Karlsruhe Institute of Technology

Approximate Shortest Gentle Paths on Terrains

Study Thesis of

Leonie Sautter

At the Department of Informatics
Institute of Theoretical Informatics

Reviewer:	Dorothea Wagner
Advisor:	Martin Nöllenburg
Second advisor:	Siu-Wing Cheng

Duration: 16th of October 2013 – 16th of April 2014

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, the 2nd of April 2014

.....

(**Leonie Sautter**)

Abstract

A path on a polygonal terrain is gentle if it does not ascend or descend too steeply, i.e., its gradient is below a given threshold θ at every point. The constriction to gentle paths makes the shortest path problem more realistic as vehicles usually can not traverse arbitrarily steep slopes. We give a $(1 + \varepsilon)$-approximation algorithm for the shortest gentle path problem. The running time lies in $\mathcal{O}\left(n^{5.5} \log \frac{n}{\varepsilon}\right)$ where n denotes the number of vertices of the terrain and is thus independent of the geometry of the terrain. Furthermore we discuss the open question whether the shortest gentle path problem is NP-hard and since we could not prove hardness develop ideas for an exact algorithm.

Zusammenfassung

Ein Pfad auf einem polygonalen Terrain heißt flach wenn er nicht zu stark ansteigt oder fällt, das heißt wenn seine Steigung für alle Punkte unter einer gegebenen Schranke θ bleibt. Die Beschränkung auf flache Pfade macht das Problem einen kürzesten Weg zu finden realistischer, da Fahrzeuge in der Realität keine beliebig steilen Strecken befahren können. In dieser Arbeit wird ein Approximationsalgorithmus mit einem relativen Fehler ε für das Kürzeste-Flache-Wege-Problem vorgestellt, der unabhängig von der Geometrie des Terrains eine Laufzeit von $\mathcal{O}\left(n^{5,5} \log \frac{n}{\varepsilon}\right)$ hat. Dabei bezeichnet n die Anzahl der Knoten des Terrains. Zusätzlich wird die offene Frage diskutiert, ob das Kürzeste-Flache-Wege-Problem NP-schwer ist. Da dies in dieser Arbeit nicht gezeigt werden kann, beschäftigt sich ein Kapitel mit der Entwicklung eines exakten Algorithmus.

Contents

1. Introduction

Shortest path problems have intrigued computer scientists for a long time. They are algorithmically interesting and have been well researched. But there are also many very different variations of the problem. One distinction between several variations is the type of search space in which we search for the shortest path. The most commonly known problem is probably the shortest path problem on weighted graphs for which we have many algorithms among them Dijkstra's famous algorithm [Dij59]. In this thesis, however, we will not consider graphs as our search space but terrains. A terrain can be seen as a landscape with hills and valleys which is modeled in our case by a polygonal surface.

In common two-dimensional topographic maps the measured distance between two points represents only the horizontal distance between these points in the real world. Thus in many common geographical shortest path problems the gradient and the vertical distance that has to be covered by a path are simply omitted. But in some conceivable cases the elevation gain is quite important maybe even more so than the horizontal distance. For example the difficulty of a hike in the Alps might be measured just by the difference in altitude that has to be covered or a robot might not be able to ascend very steep slopes. The shortest gentle path problem on terrains takes the gradient of a path as an important factor into account.

The terrain in our case is given as the surface of a triangulated mesh (see Figure 1.1 for an example). The shortest gentle path problem searches for the shortest path whose gradient is at no point outside a boundary called the slope constraint, i.e., a path that is nowhere too steep, neither rising nor falling.

There are some publications on the topic of this problem available as well as on the shortest anisotropic path problem which is a generalization we will describe in section 1.1 but it is still not known if the problem is NP-hard. For some related problems that are also special cases of the shortest anisotropic path problem, polynomial approximation algorithms have been developed, e.g. [LMS99] and [ALM09]. Until now there were only practical approximation approaches for the shortest gentle path problem but no algorithm with a polynomial worst case runtime. For example Liu and Wong presented a method to simplify the terrain such that the shortest gentle path problem can be solved faster [LW11]. They demonstrated the efficiency of their algorithm by the means of experiments

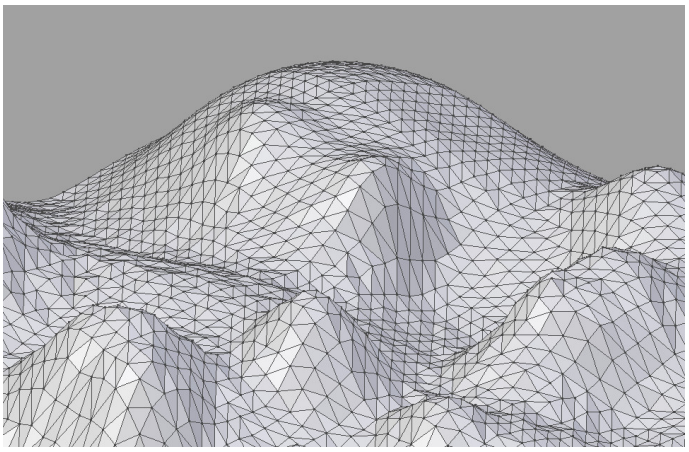


Figure 1.1: Example of a terrain modeled by a mesh.

but not by proving a worst case runtime.

The shortest gentle path problem is closely related to the shortest descending path problem, in which paths are not allowed to ascend at any point, since the problems both have terrains as the underlying structure and the only constraint on the paths is that the gradient has to stay in a certain interval. Thus we first took a look at the proposed algorithms for shortest descending paths. The problem is NP-hard and as such cannot be solved exactly by a polynomial algorithm unless P = NP. The currently best approximation algorithm that is independent of the terrain geometry was proposed by Cheng and Jin [CJ12]. It computes a $(1+\varepsilon)$-approximate shortest descending path in $\mathcal{O}\left(n^4 \log \frac{n}{\varepsilon}\right)$ time for any $\varepsilon \in (0,1)$. As it turns out the ideas of the algorithm can be used to find an approximation algorithm for the shortest gentle path problem too.

In Chapter 2 we introduce terminology, formally define the problem, prove some properties of shortest gentle paths and describe the previously mentioned algorithm for the shortest descending path problem [CJ12]. Its basic idea is to solve a similar but easier problem exactly and then use the solution to find lower and upper limits for the solution of the original problem. These limits are then used to bound the error that is made in the approximation algorithm. The approximation algorithm itself uses a structure called sequence tree that was first introduced by Chen and Han and upholds a certain property that ensures that the tree does not grow exponentially [CH90]. With the help of this tree the algorithm can iterate through all prospective solutions and find an approximate shortest descending path.

In Chapter 3 the algorithm for shortest descending paths is adapted to the shortest gentle path problem. This results in a $(1+\varepsilon)$-approximation algorithm that has a runtime which lies in $\mathcal{O}\left(n^{5.5} \log \frac{n}{\varepsilon}\right)$.

Since the shortest gentle path problem is not known to be hard there could also be a polynomial algorithm that solves the problem exactly. In Chapter 4 we formulate an idea for such an exact algorithm for the shortest gentle path problem. The basic concept is to trace a path from the origin to the destination by characterizing the bend angles at each edge of the terrain. For the shortest descending path problem this idea was implemented by Ahmed [Ahm09]. Unfortunately the adaption to the shortest gentle path problem is quite complex and results ultimately in an exponential worst case runtime for the exact algorithm.

1.1 Related Work

Many different shortest path problems on terrains have been studied. Without any restrictions the shortest path on a terrain can be found in $\mathcal{O}(n^2)$ time, n being the number of vertices in the terrain, as shown by Chen and Han [CH90]. In their paper they introduced a data structure called sequence tree and a property called one-angle-one-split for this structure that allowed them to control the running time of their algorithm. This data structure and property will also be used in our thesis to the same effect.

The weighted region problem is a generalization of the general shortest path problem as it adds a weight to each face of the terrain. The cost of a path through a face is then measured by the length multiplied with the weight of the face. For this problem Mitchell and Papadimitriou found the first approximation algorithm that runs in $\mathcal{O}(n^8 L)$ time, with L being the precision of the problem instance [MP91]. It uses a continuous Dijkstra approach while most later algorithms use Steiner points to discretise the search space and convert the terrain into a weighted graph. Alexandrov, Maheshwari and Sack use a Steiner point approach to find a $(1+\varepsilon)$-approximation in $\mathcal{O}(C(T)\frac{n}{\sqrt{\varepsilon}} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$ time, where $C(T)$ describes geometric parameters and the weights of the faces of the surface [AMS05]. Zun and Reif get a runtime of $\mathcal{O}(\frac{n}{\varepsilon}(\log \frac{1}{\varepsilon} + \log n) \log \frac{1}{\varepsilon})$ that is independent of the geometry of the terrain using a similar approach [SR06].

The anisotropic path problem adds extra flexibility to the weighted region problem. Here the weight of a path within a face depends also on the direction of the path. With this problem it is possible to model the effects that friction or gravity will have on a vehicle that moves along a path. It is also possible to model the shortest gentle path problem as a special anisotropic path problem because the slope constraint is actually a constraint on the possible directions of paths within each face. The most widely used model was introduced by Rowe and Ross [RR90]. It takes many criteria into account, for example inclines that are too steep, limits of breaks, or hillsides that are so steep that they cannot even be traversed on a horizontal path because the danger of overturning or slipping is to great. Impermissible directions that arise due to these criteria can be covered by so-called switchback paths that zigzag to and fro using directions that are only just permissible. These paths we will also use in our algorithm for the shortest gentle path problem. Contrary to our algorithm all approximation algorithms for the model of Rowe and Ross are based on a Steiner point approach (see [LMS99] and [SR05] for examples).

On the topic of gentle paths only few papers have been published. In [LW11] Liu and Wong describe a method to simplify the surface. They then use a classic breadth-first search strategy to find the shortest gentle path on the simplified surface. The speedup in comparison to an exact algorithm for the shortest gentle path problem and depending on the required accuracy of the solution was measured in experiments. Opposed to this experimental approach, Ahmed et al. [ALM09] describe a fully polynomial-time approximation scheme (FPTAS) that solves the shortest gently descending path problem. This problem combines the property that a path is not allowed to ascend with the constraints of gentle paths and is thus a more realistic model for descending paths as very steep descends are forbidden. Again they preprocess the terrain and discretise it by the use of Steiner points. The running time of the algorithm lies in $\mathcal{O}\left(\frac{n^2 X}{\varepsilon} \log\left(\frac{nX}{\varepsilon}\right)\right)$ where X is a factor that depends on the terrain geometry.

As another problem with a slope constraint we also considered the shortest descending

path problem. There are some exact algorithms for special terrains (e.g. concave or convex) but on general terrains we have only approximation algorithms. Ahmed et al. [ADL+10] present a $(1+\varepsilon)$-approximation algorithm with a running time of $\mathcal{O}\left(\frac{n^2 L}{\varepsilon h}\log^2\frac{nL}{\varepsilon h}\right)$ where L is the length of the longest edge in the terrain and h is the smallest height of a triangle of the terrain. With a running time of $\mathcal{O}\left(n^4\log\frac{n}{\varepsilon}\right)$ Cheng and Jin's algorithm [CJ12] has a higher dependency on n but is independent of the terrain geometry.

2. Preliminaries

In this chapter we introduce some basic terminology, formally define the shortest gentle path problem and prove some properties of shortest gentle paths. Furthermore we describe Cheng and Jin's algorithm for the shortest descending path problem [CJ12] as we will adapt it later to the shortest gentle path problem.

2.1 Paths on Terrains

A *terrain* is a two dimensional surface consisting of polygons in the three dimensional space $\mathbb{R}^3$ that projects injectively onto the horizontal plane. As input for the algorithm we are given a terrain $\mathcal{T}$ with n vertices and two points in the terrain, the source s and the destination t. The three coordinates of a point x are called x_1, x_2 and x_3 with x_3 being the height of the point.

From here on only triangulated terrains are considered, but as every terrain with n vertices can be triangulated in $\mathcal{O}(n)$ time this is no real limitation. If s and t are not already vertices of $\mathcal{T}$, they are added as such and connected to all other vertices of their respective faces.

All paths from s to t are directed and can be divided into segments. A *segment* is a maximal continuous subpath that is contained in one face. The endpoints of segments are always called *nodes* in contrast to the vertices of the terrain. They always lie on edges of $\mathcal{T}$.

The *face sequence* of a path is a list of the faces the path traverses in this order. If a segment lies completely on the edge between two faces we can randomly choose which face gets represented in the face sequence. Additionally if the path passes through a vertex we pretend that the path makes a short detour around the vertex and add the faces that are traversed by this detour (see Figure 2.1). In this thesis we will also use the term *edge sequence* that is defined as the sequence of the common edges between each pair of neighboring faces in the face sequence (see Figure 2.2).

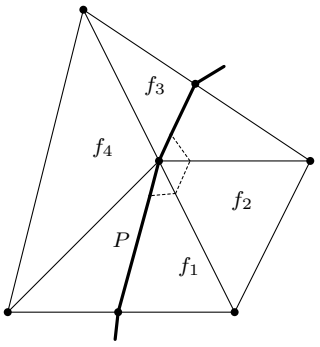


Figure 2.1: Detour around a vertex. In this case f_1, f_2 and f_3 are added to the face sequence for the path P.

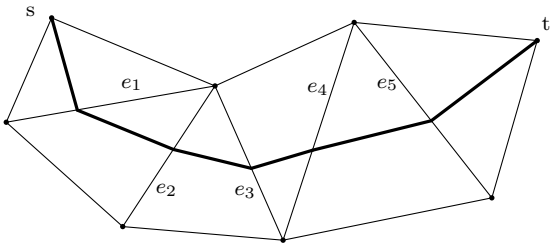


Figure 2.2: Example of a path with the edge sequence $(e_1, e_2, e_3, e_4, e_5)$.

2.2 Shortest Gentle Paths

A path is called *gentle* if at every point its absolute gradient is less than or equal to the given threshold $\theta > 0$. For a subpath with endpoints x and y that is a line segment, its absolute gradient, also called slope, is $sl(\overline{xy}) = \left|\arcsin\left(\frac{|y_3 - x_3|}{\|y-x\|_2}\right)\right|$.

Shortest Gentle Path Problem (SGP Problem).
Given a terrain $\mathcal{T}$, a threshold $\theta > 0$ and two points s and t on $\mathcal{T}$, find the shortest path from s to t on the terrain $\mathcal{T}$ that is gentle with respect to θ.

For a given edge sequence σ a *locally shortest gentle path (LSGP)* between two points s and t on the terrain is the shortest path between s and t that has edge sequence σ and is gentle.

Now we want to characterize shortest gentle paths and prove the existence of such paths. Thus we first consider the case that the two points s and t lie on the same face of $\mathcal{T}$. Now we have two possible cases: Either the line segment $\overline{st}$ between s and t is gentle or it is too steep.

In the first case $\overline{st}$ is obviously the SGP between s and t. In the second case we can use a gentle path that zig-zags forth and back along the direction of the line segment from s to t and has gradient $-\theta$ if $s_3 > t_3$ or gradient θ if $s_3 < t_3$ at each point. Such a path is called a *switchback* path (see Figure 2.3). If we find such a switchback path, it must be a shortest gentle path because using a gentle path with the maximal possible inclination at every point we need at least a path of length $\frac{|s_3 - t_3|}{\sin(\theta)}$ to overcome the height difference

between the endpoints. This length is exactly the length of the switchback path since it has always this exact gradient and proceeds from s to t.

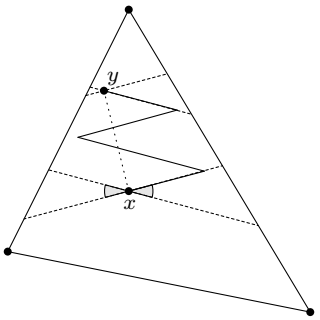


Figure 2.3: Example of a switchback path.

With the previously made observations we can formulate a Lemma that describes the behaviour of shortest gentle paths.

Lemma 1. *A shortest gentle path always consists of a sequence of line segments.*

Proof. Assume there is a shortest gentle path P that does not only consist of line segments. Then there must be a subpath between a point x and a point y that is not made up of straight line segments such that it is contained in one face of the terrain. If the straight line segment from x to y is gentle, as mentioned before, this is the only possible shortest gentle path between the two points and thus P could not be a shortest gentle path in contradiction to the assumption. Now if the line segment from x to y is too steep we already know that a switchback path that consists of a sequence of straight line segments is a possible solution. Since the switchback path has the maximal gradient at every point every other path from x to y that has a lower gradient at some point must be inevitably longer. Thus P cannot be a shortest gentle path which is a contradiction to the assumption. □

Now we still need to show that such a switchback path always exists. The *admissible directions* for each point x on the terrain are defined as all directions such that there is a line segment from x with this direction that lies on the terrain and is gentle (see Figure 2.4). Since θ is always greater than 0, each inner point of a face has either two symmetrical ranges of admissible directions or all directions are admissible. If x lies on an edge of a face but not on a vertex there is at least one range of admissible directions within this face. Only if x is a vertex of $\mathcal{T}$ there might be no admissible direction. In this case there is no SGP from x to t.

Lemma 2. *For every two points x and y in a face f of $\mathcal{T}$ that are not vertices and every slope constraint $\theta < 90°$, an SGP between them exists that lies completely inside this face.*

Proof. Without loss of generality let $x_3 \leq y_3$. If the line segment between x and y is gentle this is the SGP. Otherwise we want to find a switchback path. For every point in the interior of the triangle the only possible directions with slope exactly θ are the two boundaries of the ranges of admissible directions that have a positive gradient. If a point lies on an edge of the triangle, there is still at least one such boundary direction.

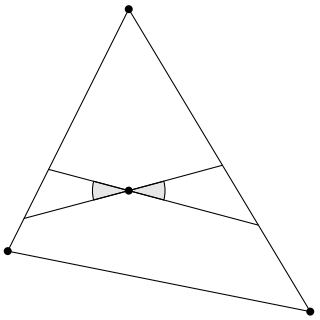


Figure 2.4: Admissible directions for a point on a face of $\mathcal{T}$.

From x on we can arbitrarily choose one of these boundary directions. Then we follow this direction until we cross a boundary line of the ranges of admissible directions of y or we reach an edge of the face. If we reached an edge we stop the line segment just before reaching it and take the other boundary direction from there. If we continued the line segment up to the edge, the second boundary direction might lie outside the current face. We repeat the process with the new directions until we reach a boundary line of y (see Figure 2.3). This will always happen because, if the path reaches the height of y it must either have crossed one of the boundary lines or must go exactly to y. Furthermore, the height increases proportional to the length of the path and the path can always be extended with another switchback. When a boundary line of y is reached, we can just follow it straight to y. Since we always used line segments with gradient θ we constructed a switchback path and thus found an SGP. □

The length of one segment of an SGP depends only on its beginning and end point x, and y and can be expressed as

$$l = \max\left(\frac{|x_3 - y_3|}{\sin(\theta)}, \|x - y\|\right)$$

That is true because either the direct connection between the two points is admissible or there is a switchback path between the points. In the first case the path has a gradient with an absolute value that is smaller or equal to θ. Thus a path that would have gradient $\pm\theta$ at any point would be shorter. In the second case the switchback path is obviously longer than the line segment between x and y. Therefore we can use the maximum of both lengths to calculate the length of the SGP.

Using Lemma 2 we can also say that for every pair of points s and t on a terrain $\mathcal{T}$, every threshold $0 < \theta < 90°$ and every possible edge sequence σ there exists a gentle path and with that also an SGP. The only exception is if either s or t are at a vertex that has no admissible directions in the first resp. last face in the face sequence.

For every gentle path we define the corresponding *simplified path* as the path that we get by substituting every segment by the direct connection between the nodes at the ends of the segment. Note that the simplified path is not always gentle.

Now we can solve the shortest gentle path problem by solving the general shortest path problem for a special norm that represents the length of a segment of an SGP between two

points. This norm is defined as $\|\mathbf{x}\|_s = \max\left(\frac{|\mathbf{x}_3|}{\sin(\theta)}, \|\mathbf{x}\|_2\right)$ with $\mathbf{x} \in \mathbb{R}^3$ being the vector between the endpoints of the segment. Doing this we get a simplified path that can be converted back to an SGP. We only have to test if an SGP exists by testing if there is an admissible direction for s and t.

Lemma 3. $\|\mathbf{x}\|_s = \max\left(\frac{|\mathbf{x}_3|}{\sin(\theta)}, \|\mathbf{x}\|_2\right)$ *with* $\mathbf{x} \in \mathbb{R}^3$ *is a norm.*

Proof. To prove that $\|\cdot\|_s$ is a norm we need to show that the following conditions are satisfied for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3,\ \alpha \in \mathbb{R}$:

$$\begin{aligned} |\alpha| \cdot \|\mathbf{x}\|_s &= \|\alpha \cdot \mathbf{x}\|_s \\ \|\mathbf{x}\|_s = 0 &\Rightarrow \mathbf{x} = 0 \\ \|\mathbf{x} + \mathbf{y}\|_s &\leq \|\mathbf{x}\|_s + \|\mathbf{y}\|_s \end{aligned}$$

The first two inequalities are trivial, the third is proven below.

$$\begin{aligned} \|\mathbf{x}+\mathbf{y}\|_s &= \max\left(\frac{|\mathbf{x}_3+\mathbf{y}_3|}{\sin(\theta)}, \|\mathbf{x}+\mathbf{y}\|_2\right) \leq \max\left(\frac{|\mathbf{x}_3|}{\sin(\theta)} + \frac{|\mathbf{y}_3|}{\sin(\theta)}, \|\mathbf{x}\|_2 + \|\mathbf{y}\|_2\right) \\ &\leq \max\left(\frac{|\mathbf{x}_3|}{\sin(\theta)} + \frac{|\mathbf{y}_3|}{\sin(\theta)},\ \|\mathbf{x}\|_2 + \|\mathbf{y}\|_2,\ \frac{|\mathbf{x}_3|}{\sin(\theta)} + \|\mathbf{y}\|_2,\ \|\mathbf{x}\|_2 + \frac{|\mathbf{y}_3|}{\sin(\theta)}\right) \\ &= \max\left(\frac{|\mathbf{x}_3|}{\sin(\theta)}, \|\mathbf{x}\|_2\right) + \max\left(\frac{|\mathbf{y}_3|}{\sin(\theta)}, \|\mathbf{y}\|_2\right) = \|\mathbf{x}\|_s + \|\mathbf{y}\|_s \end{aligned}$$

□

In Chapter 3 we describe an approximation algorithm for the SGP problem and Chapter 4 contains ideas for an exact SGP-algorithm.

2.3 The Approximation Algorithm for Shortest Descending Paths

This section describes the approximation algorithm for the shortest descending path problem as described by Cheng and Jin in [CJ12]. The algorithm for shortest gentle paths in Chapter 3 is based on this approach. Here we will only give an overview over the main ideas of the algorithm that are needed to understand the algorithm in Chapter 3. The details and proofs can be found in [CJ12].

A shortest descending path (SDP) between a source s and a destination t on a terrain is a shortest path that never increases in height on the course from s to t.

The algorithm first computes a lower and upper bound for the length of an SDP and then uses these bounds to limit the error and the runtime of the actual approximation algorithm. To find these bounds the SDP-Problem is solved using the maximum norm instead of the Euclidean norm. This problem will be called L_∞SDP and can be solved exactly in $\mathcal{O}(n^3 \log n)$ time. Using the inequality of the Euclidean and the maximum norm $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{3}\|\mathbf{x}\|_\infty$ for $\mathbf{x} \in \mathbb{R}^3$ we get $\mathsf{opt}_\infty \leq \mathsf{opt}_2 \leq \sqrt{3}\mathsf{opt}_\infty$ with opt_∞ denoting the length of an SDP for the maximum norm and opt_2 the one for the Euclidean norm.

The L_∞SDP and the approximate SDP are computed similarly. In both cases we iterate over all possible edge sequences, compute the lengths of the corresponding locally shortest descending paths and return the shortest one. In the case of the L_∞SDP we can compute the local paths exactly and thus get an exact solution; for the general SDP we can only approximate the local paths and thus get an approximate solution.

2.3.1 Computing the L_∞SDP exactly

To compute the L_∞SDP we define the L_∞LSDP (the locally shortest descending path) analogue to the LSGP as a shortest descending path that has a given edge sequence. An L_∞LSDP can easily be found in polynomial time using linear programming. Cheng and Jin also describe a combinatorial algorithm that reduces the runtime to compute all required L_∞LSDPs in $\mathcal{O}(n^3 \log n)$. That algorithm will not be described in this thesis. None the less it can also be used to reduce the runtime of the our algorithm described in Section 3.1.1.

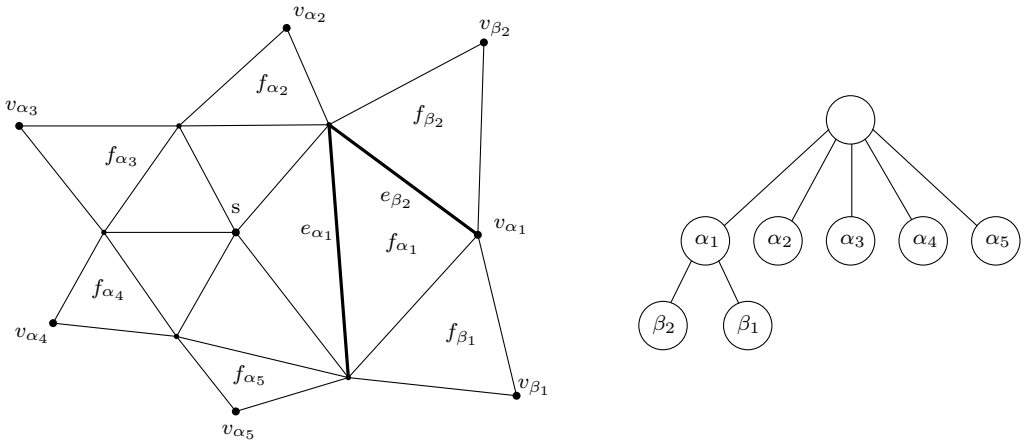
Knowing how to compute an L_∞LSDP we can iterate over all possible edge sequences by using a structure called *sequence tree* that was first proposed by Chen and Han [CH90]. The basic idea is to grow a tree of all possible edge sequences in breadth first manner from s outwards until the maximum depth, which is the number of faces in $\mathcal{T}$, is reached. The tree can stop growing at this point because if we assume there were an SDP that visits at least one face twice, we could always find a shorter path that visits this face only once by shortcutting the loop the SDP would have to make. This path is also descending and thus a shortest descending path visits each face at most once.

Each node α in the sequence tree corresponds to a *face corner* (f_α, v_α), the corner of f_α at vertex v_α. The edge e_α denotes the edge of f_α that is opposite to v_α. The initial tree has only one root node. Then for each possible first edge in the edge sequence we grow a child node that represents the face corner opposite of the edge in the face that s does not belong to (see Figure 2.5: For e_{α_1} we grow the child α_1 corresponding to $(f_{\alpha_1}, v_{\alpha_1})$).

From this level on, every leaf node corresponding to a face corner (f_α, v_α) can grow exactly two children. This is true because we can leave the face f_α only through one of its edges through which we did not enter. These edges correspond to a face corner of the next face we enter through them. The child node that corresponds to the edge that follows v_α directly in clockwise (resp. anticlockwise) order around the border of f_α is called the right (resp. left) child. Since every node α except the root node corresponds to an edge e_α the path from the root node to α in the sequence tree corresponds to a distinct edge sequence that we can also associate with α.

Unfortunately the tree grows exponentially, therefore a way to prune paths that are not relevant is needed. For this purpose we use the so-called *one-corner one-split property*: One face corner may correspond to multiple nodes in the sequence tree but only one of these nodes may have two children. This property is enforced while growing the tree. Thus when growing two children for a node would violate the one-corner one-split property either one of these children is pruned or one of the children of the conflicting node is pruned. By enforcing this property it is assured that at most $O(n^2)$ nodes are created during the process of growing as shown by Chen and Han [CH90].

To find out which of the children we can safely prune, we use the concept of *dominance*. A node α corresponding to the face corner (f, v) and the edge sequence σ_α dominates



(a) The relevant part of the terrain (b) The corresponding sequence tree

Figure 2.5: The edge sequence corresponding to node β_2 in (b) is $(e_{\alpha_1}, e_{\beta_2})$.

a node β corresponding to the same face corner (f, v) and the edge sequence σ_β if the L_∞LSDP P_α from s to v with edge sequence σ_α is shorter than the L_∞LSDP P_β from s to v with edge sequence σ_β (see Figure 2.6). If both path lengths are the same, the node with the shorter corresponding edge sequence dominates the other. To decide which child of β should be pruned we need to determine if β is *dominated on the left or right*. Let e_1 be the first edge in the longest common suffix of σ_α and σ_β. Let further e_α be the edge before e_1 in σ_α and e_β the edge before e_1 in σ_β. If e_1 is the first edge in σ_α or σ_β take the vertex s instead of e_α or e_β respectively. If (e_1, e_α, e_β) lie in clockwise (resp. anticlockwise) order around their common face, α dominates β on the right (resp. left) (see Figure 2.6).

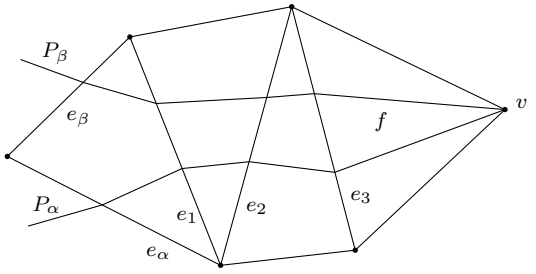


Figure 2.6: In the picture the node α that corresponds to the edge sequence $\sigma_\alpha = (\ldots, e_\alpha, e_1, e_2, e_3)$ and the L_∞LSDP P_α dominates β that corresponds to the edge sequence $\sigma_\beta = (\ldots, e_\beta, e_1, e_2, e_3)$ and the L_∞LSDP P_β on the right.

If α dominates β on the right (resp. left) the right (resp. left) child γ of β can be pruned and thus the one-corner one-split property is restored. The complete proof of this can be found in [CJ12, Lemma 3.1]. The idea is to show that the L_∞LSDP Q_β with edge sequence σ_β to any point r on the edge e_γ that lies directly after v in clockwise (resp. counterclockwise) order around f is at least as long as the L_∞LSDP Q_α to r with edge sequence σ_α (see Figure 2.7).

The complete algorithm for the L_∞SDP problem works like this: During the execution of the algorithm we store at every face corner the corresponding tree node that is currently not dominated by any other node if it exists. Then we start with the initial sequence tree. In each round we grow all possible children for every leaf in the tree and compute the

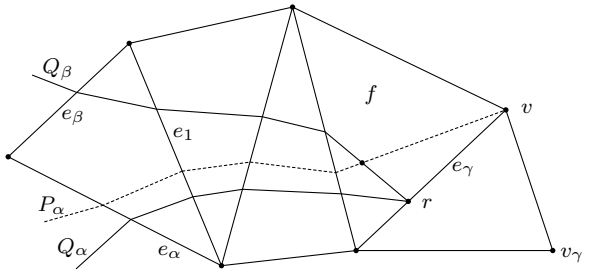


Figure 2.7: The proof of $|Q_\alpha| \leq |Q_\beta|$ uses that the L_∞LSDP Q_β must intersect P_α at some point.

corresponding L_∞LSDPs. When growing a child α we check if there is already a node stored for the current face corner. If that is the case, we find out which of the nodes dominates the other and prune the tree accordingly. Then the record stored at the face corner gets updated if necessary. When the number of levels equals the number of faces in $\mathcal{T}$ we stop growing the tree and return the minimum length of all L_∞LSDPs that terminate in t.

The runtime of the complete L_∞SDP algorithm is dominated by the time needed to compute the lengths of all the L_∞LSDPs. This runtime is, as mentioned earlier, $\mathcal{O}(n^3 \log n)$. Thereby we get the following theorem:

Theorem 1 ([CJ12])**.** *Let s and t be two points on a terrain with n vertices. The L_∞ shortest descending path from s to t can be computed in $\mathcal{O}(n^3 \log n)$ time.*

2.3.2 Approximating the SDP

Before describing the actual approximation algorithm we can already use the bounds that we found to restrict the search for an SDP to a part of the terrain. Since we have $\mathsf{opt}_\infty \leq \mathsf{opt} \leq \sqrt{3}\mathsf{opt}_\infty$, every $(1+\varepsilon)$-approximation of an SDP with $\varepsilon \in (0,1)$ has independently of the concrete value of ε a length of at most $2\mathsf{opt} \leq 2\sqrt{3}\mathsf{opt}_\infty$. Thus we can clip the terrain at the edges of a cube with its center at s and side length $4\sqrt{3}\mathsf{opt}_\infty$. Now every $(1+\varepsilon)$-approximation of an SDP has to lie completely in the clipped terrain. The triangulated resulting terrain is referred to as $\mathcal{T}^*$. Furthermore no edge in an SDP can be longer than $12\mathsf{opt}_\infty$ as this is the length of the diagonal of the cube that includes the whole clipped terrain. These bounds are used to make the runtime independent of the terrain geometry.

To approximate the length of an SDP we use the same approach as before. Since it is difficult to compute an exact LSDP we will approximate the lengths of the LSDPs needed in the algorithm. For this purpose we use an algorithm of Ahmed [Ahm09, Section 3.3.7] that gives us an approximate LSDP with an *additive* error of at most $\frac{|\sigma|\varepsilon\mathsf{opt}_\infty}{\kappa n^2 m(m+1)}$ and a runtime of $\mathcal{O}(|\sigma|^2 \log\left(\frac{n}{\varepsilon}\right))$ where σ is the given edge sequence, m is the number of faces in $\mathcal{T}^*$ and κ is a constant chosen such that fewer than κn^2 nodes are ever created during the construction of the sequence tree.

For pruning Cheng and Jin do not use the actual length of the approximate LSDPs but rather a newly defined notion of the quasi-length of a path. The quasi-length is not only dependent on the actual length of the path but also on the length of its edge sequence.

Define the *quasi-length* of a path P in $\mathcal{T}^*$ as $\|P\| + \frac{\varepsilon \mathsf{opt}_\infty}{m+1}|\sigma_P|$. The *quasi-dominance* between nodes is defined analogously to the dominance for L_∞SDP, the only difference being that the notion of quasi-length is used instead of just the length.

The rest of the algorithm works exactly as before and after the tree stops growing the minimum length of a path from s to t that corresponds to a node in the tree is returned. The proof of correctness of the algorithm will not be made here but can be found in [CJ12]. With the constants in the additive error of the LSDP approximation and the definition of quasi-length, we get a $(1+\varepsilon)$-approximate SDP in $\mathcal{O}\left(n^4 \log\left(\frac{n}{\varepsilon}\right)\right)$ time.

Theorem 2 ([CJ12])**.** *Let s and t be two points on a terrain of n vertices. For any $\varepsilon \in (0,1)$, a $(1+\varepsilon)$-approximate shortest descending path from s to t can be computed in $\mathcal{O}\left(n^4 \log\left(\frac{n}{\varepsilon}\right)\right)$ time.*

3. Approximate Shortest Gentle Paths

In this chapter the adaption of Cheng and Jin's algorithm [CJ12] for the shortest descending path problem to the shortest gentle path problem is described. With this algorithm we can compute an approximate gentle path with a relative error of ε in $\mathcal{O}\left(n^{5.5} \log \frac{n}{\varepsilon}\right)$ time.

As in the paper by Cheng and Jin we find a lower and upper bound for the length of the shortest gentle path. Then we can use these bounds to clip the terrain and for our error assessment. Having found the bounds we first solve the local problem, that is to say we compute the shortest gentle path that has a given edge sequence. Then we iterate over all possible edge sequences with the help of a sequence tree.

3.1 Lower Bound

The idea to finding a lower bound for the SGP is to find a lower bound for the length of a segment of a shortest gentle path and solve the problem using this lower bound instead of the actual segment length. Remember that the exact length of a segment is

$$\|\mathbf{x}\|_s = \max\left(\|\mathbf{x}\|_2, \frac{|x_3|}{\sin(\theta)}\right).$$

Now we define a new norm $\|\cdot\|_b$ as

$$\|\mathbf{x}\|_b := \max\left(\|\mathbf{x}\|_\infty, \frac{|x_3|}{\sin(\theta)}\right) = \max\left(|x_1|, |x_2|, \frac{|x_3|}{\sin(\theta)}\right) \quad \text{for } \mathbf{x} = (x_1\ x_2\ x_3)^T \in \mathbb{R}^3.$$

This is a norm because the following conditions are satisfied for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3,\ \alpha \in \mathbb{R}$:

$$|\alpha| \cdot \|\mathbf{x}\|_b = \|\alpha \cdot \mathbf{x}\|_b$$

$$\|\mathbf{x}\|_b = 0 \Rightarrow \mathbf{x} = 0$$

$$\|\mathbf{x} + \mathbf{y}\|_b \leq \|\mathbf{x}\|_b + \|\mathbf{y}\|_b$$

Obviously $\|\mathbf{x}\|_b \leq \|\mathbf{x}\|_s$ and $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_b$ are true. Furthermore with the definitions of $\|\cdot\|_s$ and $\|\cdot\|_b$ and the inequality $\|\mathbf{x}\|_2 \leq \sqrt{3}\|\mathbf{x}\|_\infty$ we get the following inequalities:

$$\|\mathbf{x}\|_b \leq \|\mathbf{x}\|_s \leq \sqrt{3}\|\mathbf{x}\|_b$$

Thus we found a lower and upper bound for an SGP if we can solve the simple shortest path problem with respect to the newly defined norm $\|\cdot\|_b$. We call this problem the L_b-shortest gentle path (L_bSGP) and the corresponding local problem L_bLSGP.

3.1.1 Local Problem

As in Chapter 2 we start by solving the local problem L_bLSGP, that means we compute a shortest path for a given edge sequence $\sigma = (e_1, \ldots, e_k)$. Let u_j and v_j be the endpoints of e_j for $j \in \{1 \ldots k\}$. Then

$$p_j = \begin{cases} s & \text{if } j = 0 \\ (1-\zeta_j)u_j + \zeta_j v_j & \text{if } j \in \{1 \ldots k\} \\ t & \text{if } j = k+1 \end{cases}$$

with $\zeta_j \in [0,1]$ can be used to describe the endpoints of segments of the path from s to t. With l_j being an upper bound of the segment length the local problem using $\|\cdot\|_b$ can now be expressed as:

$$\min \sum_{j=0}^{k} l_j \quad \text{subject to} \quad \|p_{j+1} - p_j\|_b \leq l_j$$

or in the long form:

$$\min \sum_{j=0}^{k} l_j$$

subject to

$$-l_j \leq \frac{p_{j+1,3} - p_{j,3}}{\sin(\theta)} \leq l_j,$$

$$-l_j \leq p_{j+1,1} - p_{j,1} \leq l_j$$

and

$$-l_j \leq p_{j+1,2} - p_{j,2} \leq l_j.$$

The constraints in the local problem are all linear and thus the exact solution can be computed using linear programming techniques. This can be done in $\mathcal{O}(n^{3.5}L)$ time with an algorithm by Karmarkar [Kar84] where n is the number of variables and L is the number of bits of the input.

The local problem can also be solved using the combinatorial algorithm used by Cheng and Jin [CJ12]. There are some modifications needed to adopt the algorithm for the shortest gentle path problem. These modifications and the original algorithm can be found in Appendix A.

With this algorithm it takes only $\mathcal{O}(n^3 \log n)$ total time to compute the solution to all local problems arising during the construction of the sequence tree.

3.1.2 Sequence Tree

To be able to use the sequence tree approach like Cheng and Jin [CJ12], we have to show that after a constant number of levels the sequence tree represents all required edge sequences. For this we will prove the following lemma.

Lemma 4. *For each start and end point for which an L_bSGP exists there also exists an L_bSGP between them that visits no face more than once.*

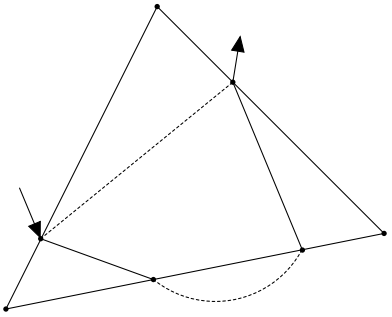


Figure 3.1: Example of how to shortcut the path.

Proof. Assume an L_bSGP visits a face more than once. Then we have a point where the path enters this face first and a point where a path exits the face last (see Figure 3.1). Because of the triangle inequality the b-norm of the direct connection is less than or equal to the b-norm of the subpath of the L_bSGP that lies between these two points. Thus the path we get if we substitute this subpath by the direct connection between the two points is also an L_bSGP and visits the face only once. By repeating this substitution until every face is visited at most once we get an L_bSGP with the desired attributes. □

Now we can stop growing the sequence tree as soon as the number of levels equals the number of faces in the terrain since by then we find at least one L_bSGP for every start and end point between which an L_bSGP exists. As described in Chapter 2 we need to enforce a one-corner one-split property to keep the size of the sequence tree manageable. Again we use the concept of dominance to prune irrelevant parts of the tree.

Definition 1. A node α corresponding to the face corner (f, v) and the edge sequence σ_α *dominates* a node β corresponding to the same face corner (f, v) and the edge sequence σ_β if the L_bLSGP P_α from s to v with edge sequence σ_α is shorter than the L_bLSGP P_β from s to v with edge sequence σ_β (see Figure 3.2). If $P_\alpha = P_\beta$, α dominates β if $|\sigma_\alpha| \leq |\sigma_\beta|$. Ties are broken arbitrarily.

Let α dominate β. Furthermore let e_1 be the first edge in the longest common suffix of σ_α and σ_β and e_α and e_β the predecessors of e_1 in σ_α and σ_β respectively. If e_1 is the first edge in σ_α (resp. σ_β) let e_α (resp. e_β) be s. If e_1, e_α and e_β lie in clockwise order around their common face, α *dominates* β *on the right.* Otherwise α *dominates* β *on the left.*

If a node is dominated on the right (resp. left) by any other node corresponding to the same face corner we will prune its right (resp. left) subtree or not grow the right (resp. left) child in the first place. Now we have to show that we do not prune a relevant subtree, i.e., we do not prune a node or an ancestor of a node that represents a shortest gentle path from s to t. If node α dominates node β on the right (resp. left) we show that α cannot be a descendant of β and show that if we would grow the right child for α and β and would compare the lengths of the L_bLSGPs up to any point on the corresponding edge, the length of the L_bLSGP with edge sequence σ_α would always be at least as short as the length of the L_bLSGP with edge sequence σ_β.

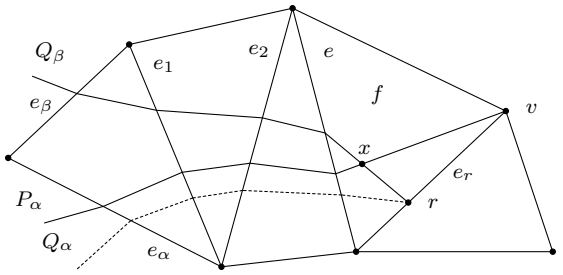


Figure 3.2: Path Q_β must intersect path P_α.

The following Lemma is similar to Lemma 3.1 in [CJ12]. There are only slight modifications needed in the proof due to the differences between the underlying problems.

Lemma 5. *Let α and β be two nodes of the sequence tree with corresponding edge sequences σ_α and σ_β that correspond to the same face corner (f, v) and edge e such that α quasi-dominates β on the right (resp. left). Let e_r be the edge that follows e immediately in anticlockwise (resp. clockwise) order around the boundary of f.*

(i) α is not a descendant of β.

(ii) For every point $r \in e_r$ and every L_bLSGP Q_β with edge sequence σ_β from s to r, the L_bLSGP Q_α with edge sequence σ_α from s to r satisfies $\|Q_\alpha\|_b \leq \|Q_\beta\|_b$; if $\|Q_\alpha\|_b = \|Q_\beta\|_b$, then $|\sigma_\alpha| \leq |\sigma_\beta|$.

Proof. Part (i): Assume that α is a descendant of β. Then σ_β is a prefix of σ_α. This means that the two paths have the same edge sequence until they enter face f. As above let P_α be the L_bLSGP from s to v with edge sequence σ_α and P_β be the L_bLSGP from s to v with edge sequence σ_β. Then P_β terminates in v while P_α exits f through another edge only to come back to f later. This is a contradiction to the assumption that $\|P_\alpha\|_b \leq \|P_\beta\|_b$ since we could shortcut P_α such that it terminates in v directly after it first enters f. Then it would be shorter than P_β which is an L_bLSGP for this edge sequence.

Part (ii): W.l.o.g. assume that α dominates β on the right. Since Q_β enters the face that is bounded by e_α, e_β and e_1 on the left of P_α and exits f on the right of it, it has to cross

P_α somewhere in between at a point called x (see Figure 3.2). If there is more than one intersection in between, let x be the last one along P_α.

Then we have

$$\|Q_\alpha\|_b \leq \|P_{\alpha,[s,x]}\|_b + \|Q_{\beta,[x,r]}\|_b$$

and

$$\|P_\beta\|_b \leq \|Q_{\beta,[s,x]}\|_b + \|P_{\alpha,[x,v]}\|_b$$

added, this gives

$$\|Q_\alpha\|_b + \|P_\beta\|_b \leq \|Q_\beta\|_b + \|P_\alpha\|_b$$

Since α dominates β we have either $\|P_\alpha\|_b < \|P_\beta\|_b$ or $\|P_\alpha\|_b = \|P_\beta\|_b$ and $|\sigma_\alpha| \leq |\sigma_\beta|$. In the first case $\|Q_\alpha\|_b < \|Q_\beta\|_b$ and in the second case $\|Q_\alpha\|_b \leq \|Q_\beta\|_b$ and $|\sigma_\alpha| \leq |\sigma_\beta|$. □

Theorem 3. *Let s and t be two points on a terrain with n vertices. The L_b shortest gentle path from s to t can be computed in $\mathcal{O}(n^3 \log n)$ time.*

Proof. For every leaf that is created in the sequence tree we need to compute the corresponding L_bLSGP. This can be done in $\mathcal{O}(n^3 \log n)$ time for all nodes ever created during the algorithm as we have seen in Chapter 3.1.1. Then we still need to check for dominance on the left or right. For this we store at each face corner the node, if it exists, that is currently not dominated by any other. Now we can check for dominance in $\mathcal{O}(1)$.

To check if a node is dominated on the left or right we have to follow both paths backwards until the paths cross different edges and prune the tree accordingly. As we have seen before the sequence tree has at most as many levels as there are faces in the terrain and the number of faces in the terrain is obviously in $\mathcal{O}(n)$. Thus for all paths that correspond to nodes in the sequence tree the number of edges in the edge sequence is also in $\mathcal{O}(n)$. Then we can also check for dominance on the left or right in $\mathcal{O}(n)$.

Chen and Han [CH90] then proved that for a tree with the one-corner one-split property only $\mathcal{O}(n^2)$ nodes are created during its construction. For all these nodes we have to do the check for dominance.

So altogether the algorithm needs $\mathcal{O}(n \cdot n^2 + n^3 \log n) = \mathcal{O}(n^3 \log n)$ time. Since with the exception of pruning all possible edge sequences are tried during the execution of the algorithm and since we have shown in Lemma 5 that we do not prune a shortest gentle path the algorithm will always find a correct solution in the given running time. □

3.2 Approximation Algorithm

Now that we found the necessary bounds we can start with the original SGP problem. For the L_bSGP opt_b and the SGP opt we have $\mathsf{opt}_b \leq \mathsf{opt} \leq \sqrt{3}\mathsf{opt}_b$. Those are exactly the same bounds for the b-norm as used by Cheng and Jin for the SDP problem and the maximum norm [CJ12]. Thus every argument using these bounds can be done similarly with opt_b instead of opt_∞. In particular we can clip the terrain at the boundary of a cube centered at s with a side length of $4\sqrt{3}\mathsf{opt}_b$.

3.2.1 Local Problem

Similar to Chapter 3.1.1 the local problem can be described as follows:

$$\min \sum_{j=0}^{k} l_j$$

subject to

$$-l_j \leq \frac{p_{j+1,3} - p_{j,3}}{\sin(\theta)} \leq l_j,$$

and

$$\|p_{j+1} - p_j\|_2 \leq l_j$$

with

$$p_j = \begin{cases} s & \text{if } j = 0 \\ (1-\zeta_j)u_j + \zeta_j v_j & \text{if } j \in \{1 \dots k\} \\ t & \text{if } j = k+1 \end{cases}$$

and k denoting the number of edges in the edge sequence. Furthermore u_j, v_j are the endpoints of the edge on which p_j lies and $\zeta_j \in [0, 1]$.

3.2.1.1 Approximation of the LSGP with Second Order Cone Programming

Second order cone programming is a special case of semidefinite programming that is itself a special case of convex programming. All linear and convex quadratic programs can be described as second order cone programs.

Definition 2. Second order cone program (SOCP)
An SOCP is a convex optimization problem of the following form:

$$\begin{array}{ll} & \min f^T x \\ \text{subject to} & \|A_i x + b_i\| \leq c_i^T x + d_i \\ \text{with} & i = 1, \dots, N, \quad x, f, c_i \in \mathbb{R}^n, \quad A_i \in \mathbb{R}^{n_i \times n}, \quad b_i \in \mathbb{R}^{n_i} \quad \text{and} \quad d_i \in \mathbb{R} \end{array}$$

The name derives from the fact that all constraints are given in the form of second order cones. A second order cone is defined as $K = \{(x, y) : \|x\| \leq y\} \quad x \in \mathbb{R}^p,\ y \in \mathbb{R}$.

The LSGP problem can be formulated as an SOCP as seen below.

$$\min f^T x$$

subject to

$$0 \leq \begin{pmatrix} 1 & \frac{u_{j,3}-v_{j,3}}{\sin\theta} & \frac{v_{j+1,3}-u_{j+1,3}}{\sin\theta} \end{pmatrix} \begin{pmatrix} l_j \\ \zeta_j \\ \zeta_{j+1} \end{pmatrix} + \frac{u_{j+1,3} - u_{j,3}}{\sin\theta}$$

$$0 \leq \begin{pmatrix} 1 & -\frac{u_{j,3}-v_{j,3}}{\sin\theta} & -\frac{v_{j+1,3}-u_{j+1,3}}{\sin\theta} \end{pmatrix} \begin{pmatrix} l_j \\ \zeta_j \\ \zeta_{j+1} \end{pmatrix} - \frac{u_{j+1,3} - u_{j,3}}{\sin\theta}$$

$$l_j \geq \left\| \begin{pmatrix} u_j - v_j & v_{j+1} - u_{j+1} \end{pmatrix} \begin{pmatrix} \zeta_j \\ \zeta_{j+1} \end{pmatrix} + u_{j+1} - u_j \right\|_2$$

with

$$f^T = (\underbrace{1 \dots 1}_{\text{k}} \underbrace{0 \dots 0}_{\text{k+1}}), \quad x \in \mathbb{R}^{2k+1}, \quad x^T = (l_0 \dots l_k \; \zeta_1 \dots \zeta_k).$$

The dimensions are

$$n = 2k+1, \quad N = 3(k+1) \quad \text{and} \quad n_i = \begin{cases} 3 & \text{for } i = 2k+3, \dots, 3k+3 \\ 0 & \text{otherwise} \end{cases}.$$

Now the LSGP can be approximated using a general SOCP algorithm. Here we use an algorithm presented by Lobo et al. [LVBL98] that gives a $(1+\delta)$ -approximation for a given SOCP and runs in $\mathcal{O}(k^{3.5} \log \frac{1}{\delta})$ as pointed out by Ahmed and Lubiw [AL09].

The algorithm uses the primal dual potential reduction method which is a specialization of the interior point method. As with interior point methods the algorithm uses a sequence of barrier functions that tend to infinity at the edges of the feasible space. Then the optimization problem can be reduced to optimizing a sequence of functions with no additional constraints by multiplying each barrier function with the objective function.

Cheng and Jin use an additive error bound for the local problem to limit the overall error [CJ12]. The error we get from the SOCP algorithm is relative but it is also possible to limit the additive error by using the bounds we already computed. We have an upper bound for the length of the LSGP and also for the lengths of edges in the terrain. Now we need an upper bound for the length of a segment in the LSGP. This upper bound also depends on the biggest height difference in a triangle of the terrain and the parameter θ.

The biggest difference in height between two points in the terrain is limited by $4\sqrt{3}\mathsf{opt}_b$. Thus the length of a segment is at most

$$\max\left(12\mathsf{opt}_b, \frac{4\sqrt{3}\mathsf{opt}_b}{\sin\theta}\right) = 4\sqrt{3}\mathsf{opt}_b \cdot \max\left(\sqrt{3}, \frac{1}{\sin\theta}\right).$$

So if the length of the face sequence is $|\sigma|$ then the length of the LSGP is limited by

$$4\sqrt{3}|\sigma|\mathsf{opt}_b \cdot \max\left(\sqrt{3}, \frac{1}{\sin\theta}\right).$$

That gives

$$\mathsf{opt} \leq (1+\delta)\mathsf{opt} \leq \mathsf{opt} + \delta \cdot 4\sqrt{3}|\sigma|\mathsf{opt}_b \cdot \max\left(\sqrt{3}, \frac{1}{\sin\theta}\right).$$

So the additive error is limited by

$$\delta \cdot 4\sqrt{3}|\sigma|\mathsf{opt}_b \cdot \max\left(\sqrt{3}, \frac{1}{\sin\theta}\right).$$

Theorem 4. *Let s and t be two points on a terrain with n vertices and σ an edge sequence for a path from s to t. For any $\delta \in (0,1)$ an approximate LSGP with an additive error of at most $4\sqrt{3} \cdot \delta|\sigma|\mathsf{opt}_b \cdot \max\left(\sqrt{3}, \frac{1}{\sin\theta}\right)$ can be computed in $\mathcal{O}\left(|\sigma|^{3.5} \log \frac{1}{\delta}\right)$ time.*

3.2.2 Sequence Tree

Remember that in Lemma 4 we showed that in the case of the b-norm we can stop growing the sequence tree after at most m levels when m is the number of faces in the terrain. Note that the same proof also works if we substitute the Euclidean norm for the b-norm in Lemma 4. Define $\mu = \frac{\mathsf{opt}_b\varepsilon}{\kappa n^2 m(m+1)}$ and $\tilde{\mu} = \frac{\mathsf{opt}_b\varepsilon}{m+1}$ just as in Cheng and Jin do. The constant κ is chosen such that the number of nodes created in the sequence tree is at most κn^2. Analogously to Cheng and Jin we define the notions of quasi-length and quasi-dominance.

Definition 3. The *quasi-length* of a path P with edge sequence σ_P is defined as $\|P\| + |\sigma_P| \cdot \tilde{\mu}$.

A node α corresponding to the face corner (f, v) and the edge sequence σ_α *quasi-dominates* a node β corresponding to the same face corner (f, v) and the edge sequence σ_β if the quasi-length of an approximate LSGP P_α from s to v with edge sequence σ_α and an additive error of at most $|\sigma_\alpha|\mu$ is at most the quasi-length of an approximate LSGP P_β from s to v with edge sequence σ_β and an additive error of at most $|\sigma_\beta|\mu$ (see Figure 3.2). Ties are broken arbitrarily.

Let α quasi-dominate β. Furthermore let e_1 be the first edge in the longest common suffix of σ_α and σ_β and e_α and e_β the predecessors of e_1 in σ_α and σ_β respectively. If e_1 is the first edge in σ_α (resp. σ_β) let e_α (resp. e_β) be s. If e_1, e_α and e_β lie in clockwise order around their common face, *α quasi-dominates β on the right.* Otherwise *α quasi-dominates β on the left.*

The algorithm works exactly as in the b-norm case with the difference that we check for quasi-dominance instead of normal dominance. As in Lemma 5 for the b-norm case we prove that we do not prune important paths using the notion of quasi-dominance. The following Lemma and its proof are again mostly analogous to Cheng and Jin [CJ12].

Lemma 6. *Let α and β be two nodes of the sequence tree with corresponding edge sequences σ_α and σ_β that correspond to the same face corner (f, v) and edge e such that α quasi-dominates β on the right (resp. left). Let e_r be the edge that follows e immediately in anticlockwise (resp. clockwise) order around the boundary of f.*

(i) α is not a descendant of β.

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

Proof. As we have seen in Theorem 4 we can compute approximate LSGPs P_α and P_β from s to v with edge sequences σ_α and σ_β respectively. As the error depends on δ which we can choose freely between 0 and 1 we can find P_α and P_β such that they have an additive error of at most $|\sigma_\alpha|\mu$ and $|\sigma_\beta|\mu$ respectively.

Part (i): Assume that α is a descendant of β. Then σ_β is a prefix of σ_α. This means that the two paths have the same edge sequence until they enter face f. Then P_β terminates in v while P_α exits f through another edge only to come back to f later. We shortcut the path P_α by taking the shortest path directly to v after P_α entered f for the first time. For the resulting path P_γ the edge sequence σ_γ is equal to σ_β. Obviously we have $\|P_\gamma\| < \|P_\alpha\|$ and since α quasi-dominates β also $\|P_\alpha\| + |\sigma_\alpha|\tilde{\mu} \leq \|P_\beta\| + |\sigma_\beta|\tilde{\mu}$. Because $|\sigma_\beta| + 1 \leq |\sigma_\alpha|$ we get

$$\|P_\gamma\| + (|\sigma_\beta| + 1)\tilde{\mu} < \|P_\alpha\| + |\sigma_\alpha|\tilde{\mu} \leq \|P_\beta\| + |\sigma_\beta|\tilde{\mu}.$$

Altogether we have $\|P_\gamma\| + \tilde{\mu} < \|P_\beta\|$. Let l be the length of an LSGP from s to v with edge sequence σ_β. Then we get $\|P_\beta\| \leq l + |\sigma_b|\mu$ and with that $\|P_\gamma\| + \tilde{\mu} - |\sigma_b|\mu < l$. Since $\tilde{\mu} = \kappa n^2 m\mu > m\mu \geq |\sigma_\beta|\mu$ we get $\|P_\gamma\| < l$ which is a contradiction since l is the length of an LSGP with the same start and endpoint and the same edge sequence as P_γ.

Part (ii): W.l.o.g. we assume that α quasi-dominates β on the right. As in Lemma 5 we know that Q_β and P_α intersect in a point called x (see Figure 3.2).

Then we have

$$\|Q_\alpha\| \leq \|P_{\alpha,[s,x]}\| + \|Q_{\beta,[x,r]}\|$$

and

$$\|P_\beta\| \leq \|Q_{\beta,[s,x]}\| + \|P_{\alpha,[x,v]}\| + |\sigma_\beta|\mu$$

added, this gives

$$\|Q_\alpha\| + \|P_\beta\| \leq \|Q_\beta\| + \|P_\alpha\| + |\sigma_\beta|\mu$$

Since α dominates β we have $\|P_\alpha\| \leq \|P_\beta\| + (|\sigma_\beta| - |\sigma_\alpha|)\tilde{\mu}$. By inserting this into the previous inequality we get $\|Q_\alpha\| \leq \|Q_\beta\| + (|\sigma_\beta| - |\sigma_\alpha|)\tilde{\mu} + |\sigma_\beta|\mu$. □

Now the algorithm can construct the sequence tree while enforcing the one-corner one-split property. When the tree has reached a depth of m the algorithm stops growing the tree and returns the shortest approximate LSGP that was computed during the construction of the tree and that terminates in t.

3.2.3 Analysis

Theorem 5. *Let s and t be two points on a terrain with n vertices. For any $\varepsilon \in (0, 1)$ a $(1+\varepsilon)$-approximate shortest gentle path from s to t can be computed in $\mathcal{O}\left(n^{5.5} \log \frac{n}{\varepsilon}\right)$ time.*

Proof. With Theorem 3 we know that we can compute the length opt_b of an L$_b$SGP in $\mathcal{O}(n^3 \log n)$ time. An approximate LSGP can be computed in $\mathcal{O}\left(|\sigma|^{3.5} \log \frac{1}{\delta}\right)$ as we have seen in Theorem 4. We have $|\sigma| \leq n$ for all LSGPs that are computed during the execution of the algorithm because the sequence tree has only n levels and thus the edge sequence cannot be longer. Thus the runtime is also in $\mathcal{O}\left(n^{3.5} \log \frac{1}{\delta}\right)$.

The additive error in the approximation of the length of an LSGP has to be at most $|\sigma|\mu$ for the analysis to work analogously to [CJ12]. Thus we have

$$|\sigma|\mu = \delta|\sigma|\mathsf{opt}_b 4\sqrt{3}\max\left(\sqrt{3}, \frac{1}{\sin\theta}\right)$$

$$\Leftrightarrow \quad \frac{|\sigma|\mathsf{opt}_b\varepsilon}{\kappa n^2 m(m+1)} = \delta|\sigma|\mathsf{opt}_b 4\sqrt{3}\max\left(\sqrt{3}, \frac{1}{\sin\theta}\right)$$

$$\Leftrightarrow \quad \delta = \frac{\varepsilon}{4\sqrt{3}\max\left(\sqrt{3}, \frac{1}{\sin\theta}\right)\kappa n^2 m(m+1)}\,.$$

Then for each LSGP the runtime is

$$\mathcal{O}\left(n^{3.5}\log\frac{1}{\delta}\right) = \mathcal{O}\left(n^{3.5}\log\left(\frac{4\sqrt{3}\max(\sqrt{3}, \frac{1}{\sin\theta})\kappa n^2 m(m+1)}{\varepsilon}\right)\right) = \mathcal{O}\left(n^{3.5}\log\frac{n}{\varepsilon}\right).$$

With the one-corner one-split property only $\mathcal{O}(n^2)$ nodes are ever created during the construction of the sequence tree. For each of these nodes the associated LSGP must be approximated. Since the construction and pruning of the sequence tree can be done in $\mathcal{O}(n^3)$ time just as in the b-norm case we get a total runtime of $\mathcal{O}\left(n^{5.5}\log\frac{n}{\varepsilon}\right)$.

The proof of correctness is exactly the same as in [CJ12] as we used the same values for μ and $\tilde{\mu}$. The original proof can be found in Appendix B. □

4. Towards an Exact Algorithm

To solve the shortest gentle path problem exactly, we only need to be able to compute the exact locally shortest gentle path. With the possibility to compute LSGPs we can use the sequence tree approach as before to find an exact SGP. Thus this chapter will focus only on the problem to find a shortest gentle path with a given edge sequence.

The basic concept is to trace a path from the origin to the destination by characterizing the bend angles at each edge of the edge sequence. For the shortest descending path problem such an approach was implemented by Ahmed [Ahm09]. We try to use the same idea for the shortest gentle path problem and find that there are indeed restrictions on the behavior of shortest gentle paths at terrain edges but also that one segment does not always determine a unique direction for the next one. Instead for one segment there might be a whole interval of directions the next segment could have.

We use this knowledge about shortest gentle paths to describe an algorithm that traces a shortest gentle path from a point s to a point t on the terrain through a given sequence of faces. The approach is to look at each edge in the sequence successively and compute for every point on the current edge the locally shortest gentle path to it. For each edge we save in the function l the lengths of the LSGPs from s to every point on this edge. This length function is continuous and piecewise differentiable. To find the length function for the next edge we use the knowledge we gained about the possible bend angles. We handle each interval on the first edge for which l is differentiable individually. For each point in each of these intervals we look at the direction of the path segment that terminates in that point and expand this path to the current edge, such that the two segments together are an LSGP (see Figure 4.1 for an example). As before we always use the simplified path and compute the length with $\|\cdot\|_s$. Being given a simplified path we can always construct a gentle path with the same length. If we get several LSGPs for one point on the second edge we have to compare the lengths of the LSGPs and save only the shortest one. That means that we have to take the lower envelope of all the length functions we get for the different intervals on the first edge. This gives us the length function for the second edge. The same procedure is repeated for all edges the path has to cross. To find the shortest gentle path from s to t we then just have to add the length of a shortest gentle path from each point on the last edge to t to the computed length function of the last edge and find the minimum of the resulting function.

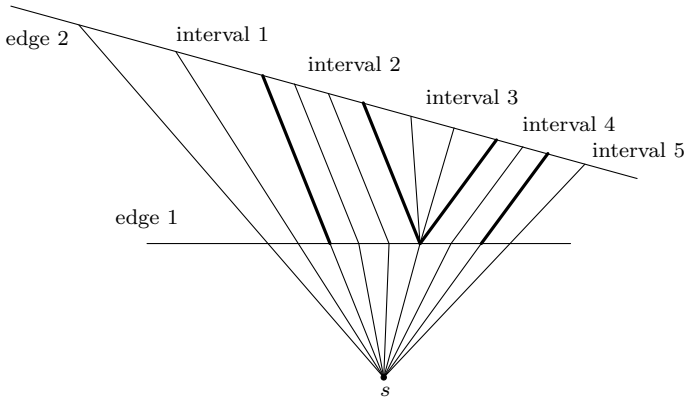


Figure 4.1: Example paths from s to the second edge. In this case we have only one interval on the first edge and five intervals on the second edge (bounded by thick drawn paths). Over each of these intervals the length function is differentiable.

In most of the cases of different intervals that arise during the execution of the algorithm it is easily possible to trace the paths to the next edge. Just in one case we cannot compute the length function for the next edge from the length function we have for the interval on the previous edge (see Case 5 in Chapter 4.3). Here we have to use an approximation but otherwise all functions can be computed exactly. Unfortunately this algorithm has an exponential running time in the worst case.

4.1 Preliminaries

First we show that LSGPs from the same origin need not cross.

Lemma 7. *Let s be a vertex of the terrain and σ an edge sequence. For any two points c and d on the final edge we can find LSGPs with the edge sequence σ to c and d respectively that do not cross.*

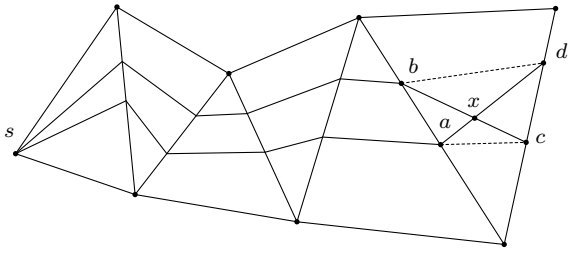


Figure 4.2: Two intersecting paths with the same edge sequence.

Proof. First we assume that we have found two LSGPs with the same edge sequence that do cross. Then we consider the first crossing point x on the paths seen from s. Let the entry and exit points on the edges of the face in which x lies be labeled a to d as seen in the picture. We can assure that the paths cross exactly in x even if the direct path between a

and d or b and c is too steep for a gentle path. Then with the triangle inequality we get:

$$\|x - a\|_s + \|c - x\|_s \geq \|c - a\|_s$$

and

$$\|x - b\|_s + \|d - x\|_s \geq \|d - b\|_s$$

Adding these two inequalities gives

$$\|c - b\|_s + \|d - a\|_s \geq \|c - a\|_s + \|d - b\|_s.$$

Thus since our paths are LSGPs the paths $s \to a \to c$ and $s \to b \to d$ must also be LSGPs. For further crossings we do exactly the same and thus get two LSGPs that do not cross. □

Lemma 8. *Let s be a vertex of the terrain and σ an edge sequence. Then there exists an LSGP to every inner point of the final edge in the edge sequence iff there is at least one possible initial direction for a gentle path starting in s.*

Proof. If there is a gentle path with the given edge sequence, then an LSGP with this edge sequence exists also. With Lemma 2 we can always find a gentle path for a given edge sequence if there is a valid initial direction. □

Lemma 9. *Let s be a vertex of the terrain and σ an edge sequence. Further let P denote an LSGP from s to a point x on an edge $\overline{uv}$ of the edge sequence and a and b two inner points on the next edge in the edge sequence $\overline{vw}$. If P can be extended to a and b then it can also be extended to any point on the line segment $\overline{ab}$.*

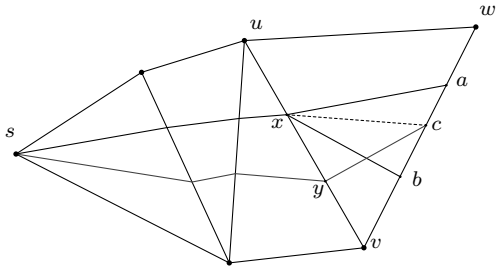


Figure 4.3: The set of points on $\overline{vw}$ to which p can be extended forms an interval.

Proof. Let c be a point on the line segment $\overline{ab}$. Then there is with Lemma 8 at least one LSGP from s to c with the edge sequence σ. Let the intersection of this path with $\overline{uv}$ be called y (see Figure 4.3). If $x \neq y$, y has to be to the left or to the right of x. W.l.o.g. assume that y lies to the right side of x. Then we can use Lemma 7 to show that the path $s \to x \to c$ must also be an LSGP. This argument is independent on the exact position of c and thus we get that p can be extended to every point on $\overline{ab}$. □

Thus the points to which one LSGP can be extended form one interval on the next edge.

Let σ be an edge sequence of length k with the edges $\overline{u_i v_i}$ for $i \in \{1 \dots k\}$. The function $g(\zeta_1, \dots, \zeta_k) = \|p_1 - s\|_s + \sum_{i=1}^{k-1} \|p_{i+1} - p_i\|_s$ with $p_i = \zeta_i u_i + (1 - \zeta_i) v_i$ and $\zeta_i \in [0, 1]$ describes the length of all gentle paths that obey the edge sequence σ. The function $l(\zeta_k) = \inf_{\zeta \in C} g(\zeta, \zeta_k)$ with $\zeta = (\zeta_1, \dots, \zeta_{k-1})$ and $C = [0, 1]^{k-1}$ describes the length of an LSGP to each point on the last edge $\overline{u_k v_k}$.

Lemma 10. *The function $l(\zeta_k)$ is convex.*

Proof. The function g is convex since $|x|$ and $\|x\|_2$ are convex and sums and maxima of convex functions are also convex. We have also given that if a function $g(x, y)$ is convex in (x, y), then the function $f(x) = \inf_{y \in C} g(x, y)$ is also convex iff C is a convex set. Since $C = [0, 1]^{k-1}$ is a convex set, $l(\zeta_k) = \inf_{\zeta \in C} g(\zeta, \zeta_k)$ is also convex. □

Lemma 8 showed that if there is a possible initial direction for a gentle path the length function l is defined for all $\zeta_k \in [0, 1]$.

4.2 Path Tracing from One Point

Given the last segment of a traced path, our goal is now to extend this path to the next edge in the edge sequence. This is done such that the last and current segment together form an LSGP.

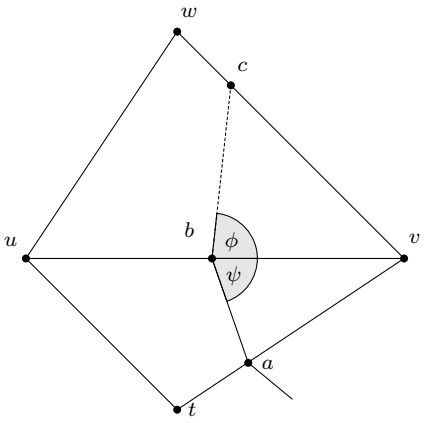


Figure 4.4: The path to b is extended to the edge $\overline{vw}$.

Given are two neighboring triangles of the terrain with vertices t, u, v, w as seen in Figure 4.4. The path enters the first triangle at the edge $\overline{tv}$ and transfers to the second triangle across edge $\overline{uv}$. This path segment is given. Let $\overline{vw}$ be the next edge in the edge sequence. The last given segment of the traced path has nodes a on $\overline{tv}$ and b on $\overline{uv}$. The angles $\angle abv$ and $\angle vbc$ will be called ψ and ϕ. Now we try to find all points c on $\overline{vw}$ such that the path from a to c via b is an LSGP.

First we determine the intervals on $\overline{vw}$ for which the direct path would be too steep. These intervals will be called steep. The rest of the line segment can then be reached by direct gentle paths and will be called flat. The intervals are found by computing the points c

on $\overline{vw}$ for which the line segment $\overline{bc}$ has slope θ (see Figure 4.5). To do this we solve the equations $\sin\theta = \frac{|b_3-c_3|}{\|c-b\|_2}$ and $c = v + s \cdot \overrightarrow{vw}$ with $s \in [0,1]$ for c which results in a quadratic equation. Thus we can get up to two such points c and with that up to three intervals on $\overline{vw}$. By testing a point in one interval we can determine which intervals are the steep ones and which are flat.

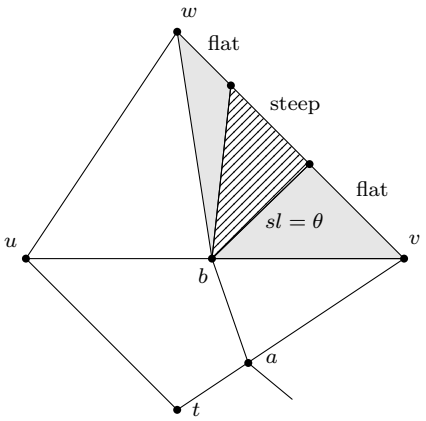


Figure 4.5: Example for steep and flat intervals.

The problem can now be described as follows:

Problem 1. Given are a terrain $\mathcal{T}$ and in the terrain two adjacent faces with the vertices t, u, v and u, v, w. Furthermore a point a on $\overline{tv}$ and a point b on $\overline{uv}$ that is parametrized by $b = \zeta u + (1-\zeta)v$ with $\zeta \in [0,1]$.

The task is to find all points c on $\overline{vw}$ such that the shortest gentle path from a to c via b is also a shortest gentle path from a to c.

For each such c we know that $\|b-a\|_s + \|c-b\|_s = \min_{x \in \overline{uv}} \|x-a\|_s + \|c-x\|_s$. Define $l(\lambda, \xi) = \|x-a\|_s + \|c-x\|_s$ with $x = \lambda u + (1-\lambda)v$ and $c = \xi w + (1-\xi)v$. Now we want to find all possible values for ξ such that $l(\zeta, \xi) = \min_{\lambda \in (0,1)} l(\lambda, \xi)$. To do that we set the derivative at ζ equal to zero: $\frac{dl}{d\lambda}(\zeta, \xi) = 0$. There are only a few cases for which the derivative is not defined. These cases will be handled individually; They are: $sl(\overline{ab}) = \theta$, $sl(\overline{xc}) = \theta$ or $\zeta \in \{0,1\}$ which means that $b = u$ or $b = v$.

In the following we will consider several cases. In Cases 1 to 3 we will assume that b lies in the interior of $\overline{uv}$. The case that $b = u$ or $b = v$ will be handled separately. We divide the cases by the gradient of the segment $\overline{ab}$. We consider $sl(\overline{ab}) > \theta$, $sl(\overline{ab}) < \theta$ and $sl(\overline{ab}) = \theta$.

If $sl(\overline{ab}) \neq \theta$, the derivative $\frac{dl}{d\lambda}(\zeta, \xi)$ is not defined if $\overline{bc}$ has gradient θ. Otherwise it is defined and we can use it to get candidates for c. Here we differentiate again between several cases depending on the gradient of the segment $\overline{bc}$. We solve the equation for each case separately and thus get possible values for ξ for each of the cases. Now we still have to check for all of the candidates if the case we computed them for occurs e.g. we test if the candidate we computed in the flat case also lies in a flat interval. If we found no candidates we still have to test if we get a minimum for ξ with $sl(\overline{bc}) = \theta$. This test is done by evaluating l for one point on each side of b on $\overline{uv}$. Since l is convex we know that if l is larger for the two points than for b, we have a minimum at b and thus ξ is a candidate.

Case 1: Line Segment $\overline{ab}$ is Steep – $sl(\overline{ab}) > \theta$

In this case $l(\lambda, \xi) = \frac{|x_3 - a_3|}{\sin\theta} + \max(\|x - c\|_2, \frac{|x_3 - c_3|}{\sin\theta})$ if λ lies in a sufficiently small neighborhood of ζ. Now we want to find all ξ such that $\frac{dl}{d\lambda}(\zeta, \xi) = 0$. To calculate the derivative we look at two different cases again.

Case 1.1: Line Segment $\overline{bc}$ is Steep – $l = \frac{|x_3 - a_3|}{\sin\theta} + \frac{|x_3 - c_3|}{\sin\theta}$

(a) $a_3 < x_3 < c_3$ **or** $a_3 > x_3 > c_3$ We have

$$\begin{aligned} & l(\lambda, \xi) = \frac{1}{\sin\theta}(-a_3 + \xi w_3 + (1 - \xi)v_3) \\ \Rightarrow \qquad & \frac{dl}{d\lambda} = 0 \end{aligned}$$

Thus in this case all possible values for c are candidates.

(b) $x_3 > a_3, c_3$ **or** $x_3 < a_3, c_3$ We have

$$\begin{aligned} & \frac{dl}{d\lambda} = \pm\frac{2(u_3 - v_3)}{\sin\theta} \\ \Rightarrow \qquad & \frac{dl}{d\lambda} = 0 \;\Leftrightarrow\; u_3 = v_3 \end{aligned}$$

This means that in this case there are no candidates except when $u_3 = v_3$. Then all possible values for c are candidates.

Case 1.2: Line Segment $\overline{bc}$ is Flat – $l = \frac{|x_3 - a_3|}{\sin\theta} + \|x - c\|_2$

We have

$$\begin{aligned} \frac{dl}{d\lambda} &= \pm\frac{u_3 - v_3}{\sin\theta} + \frac{1}{\|x - c\|_2} \cdot \sum_{i=1}^{3}(x_i - c_i)(u_i - v_i) \\ &= \pm\frac{u_3 - v_3}{\sin\theta} + \frac{< x - c, u - v >}{\|x - c\|_2} \\ &= \pm\frac{u_3 - v_3}{\sin\theta} + \|u - v\|_2 \cdot \cos\phi \end{aligned}$$

Therefore

$$\frac{dl}{d\lambda} = 0 \quad \Rightarrow \quad \cos\phi = \pm\frac{u_3 - v_3}{\sin\theta \cdot \|u - v\|_2}$$

The "±" in this case is "+" if $x_3 > a_3$ i.e. segment $\overline{ax}$ is ascending and "−" if $x_3 < a_3$ i.e. segment $\overline{ax}$ is descending. Thus there is at most one candidate in this case which only depends on u and v and is completely independent of the location of a. Note that we only

get a candidate if $\overline{uv}$ is gentle as otherwise the absolute value of the fraction in the last equation would be greater than 1.

Case 2: Line Segment $\overline{ab}$ is Flat – $sl(\overline{ab}) < \theta$

In this case we have $l(\lambda, \xi) = \|x - a\|_2 + \max(\|x - c\|_2, \frac{|x_3 - c_3|}{\sin\theta})$ if λ lies in a sufficiently small neighborhood of ζ. And again we have two cases depending on the maximum:

Case 2.1: Line Segment $\overline{bc}$ is Steep – $l = \|x - a\|_2 + \frac{|x_3 - c_3|}{\sin\theta}$

We have

$$\begin{aligned}\frac{dl}{d\lambda} &= \frac{1}{\|x-a\|_2} \cdot \sum_{i=1}^{3}(x_i - a_i)(u_i - v_i) \pm \frac{u_3 - v_3}{\sin\theta} \\ &= \frac{< x - a, u - v >}{\|x-a\|_2} \pm \frac{u_3 - v_3}{\sin\theta} \\ &= \|u - v\|_2 \cdot \cos\psi \pm \frac{u_3 - v_3}{\sin\theta}\end{aligned}$$

Therefore

$$\frac{dl}{d\lambda} = 0 \qquad \Rightarrow \qquad \cos\psi = \pm\frac{u_3 - v_3}{\sin\theta \cdot \|u - v\|_2}$$

Thus there are no candidates in this case, except the incoming angle ψ has the exact value given above. In this case all values for c are candidates.

Case 2.2: Line Segment $\overline{bc}$ is Flat – $l = \|x - a\|_2 + \|x - c\|_2$

We have

$$\begin{aligned}\frac{dl}{d\lambda} &= \frac{1}{\|x-a\|_2} \cdot \sum_{i=1}^{3}(x_i - a_i)(u_i - v_i) + \frac{1}{\|x-c\|_2} \cdot \sum_{i=1}^{3}(x_i - c_i)(u_i - v_i) \\ &= \frac{< x - a, u - v >}{\|x-a\|_2} + \frac{< x - c, u - v >}{\|x-c\|_2} \\ &= \|u - v\|_2 \cdot \cos\psi + \|u - v\|_2 \cdot \cos\phi \\ &= \|u - v\|_2 \cdot (\cos\psi + \cos\phi)\end{aligned}$$

Therefore

$$\frac{dl}{d\lambda} = 0 \qquad \Rightarrow \qquad \cos\psi = -\cos\phi = \cos(180^\circ - \phi) \qquad \Rightarrow \qquad \psi = 180^\circ - \phi$$

The last deductions are true because $u \neq v$ and $\psi, \phi \in [0°, 180°]$. So in this case there is at most one candidate and for this candidate the incoming and outgoing angle ψ and ϕ at b add to 180°. That means the path unfolds to a straight line at b.

Case 3: Line Segment $\overline{ab}$ Has Slope θ – $sl(\overline{ab}) = \theta$

In this case the derivative is not defined. But we can still compute the derivative from the left and right side. Point c is a solution iff one of these derivatives is positive and the other negative or one of the derivatives is zero.

Case 4: Point b is a Vertex – $b \in u, v$

In this case we only have to check if the derivative of the length function is positive or negative. If $b = u$ all ξ for which $\frac{dl}{d\lambda}(\zeta, \xi) \geq 0$ are candidates; if $b = v$ all ξ for which $\frac{dl}{d\lambda}(\zeta, \xi) \leq 0$ are candidates.

4.2.1 Approach for Tracing Paths from One Point

First the flat and steep intervals are determined. After that, depending on the slope of the segment $\overline{ab}$, the candidates for point c are determined. Then the candidates that do not lie in the right segment are discarded. If there are no candidates left we test the case that the second segment has slope θ. The candidates that are left now are valid and they form an interval as shown in Lemma 9. Note that there might be no candidates at all.

In the case that we have more than one candidate left, the length of the corresponding paths can be given as a function l of the position of c within $\overline{vw}$.

In cases 1 and 2 we only get more than one candidate if $\overline{bc}$ is steep. If the endpoints of the interval are $\xi_1 w + (1 - \xi_1)v$ and $\xi_2 w + (1 - \xi_2)v$ with $\xi_1 \leq \xi_2$ the length function in these cases is

$$l(\xi) = \|b - a\|_s + \frac{|\xi w_3 + (1 - \xi)v_3 - b_3|}{\sin\theta} \qquad \text{for } \xi_1 \leq \xi \leq \xi_2$$

which is a linear function.

In the cases 3 and 4 the length function can be more complex since we can get candidates for flat and steep second segments. Thus the length function in this case is

$$l(\xi) = \|b - a\|_s + \inf\left\{\|\xi w + (1 - \xi)v - b\|_s, \frac{|\xi w_3 + (1 - \xi)v_3 - b_3|}{\sin\theta}\right\} \qquad \text{for } \xi_1 \leq \xi \leq \xi_2.$$

4.3 Path Tracing for an Interval

Now that we know how to trace one path we want to generalize this idea and trace all paths at the same time. We do this edge by edge in the sequence. So we start with the first edge in the sequence and compute the function of the length of an LSGP for all points x on this edge. For the first edge this is $\|x - s\|_s$. Now we trace LSGPs for every point x

to the next edge. Since we cannot actually do this for every point separately we split the first edge into several intervals and trace all paths for these whole intervals. The length function l is piecewise differentiable. We divide the first edge into intervals over which l is differentiable and points for which l is not differentiable. Then we trace the paths for each of the intervals and compute the corresponding length functions for the next edge. Afterwards we have to find the infimum of all the length functions we computed. The resulting function is the length function for the next edge. Now we can continue with the next edge until we reach the last one. Then we just have to compute the length of every point on the last edge to the destination t and add this to the length function of this point. Then we compute the minimum and get the length of the LSGP from s to t. Now we consider how to trace the different intervals from one edge $e_1 = \overline{uv}$ to the next edge in the edge sequence $e_2 = \overline{vw}$. The edge before e_1 in the edge sequence will be called e_0. If e_1 is the first edge we will call s e_0. As before we consider different cases depending on the last segments.

Case 1: The Segments Between e_0 and e_1 are Steep

The length function in this case is always linear. This is true because the steep segments always either originate in one common point or the segments before were also steep. Thus we have one common point from which on all segments are steep. Then the lengths of the paths only depends on the overall height difference from the common origin to the last edge and this is linear.

Case 1.1: Steep Extension

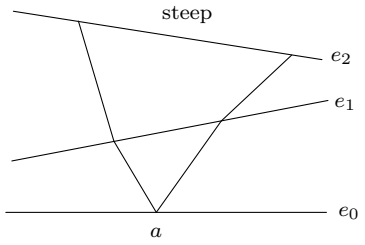


Figure 4.6: Path tracing for an interval of steep paths if the second face is also steep. The border paths have all slope θ.

As in Case 1.1 in Chapter 4.2 all points on e_2 that can be reached with a steep path are candidates if the path ascends or descends for both segments or if $u_3 = v_3$ (see Figure 4.6). Otherwise there are no candidates. The length function over the interval on e_2 is still linear and can be computed easily.

Case 1.2: Flat Extension

As in Case 1.2 in Chapter 4.2 there is at most one candidate for every point in the interval on e_1 and the outgoing angle is fixed and only depending on u and v. This means that we get a family of parallel lines and thus an interval of candidates on e_2 (see Figure 4.7)

or no candidates at all. Since the segments are parallel their length function is linear over the interval and thus the whole length function is also linear and can be computed easily.

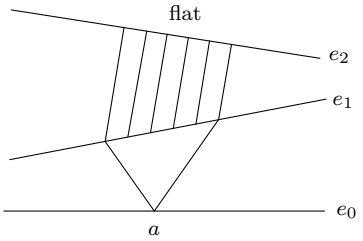


Figure 4.7: Path tracing for an interval of steep paths for Cases 1.2 and 1.3.

Case 1.3: Extension with Slope θ

If we have a steep extension all possible extensions with slope θ are already included in this case. Thus we only have to consider this case if one of the segments ascends and the other one descends and $u_3 \neq v_3$. Otherwise, if we have no steep extension it means that there are no steep paths at all and thus also no paths with slope θ. In the relevant cases we just add extensions with slope θ for all points on e_2 that have not been reached by an extension yet to the list of candidates. The length function in these cases of parallel extensions with slope θ is also linear and can be computed easily.

Case 2: The Segments Between e_0 and e_1 are Flat and Their Supporting Lines Meet in One Point

This case occurs if the paths originate in the same point y on one of the previous edges in the sequence and all segments from then on are also flat and thus with Case 2.2 in Chapter 4.2 the incoming and outgoing angles at every edge add to 180°. This means that if we unfold the terrain along these paths the paths unfold to straight lines that meet in one point. Thus the length function in this case is $k + \|b - x\|_2$ for a constant length k, a fixed point x and b on e_1. Look at Figure 4.8 for an example in which we can see all the different cases that are described below.

Case 2.1: Steep Extension

With Case 2.1 in Chapter 4.2 we know that we can extend a path only for one specific incoming angle. Since the incoming angles are different for all segments we can have at most one path that can be extended. In this case all possible steep extensions are candidates. The length function over the interval on e_2 is linear since the segments between e_1 and e_2 all originate in one point and all segments are steep.

Case 2.2: Flat Extension

Again we look at the corresponding case for the tracing of one path. In Case 2.2 in Chapter 4.2 the outgoing angle of the extension must add to 180° with the incoming angle of the

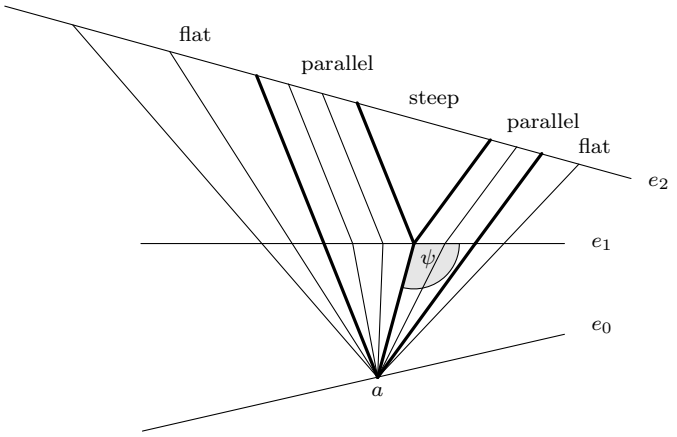


Figure 4.8: Path tracing for an interval of flat paths whose supporting lines meet in one point. The parallel paths in this figure all have slope θ. The thick paths mark the borders between the different intervals.

last segment. That means that for the whole interval on e_1 we get flat extensions such that if we unfold the terrain at the current edge the extensions with their last segments form straight lines. This means that for the edge e_2 we will get an interval of Case 2 again. If we unfold the terrain along the traced paths we get straight lines that originate in one common point. Thus the length function can be computed easily by finding the intersection z of the straight lines that contain the segments between e_1 and e_2. Then the length function is the sum of $\|c - z\|_2$ and the path length from s to y.

Case 2.3: Extension with Slope θ

If there are still points c on the edge e_2 for which we could not find a path in the previous two cases, we have to check the possibility that a path could be extended from a point b on e_1 to c with a segment that has slope θ. Note that these extensions cannot cross any of the other extensions we already traced. This is true because the paths are LSGPs from their last common point y onward and with Lemma 7 we would get a second LSGP from y through b to a point on the next edge. Since we did not find this path in the first two cases this is a contradiction. For all points b that can be extended in this way we know that the incoming angle is not the angle specified in Case 2.1 and if we consider the extension such that incoming angle equals outgoing angle this path would be steep. Thus in this case we have up to two intervals on e_1 that can be extended in this way (see Figure 4.8). The length function in this case is the length function for e_1 plus the linear function for the extension. Thus the whole length function is the sum of a constant value, a linear function and $\|b - x\|_2$ for a fixed point x and b on e_1.

Case 3: The Segments Between e_0 and e_1 are Flat and Parallel

This case only occurs if Case 1.2 occurred for the last interval. Thus the length function is linear as described there.

Case 3.1: Steep Extension

Again by looking at Case 2.1 in Chapter 4.2 we see that we can extend each segment only if the incoming angle equals a special value that depends only on u and v (see Figure 4.9). Thus if the segments have exactly this incoming angle, all possible steep extensions are candidates, otherwise there are no candidates at all. If there are candidates the length function for the interval on e_2 is obviously also linear.

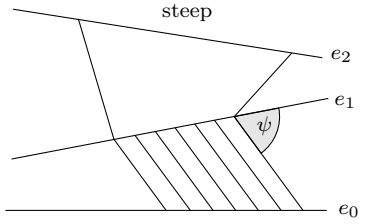


Figure 4.9: Example for a steep extension of an interval of flat parallel paths.

Case 3.2: Flat Extension

Case 2.2 in Chapter 4.2 shows that in this case the incoming and outgoing angle at e_1 must be equal (see Figure 4.10). Thus either the candidates are flat parallel lines or if the extensions would be steep, we have no candidates at all. In the former case the length function is still linear and can be computed easily.

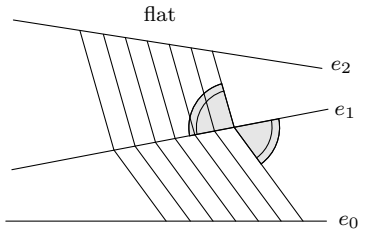


Figure 4.10: Example for a flat extension of an interval of flat parallel paths. The marked angles must be equal such that if the two faces are unfolded into one plane, the path is a line segment. The Figure can also be used as an example for Case 3.3. Now the angles need not be equal, just the extension paths must all have slope θ.

Case 3.3: Extension with Slope θ

If we have found steep extensions, all possible paths with slope θ are also included in this case. Otherwise we add if possible extensions with slope θ for all points c on e_2 to which we have found no extension yet to our candidates. The length function for a ray of parallel extensions with slope θ is also linear.

Case 4: There is Only One Last Segment and It Has Slope θ

This case can be handled just as Case 3 in Chapter 4.2. There might be up to two points on e_2 for which the length function is not differentiable. The length function for each of the intervals in between these points is either linear if it is a steep interval or $k + \|c - b\|_2$ with k denoting the length of the path up to e_1, b the node on e_1 and c a point in the interval of e_2. If the interval is flat we get the latter length function.

Case 5: The Segments Between e_0 and e_1 are Parallel and Have Slope θ

In the previous case we had just one segment with slope θ. Then we could just compute the right and left derivative and check if one is positive or zero and the other is negative or zero. In this case we try to do the same for all parallel segments. We get an interval of possible extensions for each segment. Now we have to compare the path lengths corresponding to these infinitely many intervals to get the shortest ones. In the case that the extended segment is steep it is easy to compare these lengths but if the extended segment is flat we need to use differentiation to find the shortest paths and this is only possible if the length function of the paths up to $e_0 = \overline{uv}$ is linear. Otherwise we cannot find the exact best path for each point on $e_1 = \overline{vw}$. This case occurs only if the length function up to $\overline{uv}$ is not linear which is the case if previously in the path there were segments that were flat and whose supporting lines met in one point.

Case 5.1: Segment $\overline{bc}$ is Steep

Without loss of generality we assume that segment $\overline{ab}$ is ascending. We get two derivatives one from the left and one from the right. In one we consider $\overline{ab}$ to be steep and in the other one we consider it to be flat. The derivatives are

$$\|u - v\|_2 \cdot \cos\psi \pm \frac{u_3 - v_3}{\sin\theta} \quad \text{and} \quad 0 \quad \text{or} \quad \frac{2(u_3 - v_3)}{\sin\theta}.$$

If $\overline{bc}$ is ascending we have "+" in the first derivative and 0 in the second, if $\overline{bc}$ is descending we have "−" in the first derivative and $\frac{2(u_3-v_3)}{\sin\theta}$ in the second. Now we have to review two cases: First one of the derivatives could be equal to 0 and second one of the derivatives is positive and the other negative.

If $\overline{bc}$ is ascending the second derivative is always 0. Otherwise $\frac{2(u_3-v_3)}{\sin\theta} = 0$ iff $u_3 = v_3$ and $\|u - v\|_2 \cdot \cos\psi \pm \frac{u_3-v_3}{\sin\theta} = 0$ for exactly one value of the incoming angle ψ. If one of these cases occurs it occurs for all incoming segments and independently of the outgoing angle and thus all steep extensions are candidates.

In the second case one of the derivatives has to be positive and the other one negative. Since the derivatives cannot be 0 the segment $\overline{bc}$ must be descending. We assume that the second derivative $\frac{2(u_3-v_3)}{\sin\theta}$ is positive. Then:

$$\|u - v\|_2 \cdot \cos\psi - \frac{u_3 - v_3}{\sin\theta} < 0 \quad \Leftrightarrow \quad \cos\psi < \frac{u_3 - v_3}{\|u - v\|_2 \sin\theta}$$

If the incoming angle ψ fulfills this inequality all steep paths for all points b are candidates. If $\frac{2(u_3-v_3)}{\sin\theta}$ is negative the results are similar.

In each of the cases above we get infinitely many overlapping intervals. Now we still have to decide which of the many possible paths to each point c on $\overline{vw}$ is the shortest. If the length function is linear it is possible that the paths are all equally long for one specific gradient. Otherwise the shortest paths are the whole interval for $b_{min} = \min_b \left\{ l(b) + \frac{|c_3 - b_3|}{\sin\theta} \right\}$ and for all other b a path with gradient θ parallel to the boundary of the whole interval (see Figure 4.11). Thus for steep extensions we get up to three intervals and the length functions for the extensions are linear.

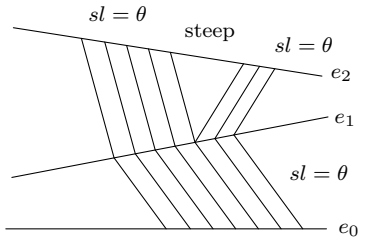


Figure 4.11: Example for a steep extension of an interval of parallel paths with slope θ.

Case 5.2: Segment $\overline{bc}$ is Flat

In this case the two derivatives (left and right) are

$$\|u - v\|_2 \cdot (\cos\psi + \cos\phi) \quad \text{and} \quad \|u - v\|_2 \cdot \cos\phi + \frac{u_3 - v_3}{\sin\theta}.$$

First we consider the case that one derivative is equal to 0. If $\|u-v\|_2 \cdot (\cos\psi + \cos\phi) = 0$ we get that $\psi = 180° - \phi$. Thus we always get an interval of parallel segments as candidates; we only have to verify that the segments are indeed flat. If $\|u - v\|_2 \cdot \cos\phi + \frac{u_3 - v_3}{\sin\theta} = 0$ we get $\cos\phi = \frac{u_3 - v_3}{\sin\theta \|u - v\|_2}$. In this case we either get an interval of parallel segments with outgoing angle ϕ or no candidates at all, if the equation cannot be fulfilled.

If none of the derivatives is equal to zero one has to be positive and the other negative. Then $\|u - v\|_2 \cdot \cos\phi$ must lie between $-\frac{u_3 - v_3}{\sin\theta}$ and $-\|u - v\|_2 \cdot \cos\psi$. Thus there is an interval of possible outgoing angles ϕ that depends only on u, v and ψ and hence is equal for all points b. Note that the boundary values of this interval correspond to the cases with one derivative equal to zero. To use only valid candidates we now intersect the interval with the interval of angles ϕ that give flat segments and call the resulting interval I. To get the best path for every point c on $\overline{vw}$ we examine the complete path lengths. For a fixed point c, the path ending there has length $f_c(\lambda) = \|c - \lambda u + (1 - \lambda)v\|_2 + l(\lambda)$ with λ being the parametrization of a point on $\overline{uv}$.

If $l(\lambda)$ is linear the derivative with respect to λ is $\frac{dl}{d\lambda} = \|u - v\|_2 \cdot \cos\phi + k$ for a constant k. Then we get $\frac{dl}{d\lambda} = 0 \quad \Leftrightarrow \quad \cos\phi = -\frac{k}{\|u - v\|_2}$. If $\phi \notin I$, the boundary value of I that is closest to the value of ϕ gives us the candidates. In both cases we get a ray of parallel segments as candidates. The length function for this segment is obviously linear.

If $l(\lambda)$ is not linear we cannot compute the derivative that easily to get the shortest paths. Also the new segments need not always be parallel. This case seems to be difficult to handle and thus we would have to discretise the interval on $\overline{uv}$. Then we have only an

finite number of length functions and can approximate the shortest paths by finding the infimum of these functions. In all other cases we can determine the exact length functions.

Case 6: There is Only One Last Segment and It Ends in u or v

This case can be handled just as in Case 4 in Chapter 4.2 which considers the same problem but with path tracing for one path. For the length function we are in the same situation as in Case 4 in this section. Thus we get up to two points for which the derivative is not defined and the intervals in between have length functions that are either linear or a composition of a linear function and the Euclidean norm.

4.4 Algorithm for the Shortest Gentle Path Problem

Once the path tracing problem for one interval is handled the overall algorithm is rather simple. As mentioned before we only need to be able to compute the locally shortest gentle path. Then we can iterate over all locally shortest gentle paths by constructing a sequence tree as we did in Chapter 3. Given an edge sequence the algorithm first computes the length function from the starting point to all points on the first edge. From here on we always trace the paths from the current edge to the next one for all intervals. For each interval on the current edge we also save the relevant information on the segments that create the interval. For a steep interval no supplementary information is needed but for other intervals we need to save some information to be able to compute the path extensions. For an interval of parallel paths we save the incoming angle while for an interval of paths with a common origin we save the crossing point of the supporting lines of the segments. Together with the length function for the current edge this is all the information we need to trace the paths from each interval to the next edge and compute the corresponding length functions.

Now we have a family of length functions for the next edge whose domains might still intersect. This means that the individual path tracing for each interval has produced more than one candidate for some points. Thus we have to compute the infimum of all length functions to find the best candidate for each point.

To this end we first compute all intersections of the length functions. Then we use an algorithm that is similar to the gift wrapping algorithm for the convex hull problem [Jar73]. We sort all intersection points from left to right. Then we start with the leftmost point. If there is more than one point, we take the lowest. This point must be part of the infimum function. Now we consider all functions that intersect at this point and find the function with the lowest gradient. This function must be part of the infimum at least up to the next intersection. This next intersection we consider now and again find the function with the lowest gradient extending from this point. Thus we can trace the infimum function through the intervals until we reach one of the rightmost points. The result is a function that gives for each point on the edge the length of an LSGP from s to this point.

When we have computed the length function for the last edge in the edge sequence we still need to add the s-norm distance to t for every point on the edge. Then the minimum of the resulting function is exactly the length of an LSGP from s to t .

Now we can iterate through all possible edge sequences with the help of the sequence tree and find the globally shortest gentle path from s to t.

4.5 Algorithm Complexity

If we could compute the exact locally shortest gentle path in polynomial time, the overall runtime would also be polynomial. Each of the steps in the algorithm can be done in polynomial time. The problem is that the number of intervals we have to consider for each edge in the edge sequence grows in the worst case exponentially. Thus the worst case runtime of the complete algorithm must also be exponential.

For example consider one interval of flat segments that have a common projected origin. This interval can generate up to five intervals on the next edge (one steep interval, two intervals of parallel segments with slope θ and two flat intervals). If the next face is flat it is possible that we get up to two new intervals of flat segments that originate at the point where the steep interval meets with the intervals of slope θ (see Figure 4.12). At the next edge these intervals can also produce five new intervals. If this development continues we get an exponential number of intervals and thus also an exponential worst case runtime.

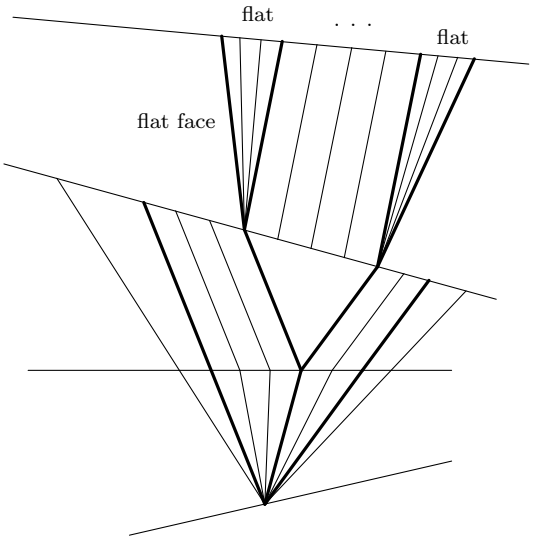


Figure 4.12: Sketch of a terrain in which the number of intervals on each edge increases exponentially.

4.6 Discussion

The shortest gentle path problem is not known to be NP-hard, neither is it known to be polynomially solvable. Thus it is still possible that a polynomial algorithm exists which solves the problem exactly. Still it seems more likely that the shortest gentle path problem is indeed NP-hard. For closely related problems such as the shortest descending path problem no exact polynomial algorithm has been found in spite of the attention this problem received during the last years. Furthermore we know that the also related weighted region problem is unsolvable in the algebraic computation model over $\mathbb{Q}$ [DCGM$^+$12]. In

the algebraic computation model any number that can be obtained from the rational numbers $\mathbb{Q}$ by applying a finite number of the operations $+$, $-$, $\cdot$, $/$ and $\sqrt[n]{\ }$ with $n \in \mathbb{N}^{\geq 2}$ can be computed exactly.

On the other hand at least at a first view the hardness of the shortest gentle path problem does not seem easy to prove. All hardness proofs for different shortest path problems we found were based on the idea presented by Canny and Reif [CR87]. They reduce the three dimensional shortest path problem with polyhedral obstacles to 3SAT. For this purpose they show how a set of obstacles with an altogether polynomial complexity can be constructed such that all paths shorter than a threshold represent a satisfying variable assignment for a previously chosen 3SAT formula.

To this end they introduce three gadgets that are each made up of a set of obstacles. A *path splitter* will double the number of shortest paths by splitting each path in two. By putting n path splitters behind one another we can thus create 2^n distinct shortest paths. Each of these paths will now represent one variable assignment. Then we have *path shufflers* that reorder the paths such that the first half of the paths corresponds to all variable assignments with a specific variable set to 1. Then we need just one obstacle, called a *literal filter*, in front of one half the paths to get elongated paths iff a specific variable is set. With these three components we can build a search space that represents an arbitrary 3SAT formula such that the shortest paths correspond to satisfying variable assignments.

The idea has been picked up and adjusted for other problems, e.g., by Gray and Evans for optimistic paths on uncertain terrains [GE04]. Unfortunately it cannot be applied to the shortest gentle path problem because we cannot reorder distinct paths on a terrain. In three dimensions it is easily possible to shuffle the 2^n paths as seen in [CR87]. Gray and Evans model the uncertainty by giving an interval for the height of each vertex instead of a fixed value. Then the paths can lie anywhere within the three dimensional space that is bordered above and below by the surfaces that we get if we set the height of every vertex to the maximal resp. minimal value. Again the third dimension of the search space makes it possible to construct the path shuffler.

For the shortest gentle path problem this is not possible because the terrain surface is strictly two dimensional. When shuffling the paths we would inevitably get intersections between them. But then the paths wouldn't be distinct anymore since they could have come from either direction before the intersection. Only if the problem has an additional constraint that makes it possible to distinguish the intersecting paths we can construct the needed path shuffler. This is the case for the shortest anisotropic path problem with few bends. The constraint on the number of bends makes it possible to intersect two distinct paths. Using this Ahmed and Lubiw proved the NP-completeness of the shortest anisotropic path problem with few bends [AL08].

We still could try to construct the needed problem instance without path shufflers. But then we would need $\mathcal{O}(2^n)$ obstacles in the form of hills or valleys to implement the literal filters which would obviously result in a terrain with an exponential number of vertices. Thus the proof of Canny and Reif [CR87] cannot be adapted for the shortest gentle path problem.

5. Conclusion

In this thesis we developed the first polynomial $(1+\varepsilon)$-approximation algorithm for the shortest gentle path problem. The runtime lies in $\mathcal{O}\left(n^{5.5}\log\frac{n}{\varepsilon}\right)$ and is as such independent of the terrain geometry. The idea for the algorithm was introduced by Cheng and Jin [CJ12] for shortest descending paths and in this thesis we adapted their algorithm to the shortest gentle path problem. In contrast to most of the approximation algorithms for anisotropic path problems this approach is not based on the simplification of the terrain with Steiner points but on Chen and Han's sequence tree [CH90] and the possibility to find close lower and upper bounds for the solution. Another advantage of the algorithm is that its runtime does not depend on the terrain geometry but only on the number of vertices in the terrain.

Since the adaption of this algorithm from the shortest descending path problem to the shortest gentle path problem worked that well it might be an interesting question if one could use the algorithm to find better approximation algorithms for other shortest path problems as well. Another natural and quite interesting question is whether the shortest gentle path problem is NP-hard. We have investigated this question but it does not seem easy to prove or refute the hardness of the problem.

As we could not establish the hardness of the problem we tried to find a polynomial algorithm that gives an exact solution for the shortest gentle path problem. This goal we could not achieve but in the process we developed specifications on how shortest gentle paths progress after they reached an edge of the terrain. These specifications are similar to the specification of the bend angles for the shortest descending path problem that was published by Ahmed [Ahm09]. Unfortunately with the shortest gentle path problem the angle of the next path segment is not always uniquely defined by the last segment as it is for the shortest descending path problem. Instead we often get a range of possible next segments. None the less these ranges give some rules to how shortest gentle paths progress through a terrain.

Bibliography

[ADL+10] 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*, vol. 8, no. 2, pp. 214–230, 2010.

[Ahm09] M. Ahmed, "Constrained shortest paths in terrains and graphs," Ph.D. dissertation, University of Waterloo, 2009.

[AL08] M. Ahmed and A. Lubiw, "Shortest anisotropic paths with few bends is NP-complete," in *The 18th Fall Workshop on Computational Geometry (FWCG)*, 2008, pp. 28–29.

[AL09] M. Ahmed and A. Lubiw, "Shortest descending paths through given faces," *Computational Geometry*, vol. 42, no. 5, pp. 464–470, Jul. 2009.

[ALM09] M. Ahmed, A. Lubiw, and A. Maheshwari, "Shortest gently descending paths," in *WALCOM: Algorithms and Computation*, ser. Lecture Notes in Computer Science, S. Das and R. Uehara, Eds. Springer, 2009, vol. 5431, pp. 59–70.

[AMS05] L. Aleksandrov, A. Maheshwari, and J.-R. Sack, "Determining approximate shortest paths on weighted polyhedral surfaces," *Journal of the ACM (JACM)*, vol. 52, no. 1, pp. 25–53, Jan. 2005.

[CH90] J. Chen and Y. Han, "Shortest paths on a polyhedron," in *Proceedings of the sixth Annual Symposium on Computational Geometry (SoCG)*, 1990, pp. 360–369.

[Cha96] T. M. Chan, "Optimal output-sensitive convex hull algorithms in two and three dimensions," *Discrete & Computational Geometry*, vol. 16, pp. 361–368, 1996.

[CJ12] S.-W. Cheng and J. Jin, "Approximate shortest descending paths," in *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2012, pp. 144–155.

[CR87] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (FOCS)*, Oct 1987, pp. 49–60.

[DCGM+12] J.-L. De Carufel, C. Grimm, A. Maheshwari, M. Owen, and M. Smid, "Unsolvability of the weighted region shortest path problem," in *European Workshop on Computational Geometry (EuroCG)*, 2012, pp. 65–68.

[Dij59] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[GE04] C. Gray and W. Evans, "Optimistic shortest paths on uncertain terrains," in *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG)*, 2004, pp. 68–71.

[Jar73] R. A. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Information Processing Letters*, vol. 2, no. 1, pp. 18–21, 1973.

[Kar84] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC)*, 1984, pp. 302–311.

[LMS99] M. Lanthier, A. Maheshwari, and J.-R. Sack, "Shortest anisotropic paths on terrains," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, J. Wiedermann, P. Emde Boas, and M. Nielsen, Eds. Springer, 1999, vol. 1644, pp. 524–533.

[LVBL98] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear algebra and its applications*, vol. 284, no. 1, pp. 193–228, 1998.

[LW11] L. Liu and R. C.-W. Wong, "Finding shortest path on land surface," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, vol. 11, 2011, pp. 433–444.

[MP91] J. S. B. Mitchell and C. H. Papadimitriou, "The weighted region problem: finding shortest paths through a weighted planar subdivision," *Journal of the ACM (JACM)*, vol. 38, no. 1, pp. 18–73, 1991.

[RR90] N. C. Rowe and R. S. Ross, "Optimal grid-free path planning across arbitrarily contoured terrain with anisotropic friction and gravity effects," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 5, pp. 540–553, 1990.

[SR05] Z. Sun and J. Reif, "On finding energy-minimizing paths on terrains," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 102–114, Feb. 2005.

[SR06] Z. Sun and J. H. Reif, "On finding approximate optimal paths in weighted regions," *Journal of Algorithms*, vol. 58, no. 1, pp. 1–32, Jan. 2006.

6. Appendix

A Combinatorial Solution to the LSGP Problem

There are only slight modifications needed to solve the local problem using the combinatorial approach described by Cheng and Jin [CJ12, Chapter 3.2] which is cited below. Afterwards the modifications to adopt the algorithm to the shortest gentle path problem are briefly explained.

A.1 Excerpt from [CJ12]: Chapter 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. For simplicity, we will not mention the edge sequence of the LSDP as it is fixed by the sequence tree.

Our method computes *all* L_∞ LSDPs from s to points on the two edges of the face f_α incident to v_α. Suppose that we have already computed all L_∞ LSDPs from s to points on e_α. Our strategy is to extend these paths to L_∞ LSDPs to 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 $p_\zeta \in ab$. 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.

Refer to Figure A.1. Let e be an edge of f_α incident to v_α. Let u be the other endpoint of e. Let u and w be the endpoints of e_α. We parametrize e_α and e by $\lambda, \tau \in [0,1]$, respectively. That is, $p_\lambda = (1-\lambda)u + \lambda w$ and $p_\tau = (1-\tau)u + \tau v_\alpha$. 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 $(v_{\alpha,z} - u_z)\tau \leq (w_z - u_z)\lambda$.

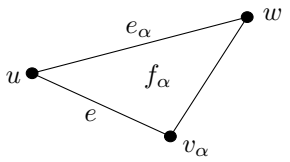
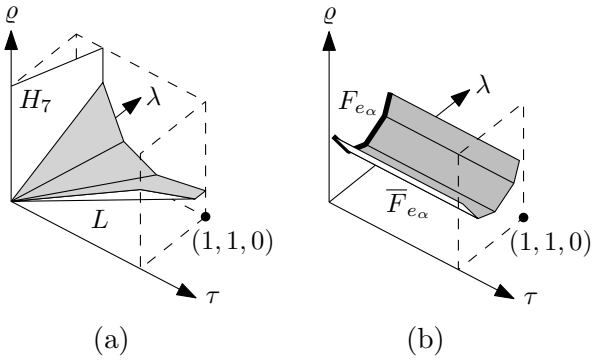


Figure A.1: The configuration of f_α.



(a) (b)

Figure A.2: In (a), the graph of L is on the positive side of H_7, and the projection of L in the $\tau\lambda$-plane is inside the unit square. In (b), F_{e_α} is shown bold in the $\lambda\varrho$-plane, and each segment on it is extended into a strip. The resulting surface is $\overline{F}_{e_\alpha}$.

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. We conclude that the graph of $\|p_\lambda p_\tau\|_\infty$ is the upper envelope of the six planes H_i for $i \in [1, 6]$.

Define the function $L(\tau, \lambda)$ to be the L_∞ length of a descending segment $p_\lambda p_\tau$. Thus, L is convex and piecewise linear. We call the halfspace $(w_z - u_z)\lambda \geq (v_{\alpha,z} - u_z)\tau$ the *positive side* of the vertical plane H_7. Then, the graph of L is the subset of the graph of $\|p_\lambda p_\tau\|_\infty$ that lies on the positive side of H_7. See Figure A.2(a) for an illustration. The graph of L consists of triangular faces that share the origin as a common vertex because H_i contains the origin for $i \in [1, 7]$.

We claim that F_e is convex and piecewise linear, which can be established by induction on the construction of the sequence tree as follows. The edges opposite s correspond to the children of the root in the initial sequence tree. 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.

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. As illustrated in Figure A.2(b), sweep each segment ℓ_i in F_{e_α} in a direction parallel to the τ-axis to obtain a strip $\ell_i \times [0,1]$. Let $\overline{F}_{e_\alpha}$ be the resulting convex surface. So $\overline{F}_{e_\alpha}(\tau,\lambda) = F_{e_\alpha}(\lambda)$. $\overline{F}_{e_\alpha} + L$ is convex and piecewise linear because it is the addition of two convex piecewise linear functions, clipped by the vertical plane H_7 and the two vertical planes that are orthogonal to the λ-axis and contain the endpoints of I_{e_α}. Since $F_e(\tau) = \min_{\lambda \in I_{e_\alpha}} \overline{F}_{e_\alpha}(\tau,\lambda) + L(\tau,\lambda)$, F_e is also convex and piecewise linear. The following two results summarize the above.

Lemma 11. *For every terrain edge e', $F_{e'}$ is a convex piecewise linear function.*

Lemma 12. *If F_{e_α} is given and it has k segments, then F_e can be constructed in $O(k \log k)$ time.*

Proof. The graph of $\overline{F}_{e_\alpha} + L$ is the upper envelope of $6k$ planes, clipped by the vertical plane H_7 and the two vertical planes that are orthogonal to the λ-axis and contain the endpoints of I_{e_α}. So $\overline{F}_{e_\alpha} + L$ can be computed by intersecting $6k+3$ halfspaces in 3D. By geometric duality, it is equivalent to computing the 3D convex hull of $6k+3$ points, which can be done in $O(k \log k)$ time [Cha96]. By sweeping a plane from $\lambda = 0$ to $\lambda = 1$, it takes $O(k)$ time to trace the polyline on the graph of $\overline{F}_{e_\alpha} + L$ that achieves the minimum value for every λ, which gives F_e. □

The critical issue is how the complexity of F_e depends on the complexity of F_{e_α} because it will affect the time to be spent in creating the children of v_α in the sequence tree. The proof of the above lemma gives a trivial upper bound of $6k+3$, where k is the complexity of F_{e_α}. This bound is not useful, as it becomes $2^{O(|\sigma_\alpha|+1)}$ 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. Then, Lemma 12 implies that computing the L_∞ LSDPs at any node of the sequence tree takes $O(n \log n)$ time. Therefore, during the construction of the sequence tree, the total time spent in constructing all L_∞ LSDPs is $O(n^3 \log n)$.

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 the graph of F_{e_α}, including the endpoints of I_{e_α}. For every $\tau \in I_e$, define

$$\lambda_\tau = \min\{\lambda : F_e(\tau) = F_{e_\alpha}(\lambda) + L(\tau,\lambda)\}.$$

Lemma 13. *For $\tau_0, \tau_1 \in I_e$, if $\tau_0 < \tau_1$, then $\lambda_{\tau_0} \le \lambda_{\tau_1}$.*

Lemma 14. *For any $\tau_0 \in I_e$, 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 the graph of L, or*

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

Consider increasing 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. Let $\mathcal{L}$ be the projection of the graph of L clipped by the plane H_7 on the $\tau\lambda$-plane. Divide η into segments at its intersections

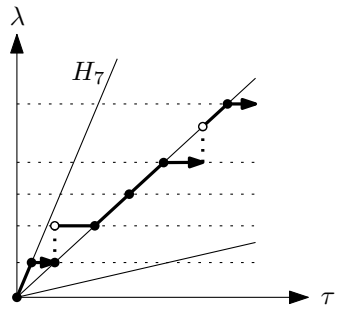


Figure A.3: The λ-values of the dotted horizontal lines are the breakpoints of F_{e_α}. The monotonic locus η shown in bold is divided into segments at its intersections with the dotted lines and the edges of $\mathcal{L}$. A segment of η is either horizontal or on an edge of $\mathcal{L}$.

with edges of $\mathcal{L}$ and points whose λ-values are breakpoints of F_{e_α}. Each segment of η corresponds to a segment of F_e. In the following, we bound the number of segments in η.

We examine η in the $\tau\lambda$-plane as τ increases. By Lemmas 13 and 14, η is monotonic, and each of segment in η either lies on an edge of $\mathcal{L}$, or is parallel to the τ-axis 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 for every $\lambda \in [\lambda_{\tau_0}, \lambda']$, $F_{e_\alpha}(\lambda) + L(\tau_0, \lambda) = F_{e_\alpha}(\lambda_{\tau_0}) + L(\tau_0, \lambda_{\tau_0})$, because λ_{τ_0} is preferred 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 A.3 shows an example.

We first bound the number of jumps η makes. Suppose η jumps from a point $(\tau_0, \lambda_{\tau_0})$ in the direction of 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 A.4. 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 term in $L(\tau_0, \lambda)$ 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 F_{e_α}, and the equation of H_j can no longer cancel the term in F_{e_α} that involves λ. On the other hand, there is no edge of $\mathcal{L}$ to follow in the interior of g. Therefore, if ℓ_0's upper endpoint is in g's interior, then by Lemma 14, 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 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 positive λ-axis in each face of $\mathcal{L}$.

Let k be the number of breakpoints of F_{e_α}. Let k' be the number of faces in $\mathcal{L}$. The above analysis shows that η makes at most k' jumps. Since η is monotonic, the upper endpoints

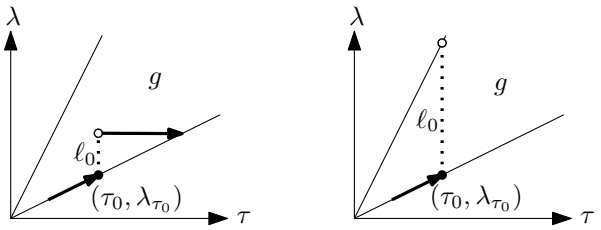


Figure A.4: The black dot is the point $(\tau_0, \lambda_{\tau_0})$. On the left, ℓ_0 stops in g's interior and then turns right in the direction of the positive τ-axis. On the right, ℓ_0 extends all the way across g.

of segments in η that lie on edges of $\mathcal{L}$ have different λ-values. 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 $k + k'$ such upper endpoints, which implies that η has at most $k + k'$ segments that lie on edges of $\mathcal{L}$.

It remains to bound the number of the segments in η that are parallel to τ-axis. Consider two such segment h_1 and h_2 of η 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. Since η does not make a jump between h_1 and h_2, the right endpoint of h_1 must 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 it does not jump into g before that. 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 parallel to τ-axis in each face. Hence, η has at most $2k'$ such segments in total that are parallel to the τ-axis..

The following result summarizes the above analysis.

Lemma 15. *If F_{e_α} has k segments, then the number of segments in F_e is at most $k+3k' = k + O(1)$, where k' is the number of faces in L.*

Since the sequence tree has $O(n)$ height, Lemma 15 implies that for each node α created, F_{e_α} has $O(n)$ segments, and so Lemma 12 implies that the L_∞ LSDP computation at α takes $O(n \log n)$ time. As $O(n^2)$ nodes are created in constructing the sequence tree, the total time needed is $O(n^3 \log n)$.

Lemma 16. *It takes $O(n^3 \log n)$ total time to compute the L_∞ LSDPs from s to v_α for all nodes α created in constructing the sequence tree.*

A.2 Modifications

If $H_5 := \frac{1}{\sin\theta} H_5$ and $H_6 := \frac{1}{\sin\theta} H_6$ then the graph of $\|p_\lambda p_\tau\|_b$ is the upper envelope of $H_1 \ldots H_6$. H_7 is not needed because the path does not need to be descending. In our case we have $I_{ab} \in \{(0,1),(0,1],[0,1),[0,1]\}$ for every edge ab in the edge sequence because it is possible that there is no gentle path from a vertex to another point in the triangle.

We assume that $I_e = [0,1]$ for all edges e and ignore the fact that there might be no gentle path with the computed length if one endpoint is a vertex. But there is always a gentle path with the same edge sequence that is only by an arbitrarily small amount longer than

the computed length and that does not go through the vertex. Thus this assumption can only cause an arbitrarily small error in the computed length.

So as in [CJ12] it takes $\mathcal{O}(n^3 \log n)$ total time to compute the solution to all local problems arising during the construction of the sequence tree.

B Excerpt from [CJ12]: Proof of the Correctness of the Approximate SDP Algorithm

Lemmata referred to in the proof of correctness:

Lemma 17. *Let $\mu = \frac{\varepsilon \mathsf{opt}_\infty}{\kappa n^2 m(m+1)}$. Given a destination v and an edge sequence σ in $\mathcal{T}$, we can compute in $\mathcal{O}(|\sigma|^2 \log(\frac{n}{\varepsilon}))$ time an approximate LSDP in $\mathcal{T}$ from s to v with edge sequence σ, so that this approximate LSDP has an additive error at most $|\sigma|\mu$.*

Lemma 18. *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 $\|Q_\alpha\| \le \|Q_\beta\| + (|\sigma_\beta| - |\sigma_\alpha|) \cdot \tilde{\mu} + |\sigma_\beta| \cdot \mu$.

Proof of correctness from [CJ12]:

Lemma 19. *For any $\varepsilon \in (0, 1)$, our algorithm returns a $(1+\varepsilon)$-approximate SDP from s to t.*

Proof. Recall that 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 $\sigma_{\gamma_0} = \mathrm{seq}(P_0)$. Lemma 17 gives an approximate LSDP P_{γ_0} such that $\|P_{\gamma_0}\| \le \|P_0\| + |\sigma_{\gamma_0}|\mu < \mathsf{opt} + m\mu < \mathsf{opt} + \varepsilon \mathsf{opt}_\infty \le (1+\varepsilon)\mathsf{opt}$. 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 18(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.

W.l.o.g., assume that α_1 quasi-dominates β_0 on the right, so the right child of β_0 is pruned. Refer to Figure B.5. 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 an ancestor of γ_0, P_0 must pass through the edge e at some point p. By Lemma 18(ii), there is a descending path Q_{α_1} from s to p with edge sequence σ_{α_1} such that

$$\begin{aligned} \|Q_{\alpha_1}\| &\le \|P_0[s,p]\| + (|\sigma_{\beta_0}| - |\sigma_{\alpha_1}|)\tilde{\mu} + |\sigma_{\beta_0}|\mu \\ &< \|P_0[s,p]\| + (|\sigma_{\beta_0}| - |\sigma_{\alpha_1}|)\tilde{\mu} + m\mu. \end{aligned} \tag{6.1}$$

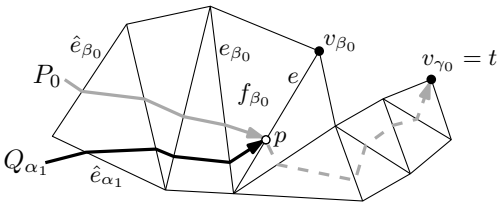


Figure B.5: P_0 consists of the grey solid and grey dashed links. The grey dashed path is not represented in the final sequence tree because α_1 quasi-dominates β_0 on the right. The grey solid path has edge sequence σ_{β_0}. Lemma 18(ii) guarantees a bold solid path Q_{α_1} from s to p with edge sequence 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}.

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 descending path $Q_{\alpha_1} \cdot P_0[p,t]$ from s to t has edge sequence σ_{γ_1}. Thus, the LSDP P_1 from s to t with edge sequence σ_{γ_1} exists, and

$$\begin{aligned} \|P_1\| &\leq \|Q_{\alpha_1}\| + \|P_0[p,t]\| \\ &< \|P_0\| + (|\sigma_{\beta_0}| - |\sigma_{\alpha_1}|)\tilde{\mu} + m\mu \quad \text{(By (6.1).)} \\ &= \|P_0\| + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_1}|)\tilde{\mu} + m\mu \\ &= \mathsf{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_1}|)\tilde{\mu} + m\mu. \end{aligned} \tag{6.2}$$

Suppose that the final sequence tree does not contain the node β_1. Then, we want to repeat the argument above the define another LSDP P_2 from s to t which is only slightly longer. In general, for $i \geq 0$, inductively assume that there is a LSDP from s to t with edge sequence σ_{γ_i} such that

$$\|P_i\| \leq \mathsf{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_i}|)\tilde{\mu} + im\mu. \tag{6.3}$$

Since $\|P_i\| \geq \mathsf{opt}$, we have $(|\sigma_{\gamma_i}| - |\sigma_{\gamma_0}|)\tilde{\mu} \leq im\mu$. For $i < \kappa n^2$, $im\mu < \kappa n^2 m\mu = \tilde{\mu}$, so $|\sigma_{\gamma_i}| - |\sigma_{\gamma_0}| < 1$. The length of an edge sequence is an integer, so $|\sigma_{\gamma_i}| \leq |\sigma_{\gamma_0}| < m$. In our algorithm, the sequence tree is grown until its height reaches m. So if the final sequence tree does not contain the node γ_i, then there must be tree nodes β_i and α_{i+1} corresponding to the same face corner such that α_{i+1} quasi-dominates β_i, σ_{β_i} is a prefix of σ_{γ_i}, and the child of β_i that would be an ancestor of γ_i is pruned due to the qusi-dominance of α_{i+1} over β_i. Then $\sigma_{\gamma_{i+1}}$ is the edge sequence obtained by replacing the prefix σ_{β_i} of σ_{γ_i} by $\sigma_{\alpha_{i+1}}$. We can define a LSDP P_{i+1} from s to t with edge sequence $\sigma_{\beta_{i+1}}$ such that, analogous to (6.2), $\|P_{i+1}\| \leq \|P_i\| + (|\sigma_{\gamma_i}| - |\sigma_{\gamma_{i+1}}|)\tilde{\mu} + m\mu$. It follows from (6.3) that $\|P_{i+1}\| \leq \mathsf{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_{i+1}}|)\tilde{\mu} + (i+1)m\mu$. Therefore, we can carry out the inductive analysis as long as $i < \kappa n^2$.

The path P_i is only defined by the pruning of β_i. Therefore, the pruning of β_i happens later than the pruning of β_{i-1} during the execution of the algorithm. Pruning can only happen after the creation of a new leaf. Thus, the sequence tree is pruned less than κn^2 times because fewer than κn^2 nodes are ever created. Hence, the inductive argument must stop at P_ℓ for some $\ell < \kappa n^2$, and the final sequence tree contains the node γ_ℓ with edge

sequence $\sigma_{\gamma_\ell} = \text{seq}(P_\ell)$. After creating γ_ℓ, we apply Lemma 17 to compute P_{γ_ℓ}. Thus,

$$\begin{aligned}
\|P_{\gamma_\ell}\| &\leq \|P_\ell\| + |\sigma_{\gamma_\ell}|\mu \\
&\leq \mathsf{opt} + (|\sigma_{\gamma_0}| - |\sigma_{\gamma_\ell}|)\tilde{\mu} + \ell m\mu + |\sigma_{\gamma_\ell}|\mu \\
&< \mathsf{opt} + |\sigma_{\gamma_0}|\tilde{\mu} + \kappa n^2 m\mu \\
&< \mathsf{opt} + (m+1)\tilde{\mu} \\
&= \mathsf{opt} + \varepsilon \mathsf{opt}_\infty \\
&\leq (1+\varepsilon)\mathsf{opt}.
\end{aligned}$$

The path returned by our algorithm has length at most $\|P_{\gamma_\ell}\|$, so it is a $(1+\varepsilon)$ approximate SDP. □