An Approximation Algorithm for Computing Shortest Paths in Weighted 3-d Domains*

Lyudmil Aleksandrov† Hristo Djidjev ‡ Anil Maheshwari §
Jörg-Rüdiger Sack ¶

February 11, 2011

Abstract

We present the first polynomial time approximation algorithm for computing shortest paths in weighted three-dimensional domains. Given a polyhedral domain $\mathcal{D}$, consisting of n tetrahedra with positive weights, and a real number $\varepsilon \in (0,1)$, our algorithm constructs paths in $\mathcal{D}$ from a fixed source vertex to all vertices of $\mathcal{D}$, whose costs are at most $1+\varepsilon$ times the costs of (weighted) shortest paths, in $O(\mathcal{C}(\mathcal{D})\frac{n}{\varepsilon^{2.5}}\log\frac{n}{\varepsilon}\log^3\frac{1}{\varepsilon})$ time, where $\mathcal{C}(\mathcal{D})$ is a geometric parameter related to the aspect ratios of tetrahedra.

The efficiency of the proposed algorithm is based on an in-depth study of the local behavior of geodesic paths and additive Voronoi diagrams in weighted three-dimensional domains, which are of independent interest. The paper extends the results of Aleksandrov et al. [4] to three dimensions.

1 Introduction

1.1 Motivation

The computation of shortest paths is a key problem arising in a number of diverse application areas including geographic information systems, robotics, computer graphics, computer-aided design, medical computing and others. This has motivated the study and subsequent design of efficient algorithms for solving shortest path problems in different settings based on the geometric nature of the problem domain (e.g., two-dimensional (2-d), three-dimensional (3-d), surfaces, presence/absence of obstacles) and the cost function/metric (e.g., Euclidean, L_p, link distance, weighted/unweighted, multi-criteria). In addition to its driver - the applications - the field has provided, and continues to do so, exciting challenges from a theoretical

*Research supported by NSERC. Preliminary results have appeared in [3].

†Bulgarian Academy of Sciences, IPP, Acad. G. Bonchev Str. Bl. 25-A, 1113 Sofia, Bulgaria. lyualeks@bas.bg

‡Los Alamos National Laboratory, Los Alamos, NM 87544, U.S.A

§School of Comp. Sci., Carleton U., Ottawa, Ontario K1S5B6, Canada. anil@scs.carleton.ca

¶School of Comp. Sci., Carleton U., Ottawa, Ontario K1S5B6, Canada. sack@scs.carleton.ca

perspective. As a result, shortest path problems have become fundamental problems in areas of Computer Science such as Computational Geometry and Algorithmic Graph Theory.

The standard 3-d Euclidean shortest path problem of computing a shortest path between pair of points avoiding a set of polyhedral obstacles, denoted as the ESP3D problem, is known to be NP-hard even when the obstacles are parallel triangles in the space. It is not difficult to see that the number of combinatorially distinct shortest paths from a source point to a destination point may be exponential in the input size. Canny and Reif [8] used this to establish the NP-hardness of the ESP3D problem, for any L_p metric, $p \geq 1$. In addition to this combinatorial hardness result, Bajaj [7] has provided an algebraic hardness argument that an exponential number of bits may be required. More recently, Mitchell and Sharir [22] gave NP-completeness proofs for the problem of computing Euclidean shortest paths among sets of stacked axis-aligned rectangles, and computing L_1-shortest paths among disjoint balls. Given the NP-hardness of the ESP3D problem, work has focused on exploiting the geometric structure of the obstacles and/or on providing approximation algorithms. We will mention some of these approaches in Section 1.4.

In many applications, the Euclidean metric does not capture adequately the nature of the problem, for instance when the problem domain is not homogeneous. This motivates the weighted versions of the shortest path problem. For example in the 2-d case, consider triangulated regions where each triangle represents a particular terrain type such as water, rock, or forest. Here different weights capture the cost of traveling a Euclidean unit-length through each face. Incorporating weights makes the solution more difficult to obtain even in 2-d, but it does provide more realistic answers. It is known that light and other types of waves (e.g., seismic and sonic) travel along the shortest paths in heterogeneous media. Hence, algorithms solving the weighted shortest path problem (WSP3D) can be used for modeling wavefront propagation in such media. In the 3-d, a number of applications are non-homogeneous in nature and can be expressed using the weighted model. Next, we list some of such potential applications.

- In *geology*, seismic refraction and reflection methods are used based on measurements of the travel time of seismic waves refracted at the interfaces between subsurface layers of different densities. As such waves propagate along shortest paths and weighted shortest path algorithms may be used to produce more accurate and more efficient estimation of subsurface layer characteristics, e.g., the amount of oil contained in the subsurface [14]. Another related application is the assessment of garbage dumps' health. When a new garbage dump is built, sensors are placed at the bottom, and when the garbage dump starts to fill, waves from the top passing through the garbage to these sensors are used in order to determine the decomposition rate of the garbage [14].

- Computation of 3-d shortest path have also been used to compute *fastest routes* for aircrafts between designated origin and destination points while avoiding hazardous, time-varying weather systems. Krozel et al. [17] investigate synthesizing weather avoidance routes in the transition airspace. Our weighted 3-d region model can be used to generalize that approach: instead of totally avoiding undesirable regions, one can assign penalty weights to them and then search for routes that minimize travel through such regions, while also avoiding unnecessarily long detours.

- In *medical applications* simulation of sonic wavefront propagation is used when performing imaging methods as photoacoustic tomography or ultrasound imaging through heterogeneous tissue [12, 27]. In radiation therapy, domain features include densities of tissue, bone, organs, cavities, or risk to radiation exposure, and optimal radiation treatment planning takes this non-homogeneity into consideration.

- The problem of time-optimum *movement planning* in 2-d and 3-d for a point robot that has bounded control velocity through a set of n polygonal regions of given translational flow velocities has been studied by Reif and Sun [24]. They state that this intriguing geometric problem has immediate applications to macro-scale motion planning for ships, submarines, and airplanes in the presence of significant flows of water or air. Also, it is a central motion planning problem for many of the meso-scale and micro-scale robots that have environments with significant flows that can affect their movement. They establish the computational hardness for the 3-d version of this problem by showing the problem to be PSPACE hard. They give a decision algorithm for the 2-d flow path problem, which has very high computational complexity, and they also design an efficient approximation algorithm with bounded error. The determination of the exact computational complexity of the 3-d flow path problem is posed as an open problem. Although, our weighted 3-d model does not apply directly to this setting, it can be used to construct initial approximations by assigning appropriate weights depending on the velocity and direction of the flows in different regions. In addition, the discretization scheme and the algorithmic techniques developed here can be used for solving the 3-d flow path problem.

1.2 Problem formulation

In this paper, we consider the following problem. Let $\mathcal{D}$ be a connected 3-d domain consisting of n tetrahedra with a positive real weight associated to each of them. The 3-d weighted shortest path problem (WSP3D) is to compute minimum-cost paths in $\mathcal{D}$ from a fixed source vertex to all vertices of $\mathcal{D}$. The cost of a path in $\mathcal{D}$ is defined as the weighted sum of the Euclidean lengths of the sub-paths within each crossed tetrahedron. We will describe and analyze an approximation algorithm for this problem that, for any real number $\varepsilon \in (0, 1)$, computes paths whose costs are at most $1 + \varepsilon$ times greater than the costs of the minimum cost paths. In Section 2, we describe our model in detail.

Note that the WSP3D problem can be viewed as a generalization of the ESP3D problem. Namely, given an instance of the ESP3D problem, one can find a large enough cube containing all the obstacles, tetrahedralize the free-space (i.e., exterior of the obstacles, but in the interior of the cube) and set equal weights to the resulting tetrahedra obtaining an instance of the WSP3D problem.

1.3 Challenges

A key difference between Euclidean shortest path computation in 2-d and 3-d weighted domain is the NP-hardness already mentioned. Underlying this is the fact that, unlike in 2-d, Euclidean 3-d shortest paths are not discrete. Specifically, in 2-d, the edges of a shortest

path (e.g., Euclidean shortest paths among obstacles in the plane) are edges of a graph, namely, the visibility graph of the obstacles including the source and the destination points. In contrast, in polyhedral 3-d domains, the bending points of shortest paths on obstacles may lie in the interior of the obstacles' edges. Moreover, in weighted 3-d settings, bending points may even belong to the interior of the faces.

Furthermore, even in the case of weighted triangulated planar domains, the (weighted) shortest path may cross each of the n cells $\Theta(n)$ times and may be composed of $\Theta(n^2)$ segments in total. Not only is the path complexity higher, but the computation of weighted shortest paths in 2-d turns out to be substantially more involved than in the Euclidean setting. In fact, there is not even an exact algorithm known, and the first $(1+\varepsilon)$ approximation algorithm due to [21] had an $O(n^8 \log(\frac{n}{\varepsilon}))$ time bound, where n is the number of triangles in the subdivision. This problem has been actively researched since then, and currently the best known algorithm for the weighted region problem on planar subdivisions (as well as on polyhedral surfaces) runs in $O(\frac{n}{\sqrt{\varepsilon}} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$ time [4]. (Also, see [4] for a detailed literature review for the planar case.)

One of the classical tools of Computational Geometry is the Voronoi Diagram. This structure finds numerous applications (see e.g., [6]). It is also a key ingredient in several efficient shortest path algorithms. Researchers have studied these diagrams under several metrics (including Euclidean, Manhattan, weighted, additive, convex, abstract) and for different types of objects (including points, lines, curves, polygons), but somehow the computation of these diagrams in media with different densities (i.e., the refractive media) remained elusive. One of the main ingredients in solving the problem studied here is to compute (partial) additive Voronoi diagrams of points in refractive media. The generic techniques of Klein [15, 16] and Lê [19] do not apply in this case, as the bisecting surfaces do not satisfy the required conditions. In this paper, we make an important step towards the understanding and computation of these diagrams.

1.4 Previous related work

By now, shortest path problems in 2-d are fairly well understood. Efficient algorithms have been developed for many problem instances and surveys are readily available describing the state of the art in the field.

In 3-d, virtually all the work has been devoted to the ESP3D problem. Papadimitriou [23] suggested the first polynomial time approximation scheme for that problem. It runs in $O(\frac{n^4}{\varepsilon^2}(L + \log(n/\varepsilon))$ time, where L is the number of bits of precision in the model of computation. Clarkson [11] provided an algorithm running in $O(n^2\lambda(n)\log(n/\varepsilon)/(\varepsilon^4) + n^2 \log n\rho \log(n \log \rho))$ time, where ρ is the ratio of the longest obstacle edge to the distance between the source and the target vertex, $\lambda(n) = \alpha(n)^{O(\alpha(n))^{O(1)}}$, and $\alpha(n)$ is the inverse Ackermann's function.

Papadimitriou's algorithm was revised and its analysis was refined by Choi et al. [9] under the bit complexity framework. Their algorithm runs roughly in $O(\frac{n^4L^2}{\varepsilon^2}\mu(X))$ time, where $\mu(X)$ represents the time (or bit) complexity of multiplying X-bit integers and $X = O(\log(\frac{n}{\varepsilon}) + L)$. In [10], the same authors further developed their ideas and proposed a precision-sensitive algorithm for the ESP3D problem. In [5], Asano et al. proposed

and studied a technique for computing approximate solutions to optimization problems and obtained another precision-sensitive approximation algorithm for the ESP3D problem with improved running time in terms of L.

Har-Peled [13] proposed an algorithm that invokes Clarkson's algorithm as a subroutine $O(\frac{n^2}{\varepsilon^2}\log\frac{1}{\varepsilon})$ times to build a data structure for answering approximate shortest path queries from a fixed source in $O(\log\frac{n}{\varepsilon})$ time. The data structure is constructed in roughly $O(\frac{n^6}{\varepsilon^4})$ time. Agarwal et al. [1] considered the ESP3D problem for the case of convex obstacles and proposed an approximation algorithm running in $O(n+\frac{k^4}{\varepsilon^7}\log^2\frac{k}{\varepsilon}\log\log k)$ time, where k is the number of obstacles. In contrast to all other algorithms discussed here, the complexity of this algorithm does not depend on the geometric features of the obstacles. In the same paper, the authors describe a data structure for answering approximate shortest path queries from a fixed source in logarithmic time.

In the weighted (non-Euclidean) 3-d case no previous algorithms have been reported by other authors. In [3], we have announced and sketched a polynomial time approximation scheme for WSP3D problem that runs in $O(\frac{n}{\varepsilon^{3.5}}\log\frac{1}{\varepsilon}(\frac{1}{\sqrt{\varepsilon}}+\log n))$ time. The run-time improves to $O(\frac{n}{\varepsilon^3}\log\frac{1}{\varepsilon}\log n)$ when all weights are equal. This algorithm can be used to efficiently solve the ESP3D problem. In this paper, we apply that approach, but develop the required details, apply new techniques, improve the complexity bounds, and provide a rigorous mathematical analysis.

1.5 Contributions of this paper

In this paper, we make several contributions to the fields of shortest path computations and the analysis of weighted 3-d regions model, as listed below.

- We provide an approximation algorithm for solving the WSP3D problem in a polyhedral domain $\mathcal{D}$ consisting of n weighted tetrahedra. The algorithm computes approximate weighted shortest paths from a source vertex to all other vertices of D in $O(\mathcal{C}(\mathcal{D})\frac{n}{\varepsilon^{2.5}}\log\frac{n}{\varepsilon}\log^3\frac{1}{\varepsilon})$ time, where $\varepsilon\in(0,1)$ is the user-specified approximation parameter and $\mathcal{C}(\mathcal{D})$ is a geometric parameter related to the aspect ratios of tetrahedra[1]. The cost of the computed paths are within a factor of $1+\varepsilon$ of the cost of the corresponding shortest paths.

  As we stated above, the ESP3D problem, i.e., the unweighted version of this problem, is already NP-hard even when the obstacles are parallel triangles in the space [8]. The time complexity of our algorithm, which is designed for the more general weighted setting, compares favorably even when applied in the Euclidean setting to the existing approximation algorithms.

- Our detailed analysis, especially the results on additive Voronoi diagrams derived in Section 2, provides valuable insights into the understanding of Voronoi diagrams in heterogeneous media. This may open new avenues, for example, for designing an algorithm to compute discretized Voronoi diagrams in such settings.

[1] See Lemma 3.1 for details on the value of $\mathcal{C}(\mathcal{D})$.

- Our approximation algorithms in 2-d have proven to be easily implementable and of practical value [18]. Our algorithm for WSP3D presented here, in spite of being hard to analyze, essentially uses similar primitives, and thus has the potential to be implementable, practical, and applicable in different areas.

- Our work provides further evidence that discretization is a powerful tool when solving shortest path-related problems in both Euclidean and weighted settings. We conjecture that the discretization methodology used here generalizes to any fixed dimension.

  Furthermore, our discretization scheme is independent of the source vertex and can be used with no changes to approximate paths from other source vertices. This feature makes it applicable for solving all pairs shortest paths problem and for designing data structures for answering shortest path queries in weighted 3-d domains.

- The complexity of our algorithm does not depend on the weights assigned to the tetrahedra composing $\mathcal{D}$, but it depends on their geometry. We analyze and evaluate that dependence in detail. Geometric dependencies arise also in Euclidean settings and in most of the previous papers. For example, in Clarkson [11], the running time of the algorithm depends on the ratio of the longest edge to the distance between the source and the target vertex. Applying known techniques (see e.g., [1]), such dependency can often be removed. Here, this would be possible provided that an upper bound on the number of segments on weighted shortest paths in 3-d is known. However, the increase in the dependency on n in the time complexity that these techniques suffer from, which is of order $\Omega(n^2)$, appears not to justify such an approach here. In our approach, the dependency on the geometry is proportional to the average of the reciprocal squared sinuses of the dihedral angles of the tetrahedra composing $\mathcal{D}$. Thus, when n is large, many tetrahedra would have to be fairly degenerate so that this average to play a major role. We therefore conjecture that in typical applications, the approach presented here would work well.

1.6 Organization of the paper

In Section 2, we describe the model used throughout this paper, formulate the problem, present some properties of shortest paths in 3-d, and derive a key result on additive Voronoi diagrams in refractive media. In Section 3, we describe our discretization scheme which is a generalization of the 2-d scheme introduced in [4]. In Section 4, we construct a weighted graph, estimate the number of its nodes and edges and prove that shortest paths in $\mathcal{D}$ can be approximated by paths in the graph so constructed. In Section 5, we present our algorithm for the WSP3D problem. In Section 6, we conclude this paper.

2 Problem formulation and preliminaries

2.1 Model

Let $\mathcal{D}$ be a connected polyhedral domain in 3-d Euclidean space. We assume that $\mathcal{D}$ is partitioned into n tetrahedra $T_1, \ldots, T_n$, such that $\mathcal{D} = \cup_{i=1}^{n} T_i$ and the intersection of any

pair of different tetrahedra is either empty or a common element (a face, an edge, or a vertex) on their boundaries. We call these tetrahedra *cells*. A positive weight w_i is associated with each cell T_i representing the cost of traveling in it. The cost of traveling along a boundary element of a cell is the minimum of the weights of the cells incident to that boundary element. We consider *paths* (connected rectifiable curves) that stay in $\mathcal{D}$. The cost of a path π in $\mathcal{D}$ is defined by $\|\pi\| = \sum_{i=1}^{n} w_i|\pi_i|$, where $|\pi_i|$ denotes the Euclidean length of the intersection $\pi_i = \pi \cap T_i$. Boundary edges and faces are assumed to be part of the cell from which they inherit their weight.

Given two distinct points u and v in $\mathcal{D}$, the shortest path problem in $\mathcal{D}$ is to find a minimum cost path $\pi(u, v)$ between u and v that stays in $\mathcal{D}$. We refer to the minimum cost paths as *shortest paths*. For a given approximation parameter $\varepsilon > 0$, a path $\pi_\varepsilon = \pi_\varepsilon(u, v)$ is an ε-approximation of the shortest path $\pi = \pi(u, v)$, if $\|\pi_\varepsilon\| \leq (1 + \varepsilon)\|\pi\|$. Without loss of generality, we may assume that the points u and v are vertices of $\mathcal{D}$, since, if they are not, we can make them such by partitioning the cells where they belong. In this paper, we present an algorithm that, for a given source vertex u and an approximation parameter $\varepsilon \in (0, 1)$, computes ε-approximate shortest paths from u to all vertices of $\mathcal{D}$.

In this setting, it is well known [21][2] that shortest paths are simple (non self-intersecting) and consist of a sequence of segments, whose endpoints are on the cell boundaries. The intersection of a shortest path with the interior of a cell, a face, or an edge is a set of disjoint segments. More precisely, each segment on a shortest path is of one of the following three types:

(i) *cell-crossing* – a segment that crosses a cell joining two points on its boundary;
(ii) *face-using* – a segment lying along a face of a cell;
(iii) *edge-using* – a segment along an edge of a cell.

We define *linear paths* to be paths consisting of cell-crossing, face-using, and edge-using segments exclusively. A linear path $\pi(u, v)$ can be represented as the sequence of its segments $\{s_1, \ldots, s_{l+1}\}$ or, equivalently, as the sequence of points $\{a_0, \ldots, a_{l+1}\}$, lying on the cell boundaries that are endpoints of these segments, i.e., $s_i = (a_{i-1}, a_i)$, $u = a_0$, and $v = a_{l+1}$. The points a_i that are not vertices of cells are called *bending points* of the path.

The local behavior of a shortest path around a bending point a, lying in the interior of a face f, is fully described by the directions of the two segments of the shortest path, s^- and s^+, that are incident to a. The direction of each of these two segments is described by a pair of angles, which we denote by (φ^-, θ^-) and (φ^+, θ^+), respectively. The *in-angle* φ^- is defined to be the acute angle between the direction normal to f and the segment s^-. Similarly, the *out-angle* φ^+ is the acute angle between the normal and the segment s^+. The angles θ^- and θ^+ are the acute angles between the orthogonal projections of s^- and s^+ with a reference direction in the plane containing the face f, respectively (see Figure 1).

It is well known that when π is a shortest path it is a linear path such that the angles (φ^-, θ^-) and (φ^+, θ^+) are related by Snell's law as follows:

[2]The 2-d case was treated there, but the arguments readily apply to the 3-d model considered in this paper.

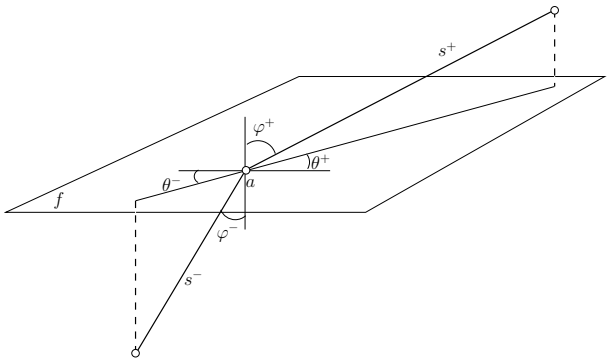

Figure 1: An illustration of the Snell's law of refraction.

Snell's Law of Refraction: *Let a be a bending point on a geodesic path π lying in the interior of a face f of D. Let the segments preceding and succeeding a in π be s^- and s^+, respectively. Let the weights of the cells containing s^- and s^+ be w^- and w^+, respectively. Then s^+ belongs to the plane containing s^- and perpendicular to f and the angles φ^- and φ^+ are related by $w^- \sin\varphi^- = w^+ \sin\varphi^+$.*

We refer to linear paths that are locally optimal (i.e., satisfy the Snell's law) as *geodesic paths*. Hence, the shortest path between pair of vertices u and v is the geodesic path of smallest cost joining them. In the following we discuss some of the implications of Snell's law on the local behavior of geodesic paths.

Hereafter, we denote by κ the ratio w^+/w^-. Without loss of generality, we assume that $w^- \geq w^+$, i.e., $\kappa \leq 1$. Let φ^* be the acute or right angle for which $\sin\varphi^* = \kappa$. We refer to this angle as the *critical angle* for the face f. From Snell's law, it follows that $\varphi^- \leq \varphi^*$. The case where $\varphi^- = \varphi^*$ deserves a special attention. In this case, φ^+ must be a right angle and therefore the segment s^+ is a face-using segment. Furthermore, if the second endpoint a_1 of s^+ is in the interior of f, then the segment following s^+ is inside the tetrahedron containing s^-, and the out-angle at a_1 is equal to φ^* (see Figure 3 (b)). In summary, if s is a face-using segment, then it is preceded and followed by segments lying in the cell with bigger weight and their corresponding in-angle and out-angles are equal to the critical angle φ^*.

In the next subsection, we study the properties of simple geodesic paths joining points in neighboring cells or in the same cell through a face-using segment. We define a function related to the cost of these geodesic paths and prove a number of properties that it possesses. These properties are essential to the design and the analysis of our algorithm.

2.2 Weighted distance function

Let F be a plane in the three-dimensional Euclidean space. We denote the two half-spaces defined by F by F^- and F^+ and assume that positive weights w^- and w^+ have been associated with them, respectively. We extend our model by assigning a weight w to F, so that $w = \min(w^-, w^+)$ if $w^- \neq w^+$, and $0 < w = w^+ (= w^-)$ if $w^- = w^+$. The latter case models the situation where the geodesic path joins two points in the same cell through a face-using segment on the boundary of that cell.

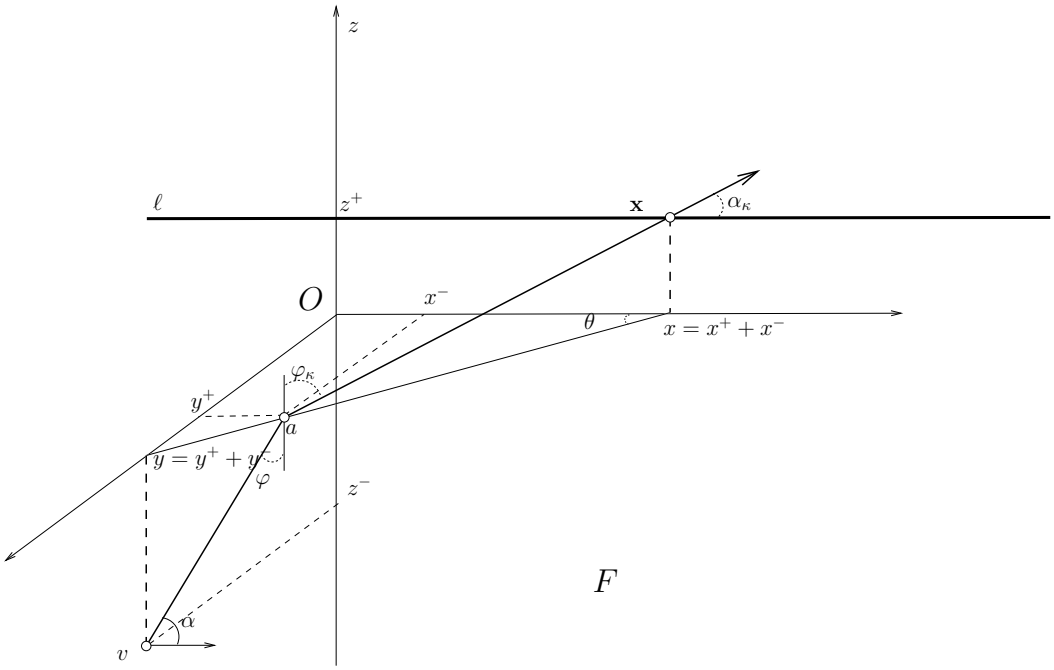

Figure 2: The geodesic path $\bar{\pi}(v, \mathbf{x})$ joining v and x is illustrated. The weighted distance function $c(v, x)$ equals to the cost of $\bar{\pi}(v, \mathbf{x})$, i.e. $c(v, x) = \|\bar{\pi}(v, \mathbf{x})\| = w^-|va| + w^+|a\mathbf{x}|$.

We refer to the half-spaces F^- and F^+ as the *lower* and the *upper* half-space, respectively. Let v be a point in the lower half-space F^- at distance z^- from F and ℓ be a line parallel to F in the upper half-space F^+ at distance z^+ from F. Let O_{xyz} be a Cartesian coordinate system such that the plane O_{xy} coincides with F, v has coordinates $(0, y, -z^-)$, and the line ℓ is described by $\{\ell : \ y = 0, \ z = z^+\}$.

We consider a point $\mathbf{x} = (x, 0, z^+)$ on ℓ and denote by $\bar{\pi}(v, \mathbf{x})$ the geodesic path between v and $\mathbf{x}$. In this setting, the geodesic path is unique and thus coincides with the shortest path. We denote the cost of this path by $c(v, x)$, where x is the x-coordinate of $\mathbf{x}$. So, for fixed l, $c(v, x)$ can be viewed as a function defined for any real x. We call c the *weighted distance function* from v to l (Figure 2).

The local structure of the geodesic path $\bar{\pi}(v, \mathbf{x})$ is governed by Snell's law. In the case where $w^- \neq w^+$, the shortest path between v and $\mathbf{x}$ consists of two segments (v, a) and $(a, \mathbf{x})$, where the bending point a is uniquely determined by Snell's law (Figure 3 (a)). In the case where $w^- = w^+$, the structure of the path $\bar{\pi}(v, \mathbf{x})$ is as follows. It is a single segment $(v, \mathbf{x})$, provided that the angle φ between $(v, \mathbf{x})$ and the direction normal to the plane F is smaller than or equal to the critical angle φ^* defined by $\sin \varphi^* = w/w^-$. Or, if $\varphi > \varphi^*$, it is in the plane perpendicular to F containing v and $\mathbf{x}$ and consists of three segments (v, a), (a, a_1) and $(a_1, \mathbf{x})$, where the acute angles between the segments (v, a) and $(a_1, \mathbf{x})$, and the direction normal to the plane F are equal to the critical angle φ^*, and the segment (a, a_1) is in F (Figure 3 (b)).

From these observations, it follows that, in all cases, weighted distance function can

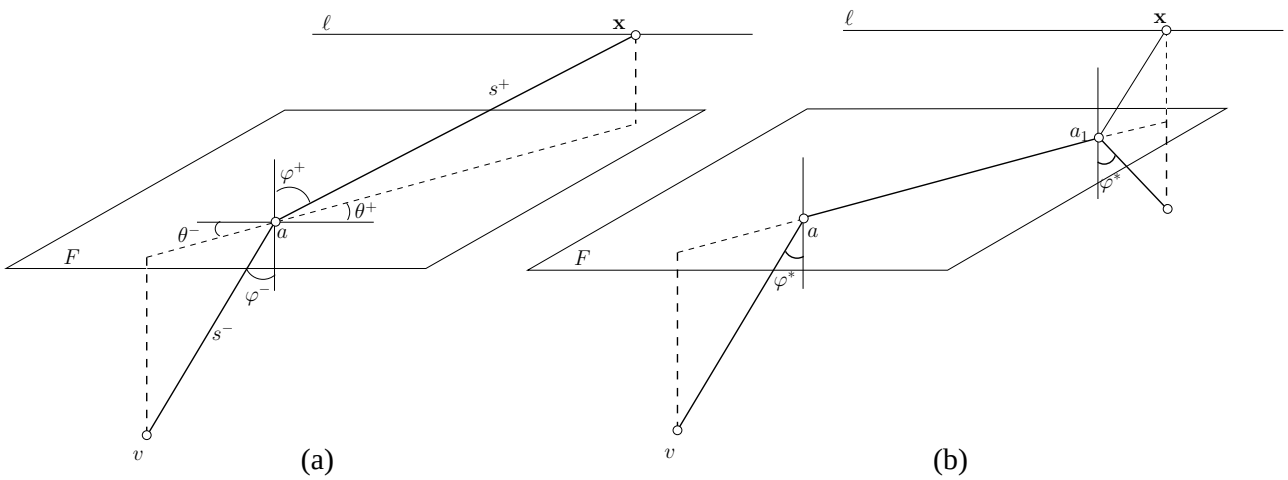

Figure 3: The diagrams illustrate the local structure of geodesic paths in different cases.

equivalently be defined by

$$c(v,x) = \|\bar{\pi}(v,\mathbf{x})\| = \min_{a,a_1 \in F} (w^-|va| + w|aa_1| + w^+|a_1\mathbf{x}|). \tag{1}$$

In the case where $w^- \neq w^+$, the minimum is achieved when $a = a_1 = (\tau x, (1-\tau)y, 0)$, where τ is the unique solution in $(0,1)$ of the equation

$$\frac{w^-\tau}{\sqrt{\tau^2(x^2+y^2)+(z^-)^2}} = \frac{w^+(1-\tau)}{\sqrt{(1-\tau)^2(x^2+y^2)+(z^+)^2}}, \tag{2}$$

where $v = (0, y, z^-)$ and $\mathbf{x} = (x, 0, z^+)$. The latter leads to an algebraic equation of degree four and it is infeasible[3] to evaluate $c(v,x)$ explicitly.

The case where $w^- = w^+$ is easier, as in that case the geodesic path is either a straight line, or a three segment path as described above and illustrated in Figure 3 (b) and the function $c(v,x)$ has an explicit representation, which is

$$c(v,x) = \begin{cases} w^-\sqrt{x^2+y^2+\bar{z}^2} & \text{if} \quad \sqrt{x^2+y^2} \leq \bar{z}/\bar{w} \\ w(\sqrt{x^2+y^2} - \bar{z}\bar{w}) & \text{if} \quad \sqrt{x^2+y^2} > \bar{z}/\bar{w}, \end{cases} \tag{3}$$

where $\bar{z} = z^- + z^+$ and $\bar{w} = \sqrt{(w^-/w)^2 - 1}$. We refer to this case as the *explicit case*. In the next lemma we state and prove some useful properties of the function $c(v,x)$.

Lemma 2.1 *For a fixed v, the weighted distance function $c(v,x)$ has the following properties:*

(a) *It is continuous and differentiable.*

(b) *It is symmetric, i.e. $c(v,x) = c(v,-x)$.*

(c) *It is strictly increasing for $x > 0$.*

[3]Although the roots of a quartic can be expressed as a rational function of radicals over its coefficients, they are too complex to be analytically manipulated and used here.

(d) *It is convex.*

(e) *It has asymptotes for $x \to \pm\infty$ as follows:*
(e1) if $w^+ < w^-$ then the asymptotes are $w^+(z^- \cot\varphi^* \pm x)$,
(e2) if $w^- < w^+$ then the asymptotes are $w^-(z^+ \cot\varphi^* \pm x)$,
(e3) in the explicit case $w^+ = w^- \geq w$ the asymptotes are $\pm wx$,
where φ^* is the critical angle.

Proof: In the explicit case ($w^- = w^+$), all these properties follow straightforwardly from the explicit representation (3). So, we consider the case $w^- \neq w^+$

From (1) and $a_1 = a = (\tau x, (1-\tau)y, 0)$ it follows that $c(v,x) = w^-\sqrt{\tau^2(x^2+y^2)+(z^-)^2} + w^+\sqrt{(1-\tau)^2(x^2+y^2)+(z^+)^2}$, where τ is the root of the equation (2). The root τ can be viewed as a function of x, which by the implicit function theorem is continuous and differentiable. Hence property (**a**) holds.

The property **(b)** follows from the observation that the value of the function $c(v,x)$ is determined by the distance between the projections of the points v and $\mathbf{x}$ on F, which is $\sqrt{y^2+x^2}$ where y is fixed.

To prove **(c)** we consider a point $\mathbf{x}' = (x', 0, z^+)$ such that $x' > x \geq 0$ and denote by τ' the corresponding root of (2). We have $c(v,x') = w^-\sqrt{\tau'^2(x'^2+y^2)+(z^-)^2} + w^+\sqrt{(1-\tau')^2(x'^2+y^2)+(z^+)^2}$. Using the fact that the function $c(v,x)$ is defined as the cost of the shortest path joining v and $\mathbf{x}$ we have

$$\begin{aligned} c(v,x) &\leq w^-\sqrt{\tau'^2(x^2+y^2)+(z^-)^2} + w^+\sqrt{(1-\tau')^2(x^2+y^2)+(z^+)^2} \\ &< w^-\sqrt{\tau'^2(x'^2+y^2)+(z^-)^2} + w^+\sqrt{(1-\tau')^2(x'^2+y^2)+(z^+)^2} = c(v,x'). \end{aligned}$$

In order to prove **(d)**, we show that for any three equidistant points $\mathbf{x}_1 < \mathbf{x}_0 < \mathbf{x}_2$ on ℓ, i.e., such that $2x_0 = x_1 + x_2$, the second finite difference $\triangle_2(c; x_1, x_0, x_2) = c(v,x_1) - 2c(v,x_0) + c(v,x_2)$ of the function $c(v,x)$ is positive. We denote by a_1 and a_2 the bending points of the shortest paths from v to $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. Let a_0' be the middle point of the segment (a_1, a_2). Then, using the definition of $c(v,x_0)$ and the convexity of the Euclidean distance function we obtain $2c(v,x_0) \leq 2(w^-|va_0'| + w^+|a'\mathbf{x}_0|) < w^-(|va_1| + |va_2|) + w^+(|a_1\mathbf{x}_1| + |a_2\mathbf{x}_2|) = c(v,x_1) + c(v,x_2)$, which implies $\triangle_2(c; x_1, x_0, x_2) > 0$ and **(d)**.

Finally, we prove **(e)**. Let us assume that $w^+ < w^-$. In this case, using Snell's law we observe that when $x \to +\infty$ the bending point $a(x)$ of the shortest path $\bar{\pi}(v,\mathbf{x})$ converges to the point $(z^- \tan\varphi^*, y, 0)$ (see Figure 2). Hence, we have $\lim_{x\to+\infty}(w^+\sqrt{(x - z^-\tan\varphi^*)^2 + y^2 + (z^+)^2} + w^-z^-/cos\varphi^* - c(v,x)) = 0$. On the other hand

$$\begin{aligned} &\lim_{x\to+\infty}(w^+\sqrt{(x - z^-\tan\varphi^*)^2 + y^2 + (z^+)^2} + w^-z^-/\cos\varphi^* - w^+(z^-\cot\varphi^* + x)) \\ &\qquad = w^+ \lim_{x\to+\infty}(\sqrt{(x - z^-\tan\varphi^*)^2 + y^2 + (z^+)^2} - (x - z^-\tan\varphi^*)) \\ &\qquad = w^+ \lim_{x\to+\infty} \frac{y^2 + (z^+)^2}{(\sqrt{(x - z^-\tan\varphi^*)^2 + y^2 + (z^+)^2} + (x - z^-\tan\varphi^*)} = 0. \end{aligned}$$

Combining these two limits we obtain $\lim_{x\to+\infty}(c(v,x) - w^+(z^-\cot\varphi^* + x)) = 0$ and thus (e1) is valid for $x \to +\infty$. The case where $x \to -\infty$ is symmetric.

In the case where $w^- < w^+$ we use Snell's law and observe that the bending point $a(x)$ of the shortest path $\bar{\pi}(v, \mathbf{x})$ converges to $x - z^+ \tan\varphi^*$, that is $\lim_{x\to+\infty}(a(x) - x - z^+ \tan\varphi^*) = 0$. Then (e2) is established analogously to (e1). □

2.3 Refractive Additive Voronoi diagram

Next we study Voronoi diagrams under the weighted distance metric defined above. Given a set S of k points $v_1, \ldots, v_k$ in F^-, called *sites*, and nonnegative real numbers $C_1, \ldots, C_k$, called *additive weights*, the *additive Voronoi diagram* for S is a subdivision of F^+ space into regions $\mathcal{V}(v_i, F^+) = \{x \in F^+ \mid \text{dist}(x, v_i) + C_i \leq \text{dist}(x, v_j) + C_j \text{ for } j \neq i\}$, where ties are resolved in favor of the site with smaller additive weight. The regions $\mathcal{V}(v_i, F^+)$ are called *Voronoi cells*. In the classic case, $\text{dist}(\cdot, \cdot)$ has been defined as the Euclidean distance. Here, for $\text{dist}(\cdot, \cdot)$, we use the weighted distance function $c(v, x)$.

Let v and v' be two points in F^-. We wish to study the intersection of the additive Voronoi diagram of v and v' with ℓ with respect to the weighted distance function. Without loss of generality, we assume that $C' = 0$ and $C \geq 0$, where C and C' are the additive weights assigned to v and v', respectively. We denote the intersection of the Voronoi cell of v with ℓ by $\mathcal{V}(v, v', \ell; C)$, or simply by $\mathcal{V}(v)$ when no ambiguity arises. We have

$$\mathcal{V}(v, v', \ell; C) = \mathcal{V}(v) = \{\mathbf{x} \in \ell \ : \ c(v, x) + C < c(v', x)\}.$$

In Theorem 2.1, we will show that if v and v' are at the same distance to F, then the Voronoi cell $\mathcal{V}(v)$ restricted to the line ℓ has a very nice structure (i.e., it is an interval). Furthermore, in Remark 2.1, we will show that if v and v' are not at the same distance to F then Theorem 2.1 does not hold. In our algorithm, presented in Section 5, we use the information about the shape of $\mathcal{V}(v)$ in order to propagate approximate shortest paths in $\mathcal{D}$ and it turns out that we need to only consider the sites that are restricted to be within a half-space of F and at the same distance to F. But, as we will see, this case in itself is mathematically challenging and provides valuable insights into the combinatorial structure of these diagrams.

Theorem 2.1 *The Voronoi cell $\mathcal{V}(v, v', \ell; C)$ is an interval on ℓ – possibly empty, finite or infinite.*

Proof: First, consider the case when $C = 0$. We denote the set of points $\mathbf{x}$ in F^+ such that $c(v, \mathbf{x}) = c(v', \mathbf{x})$ by $B(v, v')$ and observe that it is a half-plane perpendicular to F. Therefore, the set of points $\mathbf{x}$ on ℓ for which $c(v, x) = c(v', x)$ is either a single point, the whole line ℓ, or empty. Correspondingly, the Voronoi cell $\mathcal{V}(v, v', \ell; 0)$ is either a half-line, empty, or the whole line ℓ and the theorem holds for $C = 0$.

So, we assume that $C > 0$. We consider the equation $c(v', x) - c(v, x) = C$ and claim that it cannot have more than two solutions. Before we prove that claim (Claim 2.1 below), we argue that it implies the theorem.

Assume that the equation $c(v', x) - c(v, x) = C$ has at most two solutions. If it does not have any or has just one solution, then the theorem follows straightforwardly. In the case where it has exactly two solutions, the cell $\mathcal{V}(v, v', \ell; C)$ has to be either a finite interval on ℓ, or a complement of a finite interval on ℓ. From the definition of the Voronoi cell $\mathcal{V}(v, v', \ell; C)$

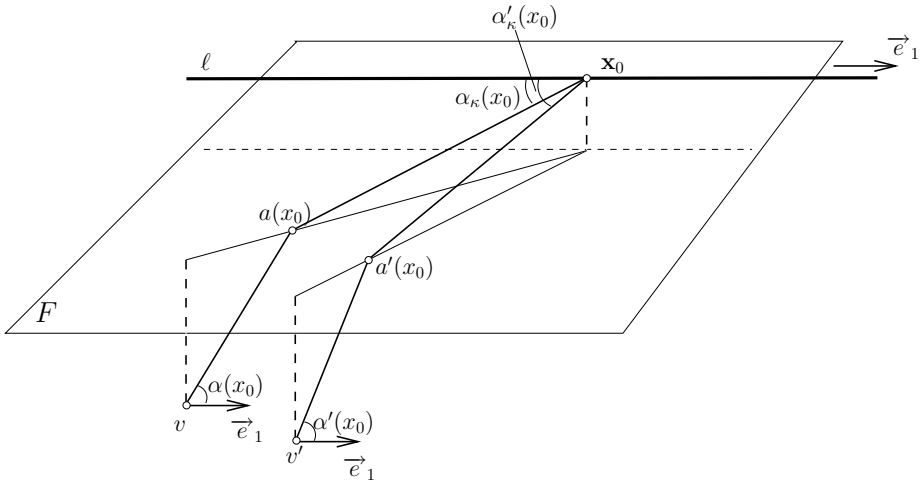

Figure 4: If the function $g(x)$ has a local extremum at the point x_0 then the angles $\alpha(x_0)$ and $\alpha'(x_0)$ must be equal.

and $C > 0$ it follows that $\mathcal{V}(v, v', \ell; C) \subset \mathcal{V}(v, v', \ell; 0)$. We know that $\mathcal{V}(v, v', \ell; 0)$ is either empty, a half-line, or the whole line. If it is either empty or a half-line then $\mathcal{V}(v, v', \ell; C)$ must be either empty or a finite interval and the theorem holds.

It remains to consider the case where $\mathcal{V}(v, v', \ell; 0)$ is the whole line ℓ. We argue that $\mathcal{V}(v, v', \ell; C)$ can not be a complement to a finite interval. We have $\mathcal{V}(v, v', \ell; 0) = \ell$ and therefore the line ℓ must be parallel to the half-plane $B(v, v')$. Furthermore, the plane containing $B(v, v')$ is a perpendicular bisector of the segment (v, v') and thus the point v' must have coordinates $(0, y', z^-)$ (Figure 2). In this setting, using Lemma 2.1(e), we observe that $c(v, x)$ and $c(v', x)$ have same asymptotes at infinity and thus $\lim_{x\to\infty}(c(v', x) - c(v, x)) = 0$. Therefore, the cell $\mathcal{V}(v, v', \ell; C)$ must be finite. The theorem follows. □

Next we establish the validity of the claim used in the proof of Theorem 2.1.

Claim 2.1 *The equation $c(v', x) - c(v, x) = C$ has at most two solutions.*

We will prove the claim by showing that the function $g(x) = c(v', x) - c(v, x)$ is unimodal, i.e., it has at most one local extremum. We establish this property in two steps. First, we prove a characterization property of a local extremum of g (Proposition 2.1). Then, we show that there may be no more than one point possessing that property.

We focus our discussion on the case $w^- \neq w^+$, since the other case is either simpler or can be treated analogously. We denote by $a(x)$ and $a'(x)$ the bending points defining the shortest paths from v and v' to $\mathbf{x}$, respectively. We assume that ℓ is oriented and denote by $\overrightarrow{e_1}$ the positive direction unit vector on ℓ. Furthermore, let $\alpha(x)$ and $\alpha'(x)$ be the angles between vectors $\overrightarrow{va(x)}$, $\overrightarrow{v'a'(x)}$ and $\overrightarrow{e_1}$, respectively (see Figure 4).

These angles are completely defined by the angles φ and θ defining the corresponding shortest paths at the bending points $a(x)$ and $a'(x)$. Precisely, we have $\cos\alpha = \sin\varphi\cos\theta$. Next, we prove that the angles $\alpha(x_0)$ and $\alpha'(x_0)$ must be equal at any local extremum x_0.

Proposition 2.1 *If x_0 is a local extremum of the function g, then $\alpha(x_0) = \alpha'(x_0)$.*

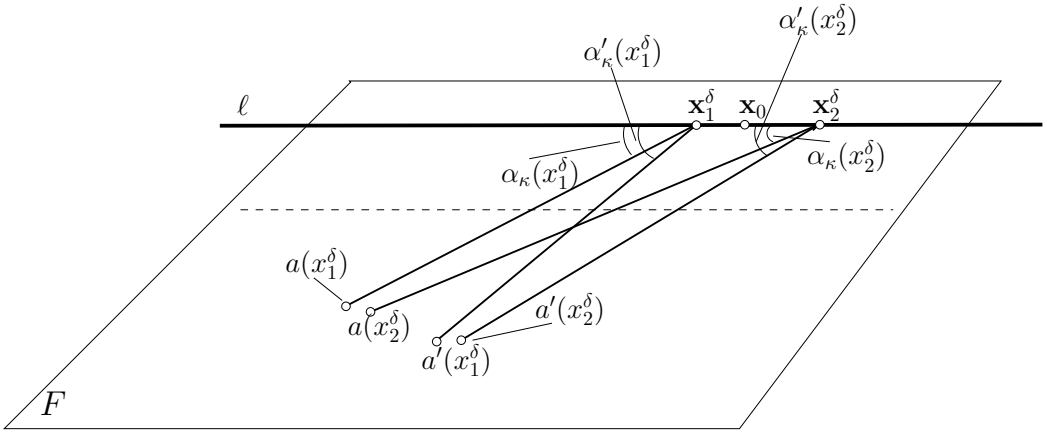

Figure 5: Illustration of the proof of inequality (5).

Proof: The proof is by contradiction. Let us assume that $\alpha(x_0) \neq \alpha'(x_0)$. We denote by $\alpha_\kappa(x_0)$ the angle between vectors $\overrightarrow{a(x_0)\mathbf{x}_0}$ and $\overrightarrow{e_1}$. Similarly, $\alpha'_\kappa(x_0)$ denotes the angle between vectors $\overrightarrow{a'(x_0)\mathbf{x}_0}$ and $\overrightarrow{e_1}$ (Figure 4). The relation $\cos\alpha = \cos\theta \sin\varphi$ and Snell's law readily imply that $\kappa \cos\alpha_\kappa(x_0) = \cos\alpha(x_0)$ and $\kappa \cos\alpha'_\kappa(x_0) = \cos\alpha'(x_0)$, where $\kappa = w^+/w^-$. Thus, we have that $\alpha_\kappa(x_0) \neq \alpha'_\kappa(x_0)$. Without loss of generality, we assume that $\alpha_\kappa(x_0) < \alpha'_\kappa(x_0)$. Under these assumptions, we show the existence of two points on $\mathbf{x}_1$ and $\mathbf{x}_2$ on ℓ, such that:

$$x_1 < x_0 < x_2, \quad g(x_1) = g(x_2), \quad \text{and} \quad |a(x_2)\mathbf{x}_2| + |a'(x_1)\mathbf{x}_1| > |a(x_2)\mathbf{x}_1| + |a'(x_1)\mathbf{x}_2|. \quad (4)$$

By the assumption that x_0 is a local extremum of g, it follows that, for any positive real number $\delta > 0$ inside the interval $(x_0 - \delta, x_0 + \delta)$, there are reals x_1^δ and x_2^δ, such that

$$x_0 - \delta < x_1^\delta < x_0 < x_2^\delta < x_0 + \delta \text{ and } g(x_1^\delta) = g(x_2^\delta).$$

On the other hand, if δ converges to zero, then $a(x_2^\delta)$ converges to $a(x_0)$, and $a'(x_1^\delta)$ converges to $a'(x_0)$. Therefore, the inequality $\alpha_\kappa(x_0) < \alpha'_\kappa(x_0)$ implies that for a small enough δ_0, the inequalities

$$\alpha_\kappa(x_1^{\delta_0}) < \alpha'_\kappa(x_1^{\delta_0}) \quad \text{and} \quad \alpha_\kappa(x_2^{\delta_0}) < \alpha'_\kappa(x_2^{\delta_0})$$

hold. From these inequalities, it follows that if we make the quadrilateral $a(x_2^\delta)a'(x_1^\delta)x_2^\delta x_1^\delta$ planar by rotation of the point $a(x_2^\delta)$ around ℓ, then the obtained planar quadrilateral will be convex (Figure 5). Therefore we have

$$|a(x_2^{\delta_0})\mathbf{x}_2^{\delta_0}| + |a'(x_1^{\delta_0})\mathbf{x}_1^{\delta_0}| > |a(x_2^{\delta_0})\mathbf{x}_1^{\delta_0}| + |a'(x_1^{\delta_0})\mathbf{x}_2^{\delta_0}|, \quad (5)$$

which proves (4) for $\mathbf{x}_1 = \mathbf{x}_1^{\delta_0}$ and $\mathbf{x}_2 = \mathbf{x}_2^{\delta_0}$.

Next, we estimate the sum $c(v, x_1) + c(v', x_2)$ we show that there may be no more than

one point possessing that property. We use (4) and obtain

$$\begin{aligned} c(v,x_1)+c(v',x_2) &= c(v,x_2)+c(v',x_1) \\ &= w^-|va(x_2)|+w^+|a(x_2)\mathbf{x}_2|+w^-|v'a'(x_1)|+w^+|a'(x_1)\mathbf{x}_1| \\ &= w^-(|va(x_2)|+|v'a'(x_1)|)+w^+(|a(x_2)\mathbf{x}_2|+|a'(x_1)\mathbf{x}_1|) \\ &> w^-(|va(x_2)|+|v'a'(x_1)|)+w^+(|a(x_2)\mathbf{x}_1|+|a'(x_1)\mathbf{x}_2|) \\ &= w^-|va(x_2)|+w^+|a(x_2)\mathbf{x}_1|+w^-|v'a'(x_1)|+w^+|a'(x_1)\mathbf{x}_2|. \end{aligned}$$

On the other hand, by the definition of the weighted distance function (1), we have the inequalities

$$c(v,x_1)\le w^-|va(x_2)|+w^+|a(x_2)\mathbf{x}_1| \quad \text{and} \quad c(v',x_2)\le w^-|v'a'(x_1)|+w^+|a'(x_1)\mathbf{x}_2|,$$

that contradict the previous strict inequality. Therefore, the angles $\alpha_\kappa(x_0)$ and $\alpha'_\kappa(x_0)$ and consequently $\alpha(x_0)$ and $\alpha'(x_0)$ must be equal. □

Our next step is to show that there cannot be two points on ℓ satisfying Proposition 2.1. To do that, we study in more detail the relationship between the position of points v and $\mathbf{x}$, and the angle $\alpha(x)$. We observe that, for any fixed y (the y-coordinate of v), there is a one-to-one correspondence between the real numbers x and the angles α. That is, for a fixed v, there is a one-to-one correspondence between the points $\mathbf{x}$ on ℓ and the angle between the shortest path $\bar{\pi}(v,\mathbf{x})$ and the positive direction on ℓ. Hence, we may consider and study the well defined function $x=x(y,\alpha)$. We prove the following:

Proposition 2.2 *The second mixed derivative of the function $x=x(y,\alpha)$ exists and is negative, i.e., $x_{y\alpha}<0$.*

Let us first show that Proposition 2.2 implies Claim 2.1.

Proof of Claim 2.1: We assume that Proposition 2.2 holds and will show that the function $g(x)$ has at most one local extremum. Recall that the point v has coordinates $(0,y,z^-)$ and let us denote the coordinates of the point v' by (x',y',z^-).

We first consider the case where $y=y'$. In this case, we observe that $\alpha'(x)=\alpha(x+x')$. In addition, the function $\alpha(x)$ is strictly monotone and, therefore, for any x, the angels $\alpha(x)$ and $\alpha'(x)$ are different. From Proposition 2.1 it follows that in this case the function $g(x)$ has no local extremum.

Next, we consider the case $y\neq y'$. We assume, for the sake of contradiction, that $g(x)$ has two local extrema, say x_1 and x_2. By Proposition 2.1, $\alpha(x_1)=\alpha'(x_1)$ and $\alpha(x_2)=\alpha'(x_2)$. We denote $\alpha_1=\alpha(x_1)=\alpha'(x_1)$ and $\alpha_2=\alpha(x_2)=\alpha'(x_2)$. Then, the difference x_2-x_1 can be represented, using the function $x(y,\alpha)$, in two ways

$$x_2-x_1=\int_{\alpha_1}^{\alpha_2} x_\alpha(y,\alpha)\,\mathrm{d}\alpha \qquad \text{and} \qquad x_2-x_1=\int_{\alpha_1}^{\alpha_2} x_\alpha(y',\alpha)\,\mathrm{d}\alpha. \tag{6}$$

Subtracting the last two equalities, we get

$$0=\int_{\alpha_1}^{\alpha_2} x_\alpha(y,\alpha)\,\mathrm{d}\alpha-\int_{\alpha_1}^{\alpha_2} x_\alpha(y',\alpha)\,\mathrm{d}\alpha=\int_{y'}^{y}\int_{\alpha_1}^{\alpha_2} x_{y\alpha}(y,\alpha)\,\mathrm{d}\alpha\,\mathrm{d}y. \tag{7}$$

The integral on the right side is negative since by Proposition 2.2, the derivative $x_{y\alpha}$ is negative, $\alpha_1 \neq \alpha_2$, and $y \neq y'$. Hence, we have a contradiction and Claim 2.1 follows. □

The proof of Proposition 2.2 is rather long and uses elaborate mathematical techniques and manipulations. On the other hand, the rest of the paper is independent of the details in that proof. So, we present the full proof for the interested readers in Appendix A.1.

Corollary 2.1 *Consider a plane H in F^+ parallel to F and two points v and v' in F^- lying at the same distance from F. For any non-negative constant C, the Voronoi cell $\mathcal{V}(v, v', \mathcal{H}; C) = \{x \in \mathcal{H} : c(v, x) + C < c(v', x)\}$ is convex.*

What can we say about the Voronoi cell of v in F^+? The above corollary implies that the intersection between the Voronoi cell and any plane, parallel to F, is convex. This, as such, is not sufficient to conclude that the cell is convex. We close this section with the following conjecture.

Conjecture 2.1 *In the above setting, the Voronoi cell of v in F^+, $\mathcal{V}(v, v', F^+; C) = \{x \in F^+ : c(v, x) + C < c(v', x)\}$, is convex.*

Remark 2.1 *Examples showing that equal distance of the points v and v' from the bending plane F is a necessary condition for $c(v', x) - c(v, x)$ to be unimodal in Theorem 2.1 is not difficult to construct. In fact, if we take arbitrary points v and v' in F^- at different distances from F, it is very likely that $c(v', x) - c(v, x)$ will have more than one local extrema and hence, for a proper choice of C, the equation $c(v', x) - c(v, x) = C$ will have more than two solutions. Such examples can be constructed by choosing arbitrary points v in F^- and x_1, x_2 on ℓ and then computing a point $v' \in F^-$ so that $\alpha(v, x_i) = \alpha(v', x_i)$, for $i = 1, 2$, where α's are the angles between the line ℓ and the shortest paths coming from v and v', respectively. As a result $c(v', x) - c(v, x)$ will have local extrema at x_1 and x_2.*

3 Discretization of $\mathcal{D}$

In this section, we describe the definition of a carefully chosen set of additional points placed in $\mathcal{D}$, called *Steiner points*. These Steiner points collectively form a discretization of $\mathcal{D}$, which is later used to approximate geodesic paths in $\mathcal{D}$. Steiner points are placed on the edges of $\mathcal{D}$ and on the bisectors of the dihedral angles of the tetrahedra in $\mathcal{D}$. While it may seem more natural to place the Steiner points on the faces of the tetrahedra, placing them on the bisectors proves to be more efficient, leading to a speed up of approximately ε^{-1} compared to the alternate placement. Recall that ε is an approximation parameter in $(0, 1)$. We provide a precise estimate on the number of Steiner points which depends on ε and aspect ratios of tetrahedra of $\mathcal{D}$.

3.1 Placement of Steiner points

We use the following definitions:

Definition 3.1
(a) For a point $x \in \mathcal{D}$, we define $D(x)$ to be the union of the tetrahedra incident to x. We denote by $\partial D(x)$ the set of faces on the boundary of $D(x)$ that are not incident to x.

(b) We define $d(x)$ to be the minimum Euclidean distance from x to any point on $\partial D(x)$.

(c) For each vertex $v \in \mathcal{D}$, we define a radius $r(v) = d(v)/14$.

(d) For any internal point x on an edge in $\mathcal{D}$, we define a radius $r(x) = d(x)/24$. The radius of an edge $e \in \mathcal{D}$ is $r(e) = \max_{x \in e} r(x)$.

Using radii $r(v)$ and $r(x)$ and our approximation parameter ε, we define "small" regions around vertices and edges of $\mathcal{D}$, called vertex and edge vicinities, respectively.

Definition 3.2
(a) The convex hull of the intersection points of the ball $B(v, \varepsilon r(v))$ having center v and radius $\varepsilon r(v)$ with the edges incident to v is called the vertex-vicinity *of v and is denoted by $D_\varepsilon(v)$.*

(b) The convex hull of the intersections between the "spindle" $\cup_{x \in e} B(x, \varepsilon r(x))$ and the faces incident to e is called the edge-vicinity *of e and is denoted by $D_\varepsilon(e)$.*

On each edge $e = AB$ of $\mathcal{D}$, we define a set of Steiner points as follows. Denote by AA' and BB' the intersections of e with vertex vicinities $D_\varepsilon(A)$ and $D_\varepsilon(B)$, respectively. Points A' and B' are Steiner points. All other Steiner points on e are placed between A' and B'. Let M_e be the point on e, such that $d(M_e) = \max_{x \in e} d(x)$. The point M_e is defined to be a Steiner point. Next, we define a sequence of points M_i, for $i = 0, 1, \ldots$ on $M_e A'$, by

$$M_0 = M_e \quad \text{and} \quad |M_{i-1} M_i| = \varepsilon r(M_i) \quad \text{for} \quad i = 1, 2, ... \tag{8}$$

All such points M_i between M_e and A' are defined as Steiner points. Analogously, we define the set of Steiner points on $M_e B'$. The number of Steiner points defined in this way on e is bounded by

$$C_e \frac{1}{\varepsilon} \log \frac{2}{\varepsilon}, \quad \text{where} \quad C_e < \frac{33}{\sin \alpha(e)} \log \frac{|AB|}{\sqrt{r(A) r(B)}}$$

and $\alpha(e)$ is the minimum angle between e and the faces on $\partial D(e)$. (All logarithms with unspecified base are assumed to be base 2.) The total number of Steiner points placed on the edges of $\mathcal{D}$ is bounded by $6\Gamma_1 \frac{n}{\varepsilon} \log \frac{2}{\varepsilon}$, where Γ_1 is the average of the constants C_e over all edges of $\mathcal{D}$.

The remaining Steiner points lie on the bisectors of the dihedral angles of tetrahedra in $\mathcal{D}$. Steiner points in any tetrahedron T of $\mathcal{D}$ are defined to lie on the six bisectors of the dihedral angles of T. Let the vertices of T be A, B, C, and D and let us consider one of the bisectors of the dihedral angles of T, say ABP (see Figure 6(a)). Next, we describe the placement of Steiner points in the triangle ABP.

Let the dihedral angle at AB of T be γ and let PH be the height (altitude) of ABP (see Figure 6(b)). First, we define an infinite sequence of points $P_0, P_1, \ldots$ on PH by

$$P_0 = P, \quad |P_{i-1} P_i| = \sqrt{\varepsilon/8} |H P_i| \sin(\gamma/2), \quad \text{for } i = 1, 2, \ldots \tag{9}$$

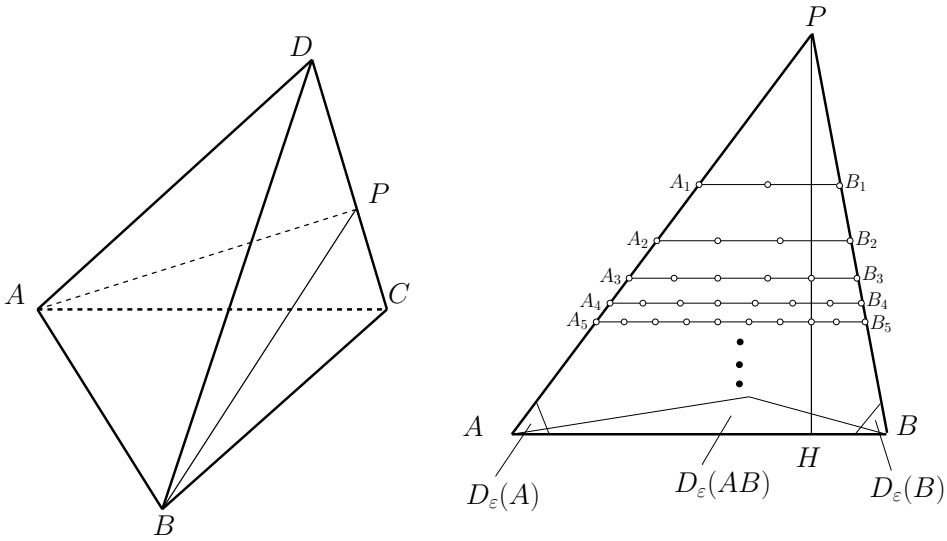

Figure 6: (a) A tetrahedron $ABCD$ and one of its bisectors ABP. (b) Placement of Steiner points on ABP.

Then, we consider the sequence of lines L_i in the plane ABP, parallel to AB, and containing P_i, for $i = 1, 2, \ldots$. Let the intersection points of these lines with AP and BP be A_i and B_i, respectively. Points A_i and B_i lying outside of the vertex vicinities $D_\varepsilon(A)$ and $D_\varepsilon(B)$ are defined to be Steiner points, respectively. The intersection points of these lines with the boundary of the union of the edge vicinity $D_\varepsilon(AB)$ and vertex vicinities $D_\varepsilon(A)$ and $D_\varepsilon(B)$, are defined to be Steiner points. On each of the segments A_iB_i, we define a set of k_i equidistantly placed Steiner points $P_{i,j}$, $j = 1, \ldots, k_i$, where

$$k_i = \left\lfloor \frac{|A_iB_i|}{|P_iP_{i+1}|} \right\rfloor \quad \text{and} \quad |P_{i,j}P_{i,j+1}| = \frac{|A_iB_i|}{k_i + 1}, \quad \text{for} \quad j = 0, 1, \ldots, k_i. \tag{10}$$

In the above expression, we have assumed that $P_{i,0} = A_i$ and $P_{i,k_i+1} = B_i$.

Definition 3.3 *The set of Steiner points in the triangle ABP consists of*
(a) all points $P_{i,j}$ outside the union $D_\varepsilon(AB) \cup D_\varepsilon(A) \cup D_\varepsilon(B)$,
(b) the intersection points of the lines L_i with the boundary of that union.

Next, we estimate the number of Steiner points placed in the triangle ABP. We denote $h = p_0 = |PH|$ and $p_i = |P_iH|$ for $i = 1, 2, \ldots$ In this notation, we have $p_{i-1} - p_i = |P_iP_{i-1}|$ and $|P_iP_{i-1}| = p_i\sqrt{\varepsilon/8}\sin(\gamma/2)$, which implies

$$p_i = h\lambda^i, \quad \text{where} \quad \lambda = (1 + \sqrt{\varepsilon/8}\sin(\gamma/2))^{-1}. \tag{11}$$

Let i_1 be the smallest index such that the line L_{i_1} is at distance smaller than $\varepsilon r(e)$ from AB. We denote by K_1 the number of Steiner points lying on lines L_i, with $i < i_1$, and by K_2 the number of the remaining Steiner points in ABP. Let us estimate the number K_1 first.

The number of Steiner points on a line L_i, with $i < i_1$, is $k_i + 2$. Using (10) and (11), we have

$$k_i = \left\lfloor \frac{|A_iB_i|}{|P_iP_{i+1}|} \right\rfloor = \left\lfloor \frac{(h-p_i)|AB|}{h(p_i-p_{i+1})} \right\rfloor = \left\lfloor \frac{(1-\lambda^i)|AB|}{h\lambda^i(1-\lambda)} \right\rfloor .$$

Thus, for the number K_1, we obtain

$$\begin{aligned} K_1 &= \sum_{i=1}^{i_1-1}(2+k_i) \le 2(i_1-1) + \frac{|AB|}{h(1-\lambda)} \sum_{i=1}^{i_1-1} \frac{1-\lambda^i}{\lambda^i} \qquad (12) \\ &= 2(i_1-1) + \frac{|AB|}{h(1-\lambda)} \left(\frac{1-\lambda^{i_1}}{(1-\lambda)\lambda^{i_1-1}} - i_1 \right) \le 2(i_1-1) + \frac{|AB|}{h(1-\lambda)^2} \frac{1-\lambda^{i_1-1}}{\lambda^{i_1-1}} . \end{aligned}$$

From the definition of i_1 and (11), we have $h\lambda^{i_1} < \varepsilon r(e) \le h\lambda^{i_1-1}$. Therefore,

$$i_1 - 1 = \left\lfloor \log_\lambda \frac{\varepsilon r(e)}{h} \right\rfloor . \qquad (13)$$

From (12) and (13), we obtain

$$K_1 < \frac{|AB|}{\varepsilon r(e)(1-\lambda)^2} + 2\log_{\lambda^{-1}} \frac{h}{\varepsilon r(e)}. \qquad (14)$$

Next, we estimate K_2, that is the number of Steiner points lying on segments A_iB_i with $i \ge i_1$. By our definition, on the segment $A_{i_1}B_{i_1}$ there is a point M_1, such that the triangle ABM_1 lies entirely inside the edge vicinity. Let M_2' be the intersection point of the boundaries of the edge vicinity $D_\varepsilon(AB)$ and the vertex vicinity $D_\varepsilon(A)$ that lies in the triangle ABP. Similarly, let M_2'' be the intersection point of the boundaries of the edge vicinity $D_\varepsilon(AB)$, the vertex vicinity $D_\varepsilon(B)$, and the triangle ABP. Furthermore, let i_2' be the smallest index such that the segment $A_{i_2'}B_{i_2'}$ is closer to AB than M_2' and similarly, let i_2'' be the smallest index so that the segment $A_{i_2''}B_{i_2''}$ is closer to AB than M_2''. All Steiner points on segments A_iB_i, with $i \ge i_1$, lie in the quadrilaterals $A_{i_1}A_{i_2'}M_2'M_1$ and $B_{i_1}B_{i_2''}M_2''M_1$. We denote the number of Steiner points in these two quadrilaterals by K_2' and K_2'', respectively.

To estimate K_2', we show an upper bound on the number of Steiner points on A_iB_i, $i_1 \le i < i_2'$ that lie inside the quadrilateral $A_{i_1}A_{i_2'}M_2'M_1$. Namely, if we denote this number by k_i' and by M_i the intersection point between A_iB_i and AM_1, we have,

$$k_i' \le 2 + \frac{|A_iM_i|}{p_i - p_{i+1}} = 2 + \frac{|A_{i_1}M_1|p_i}{p_{i_1}(p_i-p_{i+1})} = 2 + \frac{|A_{i_1}M_1|}{h\lambda^{i_1}(1-\lambda)}.$$

Thus, the number of Steiner points inside the quadrilateral $A_{i_1}A_{i_2'}M_2'M_1$ is bounded by $K_2' \le (i_2'-i_1)(2+\frac{|A_{i_1}M_1|}{h\lambda^{i_1}(1-\lambda)})$. Analogously, for the number of Steiner points inside the quadrilateral $B_{i_1}B_{i_2''}M_2''M_1$, we obtain $K_2'' \le (i_2''-i_1)(2+\frac{|B_{i_1}M_1|}{h\lambda^{i_1}(1-\lambda)})$. We sum the estimates on K_2' and K_2'', use (11), (13) and obtain

$$\begin{aligned} K_2 = K_2' + K_2'' \le (i_2' + i_2'' - 2i_1) \left(2 + \frac{|A_{i_1}B_{i_1}|}{h\lambda^{i_1}(1-\lambda)} \right) \le \\ (i_2' + i_2'' - 2i_1) \left(2 + \frac{|AB|(1-\lambda^{i_1})}{h\lambda^{i_1}(1-\lambda)} \right) < (i_2' + i_2'' - 2i_1) \left(2 + \frac{|AB|}{\varepsilon r(e)\lambda(1-\lambda)} \right). \qquad (15) \end{aligned}$$

From the definitions of the indices i'_2, i''_2, we easily derive that

$$p_{i'_2-1} > \frac{\sqrt{2}\varepsilon^2 r(e) r(A)}{|AM_e|} \cos\frac{\angle BAP}{2}, \qquad p_{i''_2-1} > \frac{\sqrt{2}\varepsilon^2 r(e) r(B)}{|BM_e|} \cos\frac{\angle PBA}{2},$$

where M_e is the point on AB where the radius $r(e)$ is achieved. These inequalities and (11) imply

$$i'_2 \leq 1 + \log_{\lambda^{-1}} \frac{h|AM_e|}{\sqrt{2}\varepsilon^2 r(e) r(A) \cos\frac{\angle BAP}{2}} \quad \text{and} \quad i''_2 \leq 1 + \log_{\lambda^{-1}} \frac{h|M_eB|}{\sqrt{2}\varepsilon^2 r(e) r(B) \cos\frac{\angle PBA}{2}}.$$

Then, we use (13) and obtain

$$\begin{aligned} i'_2 + i''_2 - 2i_1 \;\leq\; & 2 + 2\log_{\lambda^{-1}} \frac{h|AB|}{\varepsilon^2 r(e)\sqrt{r(A)r(B)}} - 2\log_{\lambda^{-1}} \frac{h}{\varepsilon r(e)} = \\ & 2 + 2\log_{\lambda^{-1}} \frac{|AB|}{\varepsilon\sqrt{r(A)r(B)}} = 2\log_{\lambda^{-1}} \frac{|AB|}{\varepsilon\lambda\sqrt{r(A)r(B)}}\ . \end{aligned} \tag{16}$$

Combining (14), (15) and (16), we obtain

$$\begin{aligned} K_1 + K_2 \leq 2\frac{|AB|}{\varepsilon r(e)\lambda(1-\lambda)} \log_{\lambda^{-1}} \frac{|AB|}{\varepsilon\lambda\sqrt{r(A)r(B)}} \;+\; & \frac{|AB|}{\varepsilon r(e)(1-\lambda)^2} \\ +\; 2\log_{\lambda^{-1}} \frac{h}{\varepsilon r(e)} + 4\log_{\lambda^{-1}} & \frac{|AB|}{\varepsilon\lambda\sqrt{r(A)r(B)}}. \end{aligned} \tag{17}$$

From the last equation, we easily derive that

$$K_1 + K_2 = C_{ABP}(T)\frac{1}{\varepsilon^2}\log\frac{2}{\varepsilon},$$

where the constant $C_{ABP}(T)$ depends on the geometry of the tetrahedron T and is bounded by (see Appendix A.2 for details)

$$C_{ABP}(T) \leq 23\frac{|AB|}{r(e)\sin^2(\gamma/2)} \log\frac{4|AB|^2 h}{r(e)r(A)r(B)}. \tag{18}$$

Our discussion is summarized in the following lemma.

Lemma 3.1 *(a) The number of Steiner points placed on a bisector ABP of a dihedral angle γ in a tetrahedron T, is bounded by $C_{ABP}(T)\frac{1}{\varepsilon^2}\log\frac{2}{\varepsilon}$, where the constant $C_{ABP}(T)$ depends on the geometric features of $\mathcal{D}$ around the edge AB and is bounded by $23\frac{|AB|}{r(e)\sin^2(\gamma/2)}\log\frac{4|AB|^2h}{r(e)r(A)r(B)}$.*

(b) The number of segments that are parallel to AB on a bisector ABP, containing Steiner points, is bounded by $C^1_{ABP}(T)\frac{1}{\sqrt{\varepsilon}}\log\frac{2}{\varepsilon}$, where $C^1_{ABP}(T) < \frac{4}{\sin(\gamma/2)}\log_2\frac{4|AB|^2h}{r(e)r(A)r(B)}$.

(c) The total number of Steiner points is bounded by $C(\mathcal{D})\frac{n}{\varepsilon^2}\log\frac{2}{\varepsilon}$, where n is the number of tetrahedra in $\mathcal{D}$ and $C(\mathcal{D})$ is the average of $C_{ABP}(T)$ over all $6n$ bisectors in $\mathcal{D}$.

By placing Steiner points in this way, in the next lemma, we show that it is possible to approximate the cell crossing segments that have their endpoints outside the vertex and the edge vicinities.

Lemma 3.2 *Let ABP be the bisector of a dihedral angle γ formed by the faces ABC and ABD of a tetrahedron $ABCD$. Let x_1 and x_2 be points on the faces ABC and ABD, respectively, that lie outside of the union $D_\varepsilon(AB) \cup D_\varepsilon(A) \cup D_\varepsilon(B)$. Then, there exists a Steiner point q on ABP, such that $\max(\angle x_2x_1q, \angle x_1x_2q) \leq \sqrt{\frac{\varepsilon}{2}}$ and $|x_1q| + |qx_2| \leq (1+\varepsilon/2)|x_1x_2|$.*

Proof: Clearly, the segment x_1x_2 intersects the bisector triangle ABP in a point x_0 lying outside the vertex vicinities $D_\varepsilon(A)$, $D_\varepsilon(B)$, and the edge vicinity $D_\varepsilon(AB)$. Recall that Steiner points in ABP are placed on a set of lines L_i parallel to AB and passing through the sequence of points P_i on the altitude PH of ABP. Let i_0 be the maximum index such that the line L_{i_0} is farther away from AB than from x_0. We define q to be the closest Steiner point to x_0 on the line L_{i_0}.

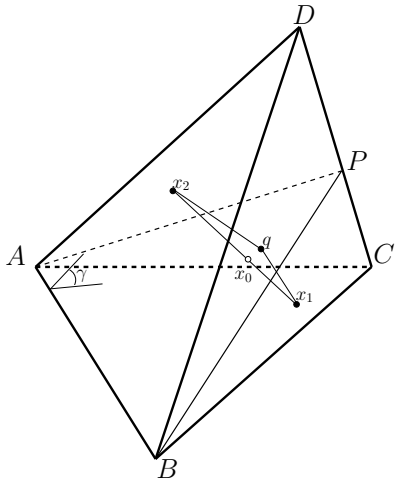

Figure 7: Illustrates Lemma 3.2.

First, we estimate the angles $\angle x_2x_1q = \angle x_0x_1q$ and $\angle x_1x_2q = \angle x_0x_2q$. By our definition of the Steiner points and Pythagorean theorem, it follows that

$$|x_0q| \leq \frac{\sqrt{5}}{4} h\lambda^{i_0} \sqrt{\frac{\varepsilon}{2}} \sin\frac{\gamma}{2}, \tag{19}$$

where h and λ are as defined above (see (11)). Let ρ be the radius of the smallest sphere containing x_0 and q and touching the face ABC. It is easily observed that

$$2\rho > (h\lambda^{i_0} - \frac{|x_0q|}{2}) \sin\frac{\gamma}{2} > \frac{8\sqrt{2}-\sqrt{5}}{8\sqrt{2}} h\lambda^{i_0} \sin\frac{\gamma}{2}\ . \tag{20}$$

If we denote the angle $\angle x_0x_1q$ by θ_1, then $\sin\theta_1 \leq \frac{|x_0q|}{2\rho}$, and using (19) and (20), we obtain

$$\sin\theta_1 \leq \frac{|x_0q|}{2\rho} < \sqrt{\frac{\varepsilon}{2}}. \tag{21}$$

The same estimate applies to angle $\angle x_0x_2q$. Hence, the first inequality of the lemma holds. Next, we prove the second inequality. We denote by θ, θ_1, and θ_2 the angles of the triangle qx_1x_2 at q, x_1 and x_2, respectively (Figure 7). By a trigonometric equality valid in any triangle, we have

$$|x_1q| + |qx_2| = \left(1 + \frac{2\sin(\theta_1/2)\sin(\theta_2/2)}{\sin(\theta/2)}\right)|x_1x_2|.$$

Thus, it suffices to prove that $\frac{2\sin(\theta_1/2)\sin(\theta_2/2)}{\sin(\theta/2)} \leq \varepsilon/2$. By (21), it follows that $\sin\theta_1$ and $\sin\theta_2$ are smaller than $\sqrt{\varepsilon/2}$ and from $\varepsilon \leq 1$ we have $\theta \geq \pi/2$. Therefore, we obtain

$$\begin{aligned}\frac{2\sin(\theta_1/2)\sin(\theta_2/2)}{\sin(\theta/2)} &= \frac{\sin\theta_1\sin\theta_2}{2\sin(\theta/2)\cos(\theta_1/2)\cos(\theta_2/2)} \leq \frac{\varepsilon}{4\sin(\theta/2)\cos(\theta_1/2)\cos(\theta_2/2)} \\ &= \frac{\varepsilon}{4\sin(\theta/2)(\sin(\theta/2)+\sin(\theta_1/2)\sin(\theta_2/2))} \leq \frac{\varepsilon}{4\sin^2(\theta/2)} \leq \frac{\varepsilon}{2}.\square\end{aligned}$$

4 Discrete paths

In this section, we use the Steiner points introduced above for the construction of a weighted graph $G_\varepsilon = (V(G_\varepsilon), E(G_\varepsilon))$. We estimate the number of its nodes and edges and then establish that shortest paths in $\mathcal{D}$ can be approximated by paths in G_ε. We follow the approach laid out in [4], but the details are substantially different, as we have to handle both the vertex and edge vicinities, as well as the bisectors in 3-d space.

The set of nodes $V(G_\varepsilon)$ consists of the vertices of $\mathcal{D}$, the Steiner points placed on the edges of $\mathcal{D}$ and the Steiner points placed on the bisectors. The edges of the graph G_ε join nodes lying on *neighboring* bisectors as defined below. A bisector is a neighbor to itself. Two different bisectors are neighbors if the dihedral angles they split share a common face. We say that a pair of bisectors sharing a face f are neighbors with respect to f. (So, a single bisector $\mathbf{b}$ is a neighbor to itself with respect to both faces forming the dihedral angle it splits.)

First, we define edges joining pairs of Steiner points on neighboring bisectors. Let p and q be nodes corresponding to Steiner points lying on neighboring bisectors $\mathbf{b}$ and $\mathbf{b_1}$, respectively, that share a common face f. We consider the shortest weighted path between p and q of the type $\{p, x, y, q\}$, where x and y belong to f (points x and y are not necessarily different). We refer to this shortest path as a *local shortest path* between p and q crossing f and denote it by $\hat{\pi}(p, q; f)$. Nodes p and q are joined by an edge in G_ε if none of the points x or y are on an edge of f. Such an edge is said to *cross* the face f. In the case where p and q lie on the same bisector, say $\mathbf{b}$, splitting an angle between faces f_1 and f_2, we define two parallel edges in G_ε joining p and q – one crossing f_1 and another crossing f_2.

The cost of an edge (p, q) in G_ε that crosses a face f is defined as the cost of the local shortest path $\hat{\pi}(p, q; f)$ and is denoted by $c(p, q; f)$, or simply by $c(p, q)$ when no ambiguity arises. Formally, we have

$$c(p, q) = c(p, q; f) = \|\hat{\pi}(p, q; f)\| = \min_{x,y \in f}(\|px\| + \|xy\| + \|yq\|). \tag{22}$$

Next, we consider a node p of G_ε lying on an edge e of $\mathcal{D}$. The node p can be either a Steiner point on e or a vertex of $\mathcal{D}$ incident to e. It is adjacent to nodes lying in tetrahedra in $D(e)$. The edges of G_ε incident to p are associated with pairs of neighboring bisectors as follows. We consider a tetrahedron t in $D(e)$, and describe edges incident to p in t. Let f_1 and f_2 be the two faces of t incident to e, and let $\mathbf{b}$ be the bisector of the dihedral angle formed by f_1 and f_2. We define edges between p and nodes lying on bisectors in t that are neighbors of $\mathbf{b}$. There are four such bisectors – two with respect to f_1 and two with respect to f_2. For a node q on a neighboring bisector $\mathbf{b_1}$ sharing, say, the face f_1 with $\mathbf{b}$, we consider the local shortest path $\hat{\pi}(p, q; f_1)$. By definition, $\hat{\pi}(p, q; f_1) = \{p, x, q\}$, where $x \in f_1$. We define an edge between p and q if and only if the point x defining the local shortest path is in the interior of f_1. The cost of the edge (p, q) equals the cost of the local shortest path $\hat{\pi}(p, q; f_1)$, i.e.,

$$c(p, q) = c(p, q; f_1) = \|\hat{\pi}(p, q; f_1)\| = \min_{x \in f_1}(\|px\| + \|xq\|).$$

We associate the edge (p, q) to $\mathbf{b}$, $\mathbf{b_1}$ and f_1 and say that it crosses f_1. Furthermore, p is joined to nodes on $\mathbf{b}$ by pair of parallel edges, provided that the corresponding local shortest paths do not touch the edges of $\mathcal{D}$ – one crossing f_1 and the other crossing f_2.

Lemma 4.1 *We have* $|V(G_\varepsilon)| = O(\frac{n}{\varepsilon^2} \log \frac{1}{\varepsilon})$ *and* $|E(G_\varepsilon)| = O(\frac{n}{\varepsilon^4} \log^2 \frac{1}{\varepsilon})$.

Proof: The estimate on the number of nodes follows directly from Lemma 3.1 and the fact that $\mathcal{D}$ has $O(n)$ vertices. The number of edges in G_ε can be estimated as follows. There are $O(n)$ faces in $\mathcal{D}$ and at most 21 pairs of neighbor bisectors with respect to a fixed face in $\mathcal{D}$. By Lemma 3.1(a), there are $O(\frac{1}{\varepsilon^4} \log^2 \frac{1}{\varepsilon})$ pairs of nodes lying on two fixed neighboring bisectors. When combined, these three facts prove the estimate on the number of edges of G_ε. □

Paths in G_ε are called *discrete paths*. The cost, $c(\pi)$, of a discrete path π is the sum of the costs of its edges. Note that if we replace each of the edges in a discrete path π by the corresponding (at most three) segments forming the shortest path used to compute its cost we obtain a path in $\mathcal{D}$ with cost $c(\pi)$. Next, we state the main theorem of this section.

Theorem 4.1 *Let* $\tilde{\pi}(v_0, v)$ *be a shortest path between two different vertices* v_0 *and* v *in* $\mathcal{D}$. *There exists a discrete path* $\pi(v_0, v)$, *such that* $c(\pi(v_0, v)) \leq (1 + \varepsilon)\|\tilde{\pi}(v_0, v)\|$.

Proof: We prove the theorem by constructing a discrete path $\pi(v_0, v)$ whose cost is as required. Recall that the shortest path $\tilde{\pi}(v_0, v)$ is a linear path consisting of cell-crossing, face-using, and edge-using segments that satisfy Snell's law at each bending point. We construct the discrete path π by successive modifications of $\tilde{\pi}$ described below as steps.

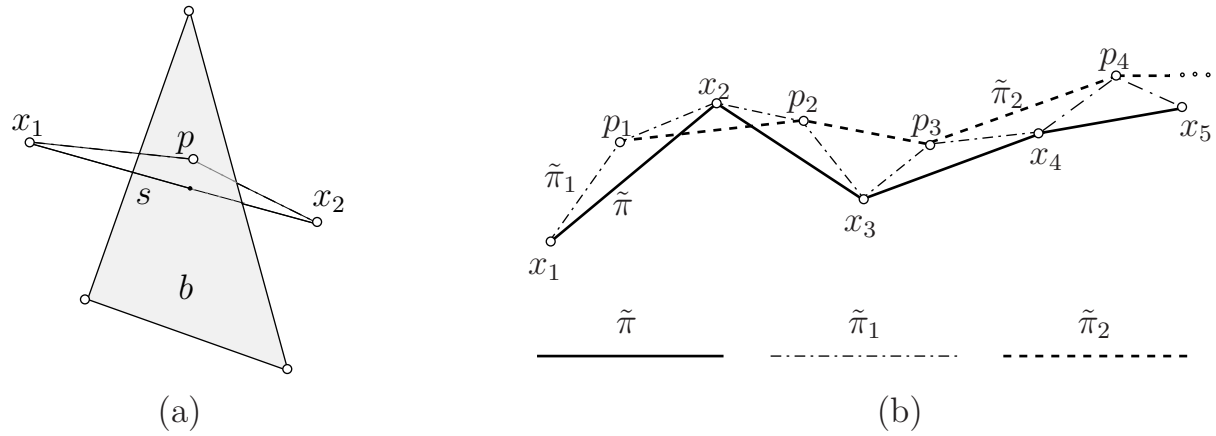

Figure 8: (a) Replacement of a cell-crossing segment $s = (x_1, x_2)$ by a two-segment path $\{x_1, p, x_2\}$. (b) Replacement of $\tilde{\pi}_1$ by a path $\tilde{\pi}_2$ joining Steiner points p_i. Note that edges $(p_i p_{i+1})$ denote local shortest paths, rather than straight-line segments.

Step 1: In this step, we replace each of the cell-crossing segments of $\tilde{\pi}$, which satisfy the conditions of Lemma 3.2, by a two-segment path through a Steiner point. Precisely, let $s = (x_1, x_2)$ be a cell-crossing segment in $\tilde{\pi}$ (Figure 8 (a)). Let f_1 and f_2 be the faces containing x_1 and x_2, respectively. Let $e = (A, B)$ be the common edge between f_1 and f_2. Assume that s is outside of the union of the edge and vertex vicinities $D_\varepsilon(e) \cup D_\varepsilon(A) \cup D_\varepsilon(B)$. We refer to such segment as *vicinity-free*[4]. Then, according to Lemma 3.2, there is a Steiner point p on the bisector b splitting the dihedral angle formed by f_1 and f_2 such that $|x_1 p| + |p x_2| \leq (1 + \varepsilon/2)|x_1 x_2|$. So, in this step, each cell-crossing and vicinity-free segment $s = (x_1, x_2)$ is replaced by two-segment path $\{x_1, p, x_2\}$, where p is the approximating Steiner point as described above. Clearly, after this step, we obtain a path joining v_0 and v, whose cost does not exceed $(1 + \varepsilon/2)\|\tilde{\pi}\|$. We denote this path by $\tilde{\pi}_1$ (see Figure 8 (b)).

Step 2: In this step, we consider the sequence of Steiner points added as new bending points along $\tilde{\pi}_1$ in Step 1. In the case where two consecutive Steiner points are split by a single bending point or a face-using segment on $\tilde{\pi}_1$, we replace the sub-path between them by the corresponding local shortest path. Precisely, assume that p_1 and p_2 are consecutive Steiner points along $\tilde{\pi}_1$ and the sub-path between them is either $\{p_1, \tilde{x}, p_2\}$ or $\{p_1, \tilde{x}, \tilde{y}, p_2\}$, $\tilde{x}$ and $\tilde{y}$ are bending points on the face f, shared by the two neighboring tetrahedra containing p_1 and p_2, respectively. So, in Step 2, we replace all such sub-paths by the local shortest paths $\hat{\pi}(p_1, p_2; f) = \{p_1, x, y, p_2\}$, using (22). We denote the obtained path by $\tilde{\pi}_2$ (Figure 8 (b)). Clearly, $\tilde{\pi}_2$ is a path joining v_0 and v, whose cost does not exceed that of $\tilde{\pi}_1$. Hence,

$$\|\tilde{\pi}_2\| \leq \|\tilde{\pi}_1\| \leq (1 + \varepsilon/2)\|\tilde{\pi}\| \ . \tag{23}$$

In the following two steps, we identify the portions of $\tilde{\pi}_2$ that lie inside the vertex and edge vicinities and replace them with discrete paths using the corresponding vertices and edges.

[4]Note that such a segment still can have an end-point in a vertex or edge-vicinity related to other vertices or edges incident to f_1 and f_2.

Step 3: Follow $\tilde{\pi}_2$ from v_0 to v and let a_0 be the last bending point on $\tilde{\pi}_2$ that lies inside the vertex vicinity $D_\varepsilon(v_0)$. Next, let b_1 be the first bending point after a_0 that is in the vertex vicinity, say $D_\varepsilon(v_1)$. Likewise, let a_1 be the last bending point in $D_\varepsilon(v_1)$. Continuing in this way, we define a sequence of, say $k+1$ for some $k \geq 1$, different vertices $v_0, v_1, \ldots, v_k = v$ and a sequence of bending points $a_0, b_1, a_1, \ldots, a_{k-1}, b_k$ on $\tilde{\pi}_2$, such that for $i = 0, \ldots, k$, points b_i, a_i are in $D_\varepsilon(v_i)$ (we assume $b_0 = v_0$, $a_k = v$). Furthermore, by our definition, portions of $\tilde{\pi}_2$ between a_i and b_{i+1} do not intersect any vertex vicinities. We partition $\tilde{\pi}_2$ into portions

$$\tilde{\pi}_2(v_0, a_0), \tilde{\pi}_2(a_0, b_1), \tilde{\pi}_2(b_1, a_1), \ldots, \tilde{\pi}_2(b_k, v). \tag{24}$$

The portions $\tilde{\pi}_2(a_i, b_{i+1})$, for $i = 0, \ldots, k-1$, are called the *between-vertex-vicinities* portions, while the portions $\tilde{\pi}_2(b_i, a_i)$, for $i = 0, \ldots, k$, are called the *vertex-vicinity* portions.

We define path $\tilde{\pi}_3$ by replacing each of the vertex-vicinities portions by a two segment path trough the corresponding vertex and show that the cost of $\tilde{\pi}_3$ is bounded by $(1 + \varepsilon/6)\|\tilde{\pi}_2\|$. Consider a between-vertex-vicinities portion $\tilde{\pi}_2(a_i, b_{i+1})$ for some $0 \leq i < k-1$. If this portion consists of a single segment (a_i, b_{i+1}), then the vertices v_i and v_{i+1} must be adjacent in $\mathcal{D}$ and we define $\tilde{\pi}_3(v_i, v_{i+1})$ to be the segment (v_i, v_{i+1}). The length of (v_i, v_{i+1}) is estimated by using the triangle inequality and the definition of the vertex-vicinities as follows:

$$\begin{aligned} |v_i v_{i+1}| \leq |v_i a_i| + |a_i b_{i+1}| + |b_{i+1} v_{i+1}| \leq |a_i b_{i+1}| + \varepsilon(r(v_i) + r(v_{i+1})) \leq \\ |a_i b_{i+1}| + \frac{\varepsilon}{14}(d(v_i) + d(v_{i+1})) \leq |a_i b_{i+1}| + \frac{\varepsilon}{7}|v_i v_{i+1}|. \end{aligned} \tag{25}$$

To estimate the cost of the segment (v_i, v_{i+1}), we observe that (a_i, b_{i+1}) lies inside a tetrahedron incident to (v_i, v_{i+1}). Thus, the weight of (v_i, v_{i+1}) is at most the weight of (a_i, b_{i+1}). This observation and (25) readily imply

$$\|\tilde{\pi}_3(v_i, v_{i+1})\| = \|v_i v_{i+1}\| \leq (1 + \frac{\varepsilon}{6})\|a_i b_{i+1}\| = (1 + \frac{\varepsilon}{6})\|\tilde{\pi}_2(a_i, b_{i+1})\|. \tag{26}$$

In the general case, where $\tilde{\pi}_2(a_i, b_{i+1})$ contains at least two segments, we follow the bending points along $\tilde{\pi}_2(a_i, b_{i+1})$ and define X to be the last bending point on the boundary $\partial D(v_i)$ (see Definition 3.1). If the path $\tilde{\pi}_2(a_i, b_{i+1})$ lies entirely in $D(v_i)$, then we set $X = b_{i+1}$. Thus, the bending points on the path $\tilde{\pi}_2$ between a_i and X lie in the tetrahedra incident to v_i. Let $\dot{w}_i$ be the minimum weight among the segments of the path $\tilde{\pi}_2(a_i, X)$ and let x be the first bending point after a_i incident to a segment, whose weight is $\dot{w}_i$. Analogously, define the bending points Y and y, by following the bending points of the backward path $\tilde{\pi}_2(b_{i+1}, a_i)$ from b_{i+1}. Note that x precedes y on the path $\tilde{\pi}_2(a_i, b_{i+1})$. We define the path $\tilde{\pi}_3(v_i, v_{i+1})$ as the concatenation of the segments (v_i, x), (y, v_{i+1}) and the portion $\tilde{\pi}_2(x, y)$, i.e.,

$$\tilde{\pi}_3(v_i, v_{i+1}) = \{(v_i, x), \tilde{\pi}_2(x, y), (y, v_{i+1})\}.$$

Next, we estimate the cost of $\tilde{\pi}_3(v_i, v_{i+1})$. First, we observe that the weight of the segment (v_i, x) cannot exceed $\dot{w}_i$. Then, we use the triangle inequality and the fact that a_i is inside the vertex vicinity $D_\varepsilon(v_i)$, obtaining

$$\begin{aligned} \|v_i x\| \;\leq\; & \dot{w}_i |v_i a_i| + \dot{w}_i |\tilde{\pi}_2(a_i, x)| \leq \dot{w}_i |v_i a_i| + \|\tilde{\pi}_2(a_i, x)\| \leq \\ & \dot{w}_i \varepsilon r(v_i) + \|\tilde{\pi}_2(a_i, x)\| \leq \dot{w}_i \frac{\varepsilon}{14} d(v_i) + \|\tilde{\pi}_2(a_i, x)\|. \end{aligned}$$

Analogously, for the cost of the segment (y, v_{i+1}), we have

$$\|yv_{i+1}\| \leq \dot{w}_{i+1}\frac{\varepsilon}{14}d(v_{i+1}) + \|\tilde{\pi}_2(y, b_{i+1})\|.$$

Using these estimates, and the way we defined the path $\tilde{\pi}_3(v_i, v_{i+1})$, the weights $\dot{w}_i$, $\dot{w}_{i+1}$, the distances $d(v_i)$, $d(v_{i+1})$, and the points X, Y, we obtain

$$\|\tilde{\pi}_3(v_i, v_{i+1}\| \leq \|\tilde{\pi}_2(a_i, b_{i+1})\| + \frac{\varepsilon}{14}(\dot{w}_i d(v_i) + \dot{w}_{i+1}d(v_{i+1})) \leq \quad (27)$$
$$\|\tilde{\pi}_2(a_i, b_{i+1})\| + \frac{\varepsilon}{14}(\|\tilde{\pi}_3(v_i, X)\| + \|\tilde{\pi}_3(Y, v_{i+1})\|) \leq \|\tilde{\pi}_2(a_i, b_{i+1})\| + \frac{\varepsilon}{7}(\|\tilde{\pi}_3(v_i, v_{i+1})\|,$$

which implies the estimate (26) in the general case. Applying the above construction to each pair of consecutive vertices in the sequence $v_0, v_1, \ldots, v_k = v$, we obtain a linear path

$$\tilde{\pi}_3(v_0, v) = \{\tilde{\pi}_3(v_0, v_1), \tilde{\pi}_3(v_1, v_2), \ldots, \tilde{\pi}_3(v_{k-1}, v)\},$$

that has no bending points inside vertex vicinities except for the vertices $v_0, v_1, \ldots, v_k = v$. We estimate the cost of this path by summing up (26), for $i = 0, \ldots, k-1$, and obtain

$$\|\tilde{\pi}_3(v_0, v)\| \leq (1 + \frac{\varepsilon}{6})\sum_{i=0}^{k-1}\|\tilde{\pi}_2(a_i, b_{i+1})\| \leq (1 + \frac{\varepsilon}{6})\|\tilde{\pi}_2(v_0, v)\|. \quad (28)$$

Observe that the path $\tilde{\pi}_3$ constructed above may contain self intersections (e.g., if one and the same vertex vicinity is visited twice by $\tilde{\pi}_2$). It is also possible that $\tilde{\pi}_3$ may contain consecutive face-using segments. Hence, at the end of Step 3, we traverse the obtained path and *compress* it. That is, we remove the loops in case of self intersections. We replace the consecutive face-using segments (which obviously lie in the same face) by the single face-using segment joining their free end-points. We denote the compressed path again by $\tilde{\pi}_3$. Clearly, compressing reduces the cost of the path and hence the estimate (28) remains true for $\tilde{\pi}_3$.

Next, in Step 4, using a similar approach as above, we further partition each vertex-vicinity-portion $\tilde{\pi}_3(v_i, v_{i+1})$ into *between-edge-vicinities* portions and *edge-vicinity* portions. Then, we replace each edge-vicinity portion by an edge-using segment plus 2 additional segments and estimate the cost of the resulting path $\tilde{\pi}_4$.

Step 4: First we define analogues of vertex and between-vertex vicinities for edges. Let (v_i, a) be the first segment of the path $\tilde{\pi}_3(v_i, v_{i+1})$. If a is not inside an edge-vicinity, we define $a_{i,0} = v_i$. Otherwise, if a is inside an edge-vicinity, say $D_\varepsilon(e_0)$, and let a' be the first bending point on the path $\tilde{\pi}_3(v_i, v_{i+1})$ after v_i lying on $\partial D(e_0)$, then we define $a_{i,0}$ to be the last bending point on $\tilde{\pi}_3(v_i, a')$ that is inside $D_\varepsilon(e_0)$. Next, let $b_{i,1}$ be the first bending point on $\tilde{\pi}_3(a_{i,0}, v_{i+1})$ that is inside an edge-vicinity, say $D_\varepsilon(e_1)$ and let b' be the first bending point on $\tilde{\pi}_3(b_{i,1}, v_{i+1})$ that is on $\partial D(e_1)$. We define $a_{i,1}$ as the last bending point on $\tilde{\pi}_3(b_{i,1}, b')$ that is in the same edge vicinity as $b_{i,1}$. Assume that, following this approach, the sequence of bending points $a_{i,0}, b_{i,1}, a_{i,1}, \ldots, a_{i,k_i-1}, b_{i,k_i}$ has been defined. They partition the portion $\tilde{\pi}_3(v_i, v_{i+1})$ into sub-portions $\tilde{\pi}_3(v_i, v_{i+1}) =$

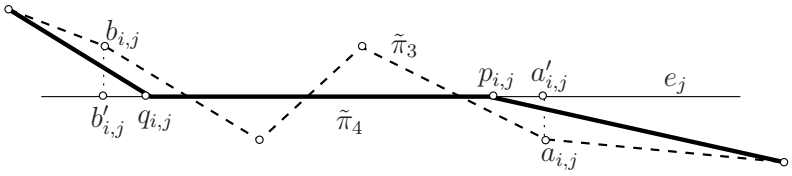

Figure 9: Replacement of subpath $\tilde{\pi}_3(b_{i,j}, a_{i,j})$ by the edge-using segment $(q_{i,j}p_{i,j})$.

$\{\tilde{\pi}_3(v_i, a_{i,0}), \ldots, \tilde{\pi}_3(a_{i,j-1}, b_{i,j}), \tilde{\pi}_3(b_{i,j}, a_{i,j}), \ldots, \tilde{\pi}_3(b_{i,k_i}, v_{i+1})\}$. Portions between $a_{i,j}$ and $b_{i,j+1}$, for $j = 0, \ldots, k_i - 1$, are called the *between-edge-vicinity* portions. Portions between $b_{i,j}$ and $a_{i,j}$, for $j = 0, \ldots, k_i$, are called the *edge-vicinity* portions ($b_{i,0} = v_i$ and $a_{i,k_i} = v_{i+1}$).

According to our construction, the bending points $a_{i,0}, b_{i,1}, a_{i,1}, \ldots, a_{i,k_i-1}, b_{i,k_i}$, defining the above partition lie inside edge vicinities. Moreover, consecutive points $b_{i,j}$ and $a_{i,j}$, for $j = 0, \ldots, k_i$, are in one and the same edge-vicinity $D_\varepsilon(e_j)$.

For $j = 0, \ldots, k_i$, let $b'_{i,j}$ and $a'_{i,j}$ be the orthogonal projections of the points $b_{i,j}$ and $a_{i,j}$ onto the edge e_j, respectively (Figure 9). Let $p_{i,j}$ and $q_{i,j}$ be the Steiner points on e_j defining the largest sub-interval of the interval $(a'_{i,j}, b'_{i,j})$ on e_j and assume that $p_{i,j}$ is between $a'_{i,j}$ and $q_{i,j}$. (In the case where the interval $(a'_{i,j}, b'_{i,j})$ contains no Steiner points, we define $p_{i,j} = q_{i,j}$ to be the closest Steiner point to $a'_{i,j}$ on e_j.) In $\tilde{\pi}_4$, the edge-using segment $(q_{i,j}p_{i,j})$ will replace in $\tilde{\pi}_3$ the subpath $\tilde{\pi}_3(b_{i,j}, a_{i,j})$. Let us estimate the resulting error. From the definition of the edge vicinity $D_\varepsilon(e_j)$, the Steiner points on e_j, and the radii $r(p_{i,j})$ and $r(q_{i,j})$, it is easy to derive that

$$|p_{i,j}a_{i,j}| \leq \frac{3}{2}\varepsilon r(p_{i,j}) \quad \text{and} \quad |b_{i,j}q_{i,j}| \leq \frac{3}{2}\varepsilon r(q_{i,j}). \tag{29}$$

Furthermore, by our construction and the fact that $(q_{i,j}, p_{i,j})$ is an edge-using segment, it follows that

$$\|q_{i,j}p_{i,j}\| \leq \|\tilde{\pi}_3(b_{i,j}, a_{i,j})\|. \tag{30}$$

Next, we modify between-edge-vicinities portions $\tilde{\pi}_3(a_{i,j}, b_{i,j+1})$, into paths $\tilde{\pi}_4(p_{i,j}, q_{i,j+1})$, joining Steiner points $p_{i,j}$ and $q_{i,j+1}$, and not intersecting any vertex or edge vicinities. We apply a construction analogous to the one used in Step 3 to define the paths $\tilde{\pi}_3(v_i, v_{i+1})$.

We fix j and consider the between-edge-vicinities portion $\tilde{\pi}_3(a_{i,j}, b_{i,j+1})$. We first consider the special case where $\tilde{\pi}_3(a_{i,j}, b_{i,j+1})$ is the segment $(a_{i,j}, b_{i,j+1})$. In this case, we observe that e_j and e_{j+1} must be edges of the tetrahedron containing the segment $(a_{i,j}, b_{i,j+1})$ and define $\tilde{\pi}_4(p_{i,j}, q_{i,j+1}) = (p_{i,j}, q_{i,j+1})$. We estimate the length of this segment using the triangle inequality, the estimates (29) and Definition 3.1 as follows

$$\begin{aligned} |p_{i,j}q_{i,j+1}| \;\leq\; & |p_{i,j}a_{i,j}| + |a_{i,j}b_{i,j+1}| + |b_{i,j+1}q_{i,j+1}| \leq \frac{3\varepsilon}{2}(r(p_{i,j}) + r(q_{i,j+1})) + |a_{i,j}b_{i,j+1}| \leq \\ & \frac{\varepsilon}{16}(d(p_{i,j}) + d(q_{i,j+1})) + |a_{i,j}b_{i,j+1}| \leq \frac{\varepsilon}{8}|p_{i,j}q_{i,j+1}| + |a_{i,j}b_{i,j+1}|. \end{aligned}$$

Using this estimate and the observation that the weight of the segment $(p_{i,j}, q_{i,j+1})$ cannot exceed the weight of $(a_{i,j}, b_{i,j+1})$, we obtain

$$\|\tilde{\pi}_4(p_{i,j}, q_{i,j+1})\| = \|p_{i,j}q_{i,j+1}\| \leq (1 + \frac{\varepsilon}{7})\|a_{i,j}b_{i,j+1}\| = (1 + \frac{\varepsilon}{7})\|\tilde{\pi}_3(a_{i,j}, b_{i,j+1})\|. \tag{31}$$

Next, we consider the general case, where $\|\tilde{\pi}_3(a_{i,j}, b_{i,j+1})\|$ consists of at least two segments. Let X be the first bending point after $a_{i,j}$ that is on the boundary $\partial D(e_j)$. If the path $\tilde{\pi}_3(a_{i,j}, b_{i,j+1})$ is entirely inside $D(e_j)$, then we set $X = b_{i,j+1}$. Furthermore, let $\dot{w}'_j$ be the minimum weight among the segments in $\tilde{\pi}_3(a_{i,j}, X)$, and let x be the first bending point after $a_{i,j}$ that is an end-point of a segment whose weight is $\dot{w}'_j$. We define the weight $\dot{w}'_{j+1}$, and the bending points Y, and y, analogously with respect to $b_{i,j+1}$ and the edge vicinity $D_\varepsilon(e_{j+1})$. It follows that the point x precedes y along $\tilde{\pi}_3(a_{i,j}, b_{i,j+1})$. We define the portion of the path $\tilde{\pi}_4$ joining $p_{i,j}$ and $q_{i,j+1}$ by $\tilde{\pi}_4(p_{i,j}, q_{i,j+1}) = \{(p_{i,j}, x), \tilde{\pi}_3(x, y), (y, q_{i,j+1})\}$ and estimate its cost. Let us first estimate the cost of the segment $(p_{i,j}, x)$. We observe that $\|p_{i,j}x\| \leq \dot{w}'_j |p_{i,j}x|$, that $|p_{i,j}, x| \leq |p_{i,j}a_{i,j}| + |\tilde{\pi}_3(a_{i,j}, x)|$ (by triangle inequality), and that the segments on the path $\tilde{\pi}_3(a_{i,j}, x)$ have weight greater than or equal to $\dot{w}'_j$. Using these observations, (29), and Definition 3.1, we obtain

$$\begin{aligned} \|p_{i,j}x\| &\leq \dot{w}'_j|p_{i,j}a_{i,j}| + \dot{w}'_j|\tilde{\pi}_3(a_{i,j}, x)| \leq \dot{w}'_j|p_{i,j}a_{i,j}| + \|\tilde{\pi}_3(a_{i,j}, x)\| \leq \\ &\frac{3\varepsilon}{2}\dot{w}'_j r(p_{i,j}) + \|\tilde{\pi}_3(a_{i,j}, x)\| = \frac{\varepsilon}{16}\dot{w}'_j d(p_{i,j}) + \|\tilde{\pi}_3(a_{i,j}, x)\|. \end{aligned} \tag{32}$$

Analogously, we have

$$\|yq_{i,j+1}\| \leq \frac{\varepsilon}{16}\dot{w}'_{j+1} d(q_{i,j+1}) + \|\tilde{\pi}_3(y, b_{i,j+1})\|. \tag{33}$$

Using the definition of the path $\tilde{\pi}_4(p_{i,j}, q_{i,j+1})$, the estimates (32) and (33), and the definition of the distances $d(p_{i,j})$ and $d(q_{i,j+1})$, the weights $\dot{w}'_j$ and $\dot{w}'_{j+1}$, and the points X and Y, we obtain

$$\begin{aligned} &\|\tilde{\pi}_4(p_{i,j}, q_{i,j+1})\| = \|\tilde{\pi}_3(a_{i,j}, b_{i,j+1})\| + \frac{\varepsilon}{16}(\dot{w}'_j d(p_{i,j}) + \dot{w}'_{j+1} d(q_{i,j+1})) \\ &\leq \|\tilde{\pi}_3(a_{i,j}, b_{i,j+1})\| + \frac{\varepsilon}{16}(\|\tilde{\pi}_3(p_{i,j}, X)\| + \|\tilde{\pi}_3(Y, q_{i,j+1})\|) \\ &\leq \|\tilde{\pi}_3(a_{i,j}, b_{i,j+1})\| + \frac{\varepsilon}{8}\|\tilde{\pi}_3(p_{i,j}, q_{i,j+1})\|, \end{aligned} \tag{34}$$

which implies estimate (31), in the general case. Finally, combining segments $(q_{i,j}, p_{i,j})$ and paths $\tilde{\pi}_4(p_{i,j}, q_{i,j+1})$, for $j = 0, \ldots, k_i$, we construct a path

$$\tilde{\pi}_4(v_i, v_{i+1}) = \{(v_i, p_{i,0}), \tilde{\pi}_4(p_{i,0}, q_{i,1}), (q_{i,1}, p_{i,1}), \tilde{\pi}_4(p_{i,1}, q_{i,2}), \ldots, \tilde{\pi}_4(p_{i,k_i-1}, q_{i,k_i}), (q_{i,k_i}, v_{i+1})\}.$$

This path has no bending points in any of the edge or vertex vicinities. Its cost can be bounded using (30) and (31) as follows

$$\begin{aligned} \|\tilde{\pi}_4(v_i, v_{i+1})\| &= \sum_{j=0}^{k_i} \|q_{i,j}, p_{i,j}\| + \sum_{j=0}^{k_i-1} \|\tilde{\pi}_4(p_{i,j}, q_{i,j+1})\| \leq \\ &\sum_{j=0}^{k_i} \|\tilde{\pi}_3(b_{i,j}, a_{i,j})\| + \sum_{j=0}^{k_i-1}(1 + \frac{\varepsilon}{7})\|\tilde{\pi}_3(a_{i,j}, b_{i,j+1})\| \leq (1 + \frac{\varepsilon}{7})\|\tilde{\pi}_3(v_i, v_{i+1})\|, \end{aligned} \tag{35}$$

where we assume $v_i = b_{i,0} = q_{i,0}$ and $v_{i+1} = a_{i,k_i} = p_{i,k_i}$.

The paths $\tilde{\pi}_4(v_i, v_{i+1})$, for $i = 0, \ldots k-1$, form a linear path $\tilde{\pi}_4(v_0, v)$, whose cost is estimated using (35), (26), and (23) by

$$\tilde{\pi}_4(v_0, v) \leq \sum_{i=0}^{k-1}(1+\frac{\varepsilon}{7})\|\tilde{\pi}_3(v_i, v_{i+1})\| = (1+\frac{\varepsilon}{7})\|\tilde{\pi}_3(v_0, v)\| \leq (1+\frac{\varepsilon}{3})\|\tilde{\pi}_2(v_0, v)\| \leq (1+\varepsilon)\|\tilde{\pi}(v_0, v)\|. \tag{36}$$

As in Step 3, it is possible for $\tilde{\pi}_4$ to contain self-intersections and consecutive face-using segments. Hence, we traverse $\tilde{\pi}_4$ and compress it by removing loops and by replacing consecutive face-using segments. The obtained path is denoted again by $\tilde{\pi}_4$, and estimate (36) is valid.

The bending points defining $\tilde{\pi}_4$ can be partitioned into two groups. The first group consists of bending points corresponding to nodes of the graph G_ε, i.e., Steiner points on bisectors, Steiner points on edges, and vertices of $\mathcal{D}$. The second group consists of the remaining bending points of $\tilde{\pi}_4$, which are bending points inside the faces of $\mathcal{D}$. We complete the proof of the theorem by showing that the sequence of the nodes in the first group defines a discrete path $\pi(v_0, v)$ whose cost $c(\pi(v_0, v)) \leq \|\tilde{\pi}_4(v_0, v)\|$. It suffices to show that any two consecutive nodes (bending points in the first group) along the path $\tilde{\pi}_4$ are adjacent in the approximation graph G_ε.

To show this, we review closely the structure of the path $\tilde{\pi}_4$. In Step 3, portions of $\tilde{\pi}_2$ related to vertex vicinities have been replaced by two segment portions through-vertices of $\mathcal{D}$. Furthermore, we observe that the segments (v_i, x) created in Step 3 are either a face-using segments or join v_i to a Steiner point on a bisector. The same applies to segments (y, v_{i+1}). Similarly, in Step 4, portions related to edge-vicinities have been replaced by three segment portions visiting corresponding edges. Again segments $(p_{i,j}, x)$ are either face-using segments or join $p_{i,j}$ to a node on a bisector, that is a neighbor of the bisector incident to the edge containing $p_{i,j}$. The same applies to the segments $(y, q_{i,j+1})$. In summary, the segments created in Steps 3 and 4 are of one of the following two types:

1. A face-using segment with one of its endpoint being a (node) vertex of $\mathcal{D}$ or a Steiner point on an edge of $\mathcal{D}$.

2. A segment joining two nodes, at least one of them being a Steiner point on an edge of $\mathcal{D}$ or a vertex of $\mathcal{D}$.

The remaining segments in $\tilde{\pi}_4$ are cell-crossing and face-using segments, whose endpoints are outside any vertex or edge vicinity. All the cell-crossing segments in $\tilde{\pi}_4$ were created during Steps 1 and 2. Hence, one of their endpoints is a (node) Steiner point on a bisector of a tetrahedron. Finally, due to the compressing, there are no consecutive face-using segments in $\tilde{\pi}_4$.

Now, let p and q be two consecutive nodes along the path $\tilde{\pi}_4$. We show that p and q are adjacent in G_ε. We consider, first, the case where at least one of the nodes, say p, is a vertex of $\mathcal{D}$. Let x be the bending point following p along the path $\tilde{\pi}_4$. By the definition of bending points adjacent to the vertices (in Step 3), we know that (p, x) is a face-using segment followed by a cell-crossing segment (x, x_1), joining x to a (node) Steiner point on a bisector lying in one of the tetrahedra incident to the face that contains (p, x). So, $q = x_1$

and q is inside a tetrahedron incident to p. Thus, p and q are adjacent in G_ε. The case where at least one of the nodes p or q is a Steiner point on an edge of $\mathcal{D}$ can be treated analogously.

Assume now that both p and q are Steiner points on bisectors. Let x and x_1 be the bending points following p along $\tilde{\pi}_4$. The point x has to be a bending point on a face of the tetrahedron containing p. The segment (x, x_1) is either a cell-crossing or a face-using segment. In the first case, q must coincide with x_1 and is adjacent to p in G_ε, since it lies in a tetrahedron that is a neighbor to the one containing p. In the second case, where (x, x_1) is a face-using segment, we consider the bending point x_2 that follows x_1 along the path $\tilde{\pi}_4$. The segment (x_1, x_2) must be a cell-crossing segment. Thus, in this case, $q = x_2$ is adjacent to p, because the tetrahedra containing p and q are neighbors.

We have shown that any pair p and q of consecutive nodes on the path $\tilde{\pi}_4$ are adjacent in G_ε. Hence, we define a discrete path $\pi(v_0, v)$ to be the path in G_ε following the sequence of nodes along $\tilde{\pi}_4$. Finally, we observe that the sub-paths of $\tilde{\pi}_4(p, q)$ joining pairs of consecutive nodes stay in the union of the tetrahedra containing these nodes and cross faces shared by the bisectors containing them. Hence, by the definition of the cost of the edges in G_ε, we have $c(p, q) \leq \|\tilde{\pi}_4(p, q)\|$. Summing these estimates, for all edges of $\pi(v_0, v)$, and using (36), we obtain, $c(\pi(v_0, v)) \leq \|\tilde{\pi}_4(v_0, v)\| \leq (1 + \varepsilon)\|\tilde{\pi}(v_0, v)\|$. □

5 An algorithm for computing SSSP in G_ε

In this section we present our algorithm for solving the Single Source Shortest Paths (SSSP) problem in the approximation graph $G_\varepsilon = (V(G_\varepsilon), E(G_\varepsilon))$. Straightforwardly, one can apply Dijkstra's algorithm, which runs in $O(|E(G_\varepsilon)| + |V(G_\varepsilon)| \log |V(G_\varepsilon)|)$ time. By Lemma 4.1 we have $|V(G_\varepsilon)| = O(\frac{n}{\varepsilon^2} \log \frac{1}{\varepsilon})$ and $|E(G_\varepsilon)| = O(\frac{n}{\varepsilon^4} \log^2 \frac{1}{\varepsilon})$. Thus, the SSSP problem in G_ε can be solved in $O(\frac{n}{\varepsilon^4} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$ time.

In the remainder of this section, we demonstrate how geometric properties of our model can be used to obtain a more efficient algorithm for solving the SSSP problem. More precisely, we present an algorithm that runs in $O(|V_\varepsilon|(\log |V_\varepsilon| + \frac{1}{\sqrt{\varepsilon}} \log^3 \frac{1}{\varepsilon})) = O(\frac{n}{\varepsilon^{2.5}} \log \frac{n}{\varepsilon} \log^3 \frac{1}{\varepsilon})$ time.

Informally, the idea is to avoid consideration of large portions of the edges of the graph G_ε when searching for shortest paths. We achieve that by applying the strategy proposed first in [25, 26] and developed further in [4] and by using the properties of the weighted distance function and additive Voronoi diagrams studied in Section 2.2. We maintain a priority queue containing candidate shortest paths. At each iteration of the algorithm, a shortest path from the source s to some node u of G_ε is found. Then, the algorithm constructs edges adjacent to u that can be continuations of the shortest path from s to u and inserts them in the priority queue as new candidate shortest paths. In general, one needs to consider all edges adjacent to u as possible continuations. In our case, we divide the edges adjacent to u into $O(\frac{1}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon})$ groups related to the segments containing Steiner points in the neighboring bisectors and demonstrate that we can consider just a constant number of edges in each group. The latter is possible due to the structure of the Voronoi cell $\mathcal{V}(u)$ of the node u in the additive Voronoi diagram related to a fixed group (see Theorem 2.1).

This section is organized as follows: In the next subsection, we describe the general structure of the algorithm. In Subsection 5.2, we show how this strategy can be applied in our case and present an outline of the algorithm. We provide details of the implementation

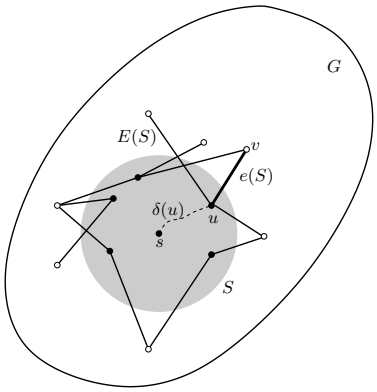

Figure 10: The sets S, S^a, and an optimal edge $e(S) = (u, v)$ in $E(S)$ are illustrated. A shortest path from s to u is illustrated by a dashed curve.

of the algorithm and analyze its complexity. Finally, at the end we establish the main result of the paper.

5.1 General structure of the algorithm

Let $G(V, E)$ be a directed graph with positive costs (lengths) assigned to its edges and s be a fixed node of G, called the *source*. A standard greedy approach for solving the SSSP problem works as follows: a subset, S, of nodes to which the shortest path has already been found and a set, $E(S)$, of edges connecting S with $S^a \subset V \setminus S$ are maintained. The set S^a consists of nodes not in S but adjacent to S. In each iteration, an *optimal* edge $e(S) = (u, v)$ in $E(S)$ is selected, with source u in S and target v in S^a (see Figure 10). The target vertex v is added to S and $E(S)$ is updated correspondingly. An edge $e = e(S)$ is optimal if it minimizes the value $\delta(u) + c(e)$, where $\delta(u)$ is the distance from s to u and $c(e)$ is the cost of e.

Different strategies for maintaining information about $E(S)$ and finding an optimal edge $e(S)$ during each iteration result in different algorithms for computing SSSP. For example, Dijkstra's algorithm maintains only a subset $Q(S)$ of $E(S)$, which, however, always contains an optimal edge. Alternatively, as in [4], one may maintain a subset of $E(S)$ containing one edge per node u in S. The target node of this edge is called the *representative* of u and is denoted by $\rho(u)$. The node u itself is called *predecessor* of its representative. The representative $\rho(u)$ is defined to be the target of the minimum cost edge in the *propagation set* $I(u)$ of u, where $I(u) \subset E(S)$ consists of all edges (u, v) such that $\delta(u) + c(u, v) < \delta(u') + c(u', v)$ for all nodes $u' \in S$ that have entered S before u. The union of propagation sets forms a subset $Q(S)$ of $E(S)$ that always contains an optimal edge. Propagation sets $I(u)$, for $u \in S$, form a partition of $Q(S)$. The propagation sets of the vertices in S form a partition of $E(S)$, which is called *propagation diagram*, and is denoted by $\mathcal{I}(S)$.

The set of representatives $R \subset S^a$ can be organized in a priority queue, where the key of the node $\rho(u)$ in R is defined to be $\delta(u) + c(u, \rho(u))$. Observe that the edge corresponding to the minimum in R is an optimal edge for S. In each iteration, the minimum key node v

in R is selected and the following three steps are carried:

Step 1. *The node v is moved from R into S. Then, the propagation set $I(v)$ is computed and the propagation diagram $\mathcal{I}(S)$ is updated accordingly.*

Step 2. *Representative $\rho(v)$ of v and a new representative, $\rho(u)$, for the predecessor u of v are computed.*

Step 3. *The new representatives, $\rho(u)$ and $\rho(v)$, are either inserted into R together with their corresponding keys, or (if they are already in R) their keys are updated.*

Clearly, this leads to a correct algorithm for solving the SSSP problem in G. The total time for the priority queue operations [5] is $O(|V| \log |V|)$. Therefore, the efficiency of this strategy depends on the maintenance of the propagation diagram, the complexity of the propagation sets, and the efficient updates of the new representatives. In the next subsection, we address these issues and provide necessary details.

5.2 Implementation details and analysis

5.2.1 Notation and algorithm outline

Our algorithm follows the general strategy as described in the previous subsection. First, we convert G_ε into a directed graph by replacing each of its edges by a pair of oppositely oriented edges with cost equal to the cost of the original edge.

Let, as above, S be the set of the nodes to which the shortest path has already been found and $E(S)$ be the set of the edges joining S with $S^a \subset V \setminus S$. We partition the edges of G_ε (and respectively $E(S)$) into groups so that the propagation sets and the corresponding propagation diagrams, when restricted to a fixed group, have a simple structure and can be updated efficiently. Then, for each node u in S, we will keep multiple representatives in R – a constant number on the average, for each group where edges incident to u participate and where its propagation set is non-empty. A node in S^a will have multiple predecessors – at most as many as the number of the groups where edges incident to it participate. We will show that the number of the groups, where edges incident to u can participate, is bounded by $O(\frac{1}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon})$ times the number of bisectors incident to u. In a fixed group, we will be able to compute new representatives in $O(\log \frac{1}{\varepsilon})$ time and update propagation diagrams in $O(\log^2 \frac{1}{\varepsilon})$ time.

Edges of G_ε joining pairs of Steiner points on bisectors are naturally partitioned into groups corresponding to ordered triples $(\mathbf{b}, \mathbf{b_1}, \mathbf{f})$, where $\mathbf{b}$ and $\mathbf{b_1}$ are neighboring bisectors with respect to the face $\mathbf{f}$ (see Section 4 for the definitions). The edges of the initial tetrahedralization $\mathcal{D}$ are assumed to belong to the bisectors incident to them. So, the group of edges corresponding to an ordered triple $(\mathbf{b}, \mathbf{b_1}, \mathbf{f})$ consists of all edges from a node on $\mathbf{b}$ to a node on $\mathbf{b_1}$ that cross $\mathbf{f}$. Recall that the nodes (Steiner points) on any bisector $\mathbf{b}$ were placed on a set of segments parallel to the edge of $\mathcal{D}$ incident to $\mathbf{b}$. In our discussion below, we refer to these segments, including the edge of $\mathcal{D}$, as *Steiner segments*. We further partition

[5] Note that we do not need a priority queue based on elaborated data structures such as Fibonacci heaps. Any priority queue with logarithmic time per operation suffices.

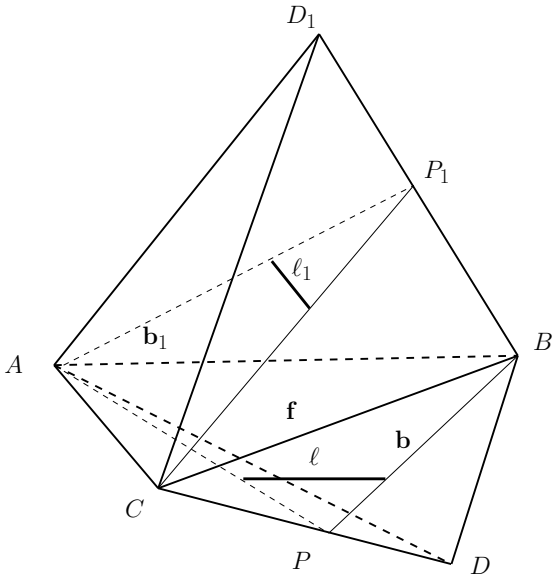

Figure 11: Two Steiner segments ℓ and ℓ_1 lying on neighboring bisectors $\mathbf{b} = \triangle ABP$ and $\mathbf{b_1} = \triangle ACP_1$ respectively, that share a face $\mathbf{f} = \triangle ABC$ are illustrated. Steiner segments on $\mathbf{b}$ and $\mathbf{b_1}$ are parallel to the shared face $\mathbf{f}$. The edges joining nodes on ℓ and ℓ_1 form the group of edges corresponding to the triple $(\ell, \ell_1, \mathbf{f})$.

the group of edges associated with the triple $(\mathbf{b}, \mathbf{b_1}, \mathbf{f})$ into subgroups corresponding to pairs of Steiner segments (ℓ, ℓ_1) on $\mathbf{b}$ and $\mathbf{b_1}$, respectively, see Figure 11 (a). In this way, the edges of G_ε are partitioned into groups corresponding to ordered triples $(\ell, \ell_1, \mathbf{f})$, where ℓ and ℓ_1 are Steiner segments parallel to $\mathbf{f}$ on two neighboring bisectors sharing $\mathbf{f}$. The group corresponding to $(\ell, \ell_1, \mathbf{f})$ is denoted by $E(\ell, \ell_1, \mathbf{f})$ and consists of all oriented edges from a node on ℓ to a node on ℓ_1 that cross $\mathbf{f}$.

A fixed bisector $\mathbf{b}$ has either three or six neighboring bisectors ($\mathbf{b}$ itself and two or five others, respectively) with respect to each of the two faces forming the dihedral angle bisected by $\mathbf{b}$. Hence, the total number of ordered triples $(\mathbf{b}, \mathbf{b_1}, \mathbf{f})$ does not exceed $36n$. By Lemma 3.1, the number of Steiner segments on any bisector is $O(\frac{1}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon})$ and thus the number of subgroups of a group corresponding to a triple $(\mathbf{b}, \mathbf{b_1}, \mathbf{f})$ is $O(\frac{1}{\varepsilon} \log^2 \frac{1}{\varepsilon})$. In total, the number of groups $E(\ell, \ell_1, \mathbf{f})$ is $O(\frac{n}{\varepsilon} \log^2 \frac{1}{\varepsilon})$.

A node u lying on a Steiner segment ℓ will have a set of representatives for each group $E(\ell, \ell_1, \mathbf{f})$ corresponding to a triple, where ℓ is the first element and where its propagation set is non-empty. We denote this set by $\boldsymbol{\rho}(u, \ell_1, \mathbf{f})$. The set of representatives $\boldsymbol{\rho}(u, \ell_1, \mathbf{f})$ will correspond to the structure of the propagation set $I(u; \ell, \ell_1, \mathbf{f})$, as we will detail in the next subsection.

Consider an iteration of our algorithm. Let v be the node extracted from priority queue R, containing all representatives. Let $\mathcal{T}(v)$ be the set of triples $(\ell, \ell_1, \mathbf{f})$ such that v lies on ℓ. First, we need to move v from S^a to S, since the distance from v to the source s has been found. Nodes that are targets of the edges originating from v need to be added to S^a. Then, we need to compute representatives of v for each group of edges, where edges originating at

v participate and where its propagation set is non-empty. Finally, we need to compute new representatives for all nodes in the set of predecessors of v, which we denote by $R^{-1}(v)$. The outline of our algorithm is as follows.

ALGORITHM: SSSP(G_ε, s)
While $S \neq V_\varepsilon$ **do**
1. $v \longleftarrow$ Extract_min(R);
2. Insert v in S and update S^a;
3. **For** each triple $(\ell, \ell_1, \mathbf{f}) \in \mathcal{T}(v)$ **do**
   3.1 Update the data structures related to the Propagation Diagram $\mathcal{I}(\ell, \ell_1, \mathbf{f})$;
   3.2 Find new representatives for nodes whose propagation set has changed in 3.1;
   3.3 Update sets of representatives $\boldsymbol{\rho}(u; \ell', \ell, \mathbf{f}')$, for all $u \in R^{-1}(v)$;
   3.4 Update R with respect to 3.2 and 3.3.

In the remainder of this section, we address the implementation of this algorithm and analyze its complexity. First, we observe that the number of iterations is $|V_\varepsilon|$. The total number of representatives cannot exceed the number of oriented edges in G_ε, which is less than $|V_\varepsilon|^2$ and so, the size of the priority queue R is bounded by $|V_\varepsilon|^2$ (later we show that it is actually $O(\frac{|V_\varepsilon|}{\sqrt{\varepsilon}} \log^2 \frac{1}{\varepsilon})$). Therefore, a single priority queue operation takes $O(\log |V_\varepsilon|)$ time and the total time for Step 1 is $O(|V_\varepsilon| \log |V_\varepsilon|)$. The total time for Step 2 is $O(|V_\varepsilon| \log \frac{1}{\varepsilon})$.

In Section 5.2.2, we describe the structure and maintenance of the data structures related to the propagation diagrams $\mathcal{I}(\ell, \ell_1, \mathbf{f})$. Computation and updates of the sets of representatives are described in Section 5.2.3. We conclude our discussion in Section 5.2.4 by summarizing the time complexity of the algorithm and by establishing our main result.

5.2.2 Implementation of Step 3.1

We consider a fixed triple (ℓ, ℓ_1, f), where ℓ and ℓ_1 are Steiner segments on neighboring bisectors $\mathbf{b}$ and $\mathbf{b_1}$ sharing $\mathbf{f}$. The propagation diagram $\mathcal{I}(\ell, \ell_1, f)$, was defined as the set consisting of the propagation sets of the active nodes on ℓ. Instead of explicitly computing the propagation diagram, we construct and maintain a number of data structures that allow efficient computation and updates of representatives.

Consider an iteration of our algorithm. Denote the currently active nodes on ℓ by $u_1, \ldots, u_k$, and assume that they are listed by their order of entering S. We denote this set by $S(\ell)$ and assume that it is stored and maintained as a doubly linked list ordered according to the position of the nodes on ℓ. In Step 3.1, we update the data structures related to the propagation diagram $\mathcal{I}(\ell, \ell_1, f)$. According to our definition, the propagation set $I(u) = I(u; \ell, \ell_1, f)$ of a node $u \in \ell$ consists of all edges (u, v_1) in $E(\ell, \ell_1, f)$ such that $\delta(u) + c(u, v_1) < \delta(u_i) + c(u_i, v_1)$, for $i = 1, \ldots, k$. Clearly, the set $I(u)$ can be viewed and described as a subset of the set of nodes v_1 on ℓ_1 that satisfy the following three conditions:

C1. The nodes u and v_1 are adjacent in G_ε by an edge that crosses $\mathbf{f}$;

C2. $\delta(u) + c(u, v_1) < \delta(u_i) + c(u_i, v_1)$, for $i = 1, \ldots, k$;

C3. The node v_1 is in S^a.

We construct and maintain separate data structures for the nodes on ℓ_1 satisfying each of these three conditions: The data structure related to **C1** is called *Adjacency Diagram* and is

denoted by $\mathcal{A}(\ell, \ell_1, f)$. It consists of sets $A(u, \ell_1)$, for all nodes u on ℓ, where the set $A(u, \ell_1)$ consists of the nodes on ℓ_1 that satisfy **C1**. This data structure is static. The data structure related to **C2** is, in fact, a dynamic additive Voronoi diagram on ℓ_1 for the active nodes on ℓ with respect to the weighted distance function $c(u, x)$ defined and studied in Section 2, see (1). Finally, the nodes on ℓ_1 that are in S^a are stored in a dynamic doubly-linked list and organized in a binary search tree with respect to their position on ℓ_1. We denote this data structure by $S^a(\ell_1)$. The lists $S(\ell)$ and $S^a(\ell_1)$ are readily maintained throughout the algorithm in logarithmic time per operation. Next, we describe in detail the construction and maintenance of these data structures.

Adjacency Diagram: The Adjacency Diagram $\mathcal{A}(\ell, \ell_1, f)$ consists of sets $A(u, \ell_1)$, for all nodes u on ℓ. We assume that the nodes on ℓ_1 are stored in an ordered list $V(\ell_1)$ according to their position on that segment. For any fixed node $u \in \ell$, the adjacency set $A(u, \ell_1)$ will be computed and stored as a sublist of the list $V(\ell_1)$. We denote this sublist by $\bar{A}(u, \ell_1)$.

We reduce the size of $\bar{A}(u, \ell_1)$ by replacing each portion of consecutive nodes in them by a pair of pointers to the first and to the last node in that portion. (Isolated nodes are treated as portions of length one.) Hence, each sublist $\bar{A}(u, \ell_1)$ is an ordered list of pairs of pointers identifying portions of consecutive nodes in the underlying list $V(\ell_1)$. The size of the sublists implemented in this way is proportional to the number of the consecutive portions they contain. Next, we discuss the structure of the lists $\bar{A}(u, \ell_1)$ and show that their size is bounded by a small constant.

According to our definitions (Section 4), an edge (u, u_1) is present in $A(u, \ell_1)$ if the local shortest path $\hat{\pi}(u, u_1; f)$ does not touch the boundary of f, where the path $\hat{\pi}(u, u_1; f)$ was defined in (22). We refer to intervals on ℓ_1 with both of their end-points being Steiner points as *Steiner intervals*. Furthermore, we say that a Steiner interval is covered by the set $A(u, \ell_1)$ if all Steiner points, including its end-points, are in $A(u, \ell_1)$. Clearly, each maximal interval covered by $A(u, \ell_1)$ corresponds to and defines a portion of consecutive nodes on ℓ_1 that are adjacent to u. Moreover, by our definition, the list $\bar{A}(u, \ell_1)$ consists of the pairs of pointers to the end-points of the maximal intervals covered by $A(u, \ell_1)$. In the next lemma, we show that there are at most seven maximal Steiner intervals covered by $A(u, \ell_1)$.

Lemma 5.1 *The number of the maximal intervals covered by $A(u, \ell_1)$ is at most seven. The corresponding ordered list $\bar{A}(u, \ell_1)$ can be computed in $O(\log K(\ell_1))$ time, where $K(\ell_1)$ denotes the number of Steiner points on ℓ_1.*

Proof: Presented in Appendix 3. □

We assume that the nodes that are end-points of the maximal Steiner intervals covered by the sets $A(u, \ell_1)$, for all nodes $u \in \ell_1$, are pre-computed in a preprocessing step and stored in the lists $\bar{A}(u, \ell_1)$ as discussed above. Lemma 5.1 implies that this preprocessing related to the group (ℓ, ℓ_1, f) takes $O(K(\ell) \log K(\ell_1))$ time, where $K(\ell)$ and $K(\ell_1)$ denote the number of the nodes on ℓ and ℓ_1, respectively. Next, we discuss the Voronoi diagram data structure related to condition **C2**.

Dynamic Additive Voronoi Diagram: We assumed that the currently active nodes, $u_1, \ldots, u_k$ on ℓ, are listed by order of their insertion into S. So, for the distances of these nodes to the source, we have $\delta(u_1) \leq \cdots \leq \delta(u_k)$. We view the distance $\delta(u_i)$ as an additive

weight assigned to the node u_i, and consider the additive Voronoi diagram of $u_1, \ldots, u_k$ on ℓ_1 with respect to the weighted distance function, introduced and studied in Section 2 and defined by (1). From the definition (see (1)), the weighted distance $c(u, x)$ for a node u on ℓ and a point $x \in \ell_1$ is given by

$$c(u, x) = c(u, x; \mathbf{f}) = \min_{a, a_1 \in F} \{w|ua| + w_f|aa_1| + w_1|a_1x|\},$$

where F is the plane containing the face $\mathbf{f}$; w, w_1 are the weights of the cells containing ℓ and ℓ_1, respectively, and w_f is the weight associated to the face f. An important observation for our discussion here is that if x is a node on ℓ_1 adjacent to u, then the cost of the edge (u, x) is $c(u, x)$.

We denote the end-points of the segment ℓ_1 by A_1 and B_1 and assume that it is oriented so that $A_1 < B_1$. For $i = 1, \ldots, k$, the Voronoi cell $\mathcal{V}(u_i)$ is defined as the set of points on ℓ_1

$$\mathcal{V}(u_i) = \{x \in (A_1, B_1) \ : \ \delta(u_i) + c(u_i, x) \leq \delta(u_j) + c(u_j, x) \text{ for } j \neq i\},$$

where ties are resolved in favor of the node that has entered S earlier. Clearly, the Voronoi diagram $\mathcal{V}(u_1, \ldots, u_k)$ is a partitioning of (A_1, B_1) into a set of intervals, where each interval belongs to exactly one of the Voronoi cells. Hence, $\mathcal{V}(u_1, \ldots, u_k)$ is completely described by a set of points $A_1 = x_0 < x_1 < \cdots < x_m < x_{m+1} = B_1$ and an assignment between the intervals (x_j, x_{j+1}), for $j = 0, \ldots, m$, and the cells of the diagram.

We assume that $\mathcal{V}(u_1, \ldots, u_k)$ is known and stored. We further assume that a node v on ℓ has been extracted by the extract-min operation in Step 1 of our algorithm. In Step 3.1, we need to add the new site v and to compute the Voronoi diagram $\mathcal{V}(u_1, \ldots, u_k, v)$. Next we show how this can be achieved in $O(\log^2 \frac{1}{\varepsilon})$ time. First, the following lemma shows that the Voronoi cell of v has a simple structure.

Lemma 5.2 *Let $u_1, \ldots, u_k$ be the active nodes on ℓ and let v be the last node inserted in S. Then the Voronoi cell $\mathcal{V}(v)$, in the Voronoi diagram $\mathcal{V}(u_1, \ldots, u_k, v)$, is either empty or consists of a single interval on ℓ_1.*

Proof: By our assumptions $\delta(u_i) \leq \delta(v)$, for $i = 1, \ldots, k$. The Voronoi cell $\mathcal{V}(v)$ can be represented as an intersection $\mathcal{V}(v) = \cap_{i=1}^{k} \mathcal{V}_i(v)$, where the sets $\mathcal{V}_i(v)$ are defined by $\mathcal{V}_i(v) = \{x \in \ell_1 \ : \ \delta(v) - \delta(u_i) + c(v, x) < c(u_i, x)\}$. By Theorem 2.1, each of $\mathcal{V}_i(v)$ is either empty or is an interval on ℓ_1, and thus the same is true for their intersection. □

Using the above lemma, we easily obtain a bound on the size of the Voronoi diagrams.

Corollary 5.1 *The number of the intervals comprising the diagram $\mathcal{V}(u_1, \ldots, u_k)$ does not exceed $2k - 1$.*

Next, we present and analyze an efficient procedure which, given the Voronoi diagram $\mathcal{V}(u_1, \ldots, u_k)$ and a new node v inserted in S, determines the Voronoi diagram $\mathcal{V}(u_1, \ldots, u_k, v)$. This includes computation of the Voronoi cell $\mathcal{V}(v)$, update of the set of points $x_1, \ldots, x_m$ describing $\mathcal{V}(u_1, \ldots, u_k)$ to another set describing $\mathcal{V}(u_1, \ldots, u_k, v)$ and update of the assignment information between intervals and Voronoi cells.

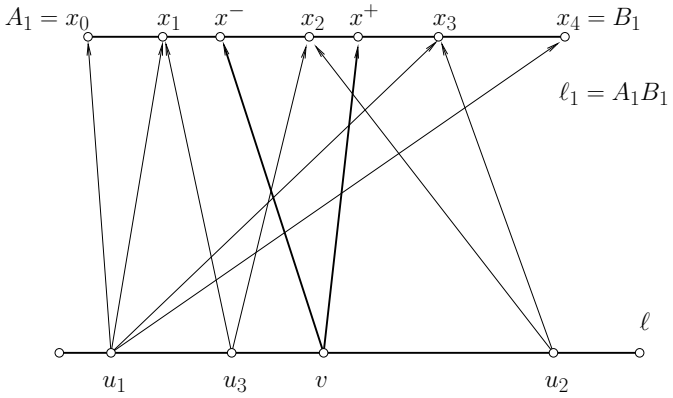

Figure 12: The figure illustrates updates of the diagram $\mathcal{V}$. The Voronoi diagram $\mathcal{V}(u_1, u_2, u_3)$ for the nodes u_1, u_2, and u_3 is characterized by the sequence $\{x_0 < x_1 < x_2 < x_3 < x_4\}$ and the assignment $\mathcal{V}(u_1) = (x_0, x_1) \cup (x_3, x_4)$, $\mathcal{V}(u_2) = (x_2, x_3)$, $\mathcal{V}(u_3) = (x_1, x_2)$. After computation of the Voronoi cell $\mathcal{V}(v) = (x^-, x^+)$ the Voronoi diagram $\mathcal{V}(u_1, u_2, u_3, v)$ is characterized by the sequence $\{x_0 < x_1 < x^- < x^+ < x_3 < x_4\}$ and the assignment $\mathcal{V}(u_1) = (x_0, x_1) \cup (x_3, x_4)$, $\mathcal{V}(u_2) = (x^+, x_3)$, $\mathcal{V}(u_3) = (x_1, x^-)$, $\mathcal{V}(v) = (x^-, x^+)$.

According to Lemma 5.2, the Voronoi cell $\mathcal{V}(v)$ is an interval, which we denote by (x^-, x^+). Let M be any of the points $x_1, \ldots, x_m$ characterizing the diagram $\mathcal{V}(u_1, \ldots, u_k)$. The following claim shows that the relative position of M with respect to the interval (x^-, x^+) can be determined in constant time.

Claim 5.1 *The relative position of M with respect to the interval (x^-, x^+) can be determined in $O(1)$ time.*

Proof: By the definition of point M, it follows that there are two nodes u_{i_1} and u_{i_2} such that $\delta(u_{i_1}) + c(u_{i_1}, M) = \delta(u_{i_2}) + c(u_{i_2}, M)$. We denote the latter value by $d(M)$ and note that $d(M) \leq \delta(u_i) + c(u_i, M)$, for $i = 1, \ldots, k$. Then, we compute the value $d(v, M) = \delta(v) + c(v, M)$ and compare it with $d(M)$.

If $d(v, M) < d(M)$, then we have $M \in (x^-, x^+)$ and thus $x^- < M < x^+$. In the case where $d(v, M) \geq d(M)$, we compute the Voronoi cell $\triangle(v)$ of v in the three cites diagram $\mathcal{V}(u_{i_1}, u_{i_2}, v)$. By Lemma 5.2, the cell $\triangle(v)$ is an interval on ℓ_1. Since M must be outside $\triangle(v)$ and $(x^-, x^+) \subset \triangle(v)$, it follows that the relative position between M and (x^-, x^+) is the same as the relative position between M and $\triangle(v)$.

The claimed time bound follows from the described procedure, which besides the constant number of simple computations, involves a constant number of evaluations of the function $c(\cdot, \cdot)$ and eventually solving of the equations $c(u_i, x) - c(v, x) = \delta(v) - \delta(u_i)$, for $i = i_1, i_2$.□

We derive the following binary search procedure, which computes the Voronoi cell $\mathcal{V}(v)$.

ALGORITHM: Voronoi cell $\mathcal{V}(v)$
Input: The sequence $X = \{A_1 = x_0 < x_1 < \cdots < x_m < x_{m+1} = B_1\}$.
Output: Points x^- and x^+ such that $\mathcal{V}(v) = (x^-, x^+)$.
A. Compute the point x^- first by the following:
1. **While** $|X| > 2$ **do** Steps 1.1 – 1.3 below
   1.1. Find the median M of the sequence X.
   1.2. Determine the relative position between M and x^-.
   1.3 **If** $x^- < M$ **then** set $X = \{x_0 < \cdots < M\}$ **else** set $X = \{M < \cdots < x_{m+1}\}$.
2. **If** $|X| = 2$ compute x^- directly.
B. Compute the point x^+ in the same way.

Once the cell $\mathcal{V}(v) = (x^-, x^+)$ has been computed, the update of diagram $\mathcal{V}(u_1, \ldots, u_k)$ to diagram $\mathcal{V}(u_1, \ldots, u_k, v)$ can be done in a natural way. The sorted sequence of points $X(u_1, \ldots, u_k, v)$ characterizing the diagram $\mathcal{V}(u_1, \ldots, u_k, v)$ is obtained from the sequence $X(u_1, \ldots, u_k) = \{x_0 < \cdots < x_{m+1}\}$ by inserting the points x^- and x^+ at their positions and by deleting points (if any) $x^- < x_{j^-+1} < \cdots < x_{j^+-1} < x^+$ lying inside the interval (x^-, x^+), where x_{j^-} and x_{j^+} are the left and the right neighbors of the points x^- and x^+, respectively. We need to delete each of the intervals (x_j, x_{j+1}), for $j = j^-, \ldots, j^+ - 1$, from the cell that contains it and to add intervals (x_{j^-}, x^-) and (x^+, x_{j^+}) to the cells that have contained intervals (x_{j^-}, x_{j^-+1}) and (x_{j^+-1}, x_{j^+}), respectively. Indeed, if the cell of some of the nodes $u_1, \ldots, u_k$ becomes empty then this node is removed from the set of active nodes in the group $E(\ell, \ell_1, f)$.

To implement all of these updates efficiently, we maintain the sequence X of points characterizing the Voronoi diagram of the currently active nodes on ℓ in an order-statistics tree, allowing us to report order statistics as well as insertions and deletions in $O(\log |X|)$ time. Based on this data structure, computation of the interval (x^-, x^+) takes $O(\log^2 |X|)$ time, since it takes $O(\log |X|)$ iterations, and each iteration takes $O(\log |X|)$ time. The update of the Voronoi diagram requires two insertions and $j^+ - j^- + 1$ deletions in X, where insertions take $O(\log |X|)$ time and deletions are done in amortized $O(1)$ time.

Let us estimate the time for the maintenance of the Voronoi diagram of the active nodes in the group $E(\ell, \ell_1, \mathbf{f})$. We denoted the total number of the nodes on ℓ by $K(\ell)$. Each of the nodes on ℓ becomes active once during the execution. Thus, each node on ℓ becomes subject of the procedure Voronoi cell exactly once. According to Corollary 5.1, the sizes of the sequences X characterizing Voronoi diagrams in the group $E(\ell, \ell_1, \mathbf{f})$ are bounded by $2K(\ell) + 1$. Therefore, the total time spent by the procedure Voronoi cell in the group $E(\ell, \ell_1, \mathbf{f})$ is $O(K(\ell) \log^2 K(\ell))$. In total, there are $O(K(\ell))$ insertions in the sequence X, and the total number of deletions, clearly, is at most the number of insertions. Hence, the total time spent for insertions and deletions is $O(K(\ell) \log K(\ell))$. Thus, the time spent for the maintenance of the Voronoi diagram in a fixed group $E(\ell, \ell_1, \mathbf{f})$ is $O(K(\ell) \log^2 K(\ell))$. Next, we discuss the computation and maintenance of a data structure that combines the adjacency diagram and the Voronoi diagram.

Propagation Diagram: As discussed above, the propagation set $I(u)$ of an active node u on ℓ is described completely by the set of nodes on ℓ_1 satisfying conditions **C1**, **C2**, and **C3**. We denote the set of nodes on ℓ_1 satisfying **C1** and **C2** with respect to u by $I'(u)$. Slightly abusing our terminology, we refer to this set again as propagation set of u. Similarly, we refer

to the set consisting of the sets $I'(u)$ for all currently active nodes as propagation diagram and denote it by $\mathcal{I}'(u_1, \ldots, u_k)$, where, as above, $u_1, \ldots, u_k$ are the currently active nodes on ℓ_1.

The difference between the originally defined propagation set $I(u)$ and the set $I'(u)$ is that the elements of $I(u)$ are the edges joining u to the nodes on ℓ_1 satisfying **C1**, **C2**, and **C3**, whereas the elements of $I'(u)$ are the nodes on ℓ_1 that satisfy **C1** and **C2**, but not necessarily **C3**. Indeed, the set $I'(u)$ is closely related to $I(u)$ and when combined with the list $S^a(\ell_1)$ describes it completely. Based on this observation, we compute and maintain the propagation diagram $\mathcal{I}'(u_1, \ldots, u_k)$ instead of the originally defined diagram.

We describe the sets $I'(u_i)$ by specifying the maximal Steiner intervals they cover. We implement these sets as ordered lists of pairs of pointers to the end-points of these intervals in the underlying list $V(\ell_1)$. The propagation sets of different active nodes do not intersect, and hence, the end-points of the maximal Steiner intervals of the propagation sets $I'(u_1), \ldots, I'(u_k)$ form a sequence, $\mathcal{I}' = \{A_1 \leq y_1 \leq z_1 \leq \cdots \leq y_{m_1} \leq z_{m_1} \leq B_1\}$, where $\ell_1 = (A_1, B_1)$. The points y_j and z_j, for $j = 1, \ldots, m_1$, are Steiner points (nodes) on ℓ_1. Any of the Steiner intervals (y_j, z_j) is a maximal Steiner interval covered by one of the sets $I'(u_1), \ldots, I'(u_k)$, whereas the Steiner points inside the intervals (z_j, y_{j+1}) do not belong to any of the sets $I'(u_i)$. Clearly, the sequence $\mathcal{I}'$ plus the assignment of the intervals (y_j, z_j) to the sets $\mathcal{I}'(u_i)$ covering them determine the diagram $\mathcal{I}'(u_1, \ldots, u_k)$. We implement sequence $\mathcal{I}'$ as an ordered list of pointers to the underlying list $V(\ell_1)$. In addition, we associate with it a binary search tree based on the position of the Steiner points on the segment ℓ_1. The diagram $\mathcal{I}'(u_1, \ldots, u_k)$ is maintained in Step 3.1 and details are as follows.

Let, as above, v be the node extracted by the extract-min operation in Step 1 in the current iteration of the algorithm. We assume that the diagram $\mathcal{I}'(u_1, \ldots, u_k)$ is known – i.e., we know the sequence $\mathcal{I}'$ as well as the assignment of the intervals (y_j, z_j) to the propagation sets $I'(u_i)$. Next, we describe the update of $\mathcal{I}'$ and the assignment information specifying $\mathcal{I}'(u_1, \ldots, u_k, v)$. By definition, $I'(v)$ consists of the nodes on ℓ_1 that lie in the Voronoi cell $\mathcal{V}(v)$ and belong to the adjacency set $A(v, \ell_1)$. By Lemma 5.2, $\mathcal{V}(v)$ is either empty or a single interval, which we have denoted by (x^-, x^+). We denote by (v^-, v^+), the largest Steiner interval inside the interval (x^-, x^+). The interval (v^-, v^+) is easily found using binary search in $O(\log K(\ell_1))$ time, where as above $K(\ell_1)$ denotes the number of Steiner points on ℓ_1. On the other hand (see Lemma 5.1), the adjacency set $A(v, \ell_1)$ consists of the nodes lying inside constant number (at most seven) of Steiner intervals, which were computed and stored as the list $\bar{A}(v, \ell_1)$. Hence, the maximal Steiner intervals specifying the propagation set $I'(v)$ can be obtained as the intersection of intervals in $\bar{A}(v, \ell_1)$ with (v^-, v^+). This is done in constant time by identifying the position of the points v^- and v^+ with respect to the elements of the list $\bar{A}(v, \ell_1)$. Clearly, the so-computed maximal Steiner intervals covered by $I'(v)$ are at most seven. We update the sequence $\mathcal{I}'$ by inserting each of the maximal Steiner intervals covered by $I'(v)$ in the same way as we inserted the interval (x^-, x^+) into the sequence X describing the Voronoi diagram. More precisely, let (y, z) be any of the maximal Steiner intervals covered by $\mathcal{I}'(v)$. We insert the points y and z at their positions in the ordered sequence $\mathcal{I}'$, and then we delete the points of $\mathcal{I}'$ between y and z. If the interval containing y is (y_j, z_j), we set new z_j to be the Steiner point preceding y on ℓ_1. Similarly, if the interval containing z is (y_j, z_j), then we set y_j to be the Steiner point

following z on ℓ_1.

At each iteration of the algorithm, the endpoints of at most seven intervals are inserted into the sequence $\mathcal{I}'$. Hence, the size of the sequence $\mathcal{I}'$ is bounded by $14K(\ell)$ and insertions in $\mathcal{I}'$ are implemented in $O(\log K(\ell))$ time. Deletions are implemented in $O(1)$ time. The total number of insertions is $O(K(\ell))$ and the total number of deletions is at most the number of insertions. Therefore, the total time spent for the maintenance of $\mathcal{I}'$ and the propagation diagram is $O(K(\ell)(\log K(\ell) + K(\ell_1)))$.

Finally, we summarize our discussion on the implementation of Step 3.1. The computations and times related to a fixed triple (ℓ, ℓ_1, f) are as follows. First, in a preprocessing step the lists $\bar{A}(u, \ell_1)$, for all nodes u on ℓ, are computed in $O(K(\ell) \log K(\ell_1))$ time (Lemma 5.1). Times spent for the maintenance of the lists $S(\ell)$ and $S^a(\ell_1)$ are $O(K(\ell) \log K(\ell)$ and $O(K(\ell_1) \log K(\ell_1)$, respectively. The time spent for maintenance of the Voronoi diagram for the active nodes on ℓ requires $O(K(\ell) \log^2 K(\ell))$ time. The time for the maintenance of the Propagation Diagram is $O(K(\ell)(\log K(\ell) + \log K(\ell_1)))$. Therefore, the total time for the implementation of Step 3.1 is

$$\begin{aligned}
&\sum_{(\ell,\ell_1,f)} \big(O(K(\ell)(\log^2 K(\ell) + \log K(\ell_1)) + O(K(\ell_1) \log K(\ell_1))\big) \\
&\leq O(\frac{1}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon}) \left(\sum_{\ell} K(\ell)(\log^2 K(\ell) + \log K(\ell_1)) + \sum_{\ell_1} K(\ell_1) \log K(\ell_1) \right) \\
&\leq O(\frac{1}{\sqrt{\varepsilon}} \log^3 \frac{1}{\varepsilon}) \left(\sum_{\ell} K(\ell) + \sum_{\ell_1} K(\ell_1) \right) = O(\frac{|V_\varepsilon|}{\sqrt{\varepsilon}} \log^3 \frac{1}{\varepsilon}),
\end{aligned}$$

where we have used Lemma 3.1 to estimate that the number of triples $(\ell, \ell_1, \mathbf{f})$ with a fixed first or second element is $O(\frac{1}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon})$, and that $\log K(\ell)$ and $\log K(\ell_1)$ are $O(\log \frac{1}{\varepsilon})$.

Lemma 5.3 *The total time spent by the algorithm implementing Step 3.1 is* $O(\frac{|V_\varepsilon|}{\sqrt{\varepsilon}} \log^3 \frac{1}{\varepsilon})$.

5.2.3 Computation and updates of set of representatives

Next, we concentrate on the computation of representatives in Steps 3.2, 3.3 and 3.4. The set of representatives $\boldsymbol{\rho}(v; \ell, \ell_1, e)$ of an active node v on ℓ in a group $E(\ell, \ell_1, \mathbf{f})$ contains one representative for each interval (y_j, z_j) in the propagation set $I(v)$. Recall that $I(v)$ consists of a set of intervals (y_j, z_j) stored in the sequence $\mathcal{I}'$, characterizing the propagation diagram of the currently active nodes on ℓ. The representative in $\boldsymbol{\rho}(v; \ell, \ell_1, e)$, corresponding to $(y_j, z_j) \in I(v)$, is the target of the minimum cost edge from v to a node in $S^a \cap (y_j, z_j)$. By Lemma 2.1, the function $c(v, x)$ is convex and thus in any interval it has a single minimum. Let $x^*(v)$ be the point on ℓ_1, where $c(v, x)$ achieves its minimum. To efficiently compute the representatives, we compute in a preprocessing step the points $x^*(v)$, for all nodes on ℓ. From the definition of the function $c(v, x)$ and Snell's law, it follows that $x^*(v)$ is the point on ℓ_1 that is closest to v. So, each of $x^*(v)$ can be computed in constant time, which leads to $O(K(\ell))$ preprocessing time for the group $E(\ell, \ell_1, \mathbf{f})$, where $K(\ell)$ is the number of nodes on ℓ. Thus, the total time for preprocessing in all groups is $O(\frac{|V_\varepsilon|}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon})$.

We have associated two data structures to the set of nodes in S^a that lie on a fixed Steiner segment ℓ_1. First, we maintain them in a doubly-linked list and second, we maintain them in a binary-search tree, with respect to their position on ℓ_1. We show that finding a representative $\rho(v) \in \boldsymbol{\rho}(v; \ell, \ell_1, e)$ takes $O(\log \frac{1}{\varepsilon})$ time. There are three situations, where we need to compute or update $\rho(v)$:

1. New representatives $\rho(v)$ are computed when v becomes active and its propagation set is non-empty. We need to compute one new representative for each maximal Steiner interval (y, z) in the propagation set $I(v)$. Recall that there are at most seven such intervals and they were computed and stored in the sequence $\mathcal{I}'$.

   To compute $\rho(v)$ in the interval (y, z), we determine the leftmost and rightmost nodes from S^a inside the interval (y, z). This is done by finding the position of the points y and z in the sequence of nodes currently in S^a. Let the leftmost and the rightmost nodes from S^a in (y, z) be y^a and z^a, respectively. Then, we determine the position of the point $x^*(v)$ with respect to y^a and z^a.

   If it is to the left of y^a, then $\rho(v) = y^a$. If it is to the right of z^a, then $\rho(v) = z^a$. If $x^*(v)$ is inside (y^a, z^a), we determine the two nodes in S^a immediately to the left and to the right of $x^*(v)$, and $\rho(v)$ is one of these two nodes. Using the binary-search tree on S^a, the nodes y^a and z^a and eventually the nodes neighboring $x^*(v)$ are determined in $O(\log \frac{1}{\varepsilon})$ time.

2. When some representative $\rho(v)$ is removed from S^a, a new representative for v is one of the neighbors of $\rho(v)$ in the doubly-linked list S^a that lie in the same interval (y, z) as $\rho(v)$. This is done in $O(1)$ time.

3. When some interval of the propagation set $I(v)$ shrinks and the current representative $\rho(v)$ is no longer inside this interval, then $\rho(v)$ is updated as follows. Let, as above, y^a and z^a be the leftmost and the rightmost nodes from S^a, respectively, in the updated interval. Then, if $\rho(v)$ lies to the left of y^a, we set $\rho(v) = y^a$. If $\rho(v)$ is to the right of z^a, we set $\rho(v) = z^a$. As above, determination of the nodes y^a and z^a is done in $O(\log \frac{1}{\varepsilon})$ time.

To complete our analysis, we need to estimate the total number of representatives which are computed by our algorithm. Each pair (representative, predecessor) relates to the edge joining them. Since such a pair can be computed at most once by the algorithm, the total number of representatives related to nodes that are vertices of $\mathcal{D}$ is bounded by the total number of edges incident to these nodes, which is $O(|V_\varepsilon|)$. It remains to estimate the number of representatives which are related to nodes that are Steiner points. Consider an iteration for a node v that is a Steiner point. There are $O(\frac{1}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon})$ triples in $\mathcal{T}(v)$, and at most nine new representatives are computed in Step 3.2. For each predecessor in $R^{-1}(v)$ that is a Steiner point, a single representative is computed. The number of predecessors $|R^{-1}(v)|$ is $O(\frac{1}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon})$. Hence, in a single iteration, $O(\frac{1}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon})$ representatives related to Steiner points are computed. Since the number of iterations is $O(|V_\varepsilon|)$ and the computation of a single representatives takes $O(\log \frac{1}{\varepsilon})$ time, we obtain that the total time for the execution of Steps 3.2 and 3.3 is $O(\frac{|V_\varepsilon|}{\sqrt{\varepsilon}} \log^2 \frac{1}{\varepsilon})$.

Finally, the number of priority queue operations executed in Step 3.4 is bounded by the number of computed representatives. Thus, the total time for Step 3.4 is $O(\frac{|V_\varepsilon|}{\sqrt{\varepsilon}} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$.

5.2.4 Complexity of the algorithm and the main result

Here, we summarize our discussion from the previous three subsections and state our main result. Step 1 of our algorithm takes $O(|V_\varepsilon| \log \frac{n}{\varepsilon})$ time. Step 2 requires $O(|V_\varepsilon|)$ time. By Lemma 5.3, Step 3.1 takes in total $O(\frac{|V_\varepsilon|}{\sqrt{\varepsilon}} \log^3 \frac{1}{\varepsilon})$ time. The total time for implementation of Steps 3.2 and 3.3 is $O(\frac{|V_\varepsilon|}{\sqrt{\varepsilon}} \log^2 \frac{1}{\varepsilon})$ and the total time for Step 3.4 is $O(\frac{|V_\varepsilon|}{\sqrt{\varepsilon}} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$. By Lemma 3.1, we have that $|V_\varepsilon| = O(\frac{n}{\varepsilon^2} \log \frac{1}{\varepsilon})$. We have thus established the following:

Theorem 5.1 *The SSSP problem in the approximation graph G_ε can be solved in $O(\frac{n}{\varepsilon^{2.5}} \log \frac{n}{\varepsilon} \log^3 \frac{1}{\varepsilon})$ time.*

Consider the polyhedral domain $\mathcal{D}$. Starting from a vertex v_0 of $\mathcal{D}$, our algorithm solves the SSSP problem in the corresponding graph G_ε and constructs a shortest paths tree rooted at v_0. According to Theorem 4.1, the computed distances from v_0 to all other vertices of $\mathcal{D}$ (and to all Steiner points) are within a factor of $1 + \varepsilon$ of the cost of the corresponding shortest paths. Using the definition of the edges of G_ε, an approximate shortest path can be output by simply replacing the edges in the discrete path with the corresponding local shortest paths used to define their costs. This can be done in time proportional to the number of segments in this path, because computation of the local shortest paths takes $O(1)$ time. The approximate shortest paths tree rooted at v_0 and containing all Steiner points and vertices of $\mathcal{D}$ can be output in $O(|V_\varepsilon|)$ time. Thus, the algorithm we described solves the WSP3D problem and the following theorem states the result.

Theorem 5.2 *Let $\mathcal{D}$ be a weighted polyhedral domain consisting of n tetrahedra and $\varepsilon \in (0, 1)$. The weighted shortest path problem in three dimensions (WSP3D), requiring the computation of approximate shortest paths from a source vertex to all other vertices of $\mathcal{D}$, can be solved in $O(\frac{n}{\varepsilon^{2.5}} \log \frac{n}{\varepsilon} \log^3 \frac{1}{\varepsilon})$ time.*

6 Conclusions

This paper generalizes the weighted region problem, originally studied in 1991 by Mitchell and Papadimitriou [21] for the planar setting, to 3-d weighted domains. We present the first polynomial time approximation scheme for the WSP3D problem. The complexity of our algorithm is independent of the weights, but depends upon the geometric features of the given tetrahedra as stated in Lemma 3.1.

There are some fairly standard techniques which can be employed here to remove the dependence on geometry (cf., [1]), provided that there is an estimate known on the maximum number of segments (i.e., the combinatorial complexity) in weighted shortest paths in three dimensions. It can be shown that the combinatorial complexity of weighted shortest paths in planar case is $\Theta(n^2)$ [21]. We conjecture that the same bound holds in three dimensions, but the proof techniques in [21] do not seem to apply here, since they use planarity. If the

combinatorial complexity of these paths in three dimensions is a polynomial in n, then we can remove the dependence on the geometry by increasing the run time by a polynomial factor in n. We do not recommend this approach, since the increase in the running time will be significant. Already, in the planar case (and in terrains), in an experimental study [18], it was shown that a constant number of Steiner points suffice to produce high-quality paths. We believe that the same holds here and this merits further investigation.

This paper also investigated additive Voronoi diagrams in heterogeneous media. We studied a fairly simple scenario and already the analysis of that was very technical and cumbersome. It is desirable to find simpler and more elegant ways to understand the combinatorics of these diagrams. Nevertheless, we believe that the discretization scheme and the algorithms presented here can be used successfully for efficient computation of approximate Voronoi diagrams in heterogeneous media.

Our algorithm does not require any complex data structures or primitives and as such should be implementable and even practical. Its structure allows Steiner points to be generated "on the fly" as the shortest path wavefront propagates though the weighted domain. This feature allows the design of more compact and adaptive implementation schemes that can be of high practical value.

One of the classical problems that motivated this study is the unweighted version of this problem, namely the ESP3D problem. There, we need to find a shortest path between a source and a target point, lying completely in the free space, avoiding three-dimensional polyhedral obstacles. We can use our techniques to solve this problem, though this will require triangulating (i.e., tetrahedralization) the free space. As outlined above, the complexity of our algorithm depends upon the geometry of these tetrahedra; so it is natural to ask whether the free space can be partitioned into nice tetrahedra? Unfortunately, there is no simple answer to this question which has been an important topic of study in computational and combinatorial geometry for several decades. Nevertheless, our algorithm provides a much simpler and so far the fastest method for solving the ESP3D problem, provided the free space is already partitioned into non-degenerate tetrahedra.

Combining the techniques of answering weighted shortest path queries on polyhedral surfaces [2] and the existence of nice separators for well-shaped meshes [20], we believe that our construction presented in this paper can be used for answering (approximate) weighted shortest path queries in 3-d.

References

[1] Pankaj K. Agarwal, R. Sharathkumar, and Hai Yu. Approximate Euclidean shortest paths amid convex obstacles. In Claire Mathieu, editor, *SODA*, pages 283–292. SIAM, 2009.

[2] Lyudmil Aleksandrov, Hristo Djidjev, Hua Guo, Anil Maheshwari, Doron Nussbaum, and Jörg-Rüdiger Sack. Algorithms for approximate shortest path queries on weighted polyhedral surfaces. *Discrete & Computational Geometry*, 44(4):762–801, 2010.

[3] Lyudmil Aleksandrov, Anil Maheshwari, and Jörg-Rüdiger Sack. Approximation algorithms for geometric shortest path problems. In *STOC*, pages 286–295, 2000.

[4] Lyudmil Aleksandrov, Anil Maheshwari, and Jörg-Rüdiger Sack. Determining approximate shortest paths on weighted polyhedral surfaces. *J. ACM*, 52(1):25–53, 2005.

[5] Tetsuo Asano, David Kirkpatrick, and Chee Yap. Pseudo approximation algorithms with applications to optimal motion planning. *Discrete & Computational Geometry*, 31(1):139–171, 2004.

[6] Franz Aurenhammer and Rolf Klein. *Handbook of Computational Geometry*, chapter Voronoi Diagrams. North Holland, 2000.

[7] Chandrajit L. Bajaj. The algebraic degree of geometric optimization problems. *Discrete & Computational Geometry*, 3:177–191, 1988.

[8] John F. Canny and John H. Reif. New lower bound techniques for robot motion planning problems. In *FOCS*, pages 49–60. IEEE, 1987.

[9] Joonsoo Choi, Jürgen Sellen, and Chee-Keng Yap. Approximate Euclidean shortest paths in 3-space. *Int. J. Comput. Geometry Appl.*, 7(4):271–295, 1997.

[10] Joonsoo Choi, Jürgen Sellen, and Chee-Keng Yap. Precision-sensitive Euclidean shortest path in 3-space. *SIAM J. Comput.*, 29(5):1577–1595, 2000.

[11] Kenneth L. Clarkson. Approximation algorithms for shortest path motion planning (extended abstract). In *STOC*, pages 56–65. ACM, 1987.

[12] B. Cox and B. Treeby. Artifact trapping during time reversal photoacoustic imaging for acoustically heterogeneous media. *IEEE Trasaction on Medical Imaging*, 29(2):387–396, 2010.

[13] Sariel Har-Peled. Constructing approximate shortest path maps in three dimensions. *SIAM J. Comput.*, 28(4):1182–1197, 1999.

[14] Philip L. Inderwiesen and Tien-When Lo. *Fundamentals of Seismic Tomography*, volume 6 of *Geophysical Monograph Series*. 1994.

[15] Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer, 1989.

[16] Rolf Klein, Elmar Langetepe, and Zahra Nilforoushan. Abstract Voronoi diagrams revisited. *Computational Geometry*, 42(9):885 – 902, 2009.

[17] J. Krozel, S. Penny, J. Prete, and J.S.B. Mitchell. Comparison of algorithms for synthesizing weather avoidance routes in transition airspace. *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*, 1:446–461, 2004.

[18] Mark Lanthier, Anil Maheshwari, and Jörg-Rüdiger Sack. Approximating shortest paths on weighted polyhedral surfaces. *Algorithmica*, 30(4):527–562, 2001.

[19] Ngoc-Minh Lê. Abstract Voronoi diagram in 3-space. *Journal of Computer and System Sciences*, 68(1):41 – 79, 2004.

[20] Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. Automatic mesh partitioning. In Alan George, John Gilbert, and Joseph Liu, editors, *Graphs Theory and Sparse Matrix Computation*, The IMA Volumes in Mathematics and its Application, pages 57–84. Springer-Verlag, 1993. Vol 56.

[21] Joseph S. B. Mitchell and Christos H. Papadimitriou. The weighted region problem: Finding shortest paths through a weighted planar subdivision. *J. ACM*, 38(1):18–73, 1991.

[22] Joseph S. B. Mitchell and Micha Sharir. New results on shortest paths in three dimensions. In *SoCG*, pages 124–133. ACM, 2004.

[23] Christos H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Inf. Process. Lett.*, 20(5):259–263, 1985.

[24] J.H. Reif and Z. Sun. Movement planning in the presence of flows. *Algorithmica (New York)*, 39(2):127–153, 2004.

[25] Zheng Sun and John Reif. Bushwhack: An approximation algorithm for minimal paths through pseudo-Euclidean spaces. In Peter Eades and Tadao Takaoka, editors, *Algorithms and Computation*, volume 2223 of *Lecture Notes in Computer Science*, pages 160–171. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-45678-3_15.

[26] Zheng Sun and John H. Reif. On finding approximate optimal paths in weighted regions. *Journal of Algorithms*, 58(1):1 – 32, 2006.

[27] T. Varslot and G. Taralsen. Computer simulation of forward wave propagation in soft tissue. *IEEE Trasactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 52(9):1473–1482, 2005.

A Appendix

A.1 Proof of Proposition 2.2

Proposition 2.2: *The second mixed derivative of the function $x = x(y, \alpha)$ is negative, i.e. $x_{y\alpha} < 0$.*

Proof: First, we consider the case where $w^- = w^+$. In this case, the function $x(y, \alpha)$ can be represented and differentiated explicitly. Recall, that the path $\bar{\pi}(v, \mathbf{x})$ in this case either consists of a single segment or is a three segment path as shown in Figure 3 (b). So, in the case where the path consists of a single segment, we have $x(y, \alpha) = y \cot \alpha$. In the case where the path consists of three segments, $x(y, \alpha) = y \cos \alpha / \sqrt{\kappa^2 - \cos^2 \alpha}$, where $\kappa = w/w^-$. The mixed derivatives $x_{y\alpha}$ of these two functions are $-1/\sin^2 \alpha$ and $-\kappa^2 \sin \alpha / (\kappa^2 - \cos^2 \alpha)^{\frac{3}{2}}$, respectively, and both are readily negative.

Next, we consider the case where $w^- \neq w^+$. We introduce some additional notation as necessary for our presentation below (Figure 2). We denote the coordinates of the bending point a of the path $\bar{\pi}(v, \mathbf{x})$ by $a = (x^-, y^+)$. Furthermore, we set $x^+ = x - x^-$ and $y^- = y - y^+$. Clearly, x^-, x^+, y^-, and y^+ can be viewed as functions of the independent variables y and α. We have $x = x^-(y^-(y, \alpha), \alpha) + x^+(y^+(y, \alpha), \alpha)$ and thereby

$$x_y = x^-_{y^-} y^-_y + x^+_{y^+} y^+_y. \tag{37}$$

We differentiate(37) with respect to α and obtain $x_{y\alpha} = A + A_1 + A_2$, where

$$A = x^-_{y^-} y^-_{y\alpha} + x^+_{y^+} y^+_{y\alpha},\ A_1 = x^-_{y^- y^-} y^-_\alpha y^-_y + x^+_{y^+ y^+} y^+_\alpha y^+_y,\ \text{and}\ A_2 = x^-_{y^- \alpha} y^-_y + x^+_{y^+ \alpha} y^+_y.$$

We complete the proof by showing that the terms A, A_1, and A_2, are negative.

We begin with the term $A = x^-_{y^-} y^-_{y\alpha} + x^+_{y^+} y^+_{y\alpha}$. From the identity $y = y^- + y^+$, it follows that $y^-_{y\alpha} + y^+_{y\alpha} = 0$ and hence, $A = (x^-_{y^-} - x^+_{y^+}) y^-_{y\alpha}$. From our notation, Snell's law, and the relation $\cos \alpha = \cos \theta \sin \varphi$, we derive the following:

$$\begin{aligned} x^+ &= \frac{z^+ \cos \alpha}{\sqrt{\kappa^2 - \sin^2 \varphi}}, \qquad y^+ = \frac{z^+ \sqrt{\sin^2 \varphi - \cos^2 \alpha}}{\sqrt{\kappa^2 - \sin^2 \varphi}}, \\ \text{and} \qquad & \\ x^- &= \frac{z^- \cos \alpha}{\sqrt{1 - \sin^2 \varphi}}, \qquad y^- = \frac{z^- \sqrt{\sin^2 \varphi - \cos^2 \alpha}}{\sqrt{1 - \sin^2 \varphi}}. \end{aligned} \tag{38}$$

First, we compute $x^-_{y^-}$ and $x^+_{y^+}$. We denote $\sin^2 \varphi$ by σ, differentiate x^+ and y^+ with respect to σ and obtain $x^+_{y^+}$ as the ratio x^+_σ / y^+_σ. We differentiate expressions (38) with respect to σ and obtain

$$\{x \in \mathcal{H} : c(v, x) + C < c(v', x)\} x^+_\sigma = \frac{z^+ \cos \alpha}{2(\kappa^2 - \sigma)^{\frac{3}{2}}}, \qquad y^+_\sigma = \frac{z^+ (\kappa^2 - \cos^2 \alpha)}{2\sqrt{\sigma - \cos^2 \alpha}(\kappa^2 - \sigma)^{\frac{3}{2}}} \tag{39}$$

and hence

$$x^+_{y^+} = x^+_\sigma / y^+_\sigma = \frac{\cos \alpha \sqrt{\sin^2 \varphi - \cos^2 \alpha}}{\kappa^2 - \cos^2 \alpha}. \tag{40}$$

Similarly,

$$x^-_\sigma = \frac{z^- \cos\alpha}{2(1-\sigma)^{\frac{3}{2}}}, \qquad y^-_\sigma = \frac{z^-(1-\cos^2\alpha)}{2\sqrt{\sigma-\cos^2\alpha}(1-\sigma)^{\frac{3}{2}}}, \qquad \text{and}$$

$$x^-_{y^-} = x^-_\sigma / y^-_\sigma = \frac{\cos\alpha\sqrt{\sin^2\varphi-\cos^2\alpha}}{1-\cos^2\alpha}. \tag{41}$$

So, for the difference $x^-_{y^-} - x^+_{y^+}$ we have

$$\begin{aligned} x^-_{y^-} - x^+_{y^+} &= \cos\alpha\sqrt{\sin^2\varphi-\cos^2\alpha}\left(\frac{1}{1-\cos^2\alpha} - \frac{1}{\kappa^2-\cos^2\alpha}\right) \\ &= (\kappa^2-1)\frac{\cos\alpha\sqrt{\sin^2\varphi-\cos^2\alpha}}{(1-\cos^2\alpha)(\kappa^2-\cos^2\alpha)}. \end{aligned} \tag{42}$$

The latter shows that

$$\operatorname{sign}(x^-_{y^-} - x^+_{y^+}) = \operatorname{sign}(\kappa^2-1). \tag{43}$$

To prove the negativity of $A = (x^-_{y^-} - x^+_{y^+})y^-_{y\alpha}$, we show that $\operatorname{sign}(y^-_{y\alpha}) = \operatorname{sign}(1-\kappa^2)$. We have

$$y^-_y = y^-_\sigma / y_\sigma = y^-_\sigma/(y^-_\sigma + y^+_\sigma) = \frac{1}{1+y^+_\sigma/y^-_\sigma} \quad \text{and thus} \quad y^-_{y\alpha} = -\frac{(y^+_\sigma/y^-_\sigma)_\alpha}{(1+y^+_\sigma/y^-_\sigma)^2}.$$

Hence, it is sufficient to show that

$$\operatorname{sign}((y^+_\sigma/y^-_\sigma)_\alpha) = \operatorname{sign}(\kappa^2-1). \tag{44}$$

So, we continue by establishing the sign of the derivative $(y^+_\sigma/y^-_\sigma)_\alpha$. We use (39) and (41) and compute the ratio y^+_σ/y^-_σ as follows

$$y^+_\sigma/y^-_\sigma = \frac{z^+(\kappa^2-\cos^2\alpha)(1-\sigma)^{3/2}}{z^-(1-\cos^2\alpha)(\kappa^2-\sigma)^{3/2}} = (z^+/z^-)BC, \quad \text{where}$$

$$B = \frac{\kappa^2-\cos^2\alpha}{1-\cos^2\alpha} \qquad \text{and} \qquad C = \left(\frac{1-\sigma}{\kappa^2-\sigma}\right)^{3/2} = \left(\frac{1-\sin^2\varphi}{\kappa^2-\sin^2\varphi}\right)^{3/2}. \tag{45}$$

Then, we compute the derivatives B_α and C_α using the expressions (45)

$$B_\alpha = \frac{\sin 2\alpha(1-\cos^2\alpha) - \sin 2\alpha(\kappa^2-\cos^2\alpha)}{(1-\cos^2\alpha)^2} = \frac{(1-\kappa^2)\sin 2\alpha}{(1-\cos^2\alpha)^2} \qquad \text{and}$$

$$C_\alpha = \frac{3}{2}\left(\frac{1-\sin^2\varphi}{\kappa^2-\sin^2\varphi}\right)^{1/2} \frac{(-\sin 2\varphi)(\kappa^2-\sin^2\varphi) - (-\sin 2\varphi)(1-\sin^2\varphi)}{(\kappa^2-\sin^2\varphi)^2}\varphi_\alpha =$$

$$\frac{3}{2}\left(\frac{1-\sin^2\varphi}{\kappa^2-\sin^2\varphi}\right)^{1/2} \frac{\sin 2\varphi(1-\kappa^2)}{(\kappa^2-\cos^2\alpha)^2}\varphi_\alpha$$

and obtain

$$
\begin{aligned}
(z^-/z^+)(y^+_\sigma/y^-_\sigma)_\alpha &= B_\alpha C + BC_\alpha \qquad (46)\\
&= \left(\frac{1-\sin^2\varphi}{\kappa^2-\sin^2\varphi}\right)^{\frac{3}{2}} \frac{(1-\kappa^2)\sin 2\alpha}{(1-\cos^2\alpha)^2} + \frac{3}{2}\left(\frac{1-\sin^2\varphi}{\kappa^2-\sin^2\varphi}\right)^{\frac{1}{2}} \frac{(\kappa^2-\cos^2\alpha)\sin 2\varphi(1-\kappa^2)\varphi_\alpha}{(1-\cos^2\alpha)(\kappa^2-\sin^2\varphi)^2}\\
&= (1-\kappa^2)\frac{\sqrt{1-\sin^2\varphi}}{(\kappa^2-\sin^2\varphi)^{3/2}(1-\cos^2\alpha)}D,\\
&\text{where}\quad D = \frac{\sin 2\alpha(1-\sin^2\varphi)}{1-\cos^2\alpha} + \frac{3\sin 2\varphi(\kappa^2-\cos^2\alpha)}{2(\kappa^2-\sin^2\varphi)}\varphi_\alpha.
\end{aligned}
$$

Omitting the positive multiplicative terms in (46), we derive that $\operatorname{sign}(y^+_\sigma/y^-_\sigma)_\alpha = \operatorname{sign}((1-\kappa^2)D)$ and continue with the evaluation of $\operatorname{sign}(D)$. We compute the derivative φ_α using the identity $y = y^- + y^+$, which implies $0 = y^-_\alpha + y^+_\alpha$. We differentiate expressions from (38) with respect to α and obtain

$$
\begin{aligned}
y^-_\alpha &= (z^-/2)\frac{(1-\sin^2\varphi)\sin 2\alpha + \sin 2\varphi(1-\cos^2\alpha)\varphi_\alpha}{(\sin^2\varphi-\cos^2\alpha)^{1/2}(1-\sin^2\varphi)^{3/2}},\\
y^+_\alpha &= (z^+/2)\frac{(\kappa^2-\sin^2\varphi)\sin 2\alpha + \sin 2\varphi(\kappa^2-\cos^2\alpha)\varphi_\alpha}{(\sin^2\varphi-\cos^2\alpha)^{1/2}(\kappa^2-\sin^2\varphi)^{3/2}},
\end{aligned}
\qquad (47)
$$

From these two, we obtain

$$
\varphi_\alpha = -\frac{I\sin 2\alpha(1-\sin^2\varphi)(\kappa^2-\sin^2\varphi)}{J\sin 2\varphi}, \quad \text{where} \qquad (48)
$$

$$
\begin{aligned}
I &= z^-(\kappa^2-\sin^2\varphi)^{1/2} + z^+(1-\sin^2\varphi)^{1/2} \qquad \text{and} \qquad (49)\\
J &= z^-(1-\cos^2\alpha)(\kappa^2-\sin^2\varphi)^{3/2} + z^+(\kappa^2-\cos^2\alpha)(1-\sin^2\varphi)^{3/2}.
\end{aligned}
$$

Next, we substitute φ_α from (48) in the expression D given in (46) and obtain

$$
D = \sin\alpha\cos\alpha(1-\sin^2\varphi)\frac{2J - 3I(\kappa^2-\cos^2\alpha)(1-\cos^2\alpha)}{J(1-\cos^2\alpha)}.
$$

The term $\sin\alpha\cos\alpha(1-\sin^2\varphi)$ and the denominator in this expression are positive and by expanding the numerator we have

$$
\begin{aligned}
&\operatorname{sign}(D) = \operatorname{sign}[2J - 3I(\kappa^2-\cos^2\alpha)(1-\cos^2\alpha)]\\
&= \operatorname{sign}\big[2z^-(1-\cos^2\alpha)(\kappa^2-\sin^2\varphi)^{3/2} + 2z^+(\kappa^2-\cos^2\alpha)(1-\sin^2\varphi)^{3/2}\\
&-3z^-(1-\cos^2\alpha)(\kappa^2-\cos^2\alpha)(\kappa^2-\sin^2\varphi)^{1/2} - 3z^+(1-\cos^2\alpha)(\kappa^2-\cos^2\alpha)(1-\sin^2\varphi)^{1/2}\big]\\
&= \operatorname{sign}\left[z^-(1-\cos^2\alpha)(\kappa^2-\sin^2\varphi)^{1/2}D^- + z^+(\kappa^2-\cos^2\alpha)(1-\sin^2\varphi)^{1/2}D^+\right],\\
&\text{where}\quad D^- = 3\cos^2\alpha - 2\sin^2\varphi - \kappa^2 \quad \text{and} \quad D^+ = 3\cos^2\alpha - 2\sin^2\varphi - 1. \qquad (50)
\end{aligned}
$$

Now, we observe that the terms multiplied by D^+ and by D^- are positive and show that D^- and D^+ are negative. We use $\cos\alpha = \cos\theta\sin\varphi$ and $\cos\alpha_\kappa = \cos\theta\sin\varphi_\kappa$, where $\sin\varphi = \kappa\sin\varphi_\kappa$ and obtain

$$\begin{aligned} D^+ &= 3\cos^2\alpha - 2\sin^2\varphi - 1 = 2\cos^2\alpha - 2\sin^2\varphi - \sin^2\alpha \\ &= 2\cos^2\theta\sin^2\varphi - 2\sin^2\varphi - \sin^2\alpha = -2\sin^2\varphi\sin^2\theta - \sin^2\alpha < 0 \\ \text{and}\quad D^- &= 3\cos^2\alpha - 2\sin^2\varphi - \kappa^2 = \kappa^2(3\cos^2\alpha_\kappa - 2\sin^2\varphi_\kappa - 1) = \\ &\kappa^2(-2\sin^2\varphi_\kappa\sin^2\theta - \sin^2\alpha_\kappa) < 0. \end{aligned}$$

From (50) and $D^+, D^- < O$, we get $\operatorname{sign}(D) = -1$. From (46) it follows that $\operatorname{sign}(y_u^+/y_u^-)_\alpha = -\operatorname{sign}(1-\kappa^2)$ and thus $\operatorname{sign}(y_{y\alpha}^-) = \operatorname{sign}(1-\kappa^2)$. The latter implies that $A < 0$.

Next, we consider the term A_1. From the identity $y_\alpha^- + y_\alpha^+ = y_\alpha = 0$, we get $A_1 = y_\alpha^-(x_{y^-y^-}^- y_y^- - x_{y^+y^+}^+ y_y^+)$. To evaluate the sign of y_α^-, we substitute φ_α from (48) in the expression (47) and by omitting positive multiplicative term, we obtain

$$\begin{aligned} &\operatorname{sign}(y_\alpha^-) = \operatorname{sign}[(J - (1-\cos^2\alpha)(\kappa^2 - \sin^2\varphi)I)\cos\alpha] \qquad (51)\\ &= \operatorname{sign}\{z^+(1-\sin^2\varphi)^{1/2}[(\kappa^2-\cos^2\alpha)(1-\sin^2\varphi) - (1-\cos^2\alpha)(\kappa^2-\sin^2\varphi)]\cos\alpha\} \\ &= \operatorname{sign}[(1-\kappa^2)\cos\alpha]. \end{aligned}$$

Next, we evaluate the sign of the difference $x_{y^-y^-}^- y_y^- - x_{y^+y^+}^+ y_y^+$. We compute $x_{y^+y^+}^+$ as follows

$$x_{y^+y^+}^+ = (x_{y^+}^+)_\sigma / y_\sigma^+ = \frac{\cos\alpha}{2(\kappa^2-\cos^2\alpha)\sqrt{\sigma-\cos^2\alpha}} \;/\; \frac{z^+(\kappa^2-\cos^2\alpha)}{2\sqrt{\sigma-\cos^2\alpha}(\kappa^2-\sigma)^{\frac{3}{2}}},$$

where we have differentiated (40) with respect to $\sigma = \sin^2\varphi$ and used (39). We compute $x_{y^-y^-}^-$ in the same way and obtain

$$x_{y^+y^+}^+ = \frac{\cos\alpha(\kappa^2-\sin^2\varphi)^{\frac{3}{2}}}{z^+(\kappa^2-\cos^2\alpha)^2} \quad \text{and} \quad x_{y^-y^-}^- = \frac{\cos\alpha(1-\sin^2\varphi)^{\frac{3}{2}}}{z^-(1-\cos^2\alpha)^2}. \qquad (52)$$

Furthermore, we have $y_y^+ = y_\sigma^+/y_\sigma = y_\sigma^+/(y_\sigma^+ + y_\sigma^-)$ and $y_\sigma^- = y_\sigma^+/(y_\sigma^+ + y_\sigma^-)$. So, we compute y_y^+ and y_y^- using (39) as follows

$$y_y^+ = \frac{z^+(\kappa^2-\cos^2\alpha)(1-\sin^2\varphi)^{\frac{3}{2}}}{J} \quad \text{and} \quad y_y^- = \frac{z^-(1-\cos^2\alpha)(\kappa^2-\sin^2\varphi)^{\frac{3}{2}}}{J}, \qquad (53)$$

where J was defined in (49). Using (52) and (53), we determine

$$\operatorname{sign}(x_{y^-y^-}^- y_y^- - x_{y^+y^+}^+ y_y^+) = \operatorname{sign}[(1/(1-\cos^2\alpha) - 1/(\kappa^2-\cos^2\alpha))\cos\alpha] = \operatorname{sign}[(\kappa^2-1)\cos\alpha].$$

The latter and (51) imply $A_1 < 0$.

Finally, we show that $A_2 = x_{y^-\alpha}^- y_y^- + x_{y^+\alpha}^+ y_y^+$ is negative too. We first compute the derivative $x_{y^+\alpha}^+$ by differentiating the expression (40) with respect to α. We have

$$x_{y^+\alpha}^+ = \frac{P_\kappa\sin\alpha + \cos\alpha\sin\varphi\cos\varphi(\kappa^2-\cos^2\alpha)\varphi_\alpha}{(\sin^2\varphi-\cos^2\alpha)^{\frac{1}{2}}(\kappa^2-\cos^2\alpha)^2}, \qquad (54)$$

where $P_\kappa = 2\cos^2\alpha(\kappa^2 - \sin^2\varphi) - \sin^2\varphi(\kappa^2 - \cos^2\alpha)$. We substitute φ_α using the expression (48) and multiply by y_y^+ using the expression (53). After simplification, we obtain

$$x^+_{y^+\alpha} y_y^+ = z^+ \sin\alpha(1-\sin^2\varphi)^{3/2} \frac{JP_\kappa - \cos^2\alpha(\kappa^2-\cos^2\alpha)(1-\sin^2\varphi)(\kappa^2-\sin^2\varphi)I}{J^2(\sin^2\varphi-\cos^2\alpha)^{\frac{1}{2}}(\kappa^2-\cos^2\alpha)}. \quad (55)$$

Analogously, we obtain the following

$$x^-_{y^-\alpha} y_y^- = z^- \sin\alpha(\kappa^2-\sin^2\varphi)^{3/2} \frac{JP_1 - \cos^2\alpha(1-\cos^2\alpha)(1-\sin^2\varphi)(\kappa^2-\sin^2\varphi)I}{J^2(\sin^2\varphi-\cos^2\alpha)^{\frac{1}{2}}(1-\cos^2\alpha)}, \quad (56)$$

where $P_1 = 2\cos^2\alpha(1-\sin^2\varphi) - \sin^2\varphi(1-\cos^2\alpha)$. We sum (55) and (56), simplify and omit the positive multiplicative terms obtaining

$$\begin{aligned} &\operatorname{sign}(x^-_{y^-\alpha} y_y^- + x^+_{y^+\alpha} y_y^+) = \quad (57)\\ &\operatorname{sign}[z^-(\kappa^2-\sin^2\varphi)^{3/2}(\kappa^2-\cos^2\alpha)Q_1 + z^+(1-\sin^2\varphi)^{3/2}(1-\cos^2\alpha)Q_\kappa],\\ &\text{where}\\ &Q_1 = JP_1 - \cos^2\alpha(1-\cos^2\alpha)(1-\sin^2\varphi)(\kappa^2-\sin^2\varphi)I \quad \text{and}\\ &Q_\kappa = JP_\kappa - \cos^2\alpha(\kappa^2-\cos^2\alpha)(1-\sin^2\varphi)(\kappa^2-\sin^2\varphi)I\ . \end{aligned}$$

We denote the expression in the square brackets by R. Finally, evaluate R and show that it is negative. First, we evaluate and simplify Q_κ and Q_1 . We substitute the expressions I and J from (49) in Q_κ and group the terms containing z^- and z^+. Then, we substitute the expression for P_κ from (54) and by simplification we get

$$\begin{aligned} Q_\kappa &= z^-(\kappa^2-\sin^2\varphi)^{3/2}[(1-\cos^2\alpha)P_k - \cos^2\alpha(\kappa^2-\cos^2\alpha)(1-\sin^2\varphi)] +\\ &\qquad z^+(1-\sin^2\varphi)^{3/2}[(\kappa^2-\cos^2\alpha)P_k - \cos^2\alpha(\kappa^2-\cos^2\alpha)(\kappa^2-\sin^2\varphi)]\\ &= z^-(\kappa^2-\sin^2\varphi)^{3/2}(\cos^2\alpha-\sin^2\varphi)(\kappa^2+\cos^2\alpha-2\kappa^2\cos^2\alpha) +\\ &\qquad z^+(1-\sin^2\varphi)^{3/2}(\cos^2\alpha-\sin^2\varphi)\kappa^2(\kappa^2-\cos^2\alpha). \end{aligned}$$

In the same way, we obtain the following representation for Q_1

$$\begin{aligned} Q_1 &= z^-(\kappa^2-\sin^2\varphi)^{3/2}(\cos^2\alpha-\sin^2\varphi)(1-\cos^2\alpha) +\\ &\qquad z^+(1-\sin^2\varphi)^{3/2}(\cos^2\alpha-\sin^2\varphi)(\kappa^2+\kappa^2\cos^2\alpha-2\cos^2\alpha). \end{aligned}$$

Substitution of Q_κ and Q_1 in (57) produces an expression for R of the form

$$\begin{aligned} R &= (z^-)^2R_1 + (z^+)^2R_2 + z^-z^+R_3, \quad \text{where}\\ R_1 &= (\kappa^2-\sin^2\varphi)^3(\kappa^2-\cos^2\alpha)(1-\cos^2\alpha)(\cos^2\alpha-\sin^2\varphi)\\ R_2 &= (1-\sin^2\varphi)^3(1-\cos^2\alpha)\kappa^2(\kappa^2-\cos^2\alpha)(\cos^2\alpha-\sin^2\varphi)\\ R_3 &= (1-\sin^2\varphi)^{3/2}(\kappa^2-\sin^2\varphi)^{3/2}(\cos^2\alpha-\sin^2\varphi)\ \times\\ &\qquad [(\kappa^2-\cos^2\alpha)(\kappa^2+\kappa^2\cos^2\alpha-2\cos^2\alpha) + (1-\cos^2\alpha)(\kappa^2+\cos^2\alpha-2\kappa^2\cos^2\alpha)]\\ &= (1-\sin^2\varphi)^{3/2}(\kappa^2-\sin^2\varphi)^{3/2}(\cos^2\alpha-\sin^2\varphi)\ \times\\ &\qquad [(\kappa^2-\cos^2\alpha)^2 + \kappa^2(1-\cos^2\alpha)^2 + (1-\kappa^2)^2\cos^2\alpha] \end{aligned}$$

From these expressions, it is clear that terms R_1, R_2, and R_3 are negative, since $\cos^2\alpha - \sin^2\varphi < 0$ and all other terms are positive. Thus, R and consequently A_2 are negative. The proposition is proved. □

A.2 Upper bound on the constant $C_{ABP}(t)$

Next, we show that

$$C_{ABP}(t) \le \frac{11|AB|}{r(e)\sin^2(\gamma/2)} \log_2 \frac{4|AB|^2 h}{r(e)r(A)r(B)}.$$

Recall that $\lambda = (1+\sqrt{\varepsilon/8}\sin(\gamma/2))^{-1}$ and $0 < \varepsilon \le 1$. We use the following two inequalities, which are easily derived from the properties of logarithms:
For any $X \ge 1$

$$\log_{\lambda^{-1}} X < \frac{3.44 \ln X}{\sqrt{\varepsilon}\sin(\gamma/2)}, \qquad \ln \frac{X}{\varepsilon} \le \ln \frac{2}{\varepsilon} \log_2 X \tag{58}$$

By our definition of the constant $C_{ABP}(t)$ and (17) it follows that

$$\begin{aligned} C_{ABP}(t) \le \frac{2\varepsilon|AB|}{r(e)\lambda(1-\lambda)\log\frac{2}{\varepsilon}} \log_{\lambda^{-1}} \frac{|AB|}{\varepsilon\lambda\sqrt{r(A)r(B)}} &+ \frac{\varepsilon|AB|}{r(e)(1-\lambda)^2\log\frac{2}{\varepsilon}} \\ + \frac{2\varepsilon^2}{\log\frac{2}{\varepsilon}} \log_{\lambda^{-1}} \frac{h}{\varepsilon r(e)} &+ \frac{4\varepsilon^2}{\log\frac{2}{\varepsilon}} \log_{\lambda^{-1}} \frac{|AB|}{\varepsilon\lambda\sqrt{r(A)r(B)}}. \end{aligned} \tag{59}$$

We estimate the terms on the right-hand side of this inequality using inequalities (58) above. For the first one, we have

$$\begin{aligned} \frac{2\varepsilon|AB|}{r(e)\lambda(1-\lambda)\log\frac{2}{\varepsilon}} \log_{\lambda^{-1}} \frac{|AB|}{\varepsilon\lambda\sqrt{r(A)r(B)}} &\le \frac{(2\sqrt{2}+1)^2\sqrt{\varepsilon}|AB|}{\sqrt{2}r(e)\sin(\gamma/2)\log\frac{2}{\varepsilon}} \log_{\lambda^{-1}} \frac{2|AB|}{\varepsilon\sqrt{r(A)r(B)}} \\ &\le \frac{3.44(2\sqrt{2}+1)^2|AB|}{\sqrt{2}r(e)\sin^2(\gamma/2)\log\frac{2}{\varepsilon}} \ln \frac{2|AB|}{\varepsilon\sqrt{r(A)r(B)}} < 25\frac{|AB|}{r(e)\sin^2(\gamma/2)} \log_2 \frac{2|AB|}{\sqrt{r(A)r(B)}}. \end{aligned} \tag{60}$$

For the second one, we have

$$\frac{\varepsilon|AB|}{r(e)(1-\lambda)^2\log\frac{2}{\varepsilon}} \le \frac{(2\sqrt{2}+1)^2|AB|}{r(e)\sin^2(\gamma/2)\log\frac{2}{\varepsilon}} < 15\frac{|AB|}{r(e)\sin^2(\gamma/2)\log\frac{2}{\varepsilon}}. \tag{61}$$

The sum of the third and fourth terms is estimated by

$$\begin{aligned} \frac{2\varepsilon^2}{\log\frac{2}{\varepsilon}} \left(\log_{\lambda^{-1}} \frac{h}{\varepsilon r(e)} + 2\log_{\lambda^{-1}} \frac{|AB|}{\varepsilon\lambda\sqrt{r(A)r(B)}} \right) &\le \frac{14\varepsilon^{1.5}}{\sin(\gamma/2)\log\frac{2}{\varepsilon}} \left(\ln\sqrt{\frac{h}{\varepsilon r(e)}} + \ln\frac{2|AB|}{\varepsilon\sqrt{r(A)r(B)}} \right) \\ \le \frac{14\varepsilon^{1.5}\ln 2}{\sin(\gamma/2)} \log_2 \frac{2|AB|\sqrt{h}}{\sqrt{r(e)r(A)r(B)}} &< 5\frac{\varepsilon^{1.5}}{\sin(\gamma/2)} \log_2 \frac{4|AB|^2 h}{r(e)r(A)r(B)}. \end{aligned} \tag{62}$$

We substitute (60), (61), and (62) in (59), use $2r(e) \le |AB|$, $r(e) \le h$ and obtain

$$C_{ABP}(t) < 23\frac{|AB|}{r(e)\sin^2(\gamma/2)} \log \frac{4|AB|^2 h}{r(e)r(A)r(B)}.$$

A.3 Proof of Lemma 5.1

Lemma 5.1 *The number of the maximal intervals covered by $A(u, \ell_1)$ is at most seven. The corresponding list $\bar{A}(u, \ell_1)$ is computed in $O(\log K(\ell_1))$, where $K(\ell_1)$ denotes the number of Steiner points on ℓ_1.*

Proof: First consider the case when the segments ℓ and ℓ_1 lie in different tetrahedra. We denote the weights of the tetrahedra containing ℓ and ℓ_1 by w^- and w^+, respectively. Let F be the plane defined by the face f and let F^- and F^+ be the two half-spaces defined by F, where we assume that ℓ is in F^- and ℓ_1 is in F^+. Furthermore, we assign weights w^- and w^+ to F^- and F^+, respectively, and we consider shortest weighted path $\bar{\pi}(u, y)$ between u, that is on ℓ, and an arbitrary point y on ℓ_1.

As discussed in Section 2, in this case, the path $\bar{\pi}(u, y)$ has the form $\bar{\pi}(u, y) = \{u, a(y), y\}$, where the point $a(y)$ lies in F and is uniquely defined by Snell's law (Figure 3 (a)). By our definition, there is an edge joining u to a Steiner point $u_1 \in \ell_1$ if and only if the point $a(u_1)$ lies in the interior of the triangle f. The interior can be represented as intersection of three half-planes defined by the lines containing the sides of f. So, we first obtain an upper bound on the number of maximal intervals covered by $A(u, \ell_1)$ in the case where f is a half-plane defined by an arbitrary line L in F.

There is one-to-one correspondence between end-points of the maximal intervals and the points y on ℓ_1 for which $a(y)$ lies on L. Hence, the number of maximal intervals covered by $A(u, \ell_1)$ can be estimated by counting the number of points y for which $a(y)$ lies on L. When the point y traverses the segment ℓ_1, the bending point $a(y)$ defines a curve in the plane F, which we denote by $\mathbf{a}(y)$. We consider Cartesian coordinate system $O_{\mu,\nu}$ in F, such that segment ℓ_1 projects onto the segment (μ_1, μ_2) of the μ-axis, and u projects onto ν-axis, say, at the point $u' = (0, \nu_0)$. We denote the μ-coordinate of the projection of y by $\mu(y)$ or simply by μ when no ambiguity arises. Then, as we discussed in Section 2, the curve $\mathbf{a}(y)$ has a representation $\mathbf{a}(\mu(y)) = (\tau\mu(y), (1-\tau)\nu_0)$, where τ is the unique solution of the equation (2).

Let $L(\mu, \nu)$ be the linear function, such that $L(\mu, \nu) = 0$ represents the line L in $O_{\mu,\nu}$. Then, a point $a(y)$ belongs to L if $L(\tau\mu(y), (1-\tau)\nu_0) = 0$. Thus, we obtain the following system of algebraic equations for τ and $\mu(y)$

$$\begin{cases} \frac{w^-\tau}{\sqrt{\tau^2(\mu^2(y)+\nu_0^2)+(z^-)^2}} = \frac{w^+(1-\tau)}{\sqrt{(1-\tau)^2(\mu^2(y)+\nu_0^2)+(z^+)^2}} \\ L(\tau\mu(y), (1-\tau)\nu_0) = 0, \end{cases} \tag{63}$$

where z^+ is the Euclidean distance from ℓ_1 to F and z^- is the Euclidean distance from u to F. Excluding τ from this system leads to a degree four algebraic equation for $\mu(y)$. Therefore, there may be no more than four intersections between $\mathbf{a}(y)$ and L and, hence, the number of the maximal intervals covered by $A(u, \ell_1)$ in the case where f is a half-plane is at most three.

In the case where f is a triangle, we denote by f_1, f_2, and f_3 the three half-planes defining f and by I_1, I_2, and I_3 the corresponding sets of maximal intervals. Then, any maximal interval $\triangle$ defined by f is obtained as an intersection $\triangle_1 \cap \triangle_2 \cap \triangle_3$, where $\triangle_i \in I_i$, $i = 1, 2, 3$. Each of the sets I_i contains at most three intervals and it is easily seen that the number of intervals that are intersections of the type $\triangle_1 \cap \triangle_2 \cap \triangle_3$ does not exceed seven.

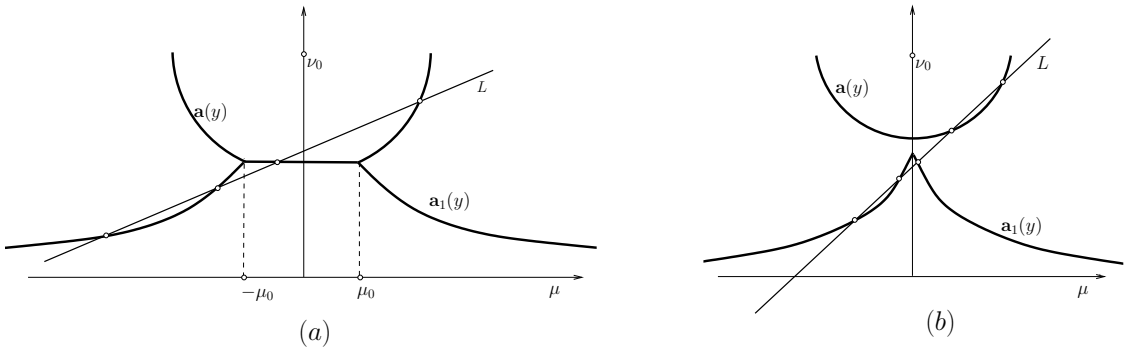

Figure 13: The figure illustrates the curves $\mathbf{a}(y)$ and $\mathbf{a}_1(y)$ in the case (a) when $\nu^- + \nu^+ > \nu_0$ and in the case (b) when $\nu^- + \nu^+ \leq \nu_0$. A line L and its intersections with the curves $\mathbf{a}(y)$ and $\mathbf{a}_1(y)$ are shown. The number of maximal intervals covered by $E(v, \ell_1)$ in the case when f is the lower half-plane defined by L, is 2 for both instances (a) and (b).

The list $\bar{A}(u, \ell_1)$ is computed by finding the position of the solutions of the systems (63) with respect to the elements of $V(\ell_1)$. This can be done by performing binary search and hence the computation of the list $\bar{A}(u, \ell_1)$ in this case takes $O(\log K(\ell_1))$ time.

Next, we consider the case where u and ℓ_1 lie in the same tetrahedron, say the one that is in F^-. If the weight w^- is smaller than w^+, then the path $\bar{\pi}(u, y)$ has the form $\{u, a(y), y\}$. The point $a(y)$ is the point in F that lies on the segment (u, y'), where y' is the point symmetric to y with respect to F. So, the curve $\mathbf{a}(y)$, in this case, is a segment. Hence, each of the sets I_i, for $i = 1, 2, 3$, in this case, is either empty or consists of a single interval. Consequently, there can be at most one interval obtained as intersection of intervals in these sets. Thus, in this case, there can be no more than one maximal interval covered by $A(u, \ell_1)$.

Finally, we consider the case where w^- is greater than w^+. In this case, the shortest path $\bar{\pi}(u, y)$ has the form $\{u, a(y), a_1(y), y\}$, where the segment $(a(y), a_1(y))$ lies in F. We discussed the structure of this path in Section 2 and illustrated it in (Figure 3 (b)). The curves $\mathbf{a}(y)$ and $\mathbf{a}_1(y)$ have explicit representations as we detail below. We set $\nu^- = z^- \tan \varphi^*$ and $\nu^+ = z^+ \tan \varphi^*$, where the critical angle φ^* is defined by $\sin \varphi^* = w^+/w^-$ and z^-, z^+ are the distances from v and ℓ_1 to F, respectively.

In this notation the curves $a(y)$ and $a_1(y)$ have the following representations in $O_{\mu,\nu}$

$$\mathbf{a}(y) = \begin{cases} \nu = \frac{z^+ \nu_0}{z^- + z^+} & \text{for} \quad |\mu| < \mu_0 \\ \nu = \nu_0 - \sqrt{(\nu^-)^2 - \mu^2} & \text{for} \quad \mu_0 \geq |\mu| \leq \nu^-, \end{cases} \tag{64}$$

$$\mathbf{a}_1(y) = \begin{cases} \nu = \frac{z^+ \nu_0}{z^- + z^+} & \text{for} \quad |\mu| < \mu_0 \\ (\nu_0 - \nu)\sqrt{(\nu^+)^2 - \nu^2} = \mu\nu & \text{for} \quad |\mu| \geq \mu_0, \end{cases}, \tag{65}$$

where $\mu_0 = \frac{z^-}{z^- + z^+}\sqrt{(\nu^- + \nu^+)^2 - \nu_0^2}$ if $\nu^- + \nu^- > \nu_0$ and $\mu_0 = 0$ if $\nu^- + \nu^- \leq \nu_0$.

In the case where $\nu^- + \nu^- > \nu_0$ (Figure 13 (b)), the curve $\mathbf{a}(y)$ is a half-circle centered at the point $(0, \nu_0)$ with radius ν^-. The curve $\mathbf{a}_1(y)$ is symmetric with respect to the ν-axis. The part of $\mathbf{a}_1(y)$ to the right of O_ν is monotonically decreasing. It is convex and approaches

the μ-axis at infinity. In the case where $\nu^- + \nu^- \leq \nu_0$ (Figure 13 (a)), the curves $\mathbf{a}(y)$ and $\mathbf{a}_1(y)$ have a common part – a horizontal segment that projects at $(-\mu_0, \mu_0)$ on the μ-axis. For $|\mu| > \mu_0$, the curves are the same as in the case $\nu^- + \nu^- > \nu_0$.

Again, we consider, first, the case when f is a half-plane defined by a line L in F. There is an edge between u and a point y on ℓ_1 if and only if the segment $(a(y), a_1(y))$ lies entirely inside f, i.e. in one of the half-planes defined by L. By a simple case analysis, we determine that in the case where f is a half-plane, the number of maximal intervals covered by $A(u, \ell_1)$ is at most 2.

Hence, in the general case when f is a triangle, each of the sets I_1, I_2, and I_3, as defined above, contains at most 2 intervals. The number of intervals that can be intersections of the type $\cap_{i=1}^{3} \triangle_i$ with $\triangle_i \in I_i$ is at most 4. The latter proves that the number of maximal intervals covered by $A(u, \ell_1)$ in the case where ℓ and ℓ_1 are in the same tetrahedron, whose weight w^- is bigger than the weight w^+ of the neighboring tetrahedron, is at most 4.

Computation of the list $\bar{A}(u, \ell_1)$ in this case is done again by binary search and takes $O(K(\ell_1))$ time. The lemma is proved. □