Approximate Shortest Path Queries on Weighted Polyhedral Surfaces *

Lyudmil Aleksandrov Hristo N. Djidjev Hua Guo Anil Maheshwari
Doron Nussbaum Jörg-Rüdiger Sack

April 3, 2006

Abstract

We consider the classical geometric problem of determining shortest paths between pairs of points lying on a weighted polyhedral surface P consisting of n triangular faces. We present query algorithms that compute approximate distances and/or approximate (weighted) shortest paths. Our algorithm takes as input an approximation parameter $\varepsilon \in (0, 1)$ and a query time parameter $\mathfrak{q}$ and builds a data structure $\mathrm{APQ}(P, \varepsilon; \mathfrak{q})$ which is then used for answering ϵ-approximate distance queries in $O(\mathfrak{q})$ time. This algorithm is source point independent and improves significantly on the best previous solution.

For the case where one of the query points is fixed we build a data structure $\mathrm{SSQ}(P, \varepsilon; a)$ that can answer ϵ-approximate distance queries from a to any query point in P in $O(\log \frac{1}{\varepsilon})$ time. This is an improvement upon the previously known solution for the Euclidean fixed source query problem. Our algorithm also generalizes the setting from previously studied unweighted polyhedra to weighted polyhedral surfaces of arbitrary genus.

Our shortest path algorithms are based on a novel separator algorithm which we introduce here and which extends and generalizes previously known separator algorithms.

1 Introduction

Motivation and problem definition Shortest path problems rank among the fundamental problems studied in computational geometry, network optimization, graph algorithms, geographical information systems (GIS), and calculus of variations. These problems arise naturally in various applications such as motion/route planning, navigation, graphics, injection molding, and computer-assisted surgery (for references see [6]). Aside from the importance of shortest paths problems in their own right, often they appear in the solutions to other problems.

Finding shortest paths with respect to the Euclidean distance sometimes provides inadequate solutions in practice. In GIS, for example, a terrain could consist of different types of regions (e.g., water, forest, rocks) which is modeled by assigning suitable weights to the regions. This leads to the weighted shortest path problem. We consider paths that stay on the surface of a connected

*Research supported by NSERC, SUN Microsystems, Stantive Computing, and a P.E.O. Scholar Award. Research was carried out in part while the first and second author were visiting Carleton University (as Adjunct Professors). The first author is from the Bulgarian Academy of Sciences, Sofia, Bulgaria. The second is from Los Alamos National Laboratory, USA. The remaining authors are from Carleton University, Ottawa, Canada.

polyhedral surface[1] P of genus g in the 3-dimensional Euclidean space consisting of n positively weighted triangular faces. The cost of a path lying inside a face is its Euclidean length multiplied by the weight of the face. The cost of a general path on P is the sum of the costs of the sub-paths within each face traversed. For a pair of points on P the path of least cost between them is called *shortest path*. The cost of the shortest path is called *distance* between its end-points.

Frequently, in applications like GIS, shortest paths queries are executed over time for a fixed domain. Efficient query algorithms are not only desirable, but often required, to provide timely answers to shortest path queries due to the relatively high time complexities for shortest path computations, in particular in weighted domains. This motivates our search for algorithms for answering approximate shortest path queries.

Throughout the paper ε is a user-specified accuracy parameter, i.e., a fixed real number in $(0, 1)$. A path whose cost divided by the cost of the shortest path is in $(1-\varepsilon, 1+\varepsilon)$ is called *ε-approximate (or simply approximate) shortest path*. The cost of an approximate path is called *approximate distance*. The approximate distance (and/or shortest path) query problem is: Preprocess the surface P so that for a pair of query points a and b approximate distance and/or an approximate shortest path between a and b can be answered efficiently. We consider the following standard two variations of these problems: The *Single Source Query* (SSQ) problem in which one of the query points is fixed and the other one is any point on P and the *All Pairs Query* (APQ) problem in which the query consists of two arbitrary points in P. To place our work in the context of the literature we next state some relevant results. (We assume that the faces containing the query points are already known.) **Previous and new results on geometric shortest path queries** Much work has been carried out to solve shortest path problems in planar graphs (see e.g., the survey [17]). Here we are interested in the geometric setting, where complexities tend to be much higher than for planar graphs in particular, for geometric weighted shortest paths computations (see [6] for references). If P is a convex surface, then a result of Dudley [11] shows that a convex set Q of size $O(\frac{1}{\varepsilon^{3/2}})$ exists, such that $P \subset Q$ and the Hausdorff distance between P and Q is $\varepsilon \cdot diameter(P)$. This was used in the algorithm of [2] to compute an ε-approximate shortest path in $O(n \log \frac{1}{\varepsilon} + \frac{1}{\varepsilon^3})$ time. Later, in [14], this algorithm was extended to answer APQs in $O(\log n/\varepsilon^{1.5} + 1/\varepsilon^3)$ time. Chazelle et al. [7] very recently presented a sublinear randomized algorithm for solving APQs on convex polyhedral surfaces. A preliminary result on APQ for weighted polyhedra has been announced in [5]. We summarize other results relevant to this paper in Table 1.

Our approach At the core of our approach is the discretization method and the Single Source Shortest Path (SSSP) algorithm developed in [6]. The discretization method constructs a graph G_ε, called approximation graph, by inserting a set of Steiner points inside the faces of P. The edges of G_ε connect nodes incident to neighbor faces and have cost equal to the cost of the shortest "local" paths between their endpoints. Next, a highly efficient SSSP algorithm is employed for finding approximate distances from a fixed vertex of G_ε to all other vertices G_ε. Here we use a modification of the graph G_ε, which has some additional edges and the edge cost is defined in slightly different way. Our algorithm for solving the SSQ problem builds a data structure $\mathrm{SSQ}(P, \varepsilon; a)$ consisting of a SSSP tree in G_ε rooted at the point a plus $O(n)$ local data structures related to the faces of P. Our algorithm for solving the APQ problem builds a data structure $\mathrm{APQ}(P, \varepsilon; \mathfrak{q})$ consisting of a collection of SSQ data structures. The choice of the SSQ data structures inserted into $\mathrm{APQ}(P, \varepsilon; \mathfrak{q})$ depends on a balanced decomposition of the surface P in terms of the number of nodes of G_ε incident to different parts.

[1] Surface P can be any polyhedral 2-manifold without assumed additional geometrical/ topological properties like convexity, being a terrain, absence of holes, etc.

Ref.	Preprocessing time	Query time	Space	Source	Convex	Weighted
[1]	$O(n^6 m^{1+\delta})$	$O((\sqrt{n}/m^{1/4}) \log n)$	$O(n^6 m^{1+\delta})$	APQ	Yes	No
[9]	$O(n^{11})$	$O(\log n)$	$O(n^{12})$	APQ	No	No
[8]	$O(n^2)$	$O(d \log n / \log d)$	$O(n \log n / \log d)$	SSQ	No	No
[15]	$O\left(n^2 \log n + \frac{n}{\epsilon} \log \frac{1}{\epsilon} \log \frac{n}{\epsilon}\right)$	$O\left(\log \frac{n}{\epsilon}\right)$	$O\left(\frac{n}{\epsilon} \log \frac{1}{\epsilon}\right)$	SSQ	No	No
Here	$\mathbf{O\left(\frac{n}{\sqrt{\epsilon}} \log \frac{1}{\epsilon} \log \frac{n}{\epsilon}\right)}$	$\mathbf{O\left(\log \frac{1}{\epsilon}\right)}$	$\mathbf{O\left(\frac{n}{\sqrt{\epsilon}} \log \frac{1}{\epsilon}\right)}$	**SSQ**	**No**	**Yes**
Here	$\mathbf{O\left(\frac{(g+1)n^2}{\varepsilon^{3/2}\mathfrak{q}} \log \frac{n}{\epsilon} \log^4 \frac{1}{\epsilon}\right)}$	$\mathbf{O(\mathfrak{q})}$	$\mathbf{O\left(\frac{(g+1)n^2}{\epsilon^{3/2}\mathfrak{q}} \log^4 \frac{1}{\epsilon}\right)}$	**APQ**	**No**	**Yes**

Table 1: We assume for our algorithms that the face containing the query point is known. The time complexity of our preprocessing algorithms inherits the dependency on geometric parameters from [6]. In [8] d is an adjustable parameter with $1 < d \leq n$.

Our contributions here are:

1. We derive an algorithm for solving the approximate SSQ problem in weighted polyhedral surfaces of arbitrary genus. See the corresponding entry in Table 1. This result improves upon the previous result in [15] in three ways. First, it works for weighted surfaces of arbitrary genus. Second, the preprocessing time is reduced by a factor of n. Third, the size of the data structure and the preprocessing time are reduced by at least a factor of $\sqrt{\varepsilon}$.

2. We present a novel algorithm for solving the approximate APQ problem in weighted polyhedral surfaces P of arbitrary genus g. See the corresponding entry in Table 1. The algorithm takes as input a query time parameter $\mathfrak{q}$ within a certain range and builds a data structure APQ$(P, \varepsilon; \mathfrak{q})$ to answer approximate distance and/or shortest path queries between arbitrary points in P in $O(\mathfrak{q})$ time.

Previous and new results on separators A number of problems in algorithmic graph theory, computational geometry, parallel computing and algorithms design have been solved by applying separator algorithms to an underlying graph. As an illustration, separators were used for: efficient message routing, design of D&Q algorithms in computational geometry, and nested dissection (Gaussian elimination). Separator results which are most relevant to this paper are summarized in Table 2.

Our contribution here is:

We designed a new partitioning algorithm (see the corresponding entry in Table 2). The algorithm partitions embedded graphs of genus g with weights and costs assigned to their vertices into components of specified weight so that their boundaries have small cost. This result extends and/or improves upon results in [3, 4, 10, 12, 13, 16]. For the design of the APQ algorithm we employed this new partitioning algorithm. Our opinion is that the algorithm may find applications for solving other problems, in particular on weighted surfaces, and in other models of computation, e.g., in parallel computing.

Organization of the paper

The remainder of this paper is organized as follows. In Section 2 we present the two novel approaches to partitioning of weighted graphs. In Section 3 we first state our approximation graph and then, in Section 4 apply it for the construction of a data structure which allows us to answer single source shortest path queries on weighted surfaces of arbitrary genus. In Section 5 then we present our solution to APQ on these surfaces. Finally, in Section 6 we conclude with a discussion of extensions and open problems. (We include a detailed proof as Appendix 7 for interested readers.)

Ref.	graph G	$c(v)$	time complexity	cost of separator S	cost of boundaries ∂R
[16]	planar	1	$O(n)$	$\|S\| = O(\sqrt{n/t})$	n/a
[10]	planar	1	$O(n)$	$\|S\| = O(\sqrt{n/t})$	n/a
[12]	planar	1	$O(n \log n)$	$\|S\| = O(\sqrt{n/t})$	$\|\partial R\| = O(\sqrt{t})$
[13]	genus-g	1	$O((n+g)\log n)$	$\|S\| = O(\sqrt{gn/t})$	n/a
[3]	genus-g	1	$O(n+g)$	$\|S\| \leq 4\sqrt{(g+1/t)n}$	n/a
[4]	planar	≥ 0	$O(n)$	$c(S) \leq 4\sqrt{2\sigma(G)/t}$	n/a
Here	**genus-g**	$\geq \mathbf{0}$	$\mathbf{O(g + n \log n)}$	$\mathbf{O(\sqrt{(g+1)\sigma(G)/t})}$	$\mathbf{O(\sqrt{(g+1)t\sigma(G)})}$

Table 2: Comparison of t-separator (see Section 2 for relevant definitions) theorems for an n-vertex graph G with non-negative vertex weights summing up to 1 and non-negative vertex cost function $c(\cdot)$, where $t \in (0,1)$. We let ∂R denote the boundary of the biggest/costliest region of the partition and $\sigma(G) = \sum_{v \in V(G)} (c(v))^2$.

2 Partitioning embedded graphs with weights and costs

In this section we present two new results on partitioning of embedded graphs with weights and costs assigned to vertices. We consider connected graphs $G = (V, E)$ that are 2-cell embedded onto an orientable surface of genus g where positive weights and costs are assigned to the vertices of G. For a subgraph G' of G, we denote by $w(G')$ and $c(G')$ the sum of the weights and sum of the costs of the vertices in G', respectively. Let t be a real number in $(0, 1)$. A set of vertices S of G is called a *t-separator* if its removal from G leaves no component of weight exceeding $tw(G)$. We denote the sum of the squares of the costs of the vertices of G by $\sigma(G)$, i.e. $\sigma(G) = \sum_{v \in V} (c(v))^2$. The lemma below follows directly from the method presented in [3] and using this we show how "low-cost" t-separators can be constructed.

Lemma 1 *Let K be an embedded and triangulated graph of genus γ with non-negative weights on its vertices and let T be a spanning tree of K. There exists a t-separator C of G that satisfies the following:* **(a)** *The separator C consists of at most $4(\gamma + 1/t)$ fundamental cycles*[2]*.* **(b)** *Any of the components of $K \setminus C$ can be adjacent to at most $2(\gamma + 1)$ cycles in C. Such a separator C can be constructed in $O(|K| \log |K|)$ time.*

Theorem 1 *Let G be an embedded graph of genus g with weights and costs assigned to its vertices. For any $t \in (0, 1)$ there exists a t-separator S whose cost is at most $4\sqrt{2(g + 1/t)\sigma(G)}$. Such a separator can be constructed in $O(|G| \log |G|)$ time.*

Proof: (Sketch) The theorem is proved by constructing a t-separator S whose cost is as required. S is constructed in two phases. In the first phase we "slice" the graph into subgraphs with "short" (in terms of cost) spanning trees using a single source shortest path (SSSP) tree T rooted at a vertex ρ. For any real $x \geq 0$ we define a set of vertices $L(x)$ called *level* as follows. A vertex v is in $L(x)$ if its distance to ρ is at least x and the distance of its predecessor in T to ρ is less than x. Let $h = \frac{1}{2\sqrt{2}} \sqrt{\frac{\sigma(G)}{(g+1/t)}}$. We apply the method described in [4] and compute a set of levels $\mathcal{L}_h$, so that their removal partitions G into components with SSSP trees of radius not exceeding $2h$. The

[2] A fundamental cycle is a cycle consisting of a single non-tree edge (v_1, v_2) plus the two paths in T from v_1 and v_2 to their lowest common ancestor.

total cost of the vertices in the set of levels $\mathcal{L}_h$ does not exceed $\sigma(G)/h$. The vertices in the levels $\mathcal{L}_h$ are inserted into S.

In the second phase, we use Lemma 1 to obtain a t-separator. Each "heavy" component K, i.e. $w(K) > tw(G)$, of the graph $G \setminus \mathcal{L}_h$ is further partitioned by fundamental cycles as stated in Lemma 1 with a parameter $t_K = tw(G)/w(K)$. The resulting separator $S(K)$ is inserted in S. By the construction in Phase I and by Lemma 1 the cost of the separator $S(K)$ is bounded by $c(S(K)) \leq 8(\gamma(K) + 1/t_K)h$, where $\gamma(K)$ is the genus of the subsurface containing K. From this inequality the stated bounds follow. □

Next, we extend the approach applied in the above construction to obtain "low-cost" t-separators, which partition the graph into components with "small-cost" boundaries. Any t-separator S naturally defines a partitioning of the vertices of G into sets inducing the connected components of $G \setminus S$ and S itself. Let $V_1, \ldots, V_k, S$ be the partitioning defined by a t-separator S. Note that a vertex in a set V_i for some $1 \leq i \leq k$ can be adjacent to vertices in $V_i \cup S$ only. The subset of vertices in S that are adjacent to vertices in V_i is called *boundary* of V_i (or of the component induced by V_i) and is denoted by ∂V_i. A partitioning $V_1, \ldots, V_k, S$ defined by S is called *B-regular* (or simply *regular*), where B is a real number, if the costs $c(\partial V_i)$ for $i = 1, \ldots, k$ are bounded by B.

Theorem 2 *Let G be an embedded graph of genus g with maximum degree three and with weights and costs assigned to its vertices. For any $t \in (0, 1)$ there exists a t-separator S, that defines a $2B$-regular partitioning of G with $B = \sqrt{(g+1)t\sigma(G)}$, whose cost is $O\left(\sqrt{(g+1)\sigma(G)/t}\right)$. Such a separator can be constructed in $O(|G| \log |G|)$ time.*

Proof: (Sketch) We set $6h = B/(g+1) = \sqrt{t\sigma(G)/(g+1)}$ and apply Phase I as described in the proof of Theorem 1 above. Thus we compute a set of levels $\mathcal{L}_h$, whose removal partitions the graph into components, whose spanning trees have radii (in terms of cost) bounded by $2h$ and the cost of vertices in $\mathcal{L}_h$ is $c(\mathcal{L}_h) \leq \sigma(G)/h$. Then we apply Phase II and obtain a set of fundamental cycles C_1, such that the set of vertices in $S_1 = \mathcal{L}_h \cup C_1$ is a t-separator for G and their cost is $c(C_1) \leq 8h(g + 1/t)$; but they may not induce $2B$-regular partitioning since there might be components in $G \setminus S_1$ with boundaries whose cost exceeds $2B$.

Let K be a component of $G \setminus S_1$, such that $c(\partial K) > 2B$. We consider the subgraph of G induced by the set of vertices $V(K) \cup \partial K$ and denote it by $\tilde{K}$. Let the genus of the subsurface containing $\tilde{K}$ be $\gamma(K)$. We assign new weights $w_1(v)$ and costs $c_1(v)$ to the vertices of $\tilde{K}$ as follows. We denote by $\partial' K$ the set of vertices in ∂K that belong to $\mathcal{L}_h$, i.e. $\partial' K = \partial K \cap \mathcal{L}_h$. The new cost of the vertices in ∂K is set to zero and the new cost of vertices in K equals to its original cost, i.e. for $v \in K$ we have $c_1(v) = c(v)$ and for $v \in \partial K$, $c_1(v) = 0$. The new weight of a vertex v in K is the sum of the costs of the vertices in $\partial' K$ that are adjacent to v. The weights of the vertices in ∂K and the vertices in K not adjacent to $\partial' K$ are set to zero. By this definition and since the maximum degree of G is three we have $w_1(\tilde{K}) \leq 3c(\partial' K)$. Then we set $t_K = (2/3)B/w_1(\tilde{K})$ and compute a t_K-separator of $\tilde{K}$ using Lemma 1. We denote this separator by $C(K)$. It can be shown that the cost of this separator is $c(C(K)) \leq 8h\gamma(K) + 6c(\partial K \cap \mathcal{L}_h)/(g+1)$. Now for any component K_1 of $\tilde{K} \setminus C(K)$ we have $c(\partial K_1) \leq 2B$.

We define separator S to be the union of S_1 and the separators $C(K)$ computed for all components K with $c(\partial K) > 2B$. Clearly, S is a t-separator that induces a $2B$-regular partitioning of G. The cost of S can be estimated as $c(S) \leq c(S_1) + \sum_{c(\partial K)>2B} c(C(K)) \leq \sigma(G)/h + 8h(g + 1/t) + \sum_{c(\partial K)>2B} 8h(\gamma(K) + 9c(\partial K \cap \mathcal{L}_h))/2B) \leq \sigma(G)/h + 8h(2g+1)/t + 6c(\mathcal{L}_h)/(g+1) < 45\sqrt{(g+1)\sigma(G)/t}$. □

3 Approximating shortest paths

First we adapt the discretization scheme presented in [6] and establish properties which are beneficial in answering shortest path queries as we will show. Let P be a polyhedral surface in the 3-dimensional Euclidean space consisting of n triangular faces $f_1, \ldots, f_n$. Each face f_i has an associated positive weight w_i, representing the cost of traveling a unit Euclidean distance inside it. The weight of an edge is the minimum of the weights of the triangles incident to that edge. The cost of a path π in P is defined as $\|\pi\| = \sum_{i=1}^{n} w_i|\pi_i|$, where $|\pi_i|$ denotes the Euclidean length of the portion of π in f_i. Given two points a and b in P a path of minimum cost joining a and b is called *shortest path* between a and b and is denoted by $a \overset{P}{\rightsquigarrow} b$. The cost of this path is referred to as *distance* between a and b and is denoted by $\text{dist}_P(a, b)$.

Let ε be a real number in $(0, 1)$. As part of our algorithms we will be constructing an *approximation graph* $G_\varepsilon = (V_\varepsilon, E_\varepsilon)$, which is a supergraph of the corresponding graph in [6], whose nodes correspond to geometric objects, namely, Steiner points and vertex vicinities of "small" radius in P. Around each vertex v of P we define a "small" star-shaped polygon $\mathcal{E}(v)$ called *vertex vicinity*; it is contained within the union of the triangles incident to v and its intersections with each of the triangles is a "small" isosceles triangle with side length $\varepsilon r(v)$, where $r(v)$ is a fraction of the distance from v to the boundary of the union of the triangles incident to v. We set $r(v)$ to be (1/8)th of this distance (see, Definition 2.1 in [6]). The nodes of G_ε are of two types depending on the object in P they represent. For each vertex of P and its vertex vicinity we define a node representing them in G_ε and call them vertex vicinity nodes. Steiner point nodes represent Steiner points inserted in P. Steiner points are placed along the bisectors of the angles of the faces of P forming a geometric progressions with ratios depending on ε and on the geometry of P as detailed in [6]. The approximation properties of Steiner points are stated in the following lemma.

Lemma 2 *[6] Let x and y be points lying on two different edges of a face f of P and outside vertex vicinities. There is a Steiner point p in f such that $|xp| + |py| \leq (1 + \varepsilon/2)|xy|$.*

Lemma 3 *[6]* **(a)** *The number of nodes in G_ε incident to a triangle f of P is bounded by $C(f)\frac{1}{\sqrt{\varepsilon}} \log \frac{2}{\varepsilon}$, where the constant $C(f)$ depends on the geometry[3] of the triangle f.* **(b)** *The total number of nodes of G_ε is less than $C(P)\frac{n}{\sqrt{\varepsilon}} \log \frac{2}{\varepsilon}$, where the constant $C(P) = \frac{1}{n}\sum_{f \in P} C(f)$.*

To define the edges of G_ε we introduce the notion of *face neighborhood* for points in P. The face neighborhood of a vertex of P is the union of the triangles incident to that vertex. The face neighborhood of a point in a face f of P consists of the union of f and its *neighboring* faces. The face neighborhood of a point a is denoted by $\mathcal{N}(a)$. A node p of G_ε is connected to all nodes, whose representations are incident with its face neighborhood $\mathcal{N}(p)$. To define costs of the edges of G_ε we use the notion of *local paths*. A path in P is called *local* if it intersects at most two faces. The cost $c(p, q)$ of an edge (p, q) in G_ε is defined as the cost of the *local shortest path* restricted to lie in the intersection of their face neighborhoods $\mathcal{N}(p) \cap \mathcal{N}(q)$.

3.1 Approximation properties of G_ε

The paths in the approximation graph G_ε are called *discrete paths*. The cost $c(\pi_G(p, q))$ of a discrete path $\pi_G(p, q)$ is the sum of the costs of its edges. For a pair of nodes p and q of G_ε, $p \overset{G}{\rightsquigarrow} q$ denotes

[3] Roughly it is about two times the sum of the reciprocals of the sinuses of the angles of f. See [6] for precise estimates.

a shortest path in G_ε between p and q. As is shown below, in general, the cost $c(p \overset{G}{\rightsquigarrow} q)$ of a shortest discrete path is an ε-approximation of the distance $\|\tilde{p} \overset{P}{\rightsquigarrow} \tilde{q}\|$, where $\tilde{p}$ and $\tilde{q}$ are points in P incident to the objects represented by p and q. A discrete path $\pi_G(p,q)$ between nodes p and q can be naturally embedded in P as follows. First, each node on that path is embedded into the object it represents, i.e. either a Steiner point or a vertex vicinity. Then each edge of $\pi_G(p,q)$ is embedded into the local shortest path between the objects representing its end-nodes. As a result we obtain a sequence of vertex vicinities joined by polygonal paths in P. Finally, we replace each vertex vicinity in $\pi_G(p,q)$ with a two segment path through the corresponding vertex of P. We refer to this embedding of a discrete path $\pi_G(p,q)$ into P as *natural embedding* and denote it by $\tilde{\pi}_G(p,q)$. By our definitions the cost in P of the natural embedding $\tilde{\pi}_G(p,q)$ of a discrete path minus the cost of its portions inside vertex vicinities equals to the cost of $\pi_G(p,q)$ in G_ε.

Theorem 3 *(follows from Theorem 4.5 in [6]) The SSSP problem in the approximation graph G_ε can be solved in $O(|V_\varepsilon| \log |V_\varepsilon|) = O(\frac{n}{\sqrt{\varepsilon}} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$ time.*

Next, we discuss how the approximation graph G_ε can be used to approximate distances and shortest paths in P. Let a and b be arbitrary points in P. If a and b lie in neighboring triangles and the shortest path $a \overset{P}{\rightsquigarrow} b$ between them is a local path (i.e. stays inside the quadrilateral formed by the union of their triangles) than we can report the exact path in constant time. So, we concentrate on the approximation of shortest paths that cross more than two faces.

Naturally, we consider paths of the form $\{a \overset{P}{\rightsquigarrow} p \overset{G}{\rightsquigarrow} q \overset{P}{\rightsquigarrow} b\}$ and then approximate $\mathrm{dist}_P(a,b)$ by the minimum of $\|a \overset{P}{\rightsquigarrow} p\| + c(p \overset{G}{\rightsquigarrow} q) + \|q \overset{P}{\rightsquigarrow} b\|$ taken over all choices of nodes p and q in G_ε. As it is shown below, we can obtain the desired approximation by taking the minimum not over all pairs of nodes in G_ε, but only over pairs p and q such that $p \in \mathcal{N}(a)$ and $q \in \mathcal{N}(b)$. Moreover we show that, it suffices to compute the local shortest paths between a and p and between b and q. We denote these local shortest paths by $a \overset{\mathcal{N}(a)}{\rightsquigarrow} p$ and $q \overset{\mathcal{N}(b)}{\rightsquigarrow} b$ and define *approximate discrete paths* between pairs of points in P as follows.

Definition 1 *A path between a pair of points a and b in P is called* approximate discrete path *if it is either a shortest local path joining a and b or a path of the form*

$$\{a \overset{\mathcal{N}(a)}{\rightsquigarrow} p \overset{G}{\rightsquigarrow} q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\}, \tag{1}$$

where $p \in \mathcal{N}(a)$, $q \in \mathcal{N}(b)$. The cost of an approximate discrete path is $\|a \overset{\mathcal{N}(a)}{\rightsquigarrow} p\| + c(p \overset{G}{\rightsquigarrow} q) + \|q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\|$ or its cost in P if it is a local path. The cost of a shortest approximate discrete path between a and b is called approximate distance *between a and b and is denoted by* $\mathrm{dist}_G(a,b)$.

Note that by this definition the approximate distance $\mathrm{dist}_G(a,b)$ between points a and b lying in neighbor triangles is the minimum of the cost of the shortest local path between a and b and the cost of any path of the form (1). The next theorem establishes the relation between approximate distances and the distances in P (proof is presented in the Appendix).

Theorem 4 *For any pair of points a and b in P one of the following two holds: either* **(a)** $(1-2\varepsilon)\mathrm{dist}_P(a,b) \le \mathrm{dist}_G(a,b) \le (1+2\varepsilon)\mathrm{dist}_P(a,b)$, *or* **(b)** $\mathrm{dist}_P(a,b) - 2\varepsilon r(v) \le \mathrm{dist}_G(a,b) \le \mathrm{dist}_P(a,b)$, *where $\varepsilon r(v)$ is the radius of $\mathcal{E}(v)$.*

If case (a) of the theorem applies then the approximate distance $\mathrm{dist}_G(a,b)$ is an approximation of the distance $\mathrm{dist}_P(a,b)$ with relative error $\frac{|\mathrm{dist}_G(a,b)-\mathrm{dist}_P(a,b)|}{\mathrm{dist}_P(a,b)}$ bounded by 2ε. Case (b) of the theorem can be viewed as an exception covering a special situation where points a and b are "close" to each other and "near" a vertex v of P, meaning that any shortest path in P between them has to intersect the vertex vicinity $\mathcal{E}(v)$ and must stay in the face neighborhood $\mathcal{N}(v)$. Furthermore, in case (b) $\mathrm{dist}_G(a,b)$ is less than $\mathrm{dist}_P(a,b)$ and it is an approximation with relative error not exceeding $\varepsilon r(v)/\mathrm{dist}_G(a,b)$. Therefore, if $r(v) \leq 2\mathrm{dist}_G(a,b)$ the quality of the approximation is 2ε, same as in case (a). If the ratio $r(v)/\mathrm{dist}_G(a,b)$ is larger than $1/\varepsilon$ then the relative error could be as big as 1. For example, if points a and b are inside the vertex vicinity $\mathcal{E}(v)$ then $\mathrm{dist}_G(a,b)$ is zero and the relative error is 1. Note, that the conditions for the occurrence of case (b) and the presence of eventually large (compare to ε) relative error are easily detected by the position of the points a and b, the structure of the approximate discrete path and the ratio $r(v)/\mathrm{dist}_G(a,b)$. Thus if the approximation is not satisfactory the exact shortest path restricted to lie inside $\mathcal{N}(v)$ can be computed. The above discussion is summarized in the next corollary.

Corollary 1 *The distance* $\mathrm{dist}_G(a,b)$ *approximates* $\mathrm{dist}_P(a,b)$ *with relative error* 2ε*, except possibly when the case (b) of Theorem 4 applies and* $r(v) > 2\mathrm{dist}_G(a,b)$*. In the latter case* $\mathrm{dist}_P(a,b)$ *can be computed directly.*

We conclude this section by a remark on the computation of approximate distances. The distance $\mathrm{dist}_G(a,b)$ between a pair of points a and b can be computed as follows. We compute $\min(\|a \overset{\mathcal{N}(a)}{\rightsquigarrow} p\| + c(p \overset{G}{\rightsquigarrow} q) + \|q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\|)$, over all pairs of nodes $p \in \mathcal{N}(a)$ and $q \in \mathcal{N}(b)$. In the case where a and b do not lie in neighbor faces this minimum is the approximate distance $\mathrm{dist}_G(a,b)$. In the case where the points a and b lie in neighbor faces we also need to consider the shortest local path between them.

4 Fixed source shortest path queries

In this section we describe an algorithm, that takes as input a point a in P, called *source*, and an approximation parameter $\varepsilon \in (0,1)$ and constructs a data structure, called *Single Source Queries* (SSQ), such that for any query point $b \in P$, called *target*, the approximate distance $\mathrm{dist}_G(a,b)$ (and/or an approximate shortest path) from a to b is computed efficiently. The algorithm uses the approximation graph G_ε. For simplicity, we assume that the point a corresponds to a node in G_ε; otherwise, we can easily augment G_ε with extra edges corresponding to local shortest paths from a to nodes in its face neighborhood $\mathcal{N}(a)$. Approximate discrete path between the node a and a point b is either a local shortest path (that can be computed in constant time) or a path of the form $\{a \overset{G}{\rightsquigarrow} p \overset{\mathcal{N}(b)}{\rightsquigarrow} b\}$, where p is a node of G_ε incident to the face neighborhood of b. Hence, the computation of $\mathrm{dist}_G(a,b)$ requires finding $\min_{p\in\mathcal{N}(b)}\{\mathrm{dist}_G(a,p) + \|p \overset{\mathcal{N}(b)}{\rightsquigarrow} b\|\}$. We know the distances $\mathrm{dist}_G(a,p)$ from a to all nodes $p \in G_\varepsilon$ (as part of preprocessing by computing SSSP tree rooted at a in G_ε) and thus our task is reduced to finding a node $p(b)$ that minimizes the above expression for a query point b. To accomplish this, for each face f of P we construct a data structure, called *Local Voronoi Diagram* in f with respect to a, and denote it by $\mathrm{LVD}(a,f)$. More precisely, let f be a face of P and let $\mathcal{N}(f)$ be its face neighborhood. Let $p_1,\ldots,p_k$ be the Steiner points and the vertices of P incident to $\mathcal{N}(f)$ and let $\delta_i = \mathrm{dist}_G(a,p_i)$ for $i = 1,\ldots,k$.

Lemma 4 *A data structure* $\mathrm{LVD}(a, f)$ *exists so that for a point* $b \in f$, $\min_{1 \le i \le k}(\delta_i + \|b \overset{\mathcal{N}(b)}{\rightsquigarrow} p_i\|)$ *and the point for which it is achieved can be computed in* $O(\log k)$ *time. The size of the data structure* $\mathrm{LVD}(a, f)$ *is* $O(k)$ *and it can be constructed in* $O(k \log k)$ *time.*

We define $\mathrm{SSQ}(P, \varepsilon; a)$ data structure to consist of SSSP tree rooted at a plus the collection of $\mathrm{LVD}(a, f)$ for all faces $f \in P$. The queries consist of a point b on P and the face[4] $f(b)$ containing b and they are answered as follows. First, we use $\mathrm{LVD}(a, f(b))$ and find the point $p(b)$ for which the minimum in Lemma 4 is achieved. Then, if a and b lie in neighbor faces, we compute the shortest local path between a and b and output the approximate distance $\mathrm{dist}_G(a, b)$, which is the smaller of the two values. If an approximation path is required we output the natural embedding of the approximate discrete path whose cost is $\mathrm{dist}_G(a, b)$. The quality of this approximation follows from Theorem 4 and Corollary 1. Hence we have the following

Theorem 5 *Given a triangulated, weighted surface* P *with* n *faces, a source point* $a \in P$ *and the set of nodes* V_ε *of* G_ε. *(For every query point* b *in* P *we assume that the face containing* b *is known.) A data structure* $\mathrm{SSQ}(P, \varepsilon; a)$ *of size* $O(|V_\varepsilon|) = O(\frac{n}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon})$ *exists, so that the approximate distance between* a *and a query point* b *in* P *can be found in* $O(\log \frac{1}{\varepsilon})$ *time. The structure SSQ can be constructed in* $O(|V_\varepsilon| \log |V_\varepsilon|) = O(\frac{n}{\sqrt{\varepsilon}} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$ *time.*

5 Arbitrary shortest path queries

In this section we describe and analyze an algorithm for constructing a data structure, called *All Pairs Queries* (APQ), such that approximate distance (and/or approximate shortest path) queries between pairs of arbitrary points in P can be answered efficiently. In addition to the weighted polyhedral surface P and the approximation parameter ε, the algorithm takes as input a query time parameter $\mathfrak{q}$ and outputs a data structure $\mathrm{APQ}(P, \varepsilon; \mathfrak{q})$, which can answer approximate distance queries in $O(\mathfrak{q})$ time. Our preprocessing algorithm uses the results of previous sections.

First we construct the dual graph P^* of P. The set of nodes of P^* corresponds to the set of faces of P and two nodes in P^* are joined by an edge if their corresponding faces are neighbors. To each node u of P^* we assign weight $w(u)$ equal to the number of nodes of G_ε, that are incident to the face $f(u)$ corresponding to u in P. Furthermore, we assign cost $c(u)$ equal to the number of nodes of G_ε, that are incident to the face neighborhood $\mathcal{N}(f(u))$. The total weight $w(P^*)$ of P^* and the value $\sigma(P^*)$, defined in Section 2, are estimated using Lemma 3 by $w(P^*) \le C(P)\frac{n}{\sqrt{\varepsilon}} \log \frac{2}{\varepsilon}$ and $\sigma(P^*) \le \Gamma(P)\frac{n}{\varepsilon} \log^2 \frac{2}{\varepsilon}$, where $C(P) = \frac{1}{n}\sum_{f \in P} C(f)$ and $\Gamma(P) \le \frac{4}{n}\sum_{f \in P} C^2(f)$. We observe that the weight of P^* and $\sigma(P^*)$ are related by $\sigma(P^*) \le 4w^2(P^*) \le n\sigma(P^*)$.

Next, we choose a value of $t = \frac{\mathfrak{q}^2}{4(g+1)\sigma(P^*)\log^2(1/\varepsilon)}$ (depending on the input query time $\mathfrak{q}$) and use Theorem 2 to construct a t-separator S, that induces regular partitioning of P^*. The separator S of P^* corresponds to a set of faces in P, which we refer to as *face separator* (or simply separator) and denote again by S. The face separator S partitions the surface P into *regions*, that are unions of faces corresponding to the connected components of $P^* \setminus S$. The boundary ∂R of a region R is the set of triangles in S, that neighbor faces in R.

Next, for each p, that is a Steiner point or vertex of P incident to a face, which is a neighbor of a face in S, compute and store $\mathrm{SSQ}(P, \varepsilon; p)$ data structure. Also for each region R and for each

[4] Otherwise point location on P would be necessary.

Steiner point or vertex of P incident to a face in R compute and store $\mathrm{SSQ}(R, \varepsilon; p)$ data structure restricted to the faces in R.

The collection of SSQ data structures and the region partitioning induced by S constitutes the $\mathrm{APQ}(P, \varepsilon; \mathfrak{q})$ data structure. We denote the genus of the surface P by g. The query time parameter $\mathfrak{q}$ will not exceed an upper bound $\bar{\mathfrak{q}} = \frac{(g+1)^{2/3} n^{1/3}}{\sqrt{\varepsilon}} \log^2 \frac{1}{\varepsilon}$. The next lemma presents our estimate on the time for the construction and the size of $\mathrm{APQ}(P, \varepsilon; \mathfrak{q})$ data structure.

Lemma 5 *For any* $\mathfrak{q} \leq \bar{\mathfrak{q}} = \frac{(g+1)^{2/3} n^{1/3}}{\sqrt{\varepsilon}} \log^2 \frac{1}{\varepsilon}$ *the construction of the data structure* $\mathrm{APQ}(P, \varepsilon; \mathfrak{q})$ *takes* $O(\frac{(g+1)n^2}{\varepsilon^{3/2}\mathfrak{q}} \log \frac{n}{\varepsilon} \log^4 \frac{1}{\varepsilon})$ *time. The size of* $\mathrm{APQ}(P, \varepsilon; \mathfrak{q})$ *is* $O(\frac{(g+1)n^2}{\varepsilon^{3/2}\mathfrak{q}} \log^4 \frac{1}{\varepsilon})$.

The APQ data structure built by the preprocessing algorithm can be used to answer approximate distance queries as outlined in Algorithm APQ_Query. Note that set A plays a critical role

ALGORITHM: **APQ_Query**

Input: The data structure $\mathrm{APQ}(P, \varepsilon; \mathfrak{q})$; query points a and b lying in faces $f(a)$ and $f(b)$, respectively.

Output: The approximate distance $\mathrm{dist}_G(a, b)$.

Set $M_0 = M_1 = M_2 = \infty$.

Step 1. If $f(a)$ and $f(b)$ are neighbor faces, then compute the local shortest path $a \overset{f(a)\cup f(b)}{\rightsquigarrow} b$ and assign its cost to M_0.

Step 2. If either of the faces $f(a)$ or $f(b)$ is in the separator S, then define A to be the set of nodes of G_ε incident to the face neighborhood $\mathcal{N}(a)$ or $\mathcal{N}(b)$, respectively.

Step 3. If neither of the faces $f(a)$ and $f(b)$ is in S, then define A to be the set of nodes of G_ε incident to the faces in the boundary $\partial R(a)$ of the region $R(a)$ containing $f(a)$.

Step 4. Use data structures $SSQ(P, \varepsilon; p')$ and compute $M_1 = \min_{p' \in A}(\mathrm{dist}_G(a, p') + \mathrm{dist}_G(p', b))$.

Step 5. If $f(b) \in R(a)$ then define A_1 to be the set of nodes of G_ε incident to the face neighborhood $\mathcal{N}(b)$. Use data structures $SSQ(R, \varepsilon; p')$ and compute $M_2 = \min_{p' \in A_1}(\mathrm{dist}_G(a, p') + \mathrm{dist}_G(p', b))$.

Set $\mathrm{dist}_G(a, b) = \min(M_0, M_1, M_2)$ and output it.

in the query algorithm. A set of nodes in G_ε is called a *separating set* for points a and b in P if any approximate discrete path of the form (1) between a and b contains a node from that set. Our query algorithm specifies a separating set A for a and b, such that for any $p' \in A$ the data structure $\mathrm{SSQ}(P, \varepsilon; p')$ is present in $\mathrm{APQ}(P, \varepsilon; \mathfrak{q})$ and then computes $\min_{p' \in A}(\mathrm{dist}_G(a, p') + \mathrm{dist}_G(p', b))$. Clearly, this minimum is the cost of the shortest approximate discrete path of the form (1). The time for this computation is $O(|A| \log \frac{1}{\varepsilon})$.

If an approximate shortest path between a and b is required we output the natural embedding of the approximate discrete path for which the minimum $\mathrm{dist}_G(a, b)$ is achieved. This can be done by using the SSSP trees stored in the corresponding SSQ data structure in time proportional to the size of this path. The next lemma establishes the correctness of the query algorithm and evaluates its running time.

Lemma 6 *The algorithm APQ_Query correctly computes the approximate distance* $\mathrm{dist}_G(a,b)$. *The running time of the algorithm is* $O(\max(\mathfrak{q}, \frac{1}{\sqrt{\varepsilon}}\log^2\frac{1}{\varepsilon})$.

Proof: The correctness of the query algorithm follows from the observation, that the cost of any approximate discrete path between a and b has to be equal to one of the values M_0, M_1, M_2, depending on the position of the faces $f(a)$ and $f(b)$ with respect to the partitioning defined by the separator S. The running time of the algorithm is dominated by the times for the execution of Steps 4 and 5. As discussed above these times are bounded by $O(|A|\log\frac{1}{\varepsilon})$ and $O(|A_1|\log\frac{1}{\varepsilon})$. By Lemma 3, $|A_1| = O(\frac{1}{\sqrt{\varepsilon}}\log\frac{1}{\varepsilon})$ and by Theorem 2 and the choice of t in the APQ_Preprocessing algorithm we obtain $|A| \leq 2\sqrt{(g+1)t\sigma(P^*)} \leq \frac{\mathfrak{q}}{\log(1/\varepsilon)}$. □
The results obtained in this section are summarized in the next theorem.

Theorem 6 *Let* P *be a weighted polyhedral surface of genus* g *and consisting of* n *triangular faces. Let* $\varepsilon \in (0,1)$ *and* $\mathfrak{q} \in (\frac{1}{\sqrt{\varepsilon}}\log^2\frac{1}{\varepsilon}, \bar{\mathfrak{q}})$, *where* $\bar{\mathfrak{q}} = \frac{(g+1)^{2/3}n^{1/3}}{\sqrt{\varepsilon}}\log^2\frac{1}{\varepsilon}$. *There exists a data structure* $\mathrm{APQ}(P,\varepsilon;\mathfrak{q})$, *such that approximate distance queries in* P *can be answered in* $O(\mathfrak{q})$ *time. The structure* $\mathrm{APQ}(P,\varepsilon;\mathfrak{q})$ *is constructed in* $O(\frac{(g+1)n^2}{\varepsilon^{3/2}\mathfrak{q}}\log\frac{n}{\varepsilon}\log^4\frac{1}{\varepsilon})$ *time and its size is* $O(\frac{(g+1)n^2}{\varepsilon^{3/2}\mathfrak{q}}\log^4\frac{1}{\varepsilon})$.

6 Extensions and Conclusions

In this paper we present novel solutions to fundamental shortest path query problems. The algorithms improve and generalize previous solutions in terms of 1) setting: a) Euclidean to weighted and b) arbitrary genus g, 2) preprocessing time, and/or 3) size of query data structure. We also develop a new graph partitioning algorithm for graphs of genus g with weights and costs on vertices which extends and/or generalizes previously known separator algorithms. Our techniques also enable us to obtain improved results with space-query time tradeoffs for the planar case, i.e. when the genus is 0 (we refer the reader to the full version of this paper).

A natural question arises whether the range $(\frac{1}{\sqrt{\varepsilon}}\log^2\frac{1}{\varepsilon}, \bar{\mathfrak{q}})$ of query time parameter $\mathfrak{q}$, in Theorem 6, can be widened, while keeping the efficiency of the algorithm. This question can be answered affirmatively by constructing a hierarchical APQ data structure, in which second level APQ data structures are built and stored for each region of the partitioning. For each region R of the partitioning construct an $\mathrm{APQ}(R,\varepsilon;\bar{\mathfrak{q}}_R)$ restricted to R, where $\bar{\mathfrak{q}}_R$ is the corresponding upper bound for R. Then in the APQ-Query algorithm we use $\mathrm{APQ}(R,\varepsilon;\bar{\mathfrak{q}}_R)$ to answer approximate shortest path queries in the case where both query points lie in R in $O(\bar{\mathfrak{q}}_R)$ in time. The latter is $O(\mathfrak{q})$ since $\bar{\mathfrak{q}} \leq \mathfrak{q}$. Our analysis shows that if $\mathfrak{q} \leq \bar{\mathfrak{q}}_1 = \frac{(g+1)^{5/9}n^{2/3}}{\varepsilon^{2/3}}\log^{7/3}\frac{1}{\varepsilon}$ the result of Theorem 6 extends to the case $\mathfrak{q} \in (\bar{\mathfrak{q}}, \bar{\mathfrak{q}}_1)$.

Next, we briefly describe how our technique can be used to build an APQ data structure with query time parameter $\mathfrak{q} = \log\frac{1}{\varepsilon}$. We build an ε-mesh consisting of $O(1/\varepsilon^2)$ additional points in each triangle f of P. The mesh is constructed so that for each point a in f there is a mesh point p which is closer to a than $\varepsilon d(p)$, where $d(p)$ is the minimum distance from p to any node of G_ε outside f. Then for each mesh point we compute, in $O(\frac{n^2}{\varepsilon^{5/2}}\log\frac{n}{\varepsilon}\log\frac{1}{\varepsilon})$ time, the SSQ data structure. For a pair of query points a, b we first find the mesh point $p(a)$ closest to a, and then use the structure $SSQ(P,\varepsilon;p(a))$ to find $\mathrm{dist}_G(p(a),b)$. Each of these two steps is carried out in $O(\log\frac{1}{\varepsilon})$ time. It can be shown that $\|ap(a)\| + \mathrm{dist}_G(p(a),b)$ is an ε-approximation of the distance $\mathrm{dist}_P(a,b)$.

Note that our algorithms inherit the geometric constants analyzed in [6]. It is an interesting open problem to determine whether eliminating these constants is inherently impossible.

References

[1] P. K. Agarwal, B. Aronov, J. O'Rourke, and C. A. Schevon. Star unfolding of a polytope with applications. *SIAM J. Comput.*, 26(6):1689–1713, 1997.

[2] P. K. Agarwal, S. Har-Peled, M. Sharir, and K.R. Varadarajan. Approximate shortest paths on a convex polytope in three dimensions. *J. Assoc. Comput.*, 44:567–584, March 1997.

[3] L. Aleksandrov and H. Djidjev. Linear algorithms for partitioning embedded graphs of bounded genus. *SIAM J. Disc. Math.*, 9(1):129–150, 1996.

[4] L. Aleksandrov, H. Djidjev, H. Guo, and A. Maheshwari. Partitioning planar graphs with costs and weights. In *ALENEX '02: Revised Papers from the 4th International Workshop on Algorithm Engineering and Experiments*, pages 98–110, London, UK, 2002. Springer-Verlag.

[5] L. Aleksandrov, M. Lanthier, A. Maheshwari, and J.-R. Sack. An ε-approximation algorithm for weighted shortest path queries on polyhedral surfaces. In *Proc. 14th Euro CG Barcelona*, pages 19–21, 1998.

[6] L. Aleksandrov, A. Maheshwari, and J.-R. Sack. Determining approximate shortest paths on weighted polyhedral surfaces. *J. ACM*, 52(1):25–53, 2005.

[7] B. Chazelle, D. Liu, and A. Magen. Sublinear geometric algorithms. *SIAM J. Comput.*, 35:627–646, 2006.

[8] J. Chen and Y. Han. Shortest paths on a polyhedron. In *Proc. 6th ACM Symposium on Computational Geometry*, pages 360–369, 1990. Full version IJCGA 6: 127-144, 1996.

[9] Y.-J. Chiang and J. S. B. Mitchell. Two-point euclidean shortest path queries in the plane. In *Proc. 10th ACM-SODA*, pages 215–224, Philadelphia, PA, USA, 1999.

[10] H. N. Djidjev. Linear algorithms for graph separation problems. In *SWAT'88, LNCS*, volume 318, pages 643–645. Springer-Verlag, Berlin, Heidelberg, 1988.

[11] R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *J. Approx. Theory*, 10(3):227–236, 1974.

[12] G. N. Frederickson. Fast algorithms for shortest paths in planar graphs. *SIAM J. Comput.*, 16:1004–1022, 1987.

[13] J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5:391–407, 1984.

[14] S. Har-Peled. Approximate shortest paths and geodesic diameters on convex polytopes in three dimensions. *DCG*, 21:216–231, 1999.

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

[16] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980.

[17] U. Zwick. Exact and approximate distance in graphs - a survey. In F. Meyer auf der Heide, editor, *Proc. ESA, LNCS 2161*, pages 33–48. Springer Verlag, Berlin, Heidelberg, 2001.

7 Appendix

Statement of Theorem 4.
For any pair of points a and b in P one of the following two holds, either

$$\textbf{(a)} \qquad (1-2\varepsilon)\mathrm{dist}_P(a,b) \le \mathrm{dist}_G(a,b) \le (1+2\varepsilon)\mathrm{dist}_P(a,b), \tag{2}$$

or

(b) There is a vertex v of P such that the points a, b are in the face neighborhood $\mathcal{N}(v)$. There is a shortest path in P between a and b that stays in $\mathcal{N}(v)$ and intersects the vertex vicinity $\mathcal{E}(v)$. Moreover

$$\mathrm{dist}_P(a,b) - 2\varepsilon r(v) \le \mathrm{dist}_G(a,b) \le \mathrm{dist}_P(a,b), \tag{3}$$

where $\varepsilon r(v)$ is the radius of $\mathcal{E}(v)$.
Proof:
Case 1: Let a and b be incident with the representations of nodes p and q of G_ε. Then by our definitions $\mathrm{dist}_G(a,b) = c(p \overset{G}{\rightsquigarrow} q)$. Following the proof of Theorem 3.2 in [6] we obtain

$$c(p \overset{G}{\rightsquigarrow} q) \le (1+\varepsilon)\mathrm{dist}_P(a,b), \tag{4}$$

which is stronger than the right part of (2). To prove the estimates on $\mathrm{dist}_G(a,b)$ from below we assume that $\mathrm{dist}_G(a,b) < \mathrm{dist}_P(a,b)$, otherwise there is nothing to prove. We consider the natural embedding of the path $p \overset{G}{\rightsquigarrow} q$. As discussed above the cost of the path $p \overset{G}{\rightsquigarrow} q$ in G_ε equals to the cost of the natural embedding in P minus the total cost of its portions inside vertex vicinities. If the total cost of these portions is less or equal $\varepsilon\mathrm{dist}_P(a,b)$ then we have

$$(1-\varepsilon)\mathrm{dist}_P(a,b) \le c(p \overset{G}{\rightsquigarrow} q), \tag{5}$$

which proves (a) in this case.

By the definitions of the vertex vicinities and the cost of the edges in G_ε the total cost of these portions is at most $(\varepsilon/2)c(p \overset{G}{\rightsquigarrow} q)$ provided that the path $p \overset{G}{\rightsquigarrow} q$ contains none or at least two vertex vicinity nodes. Using our assumption $\mathrm{dist}_G(a,b) < \mathrm{dist}_P(a,b)$ we have

$$\mathrm{dist}_P(a,b) \le c(p \overset{G}{\rightsquigarrow} q) + (\varepsilon/2)c(p \overset{G}{\rightsquigarrow} q) \le c(p \overset{G}{\rightsquigarrow} q) + (\varepsilon/2)\mathrm{dist}_P(a,b),$$

and thus the inequality (5) is valid in these cases too.

It remains to consider the case where the path $p \overset{G}{\rightsquigarrow} q$ contains exactly one vertex vicinity node, say corresponding to a vertex v of P, and the cost of the portion of the natural embedding of $p \overset{G}{\rightsquigarrow} q$ inside $\mathcal{E}(v)$ is greater than $\varepsilon\mathrm{dist}_P(a,b)$. In this case the left inequality of (3) holds.

Case 2: Consider now the case where a and b are neither Steiner points nor inside vertex vicinities. If the points a and b lie in neighbor triangles and the shortest path between them is a local path then by our definitions $\mathrm{dist}_G(a,b) = \mathrm{dist}_P(a,b)$ and the theorem holds. So, we assume below that any shortest path between a and b intersects more than two faces.

First, we prove the upper bound on $\mathrm{dist}_G(a,b)$ by constructing an approximate discrete path $\{a \overset{\mathcal{N}(a)}{\rightsquigarrow} p \overset{G}{\rightsquigarrow} q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\}$ whose cost is at most $(1+2\varepsilon)\mathrm{dist}_P(a,b)$. Let $\tilde{\pi} = a \overset{P}{\rightsquigarrow} b$ be a shortest path between a and b in P. The path $\tilde{\pi}$ consists of segments with end-points on the edges of P. We call

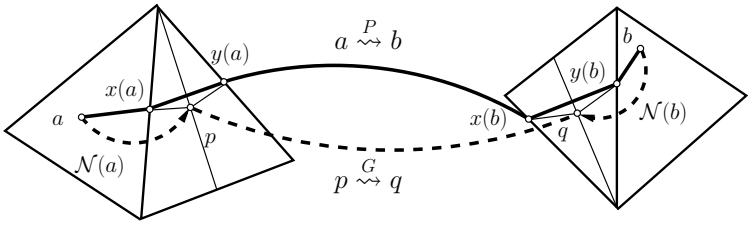

Figure 1: Illustration of bending points in $\tilde{\pi}$.

these points *bending* points of the path $\tilde{\pi}$. From our assumption the path $\tilde{\pi}$ intersects more than two faces and the definition of face neighborhoods it follows that the path $\tilde{\pi}$ is neither entirely in $\mathcal{N}(a)$ nor in $\mathcal{N}(b)$.

We denote by $y(a)$ the first bending point on $\tilde{\pi}$ when traversed from a to b, where the path exits $\mathcal{N}(a)$ (see Figure 1). Similarly, let $x(b)$ be the last bending point on $\tilde{\pi}$ when traversed from a to b, where the path enters $\mathcal{N}(b)$. Furthermore, we denote by $x(a)$ the bending point on $\tilde{\pi}$ preceding $y(a)$ and by $y(b)$ the bending point succeeding $x(b)$.

Let us consider, first, the case where none of the points $x(a)$, $y(a)$, $x(b)$, and $y(b)$ is inside vertex vicinity. By their choice $x(a)$ and $y(a)$ lie on two different sides of the face f containing the segment $(x(a), y(a))$. We define p to be a Steiner point in f such that

$$|x(a)p| + |py(a)| \leq (1 + \varepsilon/2)|x(a)y(a)|. \tag{6}$$

The existence of such a Steiner point is granted by Lemma 2. Similarly, let q be a Steiner point inside the face containing $(x(b), y(b))$ and such that

$$|x(b)q| + |qy(b)| \leq (1 + \varepsilon/2)|x(b)y(b)|. \tag{7}$$

If $x(a) = x(b)$ and $y(a) = y(b)$ we chose $p = q$. Now we have defined an approximate discrete path $\pi(a, b) = \{a \overset{\mathcal{N}(a)}{\rightsquigarrow} p \overset{G}{\rightsquigarrow} q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\}$ and estimate its cost.

We consider the case where $(x(a), y(a)) \neq (x(b), y(b))$ (Figure 1). From the definition of the points $x(a)$ and $y(b)$, and the triangle inequality we have

$$\|a \overset{\mathcal{N}(a)}{\rightsquigarrow} p\| \leq \|a \overset{P}{\rightsquigarrow} x(a)\| + \|x(a)p\| \tag{8}$$

and

$$\|q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\| \leq \|y(b) \overset{P}{\rightsquigarrow} b\| + \|qy(b)\|. \tag{9}$$

Again from the triangle inequality and the validity of (2) for the path $p \overset{G}{\rightsquigarrow} q$ we obtain

$$c(p \overset{G}{\rightsquigarrow} q) \leq (1 + \varepsilon)(\|y(a) \overset{P}{\rightsquigarrow} x(b)\| + \|py(a)\| + \|x(b)q\|). \tag{10}$$

Then the cost of the path $\pi(a, b)$ is estimated by summing (8), (10), and (9) and using (6), and (7)

$$\begin{aligned} c(\pi(a,b)) = \|a \overset{\mathcal{N}(a)}{\rightsquigarrow} p\| + c(p \overset{G}{\rightsquigarrow} q) + \|q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\| \leq \\ \|a \overset{P}{\rightsquigarrow} x(a)\| + (1+\varepsilon)\|y(a) \overset{P}{\rightsquigarrow} x(b)\| + \|y(b) \overset{P}{\rightsquigarrow} b\| + \\ (1+\varepsilon/2)\|x(a)y(a)\| + (1+\varepsilon/2)\|x(b)y(b)\| + \\ \varepsilon(\|py(a)\| + \|x(b)q\|) \leq (1+2\varepsilon)\mathrm{dist}_P(a,b) \end{aligned}$$

The case where segments $(x(a), y(a)) = (x(b), y(b))$ is simpler. The cases where some of the points $x(a)$, $y(a)$, $x(b)$, and $y(b)$ are inside vertex vicinities are handled by the same technique.

Next, we estimate $\text{dist}_G(a, b)$ from below. Again, we assume that $\text{dist}_G(a, b) < \text{dist}_P(a, b)$ otherwise statement (a) of the theorem holds. We consider any approximate discrete path of the form $\{a \overset{\mathcal{N}(a)}{\rightsquigarrow} p \overset{G}{\rightsquigarrow} q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\}$. If p is a vertex vicinity node we denote by $\tilde{p}$ the vertex of P in that vicinity. We defined the radius $r(\tilde{p})$ as (1/8)-th of the distance from $\tilde{p}$ to the boundary of its face neighborhood $\mathcal{N}(\tilde{p})$ and used $r(\tilde{p})$ to define $\mathcal{E}(\tilde{p})$. If p is a Steiner point node then $\tilde{p}$ denotes the corresponding Steiner point and we assume $r(\tilde{p}) = 0$. We use analogous notation $\tilde{q}$ and $r(\tilde{q})$ with respect to q. From our definitions and using this notation we have

$$\|a \overset{\mathcal{N}(a)}{\rightsquigarrow} p\| \geq \|a \overset{P}{\rightsquigarrow} \tilde{p}\| - \varepsilon r(\tilde{p}) \quad \text{and} \quad \|q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\| \geq \|\tilde{q} \overset{P}{\rightsquigarrow} b\| - \varepsilon r(\tilde{q}). \tag{11}$$

Then for the cost of the path $\{a \overset{\mathcal{N}(a)}{\rightsquigarrow} p \overset{G}{\rightsquigarrow} q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\}$ we obtain

$$\begin{aligned} \|a \overset{\mathcal{N}(a)}{\rightsquigarrow} p\| + c(p \overset{G}{\rightsquigarrow} q) + \|q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\| \geq \\ \|a \overset{P}{\rightsquigarrow} \tilde{p}\| + c(p \overset{G}{\rightsquigarrow} q) + \|\tilde{q} \overset{P}{\rightsquigarrow} b\| - \varepsilon(r(\tilde{p}) + r(\tilde{q})) \end{aligned} \tag{12}$$

Let us consider first the case, where

$$r(\tilde{p}) + r(\tilde{q}) \leq \|a \overset{P}{\rightsquigarrow} \tilde{p}\| + c(p \overset{G}{\rightsquigarrow} q) + \|\tilde{q} \overset{P}{\rightsquigarrow} b\|.$$

In this case, we use (12) and (5) and obtain

$$\begin{aligned} \|a \overset{\mathcal{N}(a)}{\rightsquigarrow} p\| + c(p \overset{G}{\rightsquigarrow} q) + \|q \overset{\mathcal{N}(b)}{\rightsquigarrow} b\| \geq \\ (1-\varepsilon)(\|a \overset{P}{\rightsquigarrow} \tilde{p}\| + c(p \overset{G}{\rightsquigarrow} q) + \|\tilde{q} \overset{P}{\rightsquigarrow} b\|) \geq \\ (1-\varepsilon)(\|a \overset{P}{\rightsquigarrow} \tilde{p}\| + (1-\varepsilon)\text{dist}_P(\tilde{p}, \tilde{q}) + \|\tilde{q} \overset{P}{\rightsquigarrow} b\|) \geq \\ (1-\varepsilon)^2(\|a \overset{P}{\rightsquigarrow} \tilde{p}\| + \text{dist}_P(\tilde{p}, \tilde{q}) + \|\tilde{q} \overset{P}{\rightsquigarrow} b\|) \geq (1-2\varepsilon)d_p(a, b), \end{aligned} \tag{13}$$

which implies the lower bound in (a). Consider now the case where

$$r(\tilde{p}) + r(\tilde{q}) > \|a \overset{P}{\rightsquigarrow} \tilde{p}\| + c(p \overset{G}{\rightsquigarrow} q) + \|\tilde{q} \overset{P}{\rightsquigarrow} b\|. \tag{14}$$

We show that in this case (b) holds. This inequality is possible only if one of the points $\tilde{p}$ and $\tilde{q}$ is a vertex of P, denoted by v, and the other is either a Steiner point inside the face neighborhood of v or the same vertex. Now, we use (5) for $\tilde{p}$ and $\tilde{q}$ and substitute in (14) obtaining

$$\begin{aligned} 2r(v) \geq r(\tilde{p}) + r(\tilde{q}) > \|a \overset{P}{\rightsquigarrow} \tilde{p}\| + c(p \overset{G}{\rightsquigarrow} q) + \|\tilde{q} \overset{P}{\rightsquigarrow} b\| \geq \\ \|a \overset{P}{\rightsquigarrow} \tilde{p}\| + \text{dist}_P(\tilde{p}, \tilde{q}) - 2\varepsilon r(v) + \|\tilde{q} \overset{P}{\rightsquigarrow} b\| = \text{dist}_P(a, b) - 2\varepsilon r(v), \end{aligned} \tag{15}$$

which implies $\text{dist}_G(a, b) \geq d_P(a, b) - 2\varepsilon r(v)$. Moreover, from (15) and the definition of the radius $r(v)$ it follows that the path $a \overset{P}{\rightsquigarrow} b$ must stay inside $\mathcal{N}(v)$ and must intersect $\mathcal{E}(v)$.

The cases where just one of the points a or b is Steiner point or incident to a vertex vicinity are treated analogously. □