilon h \cos \theta} \log \frac{n L}{\varepsilon h \cos \theta}\right)\), 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 \(\theta\) 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 \((1 + cO(n/\varepsilon))\)-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 \(L_\infty\) 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 \(L_\infty\) 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].
it does not happen for an SDP. Thus, for destination of $P$
viewed as crossing sequences.
unique LSDP for a given edge sequence [5]. An SDP is a shortest LSDP over all possible edge
pair of points $P$ the interior of $xy$ order; a path entering $P$ in setting seq($T$)
is descending path. A polygonal path $P$ is a descending path if it is the shortest one
among all descending paths with the same source, destination, and edge sequence. We use
the links at their crossings with the edges of $T$. Let $T$ is oriented from its source to destination, consisting of directed
segments called links. $P$ conforms to $T$ if every link is contained in an edge or a face of $T$. We assume that all polygonal paths in this paper conform to $T$, which can be enforced by splitting
the links at their crossings with the edges of $T$. The endpoints of the links are called nodes.
We use $\|P\|$ and $\|P\|_\infty$ to denote the Euclidean length and $L_\infty$ 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 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 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 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 $u_1vu_h$ and leaving $v$ through the face $u_ivu_{i+1}$ can be viewed as crossing $vu_1, vu_2, \ldots, vu_i$ or $vu_h, vu_{h-1}, \ldots, vu_{i+1}$. We use $\|seq(P)\|$ to denote the number of edges in 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.\footnote{An LSDP is allowed to go through a vertex by our definition, while this is forbidden in [5].} 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

\section{Preliminaries}

Let $T$ 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 $T$. Without loss of generality, assume that every face of $T$ is a triangle, and that the given source $s$ and destination $t$ are vertices of $T$. For every point $p$ in $T$, 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 $T$ from above.

A polygonal path $P$ in $T$ is oriented from its source to destination, consisting of directed
segments called links. $P$ conforms to $T$ if every link is contained in an edge or a face of $T$. We assume that all polygonal paths in this paper conform to $T$, which can be enforced by splitting
the links at their crossings with the edges of $T$. The endpoints of the links are called nodes.
We use $\|P\|$ and $\|P\|_\infty$ to denote the Euclidean length and $L_\infty$ 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 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 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 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 $u_1vu_h$ and leaving $v$ through the face $u_ivu_{i+1}$ can be viewed as crossing $vu_1, vu_2, \ldots, vu_i$ or $vu_h, vu_{h-1}, \ldots, vu_{i+1}$. We use $\|seq(P)\|$ to denote the number of edges in 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.\footnote{An LSDP is allowed to go through a vertex by our definition, while this is forbidden in [5].} 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
of LSDPs from $s$ to other vertices of $T$. In Chen and Han’s version, the root represents $s$ and the other internal nodes represent intervals on edges of $T$ 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 of the sequence tree corresponds to a face corner $(f_\alpha, v_\alpha)$—the corner of the face $f_\alpha$ at its vertex $v_\alpha$—and the edge $e_\alpha$ of $f_\alpha$ opposite $v_\alpha$. Let $\deg(s)$ denote the degree of $s$. For $i \in [1, \deg(s)]$, let $e_{\alpha_i}$ denote a distinct edge opposite $s$ and let $f_{\alpha_i}$ denote the face incident to $e_{\alpha_i}$ but not $s$. The initial sequence tree consists of only the root and its children $\alpha_1, \ldots, \alpha_{\deg(s)}$. Each child node $\alpha_i$ stores $e_{\alpha_i}$ and the face corner $(f_{\alpha_i}, v_{\alpha_i})$. Figures 1(a,b) show an example.

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

The sequence tree is grown by attaching child nodes to a leaf node $\alpha$. Let $e_{\beta_1}$, $e_\alpha$, and $e_{\beta_2}$ be the edges of $f_\alpha$ in anticlockwise order around the boundary of $f_\alpha$. For $i \in \{1, 2\}$, let $f_{\beta_i}$ be the face that shares $e_{\beta_i}$ with $f_\alpha$, and let $v_{\beta_i}$ be the vertex of $f_{\beta_i}$ opposite $e_{\beta_i}$. The node $\alpha$ may acquire a left child $\beta_1$ and/or a right child $\beta_2$ 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 $T$. 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 $L_\infty$ SDP

We show how to compute the minimum $L_\infty$ length of a descending path from $s$ to $t$, denoted by $\opt_\infty$, which is a constant factor approximation of the SDP length $\opt$ from $s$ to $t$ because $\opt_\infty \leq \opt \leq \sqrt{3} \opt_\infty$. 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 $\alpha$ in the sequence tree stores the edge sequence $\sigma_\alpha$. To compute an $L_\infty$ SDP from $s$ to $t$, whenever we insert a new tree node $\alpha$ into the sequence tree, we compute an $L_\infty$ LSDP from $s$ to $v_\alpha$ with edge sequence $\sigma_\alpha$. Denote this $L_\infty$ LSDP stored at $\alpha$ by $P_\alpha$. We will present two solutions for computing $P_\alpha$, one based on linear programming (Section 3.1) and another faster combinatorial algorithm (Section 3.2). We ignore the computation of $P_\alpha$ for now.

To enforce the one-corner one-split property, we need a notion of dominance among the sequence tree nodes. Let $\alpha$ and $\beta$ be two tree nodes corresponding to the same face corner.
Figure 2: Assume that $\alpha$ dominates $\beta$. The longest common suffix of $\text{seq}(P_\alpha)$ and $\text{seq}(P_\beta)$ is $(e_1, e_2, e_3, e_\alpha)$. In (a), $\alpha$ dominates $\beta$ on the right, and Lemma 3.1(ii) says that any $L_\infty$ LSDP with edge sequence $\sigma_\beta$ from $s$ to a point $r \in e$ (shown as a dashed path) cannot be shorter than the $L_\infty$ LSDP with edge sequence $\sigma_\alpha$ from $s$ to $r$. In (b), $\alpha$ dominates $\beta$ on the left, and Lemma 3.1(ii) gives a symmetric conclusion.

$(f_\alpha, v_\alpha)$. The node $\alpha$ dominates $\beta$ if $\|P_\alpha\|_\infty < \|P_\beta\|_\infty$, or $\|P_\alpha\|_\infty = \|P_\beta\|_\infty$ and $|\sigma_\alpha| < |\sigma_\beta|$. That is, the shorter of $P_\alpha$ and $P_\beta$ is preferred, and if $P_\alpha$ and $P_\beta$ have the same $L_\infty$ 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 $\alpha$ dominates $\beta$, we need to test whether $\alpha$ dominates $\beta$ on the left or right. We prune the left child of $\beta$ if $\beta$ is dominated on the left; otherwise, we prune the right child of $\beta$. The test works as follows. The general case is that $\sigma_\alpha$ and $\sigma_\beta$ are not suffixes of each other. In this case there exist edges $\hat{e}_\alpha$ and $\hat{e}_\beta$ in $\sigma_\alpha$ and $\sigma_\beta$, respectively, that immediately precede the longest common suffix of $\sigma_\alpha$ and $\sigma_\beta$.

Let $e_1$ be the first edge in the longest common suffix of $\sigma_\alpha$ and $\sigma_\beta$. Note that $\hat{e}_\alpha$, $\hat{e}_\beta$ and $e_1$ are contained in the same face $g$. Refer to Figure 2. The node $\alpha$ dominates $\beta$ 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 $\sigma_\alpha$ is a proper suffix of $\sigma_\beta$, replace $\hat{e}_\alpha$ by $s$ in the test above. Similarly, if $\sigma_\beta$ is a proper suffix of $\sigma_\alpha$, 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 $L_\infty$ 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 $L_\infty$ length.

**Lemma 3.1** Let $\alpha$ and $\beta$ be two tree nodes that correspond to the same face corner $(f_\alpha, v_\alpha)$ such that $\alpha$ dominates $\beta$ on the right (resp. left). Let $e$ be the edge that follows $e_\alpha$ immediately in anticlockwise (resp. clockwise) order around the boundary of $f_\alpha$.

(i) $\alpha$ is not a descendant of $\beta$.

(ii) For each point $r \in e$ and each $L_\infty$ LSDP $Q_\beta$ with edge sequence $\sigma_\beta$ from $s$ to $r$, every $L_\infty$ LSDP $Q_\alpha$ with edge sequence $\sigma_\alpha$ 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 $P_\alpha$ and $P_\beta$ are $L_\infty$ LSDPs from $s$ to $v_\alpha = v_\beta$ with edge sequences $\sigma_\alpha$ and $\sigma_\beta$, respectively.

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

---

4Since $\alpha$ and $\beta$ correspond to the same face corner $(f_\alpha, v_\alpha)$, $e_\alpha$ is the last edge in both $\sigma_\alpha$ and $\sigma_\beta$. Hence, $\sigma_\alpha$ and $\sigma_\beta$ have a non-empty common suffix.
Figure 3: $P_\alpha$ and $Q_\beta$ intersect at a point $p$ such that $Q_\beta[p, p]$ and $P_\alpha[p, v_\alpha]$ have the same edge sequence.

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

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

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

Let $p$ be the intersection between $P_\alpha$ and $Q_\beta$ furthest away from $s$ along $Q_\beta$. 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 $Q_\alpha$ is an $L_\infty$ LSDP from $s$ to $r$ with edge sequence $\sigma_\alpha$ by the assumption of the lemma, we obtain $\|Q_\alpha\|_\infty \leq \|P_\alpha[s, p]\|_\infty + \|Q_\beta[p, r]\|_\infty$ because $\sigma_\alpha$ is also the edge sequence of the descending path $P_\alpha[s, p]$. 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 $\alpha$ dominates $\beta$, 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 $L_\infty$ SDP algorithm works as follows. We first construct the initial sequence tree as described earlier. It consists of the root corresponding to $s$ and $\text{deg}(s)$ children, which are the leaves of the initial tree. Then we compute $L_\infty$ LSDPs from $s$ to the face corners represented by these $\text{deg}(s)$ 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 $L_\infty$ 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 $\beta$, we create a right child (resp. left child) $\gamma$ of $\beta$ if $\beta$ is not dominated on the right (resp. left) by any other tree node. After creating a new leaf $\gamma$, we compute an $L_\infty$ LSDP from $s$ to $v_\gamma$ with edge sequence $\sigma_\gamma$. If no tree node is stored at the face corner $(f_\gamma, v_\gamma)$, store $\gamma$ there. Otherwise, suppose that $\alpha$ is the tree node stored at $(f_\gamma, v_\gamma)$. If $\alpha$ dominates $\gamma$, then $\gamma$ can gain only one child in the next round of expansion. If $\gamma$ dominates $\alpha$, then store $\gamma$ at $(f_\gamma, v_\gamma)$ 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 T. After the tree stops growing, take the minimum $L_{\infty}$ LSDP length from $s$ to $v_\alpha$ among all tree nodes $\alpha$ such that $v_\alpha = t$. Return this minimum as $\text{opt}_{\infty}$.

**Lemma 3.2** Let $T(k)$ be the time to compute an $L_{\infty}$ LSDP from $s$ to any vertex with an edge sequence of length $k$. An $L_{\infty}$ 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 $T$, 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 $\gamma$, we compute an $L_{\infty}$ LSDP from $s$ to $v_\gamma$ with edge sequence $\sigma_\gamma$, which takes $T(|\sigma_\gamma|) \leq T(m)$ time. Given two tree nodes $\alpha$ and $\gamma$ that correspond to the same face corner, we check the dominance between them in $O(1)$ time. Then, we trace $\sigma_\alpha$ and $\sigma_\gamma$ 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 $L_{\infty}$ SDP from $s$ to $t$ with the shortest edge sequence. Then $P_0$ does not visit a terrain face more than once. (An arbitrary $L_{\infty}$ SDP may visit a face more than once.) So $|\text{seq}(P_0)|$ is less than the bound on the number of levels in the sequence tree. If the final sequence tree contains a tree node $\beta_0$ such that $\sigma_{\beta_0} = \text{seq}(P_0)$, 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 $\beta_0$. There must exist two nodes $\alpha_0$ and $\alpha_1$ in some intermediate sequence tree such that $\sigma_{\alpha_0}$ is a prefix of $\sigma_{\beta_0}$ and $\sigma_{\alpha_1}$ dominates $\alpha_0$. The child of $\alpha_0$ that would be an ancestor of $\beta_0$ is pruned due to the dominance of $\alpha_1$ over $\alpha_0$. By Lemma 3.1(i), $\alpha_1$ remains after this pruning. By Lemma 3.1(ii), we can replace a prefix of $P_0$ with edge sequence $\sigma_{\alpha_0}$ by a descending path $Q$ with edge sequence $\sigma_{\alpha_1}$ so that the resulting descending path $P_1$ from $s$ to $t$ is not longer than $P_0$ (in the $L_{\infty}$ metric). That is, $P_1$ is also an $L_{\infty}$ SDP from $s$ to $t$. The path $Q$ cannot be shorter than the prefix of $P_0$ replaced (in the $L_{\infty}$ metric) because $P_0$ is optimal. Then, Lemma 3.1(ii) further implies that $|\sigma_{\alpha_1}| \leq |\sigma_{\alpha_0}|$ and, therefore, $|\text{seq}(P_1)| \leq |\sigma_{\beta_0}| = |\text{seq}(P_0)|$. This implies that our algorithm can generate a node $\beta_1$ in the final sequence tree such that $\sigma_{\beta_1} = \text{seq}(P_1)$ unless some ancestor of $\beta_1$ is dominated, causing $\beta_1$ to be pruned. If $\beta_1$ 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 $L_{\infty}$ SDP $P_1$ from $s$ to $t$ with edge sequence $\sigma_{\beta_1}$ so that the final sequence tree contains the node $\beta_1$. 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 $L_{\infty}$ 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 $L_{\infty}$ LSDP by linear programming

We present a linear program that finds an $L_{\infty}$ 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 $(1 - \zeta_i)u_i + \zeta iv_i$, where $\zeta_i \in [0, 1]$. The $x$-coordinate of a point in $e_i$ can be written as $(1 - \zeta_i)u_{i,x} + \zeta iv_{i,x}$, and we similarly can do the same for the $y$- and $z$-coordinates of any
point in $c_i$. Let $u_0 = v_0 = s$. Let $u_{k+1} = v_{k+1} = w$. An $L_\infty$ LSDP from $s$ to $w$ can be found by solving the following linear program.

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

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 $\ell_i$ an upper bound on the $L_\infty$ 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 $L_\infty$ LSDP length from $s$ to $w$ with edge sequence $(e_1, e_2, \ldots, e_k)$.

### 3.2 A combinatorial $L_\infty$ LSDP algorithm

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

Suppose that we have already computed $L_\infty$ LSDPs from $s$ to all points on $c_\alpha$. Our strategy is to extend these paths to $L_\infty$ LSDPs to all points on the two edges of $f_\alpha$ incident to $c_\alpha$. 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 $L_\infty$ LSDPs lengths from $s$ to $ab$ by a function $F_{ab}: I_{ab} \rightarrow \mathbb{R}$. That is, $F_{ab}(\zeta)$ is the $L_\infty$ LSDP length from $s$ to $p_\zeta$. 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 $G(h)$, is the hypersurface $\{(x_1, x_2, \ldots, x_d, x_{d+1}) : (x_1, x_2, \ldots, x_d) \in D \land x_{d+1} = h(x_1, x_2, \ldots, x_d)\}$.

Refer to Figure 4. Let $u$ and $w$ be the other two vertices of $f_\alpha$. Let $e$ be the edge with endpoints $u$ and $v_\alpha$. We parametrize $e_\alpha$ and $e$ by $\lambda, \tau \in [0, 1]$ such that $p_\lambda = (1 - \lambda)u + \lambda w$ denotes the parametrized point on $e_\alpha$ and $p_\tau = (1 - \tau)u + \tau v_\alpha$ denotes the parametrized point on $e$. The LSDP to a point $p_3 \in e_\alpha$ is extended to a point $p_\tau \in e$ by appending the segment $p_\lambda p_\tau$. The $L_\infty$ 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 $\varrho$ which takes on real values. Define seven planes as follows.

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

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 envelopes 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 $i \in [1, 6]$.
Therefore, by the induction assumption, descending. Thus, the domain of for every $\tau$

In (b), $G$ is shown bold in the $\lambda\varrho$-plane, and each segment on it is extended into a strip. The resulting surface is $G(F_{e_\alpha})$.

Define the function $L(\tau, \lambda)$ to be the $L_\infty$ 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 $(\tau, \lambda)$'s for which $p_\lambda p_\tau$ is descending. Therefore, $G(L) \cap H^+_7$ is the graph of $\|p_\lambda p_\tau\|_\infty$ for which $p_\lambda p_\tau$ is descending. See Figure 5(a) for an illustration. $G(L) \cap H^+_7$ 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 $F_{e_\alpha}$. By the induction assumption, $F_{e_\alpha}$ is convex and piecewise linear. Refer to Figure 5(b). We sweep each segment $\ell$ in $G(F_{e_\alpha})$ in a direction parallel to the $\tau$-axis to obtain a strip $[0, 1] \times \ell$. This gives a convex surface, which is the graph of a function $F_{e_\alpha}$ such that $F_{e_\alpha}(\tau, \lambda) = F_{e_\alpha}(\lambda)$ for every $\tau \in [0, 1]$ and every $\lambda \in I_{e_\alpha}$. The domain of $F_{e_\alpha} + L$ is the convex set $[0, 1] \times I_{e_\alpha}$. Therefore, $F_{e_\alpha} + L$ 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 $(\tau, \lambda)$’s for which $p_\lambda p_\tau$ is descending. Thus, the domain of $F_e$ is the interval $I_e = \{\tau \in [0, 1] : (\tau, \lambda) \in D$ for some $\lambda\}$. $F_e$ is convex and piecewise linear because $F_e(\tau) = \min_{\lambda, (\tau, \lambda) \in D} \{F_{e_\alpha}(\tau, \lambda) + L(\tau, \lambda)\}$.

Next, we show that $F_e$ can be constructed quickly when $F_{e_\alpha}$ is given.
Lemma 3.4 If \( G(F_{ea}) \) has \( k \) segments, then \( F_e \) can be constructed from \( F_{ea} \) in \( O(k \log k) \) time.

Proof. Let \( \overline{F}_{ea} \) be defined as in the proof of Lemma 3.3. Let \( G \) denote the subset of \( G(\overline{F}_{ea} + L) \) clipped within \( H_T \) and the unbounded box \([0,1] \times I_{ea} \times [0,\infty) \). 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 \( G(L) \) onto the \( \tau\lambda \)-plane, which has \( O(1) \) size, and the projection of \( G(F_{ea}) \) onto the \( \tau\lambda \)-plane, which has \( k \) parallel segments. The function value of \( \overline{F}_{ea} + L \) 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 \( \sigma \)-value of this vertex. Then, we sweep a plane orthogonal to the \( \tau \)-axis from \( a \) to \( a' \), stopping at the \( \tau \)-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 \( \tau\rho \)-plane. The projection of this segment, triangle, or trapezoid onto the \( \tau\rho \)-plane is the portion of \( G(F_e) \) over the \( \tau \)-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 \( \overline{F}_{ea} + L \), 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 \( F_{ea} \) because it affects the time to construct the children of \( \tau \). In this case, the segment connecting \( p_{\lambda_0} \in e_a \) and \( p_\tau \in e \) must cross the segment connecting \( p_{\lambda_\tau} \in e_a \) and \( p_{\tau'} \in e \). In a convex quadrilateral, the sum of the \( L_{\infty} \) lengths of its two diagonals is at least the sum of the \( L_{\infty} \) lengths of any two opposite sides. Therefore,

\[
F_{ea}(\lambda_\tau) + L(\tau', \lambda_\tau) + F_{ea}(\lambda_{\tau'}) + L(\tau, \lambda_{\tau'}) \\
\leq F_{ea}(\lambda_\tau) + L(\tau, \lambda_\tau) + F_{ea}(\lambda_{\tau'}) + L(\tau', \lambda_{\tau'}) \\
= F_e(\tau) + F_e(\tau').
\]

(1)

We claim that the segments \( p_{\lambda_0} p_{\tau} \) and \( p_{\lambda_\tau} p_{\tau'} \) are descending. Since \( p_{\lambda_0} p_{\tau} \) and \( p_{\lambda_\tau} p_{\tau'} \) are diagonals of the convex quadrilateral \( p_{\lambda_0} p_{\lambda_\tau} p_{\tau} p_{\tau'} \), they cross at some point \( z \). The segments \( p_{\lambda_0} z, p_{\lambda_\tau} z, z p_{\tau} \) and \( z p_{\tau'} \) are descending because \( p_{\lambda_0} p_{\tau} \) and \( p_{\lambda_\tau} p_{\tau'} \) are descending. Therefore, \( p_{\lambda_0} z p_{\tau} \) and \( p_{\lambda_\tau} z p_{\tau'} \) are two descending paths, implying that the segments \( p_{\lambda_0} p_{\tau} \) and \( p_{\lambda_\tau} p_{\tau'} \) are descending.

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

Lemma 3.5 For every pair of values \( \tau_0 \) and \( \tau_1 \) in \( I_e \), if \( \tau_0 < \tau_1 \), then \( \lambda_{\tau_0} \leq \lambda_{\tau_1} \).

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 \( p_{\lambda_\tau} \in e_a \) and \( p_\tau \in e \) must cross the segment connecting \( p_{\lambda_{\tau'}} \in e_a \) and \( p_{\tau'} \in e \). In a convex quadrilateral, the sum of the \( L_{\infty} \) lengths of its two diagonals is at least the sum of the \( L_{\infty} \) lengths of any two opposite sides. Therefore,

\[
F_{ea}(\lambda_\tau) + L(\tau', \lambda_\tau) + F_{ea}(\lambda_{\tau'}) + L(\tau, \lambda_{\tau'}) \\
\leq F_{ea}(\lambda_\tau) + L(\tau, \lambda_\tau) + F_{ea}(\lambda_{\tau'}) + L(\tau', \lambda_{\tau'}) \\
= F_e(\tau) + F_e(\tau').
\]

(1)

We claim that the segments \( p_{\lambda_0} p_{\tau} \) and \( p_{\lambda_\tau} p_{\tau'} \) are descending. Since \( p_{\lambda_0} p_{\tau} \) and \( p_{\lambda_\tau} p_{\tau'} \) are diagonals of the convex quadrilateral \( p_{\lambda_0} p_{\lambda_\tau} p_{\tau} p_{\tau'} \), they cross at some point \( z \). The segments \( p_{\lambda_0} z, p_{\lambda_\tau} z, z p_{\tau} \) and \( z p_{\tau'} \) are descending because \( p_{\lambda_0} p_{\tau} \) and \( p_{\lambda_\tau} p_{\tau'} \) are descending. Therefore, \( p_{\lambda_0} z p_{\tau} \) and \( p_{\lambda_\tau} z p_{\tau'} \) are two descending paths, implying that the segments \( p_{\lambda_0} p_{\tau} \) and \( p_{\lambda_\tau} p_{\tau'} \) are descending.

It follows from our claim and the definition of \( F_e \) that \( F_{ea}(\lambda_\tau) + L(\tau', \lambda_\tau) \geq F_e(\tau') \) and \( F_{ea}(\lambda_{\tau'}) + L(\tau, \lambda_{\tau'}) \geq F_e(\tau) \). By (1), we conclude that \( F_{ea}(\lambda_\tau) + L(\tau, \lambda_\tau) = F_e(\tau) \) and \( F_{ea}(\lambda_{\tau'}) + L(\tau', \lambda_{\tau'}) = F_e(\tau') \). But the equality \( F_{ea}(\lambda_{\tau'}) + L(\tau, \lambda_{\tau'}) = F_e(\tau) \) contradicts the definition of \( \lambda_\tau \) as \( \lambda_{\tau'} < \lambda_\tau \) by assumption.
The next lemma characterizes the value $\lambda_\tau$ for every $\tau \in I_e$. We call a value $\lambda_0 \in I_{e\alpha}$ a breakpoint of $F_{e\alpha}$ if $\lambda_0$ is the $\lambda$-value of a vertex of $\mathcal{G}(F_{e\alpha})$, including the endpoints of $I_{e\alpha}$. The breakpoints of $F_e$ are similarly defined.

**Lemma 3.6** For every value $\tau_0$ in $I_e$, at least one of the following three possibilities hold:

(i) $\lambda_{\tau_0}$ is a breakpoint of $F_{e\alpha}$,

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

(iii) $\tau_0$ and $\lambda_{\tau_0}$ satisfy the equation of the plane $H_{\tau}$.

**Proof.** Suppose that $\lambda_{\tau_0}$ is not a breakpoint of $F_{e\alpha}$. Then, $\lambda_{\tau_0}$ lies between two successive breakpoints $\lambda_i < \lambda_{i+1}$. Let $\mathcal{F}_{e\alpha}$ be defined as in the proof of Lemma 3.3. Let $q = a_i \lambda + b_i$ be the support plane of the strip in $\mathcal{G}(\mathcal{F}_{e\alpha})$ that lies above $[\lambda_i, \lambda_{i+1}]$. Add $q = a_i \lambda + b_i$ 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 $H^\perp_{\tau}$. Observe that $\mathcal{U}' \subseteq \mathcal{G}(\mathcal{F}_{e\alpha} + L)$.

By our definition of $\lambda_{\tau_0}$, $(\tau_0, \lambda_{\tau_0})$ must be the first two coordinates of some point on an edge of $\mathcal{U}'$. If $\tau_0$ and $\lambda_{\tau_0}$ do not satisfy the equation of $H_{\tau}$, 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 $q = a_i \lambda + b_i$ 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 $F_{e\alpha}$ 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 $\tau$ from the smaller endpoint of $I_e$ to the larger endpoint. As $\tau$ increases, the point $(\tau, \lambda_\tau)$ traces a locus $\eta$ in the $\tau\lambda$-plane. Project the subset of $\mathcal{G}(L)$ that lies in $H^\perp_{\tau}$ onto the $\tau\lambda$-plane. Let $\mathcal{L}$ denote the projected image.

By Lemmas 3.5 and 3.6, $\eta$ is monotone with respect to both the $\tau$-axis and the $\lambda$-axis. Moreover, for each segment in $\eta$, either it lies on an edge of $\mathcal{L}$, or it is parallel to the $\tau$-axis.
and has the $\lambda$-value of a breakpoint of $F_{e_0}$. Notice that $\eta$ could be disconnected. While tracing $\eta$, we may jump from a point $(\tau_0, \lambda_{m_0})$ in the direction of positive $\lambda$-axis to a point $(\tau_0, \lambda')$. It happens when $F_{e_0}(\lambda) + L(\tau_0, \lambda) = F_{e_0}(\lambda_{m_0}) + L(\tau_0, \lambda_{m_0})$ for every $\lambda \in [\lambda_{m_0}, \lambda']$, because the definition of $\lambda_\tau$ prefers $\lambda_{m_0}$ to all such larger $\lambda$. In summary, $\eta$ follows an edge of $\mathcal{L}$, or moves in the direction of positive $\tau$-axis, or makes a jump in the direction of positive $\lambda$-axis. Figure 6 shows an example. We break $\eta$ into segments, each being a maximal connected subset that lies on an edge of $\mathcal{L}$ and between two successive breakpoints of $F_{e_0}$, or that is parallel to the direction of the positive $\tau$-axis and lies inside a face of $\mathcal{L}$. Each segment of $\eta$ projects to a subset of $I_\alpha$ that lies between two successive breakpoints of $F_{e_0}$. Therefore, the number of segments in $\eta$ is an upper bound on the complexity of $\mathcal{G}(F_e)$.

We first bound the number of jumps $\eta$ makes. Suppose that $\eta$ jumps from a point $(\tau_0, \lambda_{m_0})$ in the direction of the positive $\lambda$-axis in a face $g$ of $\mathcal{L}$. Let $\ell_0$ be the segment in $g$ covered by this jump, i.e., $(\tau_0, \lambda_{m_0})$ is the lower endpoint of $\ell_0$, and $\ell_0$ is parallel to the $\lambda$-axis. Refer to Figure 7. As explained earlier, $F_{e_0}(\lambda) + L(\tau_0, \lambda)$ remains the same for all $(\tau_0, \lambda) \in \ell_0$. In other words, $F_{e_0}$ must cancel the component of $L(\tau_0, \lambda)$ that involves $\lambda$. 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 $\lambda$ in the equation of $H_j$ is fixed. This implies that $\ell_0$ can continue to extend in $g$, while keeping $F_{e_0}(\lambda) + L(\tau_0, \lambda)$ independent of $\lambda$, until the upper endpoint of $\ell_0$ reaches a $\lambda$-value that is a breakpoint of $F_{e_0}$, or $\ell_0$ hits an edge of $g$. In the first case, $\ell_0$ cannot extend any further because we would switch to the next segment in $\mathcal{G}(F_{e_0})$, which must have a slope different from the current segment in $\mathcal{G}(F_{e_0})$ as $F_{e_0}$ is convex. Thus, the equation of $H_j$ can no longer cancel $F_{e_0}(\lambda)$ for any $\lambda$-value larger than the upper endpoint of $\ell_0$. On the other hand, there is no edge of $\mathcal{L}$ to follow in the interior of $g$. Therefore, if the upper endpoint of $\ell_0$ is in interior of $g$, then by Lemma 3.6, $\eta$ will continue from the upper endpoint of $\ell_0$ in the direction of positive $\tau$-axis until $\eta$ reaches an edge of $g$. From this point onward, $\eta$ cannot make any further jump into $g$ in the direction of positive $\lambda$-axis because $F_{e_0}$ is convex and therefore a different segment of $F_{e_0}$ cannot cancel the term in the equation of $H_j$ that involves $\lambda$. For the second case that $\ell_0$ extends all the way across $g$, $\eta$ will have to traverse a segment in the direction of the positive $\tau$-axis to enter $g$ again. Such a segment corresponds to a breakpoint of $F_{e_0}$. Therefore, by the same argument as before, $\eta$ cannot jump in $g$ again after encountering a breakpoint of $F_{e_0}$. We conclude that $\eta$ makes at most one jump in the direction of the positive $\lambda$-axis in each face of $\mathcal{L}$.

Let $k$ be the number of segments in $\mathcal{G}(F_{e_0})$. In other words, $F_{e_0}$ has $k + 1$ breakpoints. Let $k'$ be the number of faces in $\mathcal{L}$. The above analysis shows that $\eta$ makes at most $k'$ jumps. Since $\eta$ is monotone, segments in $\eta$ that lie on edges of $\mathcal{L}$ have different $\lambda$-values at their upper endpoints. Such an upper endpoint is either at some $\tau$-value that $\eta$ makes a jump, or at some $\lambda$-value that is a breakpoint of $F_{e_0}$. So there are at most $k + k' + 1$ such upper endpoints, which implies that $\eta$ has at most $k + k' + 1$ segments that lie on edges of $\mathcal{L}$.
It remains to bound the number of segments in $\eta$ that are parallel to the $\tau$-axis. Consider two such segment $h_1$ and $h_2$ in this order along $\eta$ that lie in the same face $g$ of $\mathcal{L}$. We claim that $\eta$ 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 $h_1$, $\eta$ remains on or below the lower boundary edge of $g$ until $h_2$ because $\eta$ 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 $\eta$ has two segments in a face of $\mathcal{L}$ that are parallel to the $\tau$-axis, then $\eta$ must make a jump between them. Since $\eta$ makes at most one jump in a face, we conclude that $\eta$ has at most two segments in each face that are parallel to the $\tau$-axis in each face. Hence, $\eta$ has at most $2k'$ such segments in total that are parallel to the $\tau$-axis.

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

**Proof.** Lemma 3.7 implies that for each node $\alpha$ created, $\mathcal{G}(F_{v_\alpha})$ consists of $O(|\sigma_\alpha|)$ segments. By Lemma 3.4, the $L_\infty$ LSDP computation at $\alpha$ 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 $L_\infty$ SDP length $\text{opt}_\infty$ from $s$ to $t$. Recall that $\text{opt}$ is the SDP length from $s$ to $t$ and $\text{opt}_\infty \leq \text{opt} \leq \sqrt{3} \text{opt}_\infty$. Second, we clip the terrain $\mathcal{T}$ within the box centered at $s$ and of side length $4\sqrt{3} \text{opt}_\infty$. 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 $2\text{opt} \leq 2\sqrt{3} \text{opt}_\infty$, so it lies completely inside $\mathcal{T}^*$. It thus suffices to work with $\mathcal{T}^*$. Every edge in $\mathcal{T}^*$ has length at most $12\text{opt}_\infty$. 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 $\sigma$, 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 $\sigma$, where $L$ is the length of the longest edge in $\sigma$, 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 $r'_{i+1} = v$, 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 $\sigma$ 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 $a_i b_i \subseteq q_i r_i \subseteq c_i$ 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 $(e_i, \ldots, e_k)$ 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 $i$th 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 $\sigma$, and any $\mu \in (0, 1)$, an approximate shortest descending path from $s$ to $v$ with edge sequence $\sigma$ 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 $\sigma$.

**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 $\sigma$. 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 $\|D[c_i, v]\| \leq \|P_i\| \leq (k-i) \mu$.

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, 
\[
\| P_i \| &= \| P[c, v] \| + \| c, e \| \\
\geq & |D[c_{i+1}, v]| - (k - i - 1)\mu + \| c_i \| - \| c_{i+1} \| \\
\geq & |D[c_{i+1}, v]| - (k - i - 1)\mu + (\| c_i \| - \| c_{i+1} \| ) - \| c_{i+1} \| \\
= & |D[c_i, v]| - (k - i - 1)\mu - 2 \| c_{i+1} \| \\
\geq & |D[c_i, v]| - (k - i)\mu.
\]
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 \( \kappa n^2 \) tree nodes. Set \( \mu = \varepsilon\text{opt}_\infty/(\kappa n^2m(m+1)) \), where \( n \) is the number of vertices and \( m \) is the number of faces in \( T^* \). The fact that every edge of \( T^* \) has length at most \( 12\text{opt}_\infty \) immediately yields the following result.

**Corollary 4.1** Let \( \mu = \varepsilon\text{opt}_\infty/(\kappa n^2m(m+1)) \). Given a destination \( v \) and an edge sequence \( \sigma \) in \( T^* \), we can compute in \( O(|\sigma|^2 \log(n/\varepsilon)) \) time an approximate LSDP in \( T^* \) from \( s \) to \( v \) with edge sequence \( \sigma \) 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 \( \alpha \) is created, we apply Corollary 4.1 to compute an approximate LSDP from \( s \) to \( v_\alpha \) with edge sequence \( \sigma_\alpha \). We denote this path by \( P_\alpha \) and store it at \( \alpha \). The remaining task is to define a tree pruning rule to enforce the one-corner one-split property.

Let \( \mu = \varepsilon\text{opt}_\infty/(\kappa n^2m(m+1)) \) as defined in Corollary 4.1. Define \( \bar{\mu} = \kappa n^2m\mu = \varepsilon\text{opt}_\infty/(m+1) \). For every path \( Q \) in \( T^* \), the quasi-length of \( Q \) is equal to \( |Q| + |\text{seq}(Q)| \cdot \bar{\mu} \). We define a notion of dominance among the tree nodes in order to impose the one-corner one-split property. Let \( \alpha \) and \( \beta \) be two tree nodes corresponding to the same face corner \((f_\alpha, v_\alpha)\). Therefore, \( P_\alpha \) and \( P_\beta \) are approximate LSDPs from \( s \) to \( v_\alpha = v_\beta \) with edge sequences \( \sigma_\alpha \) and \( \sigma_\beta \), respectively. Then, \( \alpha \) quasi-dominates \( \beta \) if the quasi-length of \( P_\alpha \) is less than or equal to the quasi-length of \( P_\beta \). (Break ties arbitrarily.) Assume that \( \alpha \) quasi-dominates \( \beta \). We further decide whether \( \alpha \) quasi-dominates \( \beta \) 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 \( L_\infty \) case in Section 3. Refer to Figure 2 for the possible configurations of \( P_\alpha \) and \( P_\beta \).

The next result is the analog of Lemma 3.1 in the \( L_\infty \) case.

**Lemma 4.2** Let \( \alpha \) and \( \beta \) be two tree nodes that correspond to the same face corner \((f_\alpha, v_\alpha)\) such that \( \alpha \) quasi-dominates \( \beta \) on the right (resp. left). Let \( e \) be the edge that follows \( e_\alpha \) immediately in anticlockwise (resp. clockwise) order around the boundary of \( f_\alpha \).

(i) \( \alpha \) is not a descendant of \( \beta \).

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

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

Consider (i). Suppose for the sake of contradiction that \( \alpha \) is a descendant of \( \beta \). So \( \sigma_\beta \) is a proper prefix of \( \sigma_\alpha \). \( P_\alpha \) traverses the same edges and faces as \( P_\beta \) until \( f_\beta \). Afterwards, \( P_\beta \) stops at \( v_\beta \), and \( P_\alpha \) leaves \( f_\beta \) through some point \( p \) on an edge of \( f_\beta \) other than \( e_\beta \). Shortcut
after creating a new leaf dominance. Thus, every dominance check is replaced by a quasi-dominance check. Second, there are two differences though. First, dominance among the tree nodes is replaced by quasi-

\[ \alpha \]

Since Corollary 4.1 gives an approximate LSDP a descending path from \( s \) to \( v_\beta \). Let

\[ \text{Lemma 4.2(i), } \alpha \]

we return the minimum sequence tree as described in the last inequality follows from the quasi-dominance of \( \sigma \). The last inequality follows from the quasi-dominance of \( \alpha \). Therefore, if it is shorter than \( P_\beta \), Corollary 4.1 implies that it is shorter by no more than \( |\sigma_\beta|\mu \). Thus, every dominance check is replaced by a quasi-dominance check. Second, after creating a new leaf \( \alpha \), we use Corollary 4.1 to compute an approximate LSDP \( P_\alpha \) from \( s \) to \( v_\alpha \) with edge sequence \( \sigma_\alpha \). The one-corner one-split property is enforced by pruning the sequence tree as described in the \( L_\infty \) SDP algorithm in Section 3. After the tree stops growing, we return the minimum length among all tree nodes \( \alpha \) 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 \( \mu = \varepsilon \text{opt}_\infty / (\kappa n^2 m (m + 1)) \), \( \tilde{\mu} = \kappa n^2 m \mu = \varepsilon \text{opt}_\infty / (m + 1) \), and \( m \) denotes the number of faces in \( T^* \) (which bounds the number of levels in the sequence tree). Let \( \text{opt} \) be the SDP length from \( s \) to \( t \). Let \( P_0 \) be an SDP from \( s \) to \( t \). So \( \|P_0\| = \text{opt} \) and \( |\text{seq}(P_0)| < m \) as \( P_0 \) does not visit any face more than once.

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

Suppose that there is no such node \( \gamma_0 \) in the final sequence tree. There must exist two nodes \( \beta_0 \) and \( \alpha_1 \) in some intermediate sequence tree such that \( \sigma_{\beta_0} \) is a prefix of \( \sigma_{\gamma_0} \) and \( v_{\gamma_0} \) is pruned due to the quasi-dominance of \( \alpha_1 \) above \( \beta_0 \). By Lemma 4.2(i), \( \alpha_1 \) remains after pruning. We will show that there is an almost equally good descending path from \( s \) to \( t \) whose edge sequence has \( \alpha_1 \) as a prefix.

Without loss of generality, assume that \( \alpha_1 \) quasi-dominates \( \beta_0 \) on the right, so the right child of \( \beta_0 \) is pruned. Refer to Figure 8. Let \( e \) be the edge that immediately follows \( e_{\beta_0} \) in anticlockwise order around the boundary of \( f_{\beta_0} \). Since the pruned right child of \( \beta_0 \) would be
The edge sequence of the descending path we can proceed to bound solid path has edge sequence $s$ to $s$ from an ancestor of $s$. However, it is possible that the final sequence tree does not contain such a node $s$. P

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 $a_1$ quasi-dominates $b_0$ on the right. The grey solid path has edge sequence $\sigma_{b_0}$. Lemma 4.2(ii) guarantees a bold solid path $Q_{a_1}$ from $s$ to $p$ with edge sequence $\sigma_{a_1}$. Concatenating $Q_{a_1}$ with the grey dashed path gives a descending path from $s$ to $t$ with edge sequence $\sigma_{\gamma_1}$.

an ancestor of $\gamma_0$, $P_0$ must cross the edge $e$ at some point $p$. By Lemma 4.2(ii), there is a descending path $Q_{a_1}$ from $s$ to $p$ with edge sequence $\sigma_{a_1}$ such that

$$
\|Q_{a_1}\| \leq \|P_0[s, p]\| + (|\sigma_{b_0}| - |\sigma_{a_1}|)\mu + |\sigma_{b_0}|\mu
$$

$$
< \|P_0[s, p]\| + (|\sigma_{b_0}| - |\sigma_{a_1}|)\mu + m\mu. \quad (3)
$$

Let $\sigma_{\gamma_1}$ be the concatenation of $\sigma_{a_1}$ and the suffix of $\sigma_{\gamma_0}$ after $\sigma_{b_0}$. So $|\sigma_{\gamma_0}| - |\sigma_{\gamma_1}| = |\sigma_{b_0}| - |\sigma_{a_1}|$. The edge sequence of the descending path $Q_{a_1} \cdot P_0[p, t]$ is $\sigma_{\gamma_1}$. This implies that the LSDP from $s$ to $t$ with edge sequence $\sigma_{\gamma_1}$ exists. Denote it by $P_1$. It satisfies the following relation.

$$
\|P_1\| \leq \|Q_{a_1}\| + \|P_0[p, t]\|
$$

$$
< \|P_0\| + (|\sigma_{b_0}| - |\sigma_{a_1}|)\mu + m\mu \quad (\cdot \ (3))
$$

$$
= \|P_0\| + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_1}|)\mu + m\mu
$$

$$
= \text{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_1}|)\mu + m\mu. \quad (4)
$$

If the final sequence tree contains a tree node $\gamma_1$ such that $v_{\gamma_1} = t$ and $\sigma_{\gamma_1} = \text{seq}(P_1)$, 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 $\gamma_1$. 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 $i \in [0, kn^2 - 1]$, if the final sequence tree does not contain a tree node $\gamma_i$ such that $v_{\gamma_i} = t$ and $\sigma_{\gamma_i} = \text{seq}(P_i)$, then there exists a descending path $P_{i+1}$ from $s$ to $t$ such that $\|P_{i+1}\| \leq \text{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_{i+1}}|)\mu + (i + 1)m\mu$.

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

$$
\|P_i\| \leq \text{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_i}|)\mu + im\mu \quad (5)
$$

for some $i \in [1, kn^2 - 1]$. Suppose that the final sequence tree does not contain a tree node $\gamma_i$ such that $v_{\gamma_i} = t$ and $\sigma_{\gamma_i} = \text{seq}(P_i)$. Observe that $\|P_i\| \geq \text{opt}$ and for $i < kn^2$, $im\mu < kn^2m\mu = \mu$. Plugging these two inequalities into (5) gives $(|\sigma_{\gamma_i}| - |\sigma_{\gamma_0}|)\mu \leq im\mu < \mu$. 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 $\gamma_i$ would have been generated if the sequence tree had not been pruned during its construction. Therefore, there must exist tree nodes $\beta_i$ and $\alpha_{i+1}$ in some intermediate sequence tree such that $\gamma_i$ is a descendant.
Corollary 4.1, computing the LSDP takes time $O(n \log(n/\varepsilon))$ 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 $\text{opt}_\infty$. 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


