Approximating Shortest Paths on Weighted Polyhedral Surfaces [*] [†]

Mark Lanthier, Anil Maheshwari and Jörg-Rüdiger Sack
School of Computer Science
Carleton University
1125 Colonel By Drive
Ottawa, ON, CANADA K1S 5B6
E-mail: [lanthier,maheshwa,sack] @scs.carleton.ca

[*]Research supported by NSERC in conjunction with our industrial partner ALMERCO Inc.

[†]A preliminary version has been presented at the 13th Annual ACM Symposium on Computational Geometry [21] with an accompanying video [23].

1 Introduction

1.1 Problem Definition

Shortest path problems are among the fundamental problems studied in computational geometry and other areas such as graph algorithms, geographical information systems (GIS) and robotics. We encountered several shortest path related problems in our R&D on GIS ¶. Let s and t be two vertices on a given possibly non-convex polyhedron $\mathcal{P}$, in $\Re^3$, consisting of n triangular faces on its boundary, each face has an associated positive weight. A Euclidean *shortest path* $\pi(s,t)$ between s and t is defined to be a path with minimum Euclidean length among all possible paths joining s and t that lie on the surface of $\mathcal{P}$. A *weighted shortest path* $\Pi(s,t)$ between s and t is defined to be a path with minimum cost among all possible paths joining s and t that lie on the surface of $\mathcal{P}$. The *cost* of the path is the sum of the lengths of all segments multiplied by the corresponding face weight.

In this paper we propose several simple and practical algorithms (schemes) to compute an approximated weighted shortest path $\Pi'(s,t)$ between two points s and t on the surface of a polyhedron $\mathcal{P}$.

1.2 Related Work

Shortest path problems in computational geometry can be categorized by a variety of factors which include the dimensionality of space, the type and number of objects or obstacles (e.g., polygonal obstacles, convex or non-convex polyhedra, ...) and the distance measure used (e.g., Euclidean, number of links, or weighted distances). Here we discuss those contributions which relate directly to our work; these are in particular 3-dimensional weighted scenarios. Due to their relevance in practice, 3-dimensional shortest paths problems have received considerable attention. The 3-dimensional Euclidean shortest path problem is stated as follows: Given a set of pairwise disjoint polyhedra in $\Re^3$ and two points s and t, compute a shortest path between s and t, that avoids the interiors of the polyhedra. Canny and Reif [7] showed that this problem is NP-Hard. The shortest path problem amidst (disjoint) convex polyhedra can be solved in time exponential in the number of polyhedral objects as was shown by Sharir [36]. For two polyhedral obstacles with a total of n vertices, Baltsan and Sharir [6] presented an $O(n^3 \log n)$ time shortest path algorithm. The computation of Euclidean shortest paths on non-convex polyhedra has been investigated by [8, 28, 30, 2]; currently, the best known algorithm is due to Chen and Han [8] and it runs in $O(n^2)$ time.

¶For information regarding our GIS work see [39].

Since most application models are approximations of reality and high-quality paths are favored over optimal paths that are "hard" or expensive to compute, approximation algorithms are suitable and necessary. Choi et al. [9] presented a refinement of work by Papadimitriou [32] who provided an ϵ-approximation of shortest paths amidst polyhedral obstacles in three dimensions. The algorithm runs in $O(n^4(L+\log(n/\epsilon))^2/\epsilon^2)$ time, where L represents the bit precision. Clarkson [10] presented an $O(\frac{n \log n}{\epsilon})$ time algorithm to build a data structure so that a path between two query points can be computed in $O(\frac{n}{\epsilon} + n \log n)$ time. He also provides an algorithm to compute an ϵ-approximation for paths amidst polyhedral obstacles in 3-space which runs in approximately $O(n^2 \log^{O(1)} n/\epsilon^4)$ time.

Recently, there have been several results on approximation algorithms for computing shortest paths on single convex and non-convex polyhedra. Hershberger and Suri [17] presented a simple linear time algorithm that computes a short path on a convex polyhedron that is at most twice the Euclidean shortest path length. Har-Peled et al. [16] extended this result to provide an algorithm to compute an ϵ-approximation of the shortest path; it runs in $O(n \min\{1/\epsilon^{1.5}, \log n\} + 1/\epsilon^{4.5} \log(1/\epsilon))$ time. Agarwal et al. [1] provided an improved algorithm that runs in $O(n \log \frac{1}{\epsilon} + \frac{1}{\epsilon^3})$ time. All of these algorithms crucially exploit the properties of convex polyhedra and hence are not easily extendible to non-convex polyhedra. Varadarajan and Agarwal [2] provide an algorithm that computes a path on a, possibly non-convex, polyhedron that is at most $7(1+\epsilon)$ times the shortest path length; it runs in $O(n^{5/3} \log^{5/3} n)$ time. They also present a slightly faster algorithm that returns a path which is at most $15(1+\epsilon)$ times the shortest path length.

All of the work mentioned so far considers the problem of computing paths using the Euclidean distance metric. Mitchell and Papadimitriou [29] introduced the *weighted region problem* in which each face has an associated weight, denoted by a real number $w_i > 0$. They presented an algorithm that computes a path between two points in a weighted planar subdivision which is at most $(1+\epsilon)$ times the shortest weighted path cost. Their algorithm requires $O(n^8 L)$ time in the worst case, where $L = \log(nNW/w\epsilon)$ is a factor representing the bit complexity of the problem instance. Here N is the largest integer coordinate of any vertex of the triangulation and W (respectively, w) is the maximum (respectively, minimum) weight of any face of the triangulation. In recent work, Johannson presents a weighted distance model for injection molding [19]. Mata and Mitchell [26] presented independently and at the same time as this work [21] an alternate algorithm that constructs a graph which can be searched to obtain an ϵ-approximate path; their algorithm runs in $O(kn^3)$ time, where $k = O(\frac{W/w}{\epsilon\theta_{min}})$, W/w is the largest to smallest weight ratio, and θ_{min} is the minimum angle of any face of $\mathcal{P}$.

1.3 Contributions

Our approach to solving the weighted shortest path problem is to discretize the input polyhedron in a natural way, by placing Steiner points along the edges of the polyhedron. We construct a graph G containing the Steiner points as vertices and edges as the interconnections between Steiner points that correspond to segments which lie completely in the triangular faces of the polyhedron. The geometric shortest path problem on polyhedra is thus stated as a graph problem so that the existing efficient algorithms and their implementations for shortest paths in graphs can be used (we use a variation of Dijkstra's algorithm as mentioned in Section 4.1).

Our discretization method falls into the class of edge subdivision algorithms. Grid-based methods as introduced e.g., by Papadimitriou [32], are instances of this class. As concluded by Choi, Sellen and Yap [9]: *"... grids are a familiar practical technique in all of computational sciences. From a complexity theoretic viewpoint, such methods have been shunned in the past as trivial or uninteresting. This need not be so, as Papadimitriou's work has demonstrated. In fact, the grid methods may be the most practical recourse for solving some intractable problems."* The results of this paper substantiate their claim.

Our contributions are:

1. Theoretical
   - The design and analysis of several simple schemes for shortest path approximations on weighted and unweighted polyhedra.
   - Proofs establishing the accuracy of the approximations. The accuracy of the approximation varies with the length of the longest edge L of $\mathcal{P}$ and the maximum weight W of the faces of $\mathcal{P}$. We present two variations on the algorithm. The first ensures that $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + W|L|$. The second, based on graph spanners, ensures that $\|\Pi'(s,t)\| \leq \beta(\|\Pi(s,t)\| + W|L|)$ for some fixed constant $\beta > 1$. This second variation produces quicker run times, but at the cost of reduced accuracy.
   - Results on arbitrary shortest path queries.
2. Experimental
   - Implementation of:
     - all of our schemes

  - point-to-point Euclidean shortest paths in a sleeve
  - graph spanners on faces of polyhedra
  - Chen and Han's algorithm (as relevant to this work)
- The algorithms:
  - are easy to implement
  - employ elementary data structures (such as heap, graph, list) and operations.
- Extensive experiments on triangular irregular networks (TINs) establish:
  - Excellent performance of the schemes in path quality and run time both being better than the theoretical worst case bounds.
  - A constant number of Steiner points (i.e., six) per edge suffice implying an $O(n \log n)$ run time as was also observed experimentally.
  - In the unweighted case, a direct comparison to Chen and Han's algorithm is possible. Here our schemes show a fast convergence to optimal in accuracy with a much improved running time over Chen and Han. Using an additional post-processing step the exact shortest paths are frequently obtained.
  - In the weighted scenario, as far as we are aware of, this work represents the first adequately documented implementation[‡]. Here also the path accuracy converges rapidly at a fast running time.

The paper is organized as follows: Section 2 introduces notation and properties that are required. Sections 3 describes our approximation algorithms for shortest paths computations. Section 4 presents the testing procedures, experimental setup and results that were obtained. Section 5 discusses an extension to our algorithm to handle shortest path queries. We conclude with Section 6.

2 Preliminaries

Let $\mathcal{P}$ be a polyhedron with n triangular faces. Let L be the longest edge of $\mathcal{P}$ and let θ_{min} be the minimum interior angle among all faces of $\mathcal{P}$. Each face $f_i, 1 \leq i \leq n$ is assigned a positive weight, $w_{f_i} > 0$, which represents the cost of traveling through that face. The cost could be determined

[‡]At the same time as this work [21], an alternate approach of [26] was proposed.

e.g., by the slope or the characteristic of terrain (water, forest, ...). Two faces are called *edge-adjacent* if they share a common edge. The weight of an edge e of $\mathcal{P}$ is the minimum weight of its adjacent faces. Let W and w denote the maximum and minimum weights of any face in $\mathcal{P}$, respectively.

Let s and t be two points on the surface of $\mathcal{P}$ (the source and target points respectively). Denote by $\text{dist}(s,t)$ the minimum distance between s and t in 3-space. A shortest Euclidean path between s and t is denoted by $\pi(s,t)$ with cost, $|\pi(s,t)|$, and it is composed of a sequence of k adjacent straight line segments (or links) $s_1, s_2, ..., s_k$. We use $'$ to denote the corresponding approximation paths, segments etc... In general, $k \neq k'$, but for analysis we will often examine approximated paths where $k = k'$ (i.e., $\pi'(s,t)$ passes through the same sequence of faces as $\pi(s,t)$).

Property 2.1 *(Sharir and Schorr [37]) A Euclidean shortest path $\pi(s,t)$ may cross a face at most once.*

Property 2.2 *(Sharir and Schorr [37]) A Euclidean shortest path does not bend in the interior of a face.*

Let $F = f_1, f_2, ..., f_k$ be a set of faces of $\mathcal{P}$ such that f_i is edge-adjacent to f_{i+1} for $1 \leq i \leq k-1$. We say that f_i and f_{i+1} are *edge-adjacent* if they share an edge e_i. An *unfolding* of two edge-adjacent faces f_i and f_{i+1} is the set of points obtained by rotating f_{i+1} about e_i until all vertices of f_{i+1} lie in the same plane as that of f_i. The unfolding is said to be a *planar unfolding* if and only if the unfolded faces do not overlap. (We say that two faces *overlap* if they have a common point in their interior). A sequence of faces F is unfolded by unfolding the faces in order from 1 to k such that f_i is in the same plane as each of f_{i-1}, f_{i-2},...,f_1, where $i > 1$. Let $F' = f'_1, f'_2, ..., f'_k$ be the sequence of faces obtained by applying such an unfolding. We call F' a *simple sleeve* if it is a planar unfolding. An unfolding which does not guarantee planarity is called a *non-simple sleeve.*

The shortest weighted cost path between s and t is denoted by $\Pi(s,t)$ with cost $\|\Pi(s,t)\|$. The cost of a segment $s_j, 1 \leq j \leq k$ passing through face $f_i, 1 \leq i \leq n$ is assumed to be $w_{f_i}|s_j|$.

Property 2.3 *(Mitchell and Papadimitriou [29]) A weighted shortest path obeys Snell's law [25] of refraction in which the path bends at the edges of $\mathcal{P}$. The amount by which it bends depends on the relative weights of the two faces adjacent to this edge (see Figure 1a)).*

Property 2.4 *(Mitchell and Papadimitriou [29]) A weighted shortest path may travel along (i.e., critically use) an edge which is cheaper and then reflect back into the face (see Figure 1b).*

Property 2.5 *(Mitchell and Papadimitriou [29]) A weighted shortest path may cross a face $\theta(n)$ times.*

We distinguish between two types of path segments of a weighted shortest path: 1) *face-crossing* segments in which the endpoints lie on different edges of a face, and 2) *edge-using* segments in which both endpoints lie on the same edge of $\mathcal{P}$.

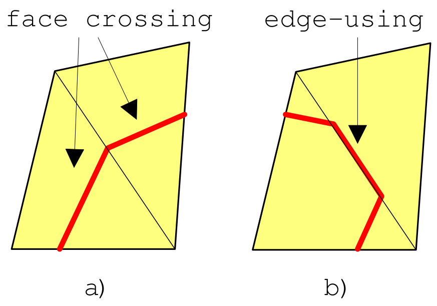


FIGURE 1. Characteristics of a weighted shortest path. a) adjacent face-crossing segments cause a bend in the path, b) edge-using segments cause a reflection back into the same face.

Let $G = (V, E)$ be a simple connected graph with vertex set V and edge set E, where each edge has a positive weight. We will make use of the following algorithms:

Algorithm 1: *Given two vertices s and t of G, a shortest path in G from s to t can be computed in $O(V^2)$ time (Dijkstra [11]). Using Fibonacci heaps, a shortest path can be computed in G from s to t in $O(V \log V + E)$ time (Fredman and Tarjan [12]).*

Observation 2.1 follows by straight forward generalization of the planar point location algorithm [35], see also [14] and [31].

Observation 2.1 *Given a polyhedron $\mathcal{P}$ with $O(n)$ faces, a data structure can be built in $O(n^3)$ time such that for a given query point q on $\mathcal{P}$, we can determine the face of $\mathcal{P}$ in which q lies in $O(\log n)$ time. In the special case in which $\mathcal{P}$ is a terrain, then more efficient planar point location algorithms can be used (e.g. see [20]).*

3 Shortest Path Approximations

A natural approach to approximating paths on a given polyhedron $\mathcal{P}$ is to choose a path which is restricted to traveling along edges of $\mathcal{P}$. That is, one could compute a graph G as follows. The vertices in G correspond to the vertices of $\mathcal{P}$ and there is an edge between two vertices in G if the corresponding vertices in $\mathcal{P}$ are connected by a polyhedral edge. The weight of an edge in G is the Euclidean length of the corresponding edge in $\mathcal{P}$ times the minimum weight of its two adjacent faces. For simplicity, assume that s and t are vertices of $\mathcal{P}$. An approximate shortest path $\Pi'(s,t)$, between s and t can now be computed by using Algorithm 1.

Although this method may not produce a good approximation in the worst case (the interested reader is referred to [21] for an example), we can bound the path cost with respect to parameters of $\mathcal{P}$. In the following theorem we present an upper bound on the approximation quality obtained by using the simple approach. Due to the dependency on θ_{min}, the approximation may be bad in the worst case which motivates the development of alternate schemes presented next. We include the proof in the appendix for the interested reader.

Theorem 3.1 *A shortest path* $\Pi(s,t)$ *between two vertices* s *and* t *of a weighted polyhedral surface* $\mathcal{P}$ *with* n *faces can be approximated by a path* $\Pi'(s,t)$ *such that* $\|\Pi'(s,t)\| \leq (\frac{2}{\sin\theta_{min}})\|\Pi(s,t)\|$*, where* θ_{min} *is the minimum interior angle of any face of* $\mathcal{P}$*. Furthermore,* $\|\Pi'(s,t)\|$ *can be computed in* $O(n \log n)$ *time.*

3.1 Improved Approximation Schemes

The approximation schemes introduced in this section discretize the polyhedron by placing Steiner points along the edges of the polyhedron. The schemes differ in the placement of Steiner points, the interconnection (edges) between Steiner points, and possible refinement (post-processing using sleeve computation). We point out that, in all of our schemes, we can store the graph as it pertains to Steiner points implicitly. In an iteration of Dijkstra's algorithm, adjacency information can be computed on the fly at a small additional cost. Thus, the graph needs neither to be precomputed nor to be stored.

We give an outline of our presentation of the three schemes. First, we describe the method of placing Steiner points on each face (i.e., the scheme), and define a graph on each face, called a *face graph.* Then we derive bounds on the portion of the shortest path going through a face. After that we compute the graph for the entire polyhedron by taking the union of face graphs for each face and prove the stated approximation bounds for paths computed by the respective schemes. We do this by showing the existence of a path approximating the shortest path within the stated bounds. The result of applying Dijkstra's algorithm may be this path or a path with equal or better cost.

Fixed Scheme:

We place m Steiner points evenly along each edge of $\mathcal{P}$, for some positive integer m. For each face $f_i, 1 \leq i \leq n$ of $\mathcal{P}$, compute a face graph G_i as follows. The Steiner points, along with the original vertices of f_i, become vertices of G_i. Connect a vertex pair v_a, v_b of G_i to form an edge $v_a v_b$ of G_i if and only if v_a and v_b correspond to Steiner points (or vertices) that lie on different edges of f_i or are adjacent on the same edge of f_i. The weight on a graph edge $v_a v_b$ is the Euclidean distance between v_a and v_b times the weight of f_i, and the magnitude of this weight is denoted as $|v_a v_b|$. For example, Figure 2 shows how 6 Steiner points and 27 edges are added to a face to form the face graph G_i where $m = 2$.

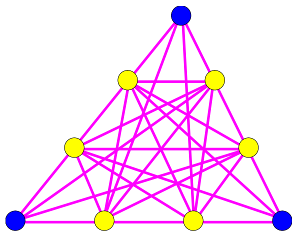


FIGURE 2. Adding Steiner points and edges to a face using the "fixed scheme".

To avoid introducing additional notation, we say that the segment corresponding to graph edge s'_j on P is also s'_j, and from the context its usage will be clear.

Claim 3.1 *Given a segment s_j crossing face f_i, there exists an edge s'_j in G_i such that $|s'_j| \leq w_{f_i} \cdot |s_j| + w_{f_i} \cdot \frac{|L|}{m+1}$.*

Proof: Each edge in $\mathcal{P}$ is divided into $m+1$ intervals which have length at most $\frac{|L|}{m+1}$. Let $s_j = ab$, where a and b are the end points of s_j lying on edges e_a and e_b of f_i, $e_a \neq e_b$, respectively. Let c (respectively, d) be Steiner point in f_i, where c (respectively, d) is closest to a (respectively, b) among Steiner points on e_a (respectively, e_b). Since c and d lie on different edges of f_i, then we know that there is an edge $e \in G_i$ joining them (see Figure 3), set $s'_j = e$. The triangle inequality ensures that $|s'_j| \leq |\overline{ca}| + |s_j| + |\overline{bd}|$. Since we chose the closer interval endpoints, then $|\overline{ca}| \leq \frac{|L|}{2(m+1)}$ and $|\overline{bd}| \leq \frac{|L|}{2(m+1)}$. Hence

$$|s'_j| \leq |s_j| + \frac{|L|}{m+1}. \tag{1}$$

Now, multiplying by w_{f_i} we have

$$w_{f_i} \cdot |s'_j| \leq w_{f_i} \cdot |s_j| + w_{f_i} \cdot \frac{|L|}{m+1}. \tag{2}$$

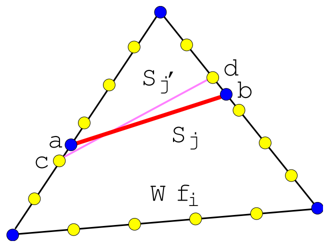


FIGURE 3. A face-crossing segment s_j of a weighted shortest path.

Notice that the weight of the graph edge s'_j equals the length of the segment s'_j on $\mathcal{P}$ times the weight w_{f_i} of the face. Moreover, the above arguments can be used to show that if s_j is edge using, then there exists a sequence of adjacent collinear Steiner edges joining the corresponding Steiner points, and we can view these collinear edges as a single segment s'_j. □

A graph G is computed by forming the union of all face graphs $G_i, 1 \leq i \leq n$. We call this particular method of graph creation the *fixed scheme*. It can be shown that all edges of G lie on the surface of $\mathcal{P}$. Hence, any path in G (i.e., our approximation) can be transformed to a path on the surface of $\mathcal{P}$. In the following lemma we show that there exists a path in G that approximates shortest path $\Pi(s,t)$.

Lemma 3.1 *Given two vertices s and t of $\mathcal{P}$, there exists a path $\Pi'(s,t)$ in G between the vertices corresponding to s and t such that $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + \frac{|L|}{m+1} \cdot k \cdot W$, where k is the number of segments of $\Pi(s,t)$.*

Proof: Let $\Pi(s,t) = \{s_1, s_2, \cdots, s_k\}$. For each $s_j \in \Pi(s,t)$, it follows from Claim 3.1 that there exists an edge $s'_j \in G$ that approximates s_j. We choose this edge s'_j such that it joins the Steiner points that are closest to the endpoints of s_j. Let $\Pi'(s,t) = \{s'_1, s'_2, \cdots, s'_k\} \in G$. Observe that since consecutive edges s_j and s_{j+1} of $\Pi(s,t)$ share a common point, then s'_j and s'_{j+1} will share a common Steiner point. This "sharing" ensures that $\Pi'(s,t)$ is a connected path. In some special cases, s'_j may degenerate to a vertex but $\Pi'(s,t)$ still remains connected.

By applying Claim 3.1 to each segment of $\Pi'(s,t)$ we have: $\sum_{i=1}^{k} |s'_i| \leq \sum_{i=1}^{k}(w_{f_{s_i}} \cdot |s_i| + w_{f_{s_i}} \cdot \frac{|L|}{m+1})$, where $w_{f_{s_i}}$ denotes the weight of the face that s_i passes through. We can rewrite this as

$$\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + \frac{|L|}{m+1} \cdot \sum_{i=1}^{k}(w_{f_{s_i}}) \leq \|\Pi(s,t)\| + \frac{|L|}{m+1} \cdot k \cdot W$$

□

Theorem 3.2 *Using the fixed scheme, we can compute an approximation $\Pi'(s,t)$ of the weighted shortest path $\Pi(s,t)$ between two points s and t on a polyhedral surface $\mathcal{P}$ such that $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + W|L|$, where L is the longest edge of $\mathcal{P}$ and W is the maximum weight among all face weights of $\mathcal{P}$. Moreover, we can compute this path in $O(n^5)$ time.*

Proof: In Lemma 3.1 we have shown that there is a path $\Pi'(s,t)$ in G that approximates a shortest path $\Pi(s,t)$ on P. We use Algorithm 1 to compute a shortest path between the vertices corresponding to s and t in G. This will result in a path in G that has either the same cost as $\Pi'(s,t)$ or even less. Since any path in G can be mapped to a path on P, we obtain an approximate path on P. From Property 2.5 it follows that $\Pi(s,t)$ may have $\theta(n^2)$ segments. In Lemma 3.1, set $k = m = \theta(n^2)$, we obtain:

$$\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + W|L|.$$

Now we analyze the time complexity of the algorithm. In the fixed scheme, each edge of $\mathcal{P}$ contributes $O(n^2)$ graph vertices and each face contributes $O(n^4)$ graph edges. Algorithm 1 is then applied which runs in $O(n^5)$ time. (The path from G can be mapped to the path on $\mathcal{P}$ in the time proportional to the number of links in the path. Although in our implementation this additional step is avoided.) □

Note: The bound of the above theorem can be rewritten as $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + \frac{W|L_T|}{|E|}$, where $|L_T|$ is the sum of all edge lengths of $\mathcal{P}$ and $|E|$ denotes the number of edges of $\mathcal{P}$. This bound uses the average edge length as opposed to the pessimistic longest edge length. In the worst case, however, this bound is the same as stated in the theorem. This bound can actually be made tighter and written as $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + \frac{W|L_\Pi|}{|E_\Pi|}$ where $|L_\Pi|$ is the length of all edges that $\Pi(s,t)$ passes through and $|E_\Pi|$ is the number of edges that $\Pi(s,t)$ passes through. These bounds hold throughout the paper.

Interval Scheme:

In the fixed scheme, we made an assumption in our analysis that each edge crossed by the shortest path was of length $|L|$. In reality there may be many edges of $\mathcal{P}$ with small length compared to $|L|$. We can improve the fixed scheme by forcing the intervals between adjacent Steiner points on an edge to be of equal length, approximately $\frac{|L|}{m+1}$. As a result, we can reduce the number of Steiner points added per edge considerably. We call this scheme, the *interval scheme.* Figure 4 shows an example of how Steiner points are added to faces using a) the fixed scheme where $m = 7$ and b) the interval scheme. As can be seen in the figure, the interval scheme allows a significant decrease in the number of Steiner points placed while maintaining the same path accuracy as with the fixed scheme.

Since the maximum length of an interval is at most $\frac{|L|}{m+1}$, the proofs of Claim 3.1 and Lemma 3.1 still apply and hence Theorem 3.2 holds for the interval scheme. Although, the worst case analysis is the same for both schemes, the interval scheme typically reduces the number of Steiner points and edges that need to be processed.

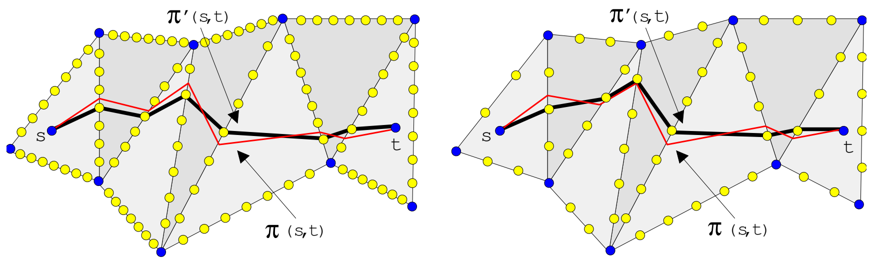


FIGURE 4. The difference in the layout of Steiner points (m=7) for a) the fixed scheme, b) the interval scheme.

Spanner Scheme:

We present here a scheme that improves upon the time complexity of the fixed and interval schemes; though the approximation achieved is not quite as good. The key idea is to compute a sparse face graph G_i for each of the faces of $\mathcal{P}$ instead of the dense graphs computed in the fixed or interval schemes. This is based on the notion of *spanners*. Clarkson [10] has shown that given a planar point set S, one can compute a sparse graph H, where vertices in H are the points in P, so that any segment joining two points in S can be approximated by a path joining the corresponding two vertices in H. A graph H is called a *β-spanner* of a point set S, if any segment joining two points in S can be approximated by a path joining the corresponding vertices in H of length at most β times the length of the segment, where $\beta > 1$ is a constant. The number and the placement of edges in H depends upon the desired accuracy bounds.

Now we briefly describe the computation of the spanners using the *cone method* of Clarkson [10]. Consider a face f_i of $\mathcal{P}$, and add Steiner points according to either the fixed or interval scheme on the boundary of f_i. Let S_i denote the set consisting of the Steiner points and the vertices of f_i. Let G_i be the required spanner graph on the points in S_i. (We also add edges between Steiner points that are adjacent on a common polyhedral edge.) Let $\mathcal{C}$ be the set of planar cones with apex at all $v_j \in S_i$ and the conical angle $\theta = \frac{\pi}{\rho}$ for an integer constant $\rho > 4$. For each $v_j \in S_i$ perform a radial sweep of the elements of S_i. During this sweep compute the element $v_{min} \in S$ that has minimal distance to v_j in each of the ρ cones and add an edge in the spanner graph G_i connecting the vertices corresponding to v_j and v_{min} (see Figure 5).

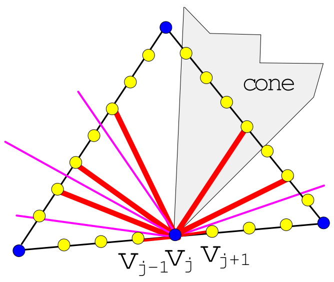


FIGURE 5. The spanner edges added from a vertex v_j with $\theta = 30°$.

Results of Clarkson [10], ensure that G_i is a β-spanner, where $\beta = \frac{1}{cos\theta - sin\theta}$. The graph consists of $O(m)$ vertices and edges (recall that $|S_i| = O(m)$). We compute a spanner for each face of $\mathcal{P}$ individually and then merge each $G_i, 1 \leq i \leq n$ to form the union G. We call this the *β-spanner scheme*. The following theorem summarizes the results for spanner scheme.

Theorem 3.3 *An approximate weighted shortest path $\Pi'(s,t)$ between two points s and t on a polyhedron $\mathcal{P}$ of n faces can be computed in $O(n^3 \log n)$ time such that $\|\Pi'(s,t)\| \leq \beta(\|\Pi(s,t)\| + W|L|)$, $\beta > 1$, where L is the longest edge of $\mathcal{P}$ and W is the maximum weight among all face weights of $\mathcal{P}$.*

Proof: In Claim 3.1 we can replace s'_j by an approximated path, say p_j, in G, where $|p_j| \leq \beta \cdot |s'_j|$, and use this value in Theorem 3.2, to obtain $\|\Pi'(s,t)\| \leq \beta(\|\Pi(s,t)\| + W|L|)$, $\beta > 1$.

Now we analyze the time complexity of the algorithm. We apply the β-spanner scheme to obtain G_i for each face f_i of $\mathcal{P}$ where $1 \leq i \leq n$. Using $O(n^2)$ Steiner points per edge, each subgraph G_i contains $O(n^2)$ graph vertices and each vertex has a constant degree. Hence, G_i contains $O(n^2)$ graph edges. The graph G_i is computed as follows. Observe that the vertices (or points) lie on the boundary of the triangular face, and they are at fixed intervals. For each vertex v_j all the cones ($O(1)$ in all) with apex at v_j can be computed in $O(1)$ time. Moreover the closest vertex to v_j in each of its cone can be computed in $O(1)$ time, by observing the relative location of vertices (or points) in the cone with respect to the perpendicular from v_j to the edges of f_i (we are omitting the technical details). This implies that each G_i can be computed in $O(n^2)$ time and G can be computed in $O(n^3)$ time. A shortest path in G can be computed by using Algorithm 1 and it runs in $O(n^3 \log n)$ time. □

Sleeve Based Schemes - Unweighted Case:

Here we describe how an approximated Euclidean (unweighted) path can be fine tuned to be of near-optimal length and sometimes be optimal. This technique is based on choosing an edge sequence based on an approximation $\pi'(s,t)$ using either the fixed or the interval scheme. We then determine a sleeve $\mathcal{S}$ by unfolding the faces along the edge sequence of $\pi'(s,t)$ and computing the shortest path $\pi_{\mathcal{S}}(s,t)$ that lies within this sleeve (using the algorithm of Lee and Preparata [24]). This path is then projected back onto $\mathcal{P}$ to obtain a refined approximation $\pi''(s,t)$. If the sleeve we choose happens to coincide with a shortest path edge sequence the final approximation is optimal.

In order to construct the sleeve, we choose a sequence of faces through which $\pi'(s,t)$ passes, i.e., for each segment s_i of $\pi'(s,t)$ we determine the face through which it passes. If s_i is not incident to a vertex of the face we append that face to our list of faces for the sleeve. If s_i passes through a vertex, say v, we must make a decision as to whether or not the edge sequence should go around v in the clockwise (CW) or counter-clockwise (CCW) orientation. The example of Figure 6 shows how this decision can affect the overall accuracy of $\pi_{\mathcal{S}}(s,t)$. If we choose the shaded faces, $\pi_{\mathcal{S}}(s,t)$ will not be as good as if we choose a CCW path around v_3 since $\pi(s,t)$ does not pass through all of the shaded faces.

We attempt to choose the good edge sequence (hence sleeve) by applying a simple heuristic for the special class of polyhedra: TINs. Let s_i and s_{i+1} be consecutive edges of $\pi'(s,t)$ such that their shared endpoint lies at a vertex v of $\mathcal{P}$. Determine the turn type (i.e., left, right or collinear) between the projections of s_i and s_{i+1} onto the XY plane. If it is a left turn, we chose a CW path around the vertex, otherwise we chose a CCW path. In Figure 6 we can see that all three vertices that were crossed result in left turns and we have chosen the CW path around each. Obviously, at v_3 the heuristic has chosen badly and we will never obtain an optimal path from $\pi'(s,t)$. However, we have observed that this heuristic performs well in practice on terrain data (see Section 4).

Given the unfolded sleeve, we compute the shortest path $\pi_{\mathcal{S}}(s,t)$ in the sleeve from s to t. Lee and Preparata [24] show that a *funnel* data structure can be used to represent a shortest path from any point s in a sleeve to a diagonal of the sleeve. The funnel consists of 4 components: a *tail* path, a *cusp* point and two convex chains of vertices forming the funnel borders.

The two convex chains meet at the cusp and are joined at their other end by the *funnel diagonal (or lid)* (see Figure 7).

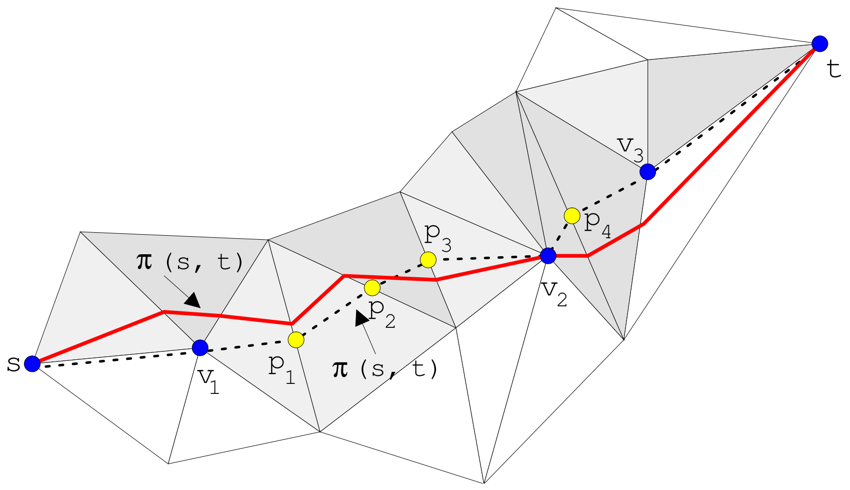


FIGURE 6. Choosing an edge sequence from the path $\pi'(s,t)$.

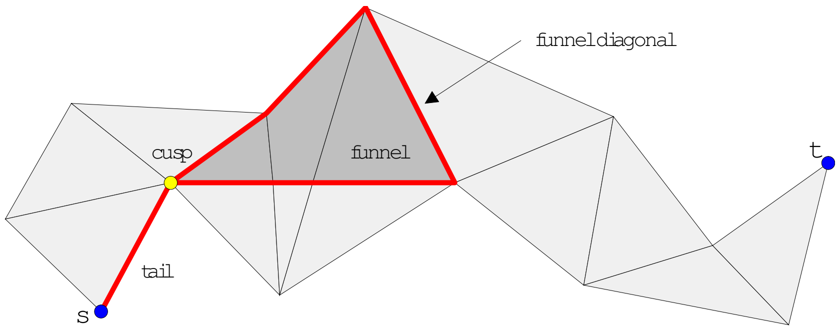


FIGURE 7. The funnel structure.

The algorithm maintains a funnel structure $\mathcal{F}$ from s while expanding by one sleeve diagonal at a time until t is reached. $\pi_{\mathcal{S}}(s,t)$ is then computed which has as most three pieces: 1) the tail path from s to $cusp(\mathcal{F})$, 2) a (possibly empty) sequence of edges along the left or right convex chain of $\mathcal{F}$ up to some point, say p and 3) a line segment from p (or $cusp(\mathcal{F})$ if piece 2 is empty) to t. We denote the consecutively computed funnels by $\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_n$ for an n-face sleeve such that $\mathcal{F}_{i+1}$ is formed from $\mathcal{F}_i$ through the expansion of one face (i.e., extending the funnel by one sleeve diagonal).

One problem is that the algorithm of Lee and Preparata [24] applies to a simple sleeve (non-intersecting). Since $\mathcal{P}$ is non-convex in general, $\mathcal{S}$ may be non-simple. We must show that even though the sleeve may be self-overlapping, it does not affect the correctness of the algorithm. We must also show that the resulting path is non-overlapping when projected back onto the surface of $\mathcal{P}$.

Property 3.1 *$\pi'(s,t)$ does not pass through a face of $\mathcal{P}$ more than once.*

Proof: The proof is by contradiction. Assume that $\pi'(s,t)$ is a shortest path in G that enters through an edge e of a face f_i at some Steiner point a and exits f_i at some Steiner point b. Now assume that $\pi'(s,t)$ enters f_i again, say through Steiner point c. Since a and c are both Steiner points on edges of the same face f_i, there exists a Steiner edge $\overline{ac}$ in G which is clearly a shortest path from a to c. Hence, the portion of $\pi'(s,t)$ from b to c cannot be a shortest path in G and we have a contradiction. □

Claim 3.2 *No two segments of $\pi_{\mathcal{S}}(s,t)$ lie in the same face of $\mathcal{P}$.*

Proof: By definition, none of $\mathcal{F}_i, 1 \leq i \leq n$ are self-intersecting. However, it is possible that a funnel may intersect the tail which it is connected to. Hence, $\pi_{\mathcal{S}}(s,t)$ may be non-simple due to the constraints that force it to pass through adjacent faces of $\mathcal{S}$. By construction of any funnel $\mathcal{F}_i$, any convex chain of segments from $cusp(\mathcal{F}_i)$ to $lid(\mathcal{F}_i)$ can be formed by line segments which pass through unique faces of $\mathcal{S}$. The left and right convex chains of $\mathcal{F}_i$ also consists of segments that lie in different faces of $\mathcal{S}$. Since the tail of any funnel is constructed by appending the convex chains of previous funnels, it is composed of segments that lie in different faces of $\mathcal{S}$. Therefore, $\pi_{\mathcal{S}}(s,t)$ contains segments that lie in different faces of $\mathcal{S}$ since it consists of pieces from $tail(\mathcal{F}_n)$, $chain(\mathcal{F}_n)$ and a path consisting of segments passing through the unique funnel faces. □.

Lemma 3.2 *The projected path $\pi''(s,t)$ is simple.*

Proof: The proof follows from Property 3.1, and Claim 3.2. □

Section 4 shows that these approximated paths are more accurate with this additional computation at a negligible increase in execution time. In our algorithm, as m increases the difference $|\pi'(s,t)| - |\pi(s,t)|$ decreases. For some value of m, $\pi'(s,t)$ will pass through the same edge sequence as $\pi(s,t)$. Since the sleeve computation unfolds the sleeve which contains both $\pi'(s,t)$ and $\pi(s,t)$, path $\pi''(s,t)$ will exactly match $\pi(s,t)$. Hence, in some instances, the edge sequence of the approximated path is identical to that of $\pi(s,t)$ and the sleeve computation produces the exact shortest path.

There is no efficient algorithm for computing shortest paths in weighted sleeves. Hence, we apply a different approach, namely that of continuously refining approximations based on a selected region of the terrain. To do this, we first compute a preliminary approximation $\pi'(s,t)$ as before using either the fixed or interval scheme. We then form $\mathcal{P}'$ as the union of all faces that $\pi'(s,t)$ intersects. If $\pi'(s,t)$ passes through a vertex v of $\mathcal{P}$, we include all faces incident to v. This union of faces forms a non-convex polyhedral surface $\mathcal{P}'$ but in general it does not form a closed polyhedron. We call this union a *buffer* around $\pi'(s,t)$. We then apply the approximation scheme on $\mathcal{P}'$ with an increased number of Steiner points per edge. As a result, we obtain a refined path. The refinement can be iterated allowing a trade-off between path accuracy and running time.

4 Experimental Results

In this section, we describe implementation issues, our testing procedures and experimental results. Due to their practical relevance, and in the context of our R&D [18], our experimental results are carried out on subclass of non-convex polyhedra: *Triangular Irregular Networks* (or TINs). A TIN is often constructed from a triangulated point set in the plane in which each point is assigned a height. In Geographic Information Systems, Cartography and related areas, shortest path problems arise frequently on terrains which are often modeled using TINs as shown in Figure 8. The algorithms presented in the previous section apply to any non-convex polyhedron. In addition to the tests explained here, we have verified that our implementation also works on 3D model data (non-convex polyhedra) which we obtained from the National Research Council of Canada (see [22]).

The section is organized as follows. In Section 4.1 we discuss implementation issues then discuss our test data suite and our procedure for testing. In Section 4.2 we then begin a discussion of the Euclidean results by first examining the path accuracy. We give a description as to how the additional sleeve computation, stretching of the TIN heights and spanner usage affects the accuracy. In Section 4.3 computation time is examined and comparisons are made with spanners. In Section 4.4, we describe the results for the weighted setting.

4.1 Implementation and Testing Procedures

Implementation Issues:

The implementation of our various algorithms involved implementing a variant of Dijkstra's algorithm, computing the unfolding of edge sequences

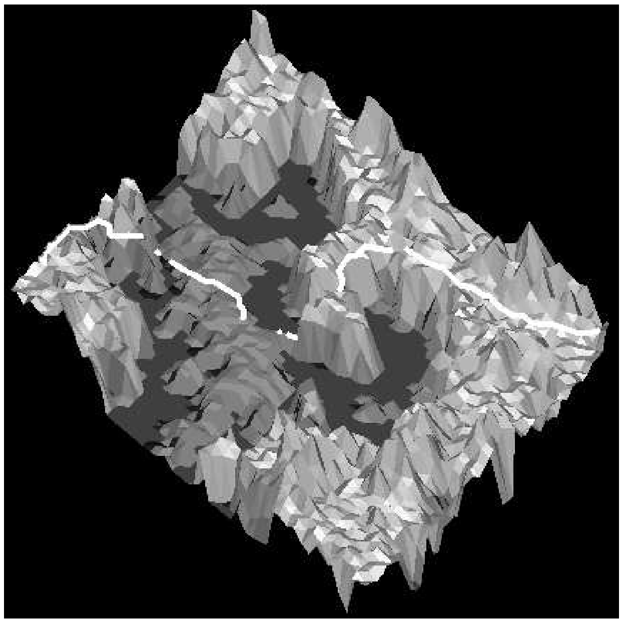


FIGURE 8. A weighted shortest path on a terrain in which traveling on water is expensive.

(only for refinement stage), computing the shortest path in a planar sleeve (subset of the algorithm to compute shortest paths in a polygon) and finally a modification to store an implicit graph representation.

The variation of Dijkstra's algorithm used was that of the well-known A* algorithm (see [15]), which incorporates a "distance-to-goal estimate" during the search. An additional weight was associated with each vertex, namely its Euclidean distance to the destination vertex. Then, during each iteration, we chose the vertex which minimized the sum of its cost from the source vertex plus the Euclidean distance to the destination vertex, over all candidate vertices.

Suri [38] points out that the Chen and Han algorithm [8] based on unfolding is sensitive to numerical problems, mainly due to the fact that 3D rotations are performed and errors accumulate along the paths and geometric structures computed. In our schemes the paths go through vertices or Steiner points (with the exception of the variation using the sleeve computation as its final step); thus reducing the chances of accumulating numerical errors. When doing the final step in which a sleeve in 3D is unfolded onto a plane, we compute an exact shortest path within the resulting sleeve. Although this is an exact path computation, it too is susceptible to numerical errors since the unfolding process may have generated discrepancies in the sleeve itself. We have shown however, that this final stage of refinement does provide significant improvement in the resulting accuracy.

Our implementation of Chen and Han's algorithm was designed to enable time and approximation quality comparisons with our algorithms. We took care of numeric stability issues as required. When using LEDA's data types [27], the running time of our implementation of Chen and Han's algorithm deteriorated drastically. Thus, by using LEDA (or similar) the comparison to our algorithms would have become worse for Chen and Han.

One final implementation issue was that of storage. Since our graphs are created based on evenly distributed points along an edge, it is possible to implicitly store the graph vertices and edges and only compute them as needed. Our initial implementation stored the entire graph and then during the execution of Dijkstra's algorithm, added all vertices to the priority queue as an initialization step. We later improved the implementation such that the graph vertices were still stored but the graph edges were not. In order to simplify the code, this improved implementation still added the vertices to the priority queue during the initial stage of Dijkstra's algorithm. It should be pointed out that a further improvement can be made by only adding the vertices to the priority queue as they are encountered during propagation. We ran tests which showed that this implicit edge representation improved the preprocessing time (i.e. graph construction) by a factor of two to three. The additional computations required to compute the edge lengths and weights during the running of Dijkstra's algorithm was negligible. Thus, we maintained almost the same query time.

Test Data:

One of the main difficulties in presenting experimental results is the lack of data, in general, and here of *benchmark* TINs. It is conceivable that different TIN characteristics could affect the performance of an algorithm. We have attempted to accommodate different characteristics by performing our tests with TINs that have different sizes (i.e. number of faces), height characteristics (i.e. smooth or spiky as modeled by accentuating the heights), and data sources (i.e. random or sampled from Digital Elevation Models (DEM)). Table 1 shows the attributes of the TINs that we tested. TINs with stretched heights were created by multiplying their heights by five. *

For the weighted domain, we used the same TINs and set the weight of

*Images of these TINs can be found in the [22]

No. of FACES	STRETCHED	DATA SOURCE
1,012	NO	DEM
1,012	YES	DEM
5,000	NO	RANDOM
5,000	YES	RANDOM
10,082	NO	DEM
10,082	YES	DEM
9,799	NO	DEM of partial Africa
9,788	NO	DEM of partial North America
9,944	NO	DEM of partial Australia
9,778	NO	DEM of partial Brazil
9,817	NO	DEM of partial Europe
9,690	NO	DEM of partial Greenland
10,952	NO	DEM of partial Italy
2,854	NO	DEM of partial Japan
9,839	NO	DEM of partial Madagascar
9,781	NO	DEM of partial Northwest Territories

TABLE 1. The data used for the experiments showing the TINs and their attributes.

each face to be the slope of the face. Thus, steeper faces have higher weight. Each edge of the TIN is given weight equal to the minimum of its adjacent faces. To determine if the results were biased due to our choice of weights, we ran additional tests in which the weights were chosen at random for each face.

Testing Procedure:

For each TIN, we computed a set of 100 random vertex pairs. We then tested each of the approximation schemes listed in Table 2. We give the legend id of each scheme as they appear in the graphs. For each test, we computed the path cost between each of the 100 vertex pairs and then obtained an “average path cost” for these pairs. We also computed the average computation time for the 100 pairs. The timing results presented here include the time required to compute the path itself, not just to produce the cost. The tests were performed in iterations based on the number of Steiner points per edge. Each scheme was tested for both weighted and unweighted scenarios (with the exception of the sleeve computations which were only computed in the unweighted case; a second approximation using a buffer was applied in the weighted case).

Graph Spanners:

In each test for graph spanners, we varied the degree of the cones from which the spanner graph was created from 1° to 40°. A 1° spanner can be

LEGEND ID	SCHEME	SLEEVE
INT	INTERVAL	NO
FIX	FIXED	NO
INTSLV	INTERVAL	YES
FIXSLV	FIXED	YES
x degree	SPANNER	NO

TABLE 2. The different approximation schemes.

viewed as the same graph which is obtained in the fixed or interval scheme.

4.2 Path Accuracy

We first examine the Euclidean shortest path problem. Figure 10 depicts the results of these tests for four terrains. The approximated path length rapidly converges to the actual path length (as computed using Chen and Han algorithm [8]). The graphs show that six Steiner points per edge suffice to obtain close-to-optimal approximations. The path accuracy observed is far better than the theoretical bound derived.

To understand this, recall that our theoretical worst-case analysis assumes that all edges intersected by the approximate path are long. In most applications this will be unlikely as short edges are common. (Long edges, if they are present, tend to be near the boundary and will therefore not be crossed by a shortest path.) We have examined edge-length histograms for all of our TIN data and show a typical histogram in Figure 9.

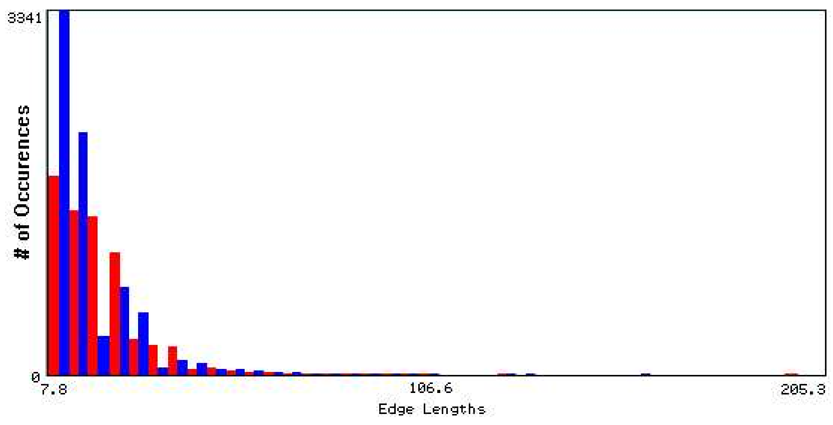


FIGURE 9. Histogram of edge lengths for one of our TINs.

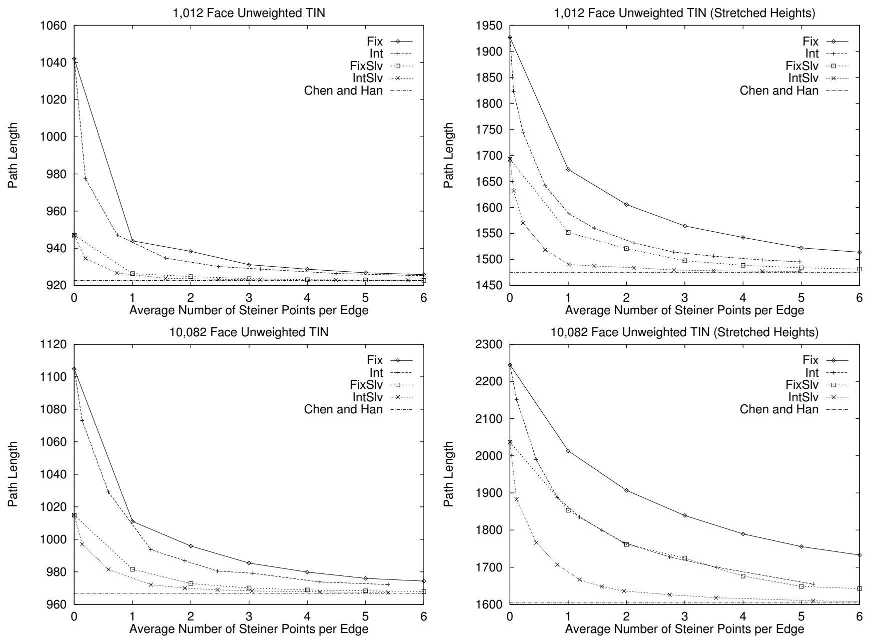


FIGURE 10. Graphs showing average path length for four selected terrains.

Sleeve Computation:

The graphs also illustrate that the additional sleeve computation helps to obtain even better approximations. We concluded that the best of our unweighted schemes is the interval scheme with the sleeve computation (IntSlv).

Effects of Stretching:

We now discus the effects on the accuracy due to stretching of the TIN. As mentioned, for this test, the heights of the vertices were multiplied by a factor of five. Figure 10 allows a comparison of accuracy for stretched versus unstretched TINs. In both instances, the approximate path length converges after only a few Steiner points per edge have been added. One may notice however, a slightly slower convergence for stretched input TINs. This is mainly due to the fact that Steiner points are placed further apart along the now longer edges. Therefore, it requires more Steiner points to reduce the interval size to that of the flatter TIN. The interval scheme performs better than the fixed scheme, since interval scheme favors placement of Steiner points on longer edges and longer edges are more likely to be crossed by the set of paths.

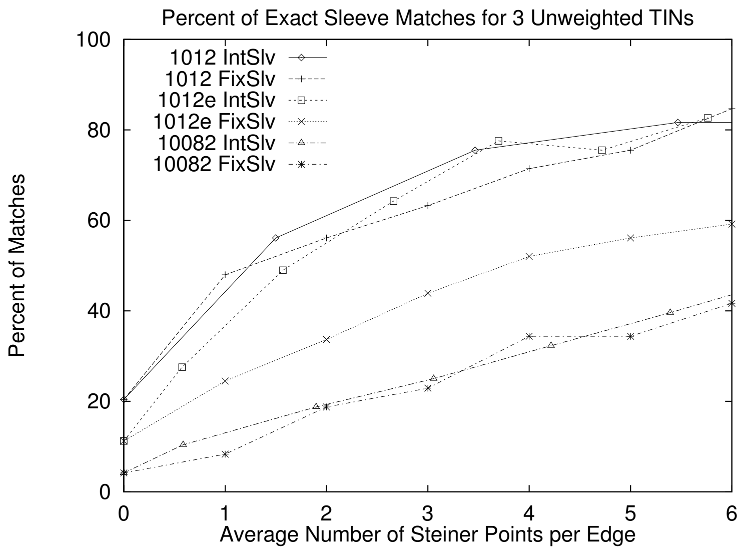
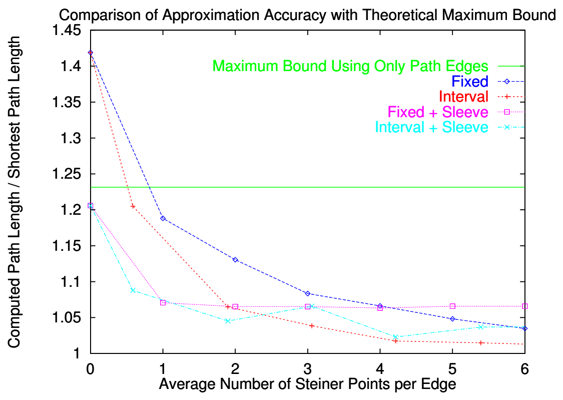


FIGURE 11. Graph showing the percentage of exact edge sequence matches for a 1012 face terrain, a 1012 face stretched terrain and a 10082 face terrain.

It is no surprise that the sleeve computation provides a significant improvement on the average path cost. Figure 11 shows the results of running sleeve match tests on a 1012 face terrain, a 1012 face stretched terrain and a 10082 face terrain. In each case, we determined the percentage of iterations that converged to the exact same edge sequence as the shortest path computed using our implementation of Chen and Han's algorithm. As can be easily seen, the smaller terrain had more sleeve matches and the stretched 1012 face terrain had less than the unstretched 1012 face terrain. In any case, we see that with six Steiner points per edge, an exact shortest path is obtained between 40% to 80% of the time.

We have proven that the worst case Euclidean approximation factor for the fixed and interval schemes is $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + W|L|$. The analysis made the pessimistic assumption that each edge crossed by the shortest path was of length $|L|$ and that all faces had maximum weight. We made the claim that terrains typically have many edges which are shorter than L and hence our worst case analysis is an over estimate (see also Figure 9). Just after the proof of Theorem 3.2, we have shown that this bound can be written in terms of the average length of edges through which $\Pi(s,t)$ passes as follows: $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + \frac{W|L_\Pi|}{|E_\Pi|}$. The graphs of Figure 12 compare this improved worst case theoretical accuracy with that of the produced accuracy for tests on the 10,082 face terrain using the fixed and interval schemes. The top graph depicts the maximum (i.e. worst case) error



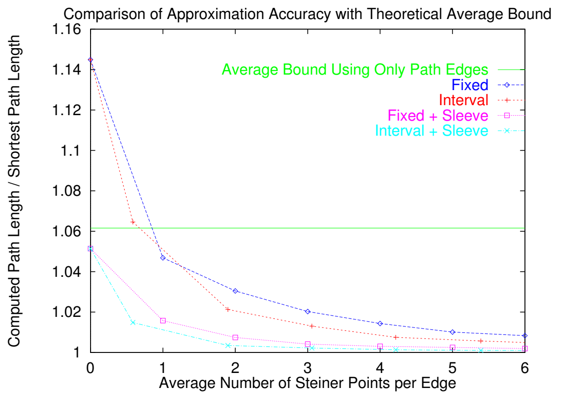


FIGURE 12. Graphs comparing the worst case (theoretical) accuracy with that of the produced accuracy for a 10,082 face terrain using the fixed and interval schemes (top: maximum error; bottom: average error).

obtained from the 100 paths tested; whereas the bottom graph depicts the average error obtained. The worst case and average case theoretical bounds using the computation of $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + W|L|$ are not shown on the graph, but are 3.220407 and 1.445150, respectively. The results confirm that the algorithm performs much better in practice than predicted by the theory.

Additional Terrains:

In order to verify that the algorithm would perform well on a variety of terrain data, we ran additional tests on 10 terrains which were constructed from DEM data from various parts of the world as shown in Table 1. Some of these results are depicted in Figure 13 and they verify that all terrains tested had similar accuracy and convergence behavior. Figure 14 shows the typical path accuracy as compared to that of Chen and Han's algorithm

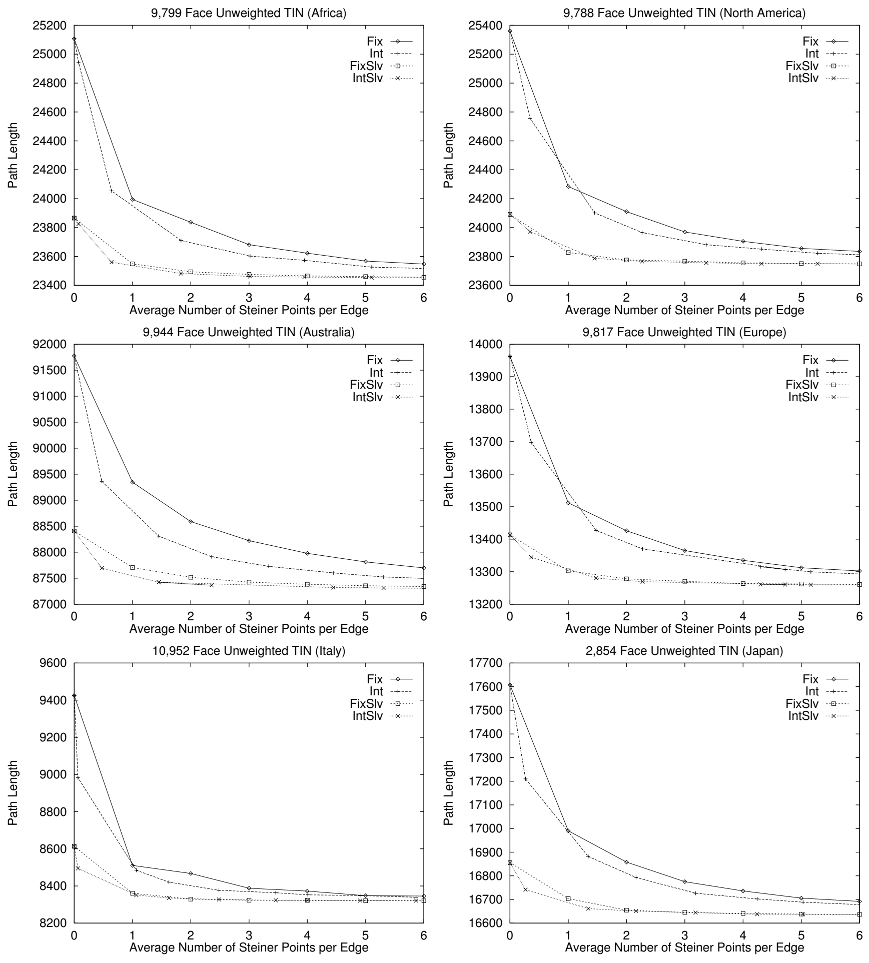


FIGURE 13. Graphs showing average path length for six real-data terrains.

for these additional TINs. The path converges quickly to near-optimal after only six Steiner points are added.

Spanners:

To determine the effects of using graph spanners on the path accuracy we ran several experiments whose set-up is described next. In each test, we varied the degree of the graph spanner cones from 1° to 40° degrees in increments of five. The 1° test essentially represents the graph without spanners allowing therefore a comparison of accuracy for schemes with and without spanners. Figure 15 depicts the results of computing shortest Euclidean paths on the 10,082 face terrain using the fixed and interval

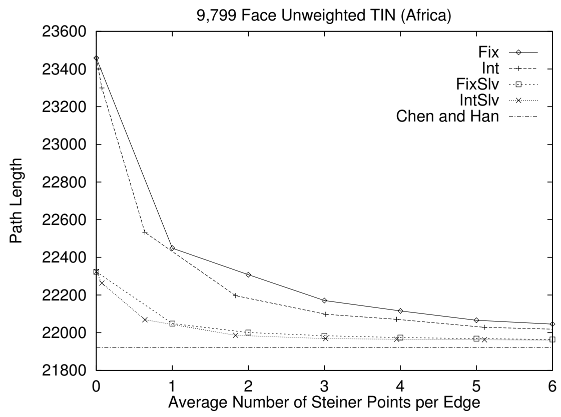


FIGURE 14. Graph comparing average path length between our schemes and that of Chen and Han for the Africa TIN.

schemes [†]. The graphs show the loss in path accuracy with increasing cone angle. As in the non-spanner schemes, the interval scheme converges quicker that the fixed scheme.

4.3 Computation Time

First we remark that we attempted to make a fair comparison between our schemes and Chen and Han's algorithm (for the unweighted case). For this, we used the same geometric primitives wherever possible.

In general, the algorithms' running times depend on the number of Steiner edges in G because we are invoking Algorithm 1. Since we are adding only a constant number of Steiner points on average per edge, the total number of edges is linear, and thus the running time of our algorithms becomes $O(n \log n)$.

Figure 16 depicts the actual run-time results of the Euclidean shortest path tests for variations of our approximation schemes on four of our original six terrains. As we ran tests for many pairs of points, we precomputed the graph G instead of building it every time on the fly. Then we measured the time it took to compute an approximate path for a query pair (source, destination). The differences between the individual schemes are not important in this graph. More importantly, the graph indicates that our algorithms are substantially faster than the one due to Chen and Han[8]. The main reason is that our algorithms do not require any complex data structures, nor do they perform expensive computations (such as 3D ro-

[†]The 100 (random) vertex-pairs considered here are different than those in Figure 10

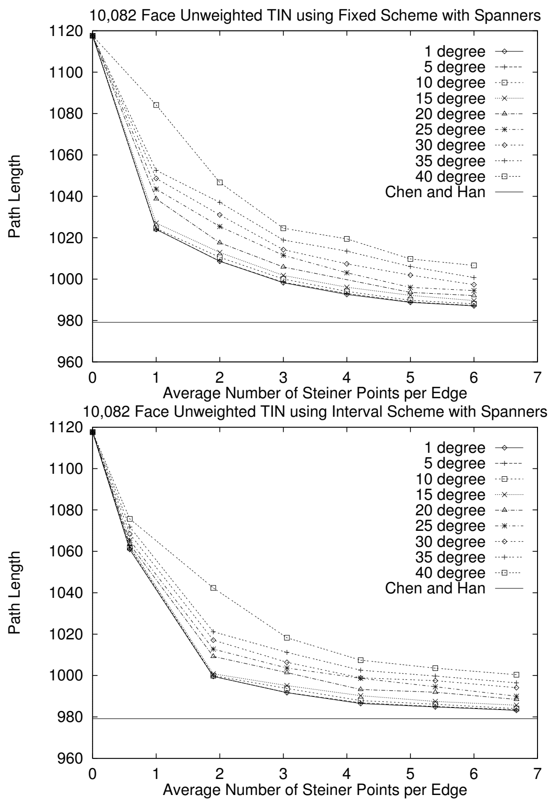


FIGURE 15. Graphs showing path accuracy obtained for a 10,082 face terrain using the fixed and interval schemes for a variety of spanner angles.

tation and unfolding). Due to the scale of the graph (resulting from the large time difference to Chen and Han), we cannot distinguish between the characteristics of the fixed and interval schemes.

Figure 17 depicts a graph showing the typical experimental run-time results obtained from our tests. Shown here are two sets of results, one for a 1,012 face TIN and one for the same TIN stretched by a factor of five. From the graph, we can see that the time required for the additional sleeve computation is negligible. In addition, there is very little difference between the fixed and interval schemes. This is mainly due to the fact that we are looking at the average number of Steiner points per edge. If we had chosen the X-axis of the graph to be "Maximum Number of Steiner points per Edge" we would see that the fixed scheme had a much slower running time than the interval scheme; however, this would have been an unfair comparison. Note as well, that the computation time increases for the stretched terrain. The timing results for six additional terrains is shown

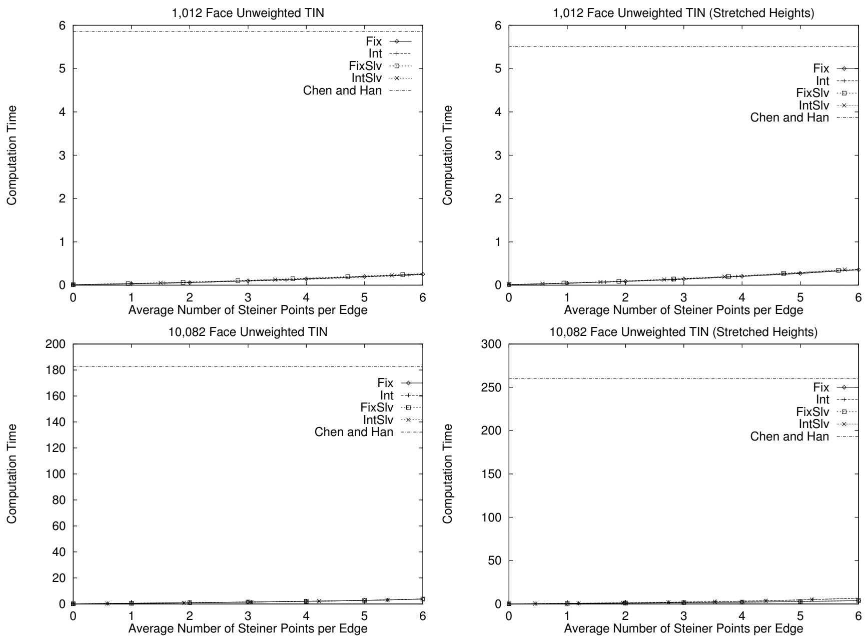


FIGURE 16. Graphs showing average computation time for four selected terrains.

in Figure 18 ; notice the similarly computed running time.

Graph Spanners:

Getting back to the spanner schemes, we can now examine their computation time. Although the path accuracy is reduced when the more sparse spanner graphs are used, the running time also decreases since there are less edges in the graph. Figure 19 shows the running time for the tests corresponding to the graphs of Figure 15. We can see that the spanners using the larger cone angle have better running time that those with smaller cone angles. In fact, we see that the graph shape goes from "quadratic" to "linear" since the number of graph edges becomes linear (times a constant) when spanners are used. We also see that there is little difference again between the fixed and interval schemes.

Clearly, the spanners allow a tradeoff between path accuracy and running time. From looking at these graphs, it is not immediately clear how the tradeoff can help make a decision as to whether or not to use a spanner and if so, what cone size to choose. The graphs of Figure 20 help determine the feasibility of the spanner scheme. The graphs show the path accuracy vs. computation time for the same data as Figure 15 and Figure 19. Here we can see which cone size provides the best path accuracy, when given

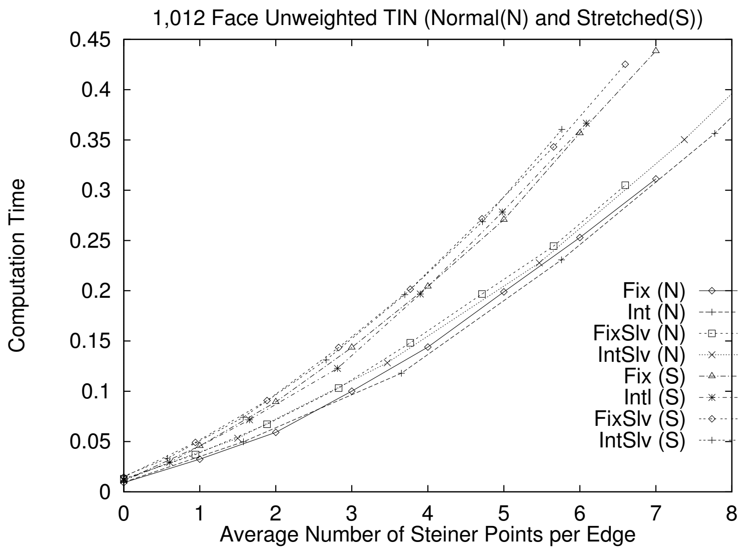


FIGURE 17. Graph showing the typical running time characteristics for the tested terrains using the fixed and interval schemes for normal(N) and stretched(S) terrains.

a certain computation time. For instance, if a path is required in 2 seconds, one can see that the 1 degree spanner provides the best accuracy for that amount of time and the 40 degree spanner provides the worst accuracy.

Examining the graphs more closely, we notice that the 5 and 10 degree spanners provide essentially the same accuracy as the 1 degree spanner. The 15 degree spanner also has similar accuracy. Figure 19 illustrates that the 10 and 15 degree spanner results can be 17% to 29% faster than the 1 degree spanner. One could therefore conclude that it may be worthwhile to implement the 5, 10 or 15 degree spanner since nearly the same accuracy can be obtained in less time. The graphs also indicate that if the allowable computation time is small, the more sparse spanner schemes do not perform well and should not be used. If the graphs could be extrapolated for larger run times, it is likely that at some point, the spanner schemes will provide a better accuracy vs. run time tradeoff. However, since good path accuracy is obtained in practice after only a few (constant) Steiner points are added per edge, the more sparse spanners become impractical.

4.4 Weighted Paths

For the weighted scenario, there is no implementation of an algorithm that determines the (true) shortest weighted path. This poses a problem when determining the accuracy of approximation. Another problem arises when

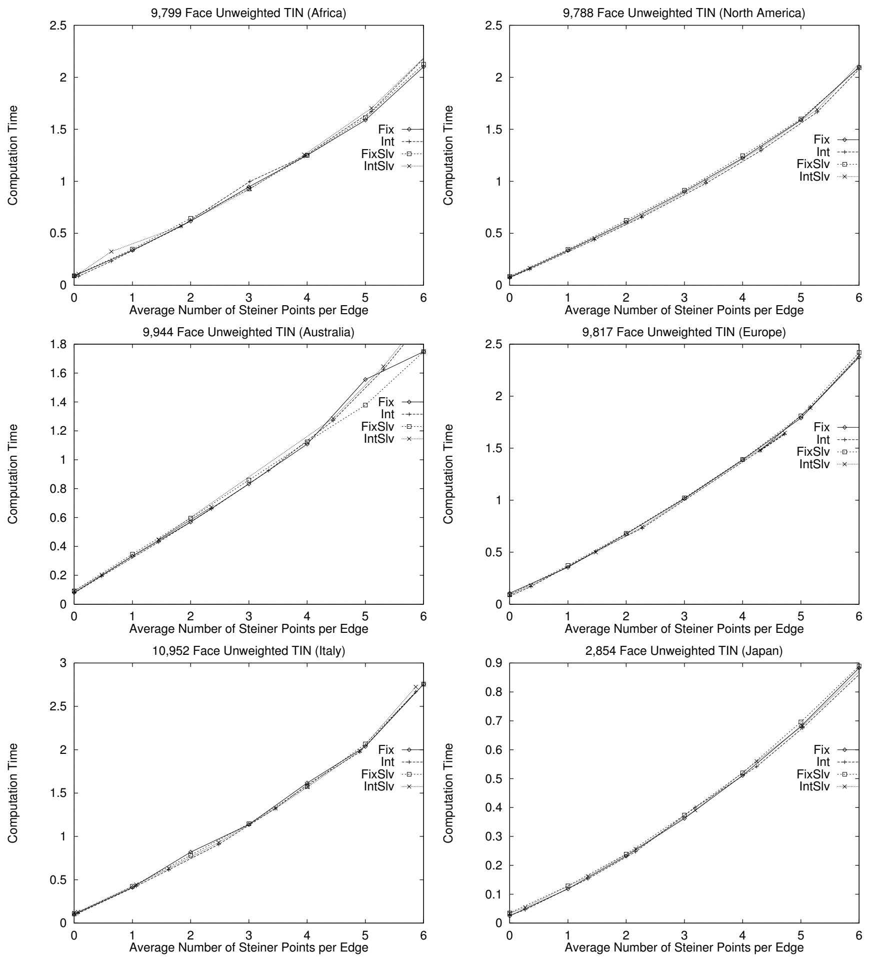


FIGURE 18. Graphs showing average computation time for six real-data terrains.

attempting to refine weighted paths by adding Steiner points. How many Steiner points are to be used during the second stage of approximation?

For our initial experiments, we used 20 Steiner points per edge in this second stage. However, with the interval scheme, even though the average number of Steiner points per edge is small (e.g., six) there may be many more than six Steiner points placed on longer edges. Therefore, the approximation accuracy may get worse during the second stage if we only allow a maximum of 20 Steiner points per edge.

Figure 21 shows the accuracy obtained through experimentation on weighted terrains with and without the second stage approximation (using 20

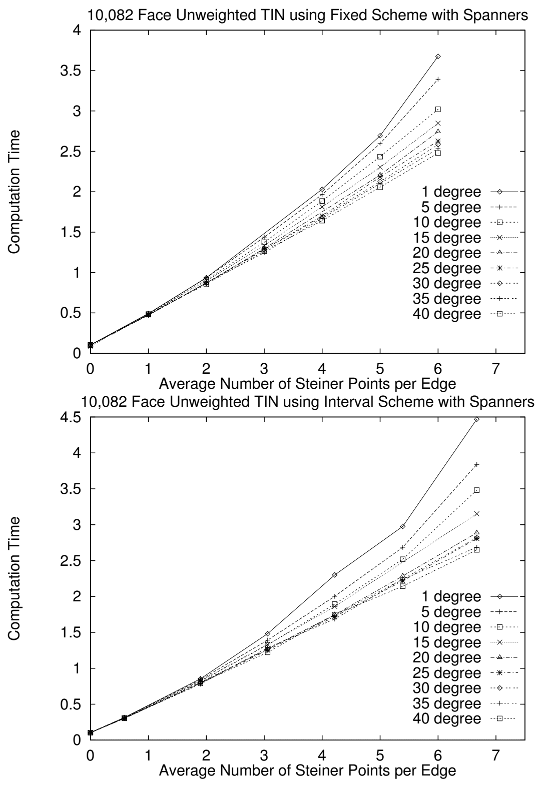


FIGURE 19. Graphs showing computation time obtained for a 10,082 face terrain using the fixed and interval schemes for a variety of spanner angles.

Steiner points per edge as mentioned). Like for the unweighted scenario, the path costs converge after only a few Steiner points have been added on each edge. Since the convergence is similar to that of the unweighted case, it is natural to conjecture that the cost of the paths converges to the actual weighted path cost. The second approximation based on the buffer technique provides an increase in accuracy, similarly. Since we use the same algorithm for unweighted and weighted scenarios, we obtained almost identical running time as shown in Figure 22. The second stage of approximation resulted in a significant increase in computation time. This is mainly due to the construction of a newly refined graph which is necessary for each query.

Time Independence from Weight Assignment:

The results just mentioned are based on terrains in which weights were assigned to each face based on slope. To show that this assignment of weights

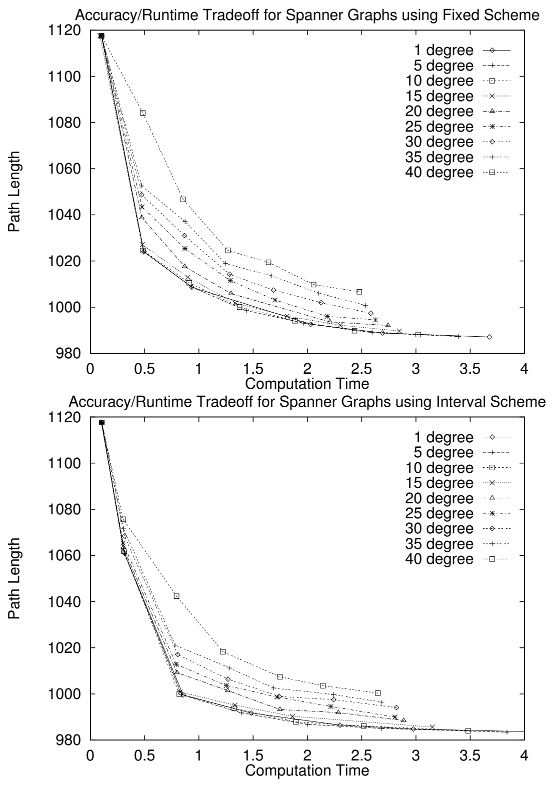


FIGURE 20. Graphs showing path accuracy vs. computation time obtained for a 10,082 face terrain using the fixed and interval schemes for a variety of spanner angles.

does not bias the results, we ran additional tests in which the weights were chosen at random for each face. [‡] The additional tests were performed on the normal and stretched versions of the 5000 face terrain with random heights and face weights. For these tests, we also changed the number of Steiner points used in the second stage of approximation. For the second stage approximation using the interval scheme, we increased the number of Steiner points per edge to produce intervals of approximately half of the size from the first stage. The fixed scheme tests were carried out as before with 20 Steiner points per edge.

Figure 23 compares the weighted path costs between terrains with slope weights and random weights. The slope weight tests were redone here with the new calculation for the second stage approximation. As can be seen

[‡]The standard gnu-c function drand48() was used to generate the random weights.

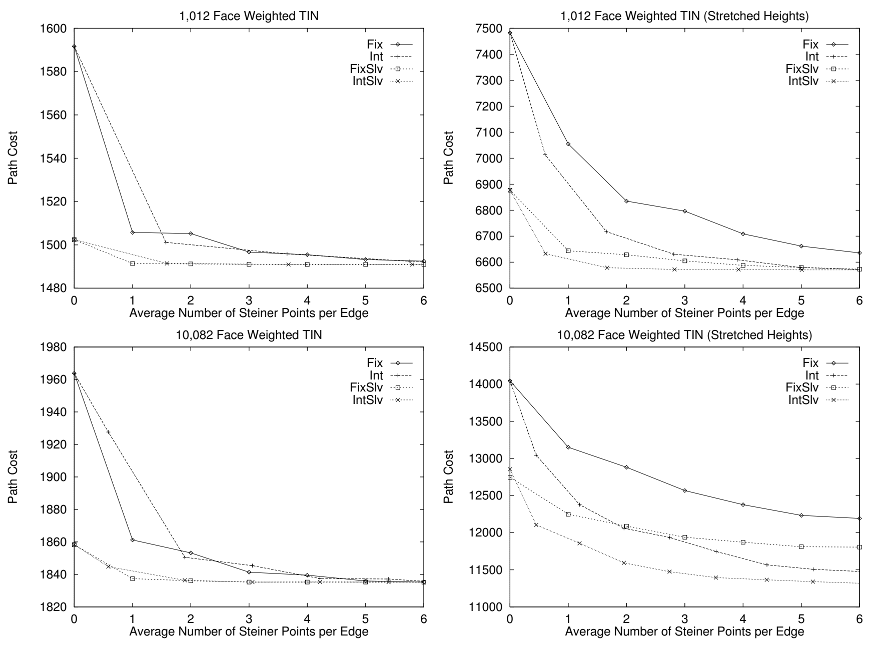


FIGURE 21. Graphs showing average path cost for four selected weighted terrains.

by the graphs, there is very little difference between the shape and convergence behavior between the two weighted scenarios. The timing graphs for these additional tests also showed very little difference.

The similar characteristic shape and scale of the graphs as compared to the unweighted case imply that our algorithm is not sensitive to the weights of the faces.

5 Shortest Path Queries

In this section, we describe algorithms for computing paths between arbitrary query points on the polyhedral surface. We consider two variations of the shortest path query problem: 1) fixed source, and 2) arbitrary queries. The preprocessing time involves running our algorithm from the previous section and building a structure for point location.

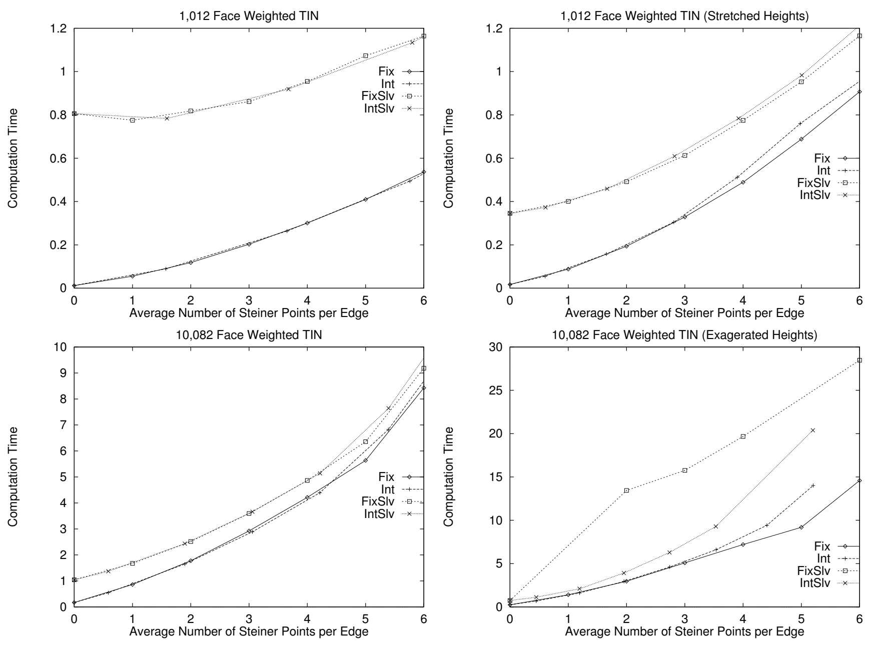


FIGURE 22. Graphs showing average computation time for four selected weighted terrains.

5.1 Fixed Source

Let us consider a fixed source (w.l.o.g. assumed to be a vertex) and allow arbitrary destination queries. Since the source is fixed, we can run Algorithm 1 as a preprocessing step. Now, if the destination query point is a vertex or a Steiner point, the cost from s to t is precomputed and thus can be reported in constant time. The path can be reported in time proportional to the number of its segments. If the destination t is not a vertex, face f_t which contains t must first be located. Then the predetermined shortest paths are used to the vertices and/or Steiner points of f_t. An approximation for $\Pi'(s,t)$ can be constructed in one of two ways through concatenation of:

1. $\Pi'(s,q_t), \overline{q_t t}$, where q_t is a Steiner point of f_t, or
2. $\Pi'(s,v_t), \overline{v_t t}$, where v_t is a vertex of f_t.

The choice of going through a vertex or a Steiner point allows for a tradeoff between path accuracy and query time. The following theorem applies to the two construction strategies above:

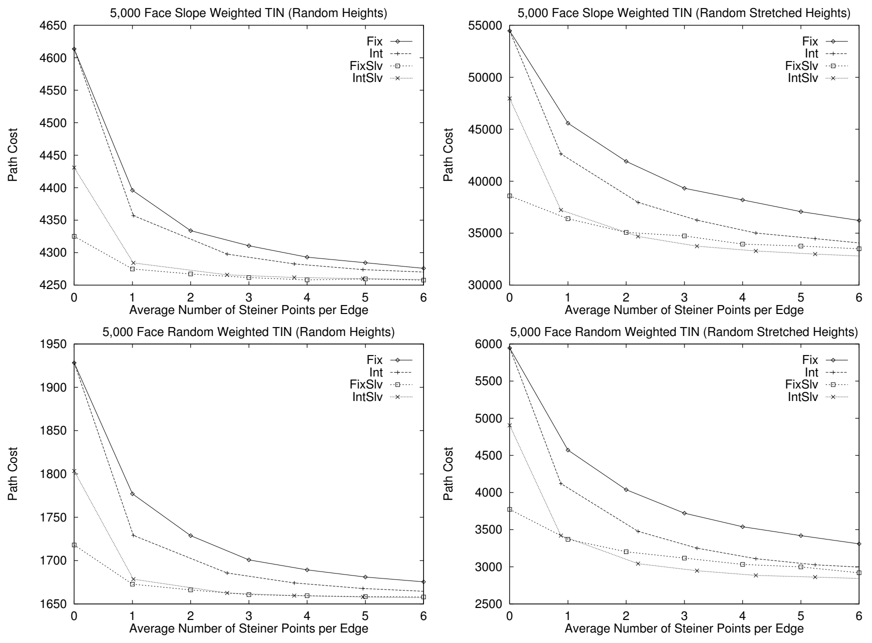


FIGURE 23. Graphs comparing the accuracy of weighted approximations for 2 weight scenarios: slope weights and random weights.

Theorem 5.1 *A given polyhedron $\mathcal{P}$ can be preprocessed, for a given source vertex s such that a path $\Pi'(s,t)$ can be computed to a query point t on $\mathcal{P}$ in*
1) $O(m + \log n)$ time such that $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1 + \frac{1}{m+1})W|L|$
2) $O(\log n)$ time such that $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1 + \frac{2}{\sqrt{3}})W|L|$.

Proof: The proof is given separately for the two construction methods.
1) The query time follows from Observation 2.1 applied to n faces with an additional $O(m)$ time for computing the minimum of $O(m)$ precomputed paths from s to Steiner points of f_t. Now consider the accuracy of the path so obtained. Let b be the point at which $\Pi(s,t)$ enters f_t and let b' be the Steiner point, closest to b, at which $\Pi'(s,t)$ enters f_t (as per our scheme). Using Theorem 3.2 we can derive the following bound on the cost of $\Pi'(s,t)$ to be $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1 + \frac{1}{m+1})W|L|$.

2) For the analysis, we will assume that v_t is the vertex of f_t that is closest to t and let L be the longest edge among the edges of f_t. It can be shown that $|\overline{v_t t}| \leq \frac{|L|}{\sqrt{3}}$, since the distance $|v_t t|$ is maximized when f_t is an equilateral triangle and t is the point of intersection of the perpendicular bisectors of each of the sides. Now from Theorem 3.2 it follows that $\|\Pi'(s,v_t)\| \leq \|\Pi(s,v_t)\| + W|L|$ Applying this to the constructed path $\Pi'(s,t)$ we ob-

tain: $\|\Pi'(s,t)\| = \|\Pi'(s,v_t)\| + \|\overline{v_t t}\| \leq \|\Pi(s,t)\| + \|\overline{v_t t}\| + W|L| + \|\overline{v_t t}\| \leq \|\Pi(s,t)\| + (1+\frac{2}{\sqrt{3}})W|L|$. □

5.2 Arbitrary Queries

Now consider the case in which both s and t are arbitrary query points. Preprocess the polyhedron by computing the shortest path from each vertex of $\mathcal{P}$ to all Steiner points. We can then construct a path in one of two ways as for the fixed source query problem. Let f_s and f_t be the faces containing s and t, respectively. Let q_s (respectively q_t) be a Steiner point of f_s and let v_s (respectively v_t) be a vertex of f_s (respectively f_t). We can construct $\Pi'(s,t)$ in one of the following ways: (A) $\overline{sq_s}, \Pi'(q_s,q_t), \overline{q_t t}$ (B) $\overline{sv_s}, \Pi'(v_s,q_t), \overline{q_t t}$ (C) $\overline{sq_s}, \Pi'(q_s,v_t), \overline{v_t t}$ (D) $\overline{sv_s}, \Pi'(v_s,v_t), \overline{v_t t}$.

In order to make the best choice of q_s, q_t, v_s and/or v_t, we compute the path for all possible pairs of Steiner points and/or vertices of f_s and f_t. In the first case, this takes $O(m^2)$ time as there are $O(m)$ possibilities for both q_s and q_t. For the second (respectively third) case, we need to check $O(m)$ possibilities for q_t (respectively q_s) and three possibilities for v_s (respectively v_t). This takes $O(m)$ time. Since there are at most nine combinations to check, the last case can be carried out in constant time.

Theorem 5.2 *A given polyhedron $\mathcal{P}$ can be preprocessed such that a path $\Pi'(s,t)$ can be computed between two query points s and t on $\mathcal{P}$ in*
1) $O(\log n + m^2)$ time such that $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1+\frac{2}{m+1})W|L|$,
2) $O(\log n + m)$ time such that $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1+\frac{2}{\sqrt{3}}+\frac{1}{m+1})W|L|$,
3) $O(\log n)$ time such that $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1+\frac{4}{\sqrt{3}})W|L|$.

Proof: The proof is similar to Theorem 5.1 where 1) uses construction method (A), 2) uses construction method (B) or (C), and 3) uses construction method (D). □

6 Conclusion

Shortest path problems belong to a class of geometric problems that are fundamental and of significant practical relevance. While realistic shortest path problems frequently arise in applications where the cost of travel is not uniform over the domain, the time, space and implementation complexities of existing algorithms even for the planar case are extremely high which motivates our study of approximation algorithms. Our experimental results show that high-quality approximations can be obtained with very good run-times. More precisely, we have provided empirical results showing that typical terrain data requires only a few (constant) Steiner points per edge. This reduces the running time to $O(n \log n)$ in practice which is orders of magnitude smaller than the best known exact shortest path algorithm. The solutions are simple and of practical value.

We also theoretically establish bounds on the approximation quality and give worst-case bounds on the run-time of our algorithms. For the unweighted scenario, we compared our accuracy to that of Chen and Han [8] and gave results indicating that our algorithm performs up to 50 times faster with a minimum observed speedup of 14 times and produces nearly identical path results. We claim that our algorithm is efficient w.r.t. accuracy versus running time and is simple to implement. Our algorithm is of particular interest also for the case of queries with unknown source and destination.

A different class of approximation is an ϵ-approximation where we would like to find a path such that $\|\Pi'(s,t)\| \leq (1+\epsilon)\|\Pi(s,t)\|$ for some $\epsilon > 0$. We can prove that our vertex-to-vertex schemes are ϵ-approximations. However, the number of Steiner points required and preprocessing cost could be high, making this scheme of little practical value. The worst case bounds, when compared in the integer coordinate model, match that of Mata and Mitchell [26]. An improved ϵ-approximation algorithm is described in [5]. Furthermore, the ϵ-approximation scheme for queries is discussed in [3]. In [4] the determination of an isotropic paths is discussed. The study of anisotropic paths adds further realism by taking into consideration the direction of travel on each face thereby e.g., eliminating paths that are too steep for vehicles to travel and preventing the vehicles from turning over.

In order to further verify the schemes, we ran tests on different sources of data: TINs created from DEMs, random data and general polyhedral models which were based on 3D range finder data. We found that all data resulted in good performance for the schemes presented here. We did observe however, that the flatter terrain data provided slightly better results than the more "spiky" terrains. In addition to the schemes presented here, we have designed additional Steiner placement and interconnection schemes. Tests have shown that these other schemes did not perform well [22]. It remains an open problem as to whether or not it is advantageous to place Steiner points within the interior of the polyhedral faces.

Several other problems remain open. For the Euclidean shortest path problem, our implementation of Chen and Han's algorithm allowed us to compare the path accuracies obtained by applying our schemes to the true value. For the weighted shortest path problem, no algorithm exists. It remains thus open to: (1) provide an algorithm and its implementation for the weighted shortest path problem and (2) establish whether our conjecture that the accuracy of our weighted shortest paths converges to the true value is correct. (The curves for weighted and unweighted look very similar suggesting correctness.)

The preprocessing time taken to answer queries efficiently, increases the internal storage space requirements and, as also observed by Mata and Mitchell [26], it remains an open question as to how the shortest path algorithms fare experimentally in external memory settings. We are presently investigating methods to overcome the storage issues these include also the use of parallel computing [22]. Furthermore, a data-base of geometric objects including e.g., benchmark TINs, should be constructed and made available to the community. This would aid in providing a good basis for experimental comparisons between algorithms and code.

Acknowledgments:

The authors would like to thank Stefan Schirra for his help in analyzing our code with respect to numerical stability issues. We would also like to thank Paola Magillo for supplying us with TINs as well as Gerhard Roth from the NRC for providing us with general 3D polyhedral model data. Finally, we would like to thank the referees for providing input resulting in this version of the paper.

7 References

[1] P.K. Agarwal, S. Har-Peled, M. Sharir, and K.R. Varadarajan, “Approximating Shortest Paths on a Convex Polytope in Three Dimensions”, *Journal of the ACM*, Vol. 44, 1997, pp 567-584.

[2] P.K. Agarwal and K.R. Varadarajan, “Approximating Shortest Paths on a Polyhedron”,*Proceedings of the 38th IEEE Symp. on Foundations of Computer Science*, 1997.

[3] L. Aleksandrov, M. Lanthier, A. Maheshwari and J.-R. Sack, “An ϵ-Approximation Algorithm for Weighted Shortest Path Queries on Polyhedral Surfaces”, *14th European Workshop on Computational Geometry*, Barcelona, Spain, March 1998, pp. 19-21.

[4] M. Lanthier, A. Maheshwari, and J.-R. Sack, ”Shortest Anisotropic Paths on Terrains”, in Proc. ICALP’99, Prague, LNCS 1644, , pp. 524-533, 1999.

[5] L. Aleksandrov, M. Lanthier, A. Maheshwari and J.-R. Sack, “An ϵ-Approximation Algorithm for Weighted Shortest Paths on Polyhedral Surfaces”, *6th Scandinavian Workshop on Algorithm Theory*, Stockholm, Sweden, July 1998.

[6] A. Baltsan and M. Sharir, “On the Shortest Paths Between Two Convex Polyhedra”, *Journal of the ACM*, **35**, January 1988, pp. 267-287.

[7] J. Canny and J. H. Reif, "New Lower Bound Techniques for Robot Motion Planning Problems", *Proceedings of the 28th IEEE Symp. on Foundations of Computer Science*, 1987, pp. 49-60.

[8] J. Chen and Y. Han, "Shortest Paths on a Polyhedron", *International Journal of Computational Geometry and Applications*, Vol. 6, 1996, pp. 127-144.

[9] J. Choi, J. Sellen and C.K. Yap, "Approximate Euclidean Shortest Path in 3-Space", *International Journal of Computational Geometry and Applications*, Vol. 7, No. 4, 1997, pp. 271-295.

[10] K.L. Clarkson, "Approximation algorithms for shortest path motion planning", *Proc. 19th Annual ACM Symp. Theory of Computing*, 1987, pp. 56-65.

[11] E.W. Dijkstra, "A Note on Two Problems in Connection with Graphs", *Numerical Mathematics 1*, 1959, pp.269-271.

[12] M.L. Fredman and R.E. Tarjan, "Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms", *J. ACM*, **34**(3), 1987, pp.596-615.

[13] C. Gold, "The Practical Generation and Use of Geographic Triangular Element Data", in *Harvard Papers on Geographic Information Systems*, 5, (1978).

[14] M. Goodrich and R. Tamassia, "Dynamic Trees and Dynamic Point Location", *Proceedings of the 23rd Annual Symposium on Theory of Computing*, 1991, pp. 523-533.

[15] P.E. Hart, N.J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions on System Science and Cybernetics*, SSC-4(2), 1968, pp. 100-107.

[16] S. Har-Peled, M. Sharir, and K.R. Varadarajan, "Approximating Shortest Paths on a Convex Polytope in Three Dimensions", *Proc. 12th Annual Symp. on Computational Geometry*, Philadelphia, PA, 1996, pp. 329-338.

[17] J. Hershberger and S. Suri, "Practical Methods for Approximating Shortest Paths on a Convex Polytope in $\Re^3$", *Proc. of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1995, pp. 447-456.

[18] D. Hutchinson, M. Lanthier, A. Maheshwari, D. Nussbaum, D. Roytenberg, J.-R. Sack, "Parallel Neighbourhood Modeling", *Proc. of the 4th ACM Workshop on Advances in Geographic Information Systems*,Minnesota, 1996, pp. 25-34.

[19] P. Johansson, "On a Weighted Distance Model for Injection Molding", Linköping Studies in Science and Technology, Thesis no. 604 LiU-TEK-LIC-1997:05, Division of Applied Mathematics, Linköping University, Linköping, Sweeden, February 1997.

[20] D.G. Kirkpatrick, "Optimal Search in Planar Subdivisions", *SIAM Journal of Computing*, Vol. 12, No. 1, 1983, pp. 28-35.

[21] M. Lanthier, A. Maheshwari and J.-R. Sack, "Approximating Weighted Shortest Paths on Polyhedral Surfaces", *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, 1997, pp. 274-283.

[22] M. Lanthier, "Shortest Path Problems on Polyhedral Surfaces", *Ph.D. Thesis in progress*, School of Computer Science, Carleton University, Ottawa, Canada, 1999.

[23] M. Lanthier, A. Maheshwari and J.-R. Sack, "Approximating Weighted Shortest Paths on Polyhedral Surfaces", video in this proceedings, *13th Annual ACM Symposium on Computational Geometry*, June 1997.

[24] D.T. Lee, and F.P. Preparata, "Euclidean Shortest Paths in the Presence of Rectilinear Barriers", *Networks*, **14**, 1984, pp. 393-410.

[25] L.A. Lyusternik, "Shortest Paths: Variational Problems", Macmillan, New York, 1964.

[26] C. Mata and J. Mitchell, "A New Algorithm for Computing Shortest Paths in Weighted Planar Subdivisions", *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, 1997, pp. 264-273.

[27] K. Mehlhorn and S. Näher, *LEDA: a platform for combinatorial and geometric computing*, Commun. ACM, **38**, 1995, pp. 96–102.

[28] J.S.B. Mitchell, D.M. Mount and C.H. Papadimitriou, "The Discrete Geodesic Problem", *SIAM Journal of Computing*, **16**, August 1987, pp. 647-668.

[29] J.S.B. Mitchell and C.H. Papadimitriou, "The Weighted Region Problem: Finding Shortest Paths Through a Weighted Planar Subdivision", *Journal of the ACM*, **38**, January 1991, pp. 18-73.

[30] J. O'Rourke, S. Suri and H. Booth, "Shortest Paths on Polyhedral Surfaces", extended abstract, Dept. Electrical Engineering and Computer Science, Johns Hopkins University, Baltimore, Maryland, September 1984.

[31] M. Overmars and A. van der Stappen, "Range Searching and Point Location among Fat Objects", *Journal of Algorithms*, Vol. 21, 1996, pp. 629-656.

[32] C.H. Papadimitriou, "An Algorithm for Shortest Path Motion in Three Dimensions", *Information Processing Letters*, **20**, 1985, pp. 259-263.

[33] T. K. Peucker, "Data Structures for Digital Terrain Modules," in *Harvard Papers on Geographic Information Systems*, 5, (1978).

[34] T. K. Peucker, R. J. Fowler, J. J. Little, and D. M. Mark, "The Triangulated Irregular Network," in *Proceedings DTM Symposium American Society of Photogrammatry - American Congress on Survey and Mapping,* pp. 24-31, (1978).

[35] F.P. Preparata, "A New Approach to Planar Point Location", SIAM Journal of Computing, Vol. 10, No. 3, 1981, pp. 473-482.

[36] M. Sharir, "On Shortest Paths Amidst Convex Polyhedra", *SIAM Journal of Computing*, **16**, 1987, pp. 561-572.

[37] M. Sharir and A. Schorr, "On Shortest Paths in Polyhedral Spaces", *SIAM Journal of Computing*, **15**, 1986, pp. 193-215.

[38] S. Suri, personal communication, 1996.

[39] Paradigm Group Web-page, School of Computer Science, Carleton University, http://www.scs.carleton.ca/˜ gis.

Appendix - Proof of Theorem 3.1

We present here a proof of Theorem 3.1. The proof makes use of two properties, a claim and a lemma which are stated first.

Property 7.1 *An edge of $\mathcal{P}$ cannot have a weight greater than its adjacent faces.*

Property 7.2 *Let ABC be a triangle. Let X be a point on $\overline{AC}$ and Y be a point on $\overline{AB}$ such that $|\overline{AX}| = \delta|\overline{AC}|$ and $|\overline{AY}| = \delta|\overline{AB}|$ for some $0 < \delta < 1$. Then $\overline{XY}$ is parallel to $\overline{BC}$. Furthermore, $|\overline{XY}| = \delta|\overline{BC}|$.*

Claim 7.1 *Let ABC be a triangle such that $\angle CAB = \theta$. Now let $\overline{XY}$ be a line segment with endpoints X and Y lying on $\overline{AC}$ and $\overline{AB}$, respectively. Given some constant $0 < \delta < 1$, the following is true:*

i) If $|\overline{AX}| \geq \delta|\overline{AC}|$ and $|\overline{AY}| \leq \delta|\overline{AB}|$ then $|\overline{AC}| \leq \frac{|\overline{XY}|}{\delta \sin\theta}$

ii) If $|\overline{AX}| \leq \delta|\overline{AC}|$ and $|\overline{AY}| \geq \delta|\overline{AB}|$ then $|\overline{AB}| \leq \frac{|\overline{XY}|}{\delta \sin\theta}$ (symmetrical to i above)

iii) If $|\overline{AX}| \geq \delta|\overline{AC}|$ and $|\overline{AY}| \geq \delta|\overline{AB}|$ then $|\overline{BC}| \leq \frac{|\overline{XY}|}{\delta}$

Proof: Let P be the point on line AB such that $|\overline{XP}|$ is minimized (i.e., $\overline{XP}$ is a perpendicular of line AB which may or may not lie on segment $\overline{AB}$). Define Y' to be the point on segment $\overline{AB}$ such that $|\overline{XY'}|$ is minimized ($Y' = A, P$, or B). Since $|\overline{XY'}| \leq |\overline{XY}|$, it is sufficient to prove the claim for $|\overline{XY'}|$. Lastly, let D and E be the points on $\overline{AC}$ and $\overline{AB}$, respectively, such that $|\overline{AD}| = \delta|\overline{AC}|$ and $|\overline{AE}| = \delta|\overline{AB}|$. (See Figure 24 where δ is set to $\frac{1}{2}$).
i) (case ii is symmetrical) By definition $|\overline{XY}| \geq |\overline{XY'}| \geq |\overline{XP}|$. If $Y' = A$ then we are done since by assumption $|\overline{AX}| \geq \delta|\overline{AC}|$. Hence $|\overline{AC}| \leq \frac{|\overline{XY}|}{\delta}$. Now if $Y' = P$, then $\sin\theta = \frac{|\overline{XP}|}{|\overline{AX}|}$ and from our assumption that $|\overline{AX}| \geq \delta|\overline{AC}|$ we have $|\overline{AC}| \leq \frac{|\overline{XP}|}{\delta\sin\theta} \leq \frac{|\overline{XY}|}{\delta\sin\theta}$.
iii) From Property 7.2 $|\overline{DE}| = \delta|\overline{CB}|$. Due to the constraint that $|\overline{AX}| \geq \delta|\overline{AC}|$ and $|\overline{AY}| \geq \delta|\overline{AB}|$ then we can infer that $|\overline{XY}| \geq |\overline{ED}| = \delta|\overline{CB}|$. Therefore, $|\overline{CB}| \leq \frac{|\overline{XY}|}{\delta}$. □

Lemma 7.1 *A Euclidean shortest path $\pi(s,t)$ between two vertices s and t of $\mathcal{P}$ can be approximated by a path $\pi'(s,t)$ consisting only of edges of $\mathcal{P}$ such that $|\pi'(s,t)| \leq \frac{2}{\sin\theta_{min}}|\pi(s,t)|$.*

Proof: We will show that for each segment s_i of $\pi(s,t)$ there is a corresponding segment s_i' of $\pi'(s,t)$ which represents an edge of $\mathcal{P}$ and is bounded by $\frac{2}{\sin\theta_{min}}|s_i|$. Let $s_i = \overline{xy}$ and $s_{i+1} = \overline{yz}$ be two consecutive segments of $\pi(s,t)$. Let x,y and z lie on edges e_x, e_y and e_z, respectively.

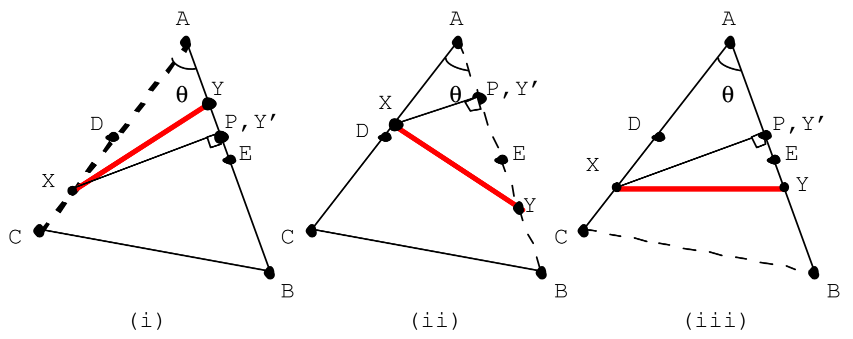


FIGURE 24. The three cases in which an edge of $\triangle ABC$ (shown dashed) has length bounded by a segment $\overline{XY}$ crossing the triangle.

Also let v_x, v_y and v_z be the vertices of e_x, e_y and e_z that are closest to x,y and z, respectively. Claim 7.1 ensures that $|\overline{v_x v_y}| \le \frac{|xy|}{2\sin\theta_{min}}$ and that $|\overline{v_y v_z}| \le \frac{|yz|}{2\sin\theta_{min}}$. Let $s'_i = \overline{v_x v_y}$ and $s'_{i+1} = \overline{v_y v_z}$ be segments of $\pi'(s,t)$ that approximate s_i and s_{i+1}, respectively. This bound applies to every segment $|s'_i|$ of $\pi'(s,t)$ such that $|s'_i| \le (\frac{2}{\sin\theta_{min}})|s_i|$. Since the cost of $\pi(s,t)$ is the sum of the cost of its segments, then $|\pi'(s,t)| \le (\frac{2}{\sin\theta_{min}})|\pi(s,t)|$. Connectivity is ensured since every pair of consecutive segments of $\pi'(s,t)$ share a vertex.

□

Theorem 3.1 *A shortest path $\Pi(s,t)$ between two vertices s and t of a weighted polyhedral surface $\mathcal{P}$ with n faces can be approximated by a path $\Pi'(s,t)$ such that $\|\Pi'(s,t)\| \le (\frac{2}{\sin\theta_{min}})\|\Pi(s,t)\|$, where θ_{min} is the minimum interior angle of any face of $\mathcal{P}$. Furthermore, $\|\Pi'(s,t)\|$ can be computed in $O(n\log n)$ time.*

Proof: The time complexity follows from Algorithm 2. From Lemma 7.1 we know that the length of an approximation segment s'_i that passes through a face is at most $\frac{2}{\sin\theta_{min}}$ times the actual shortest path segment length. In addition, s'_i is an edge of $\mathcal{P}$. From Property 7.1, s'_i cannot have a weight greater than the weight of its adjacent faces. Hence, for any segment s_i of $\Pi(s,t)$ that passes through a face, we approximate it with a segment s'_i of $\Pi'(s,t)$ with cost at most $(\frac{2}{\sin\theta_{min}})\|s_i\|$. If there are no reflected segments in $\Pi(s,t)$ (as in Figure 1b), then the bound of Lemma 7.1 also applies in the weighted case since edges of the terrain cannot have a cost higher than its incident faces. We must now show that if paths are reflected, then the bound also holds.

Let $\triangle ABC$ be a face of $\mathcal{P}$. Let p_1, p_2, p_3 and p_4 be consecutive points of $\Pi(s,t)$ joining consecutive segments of $\Pi(s,t)$. Without loss of generality,

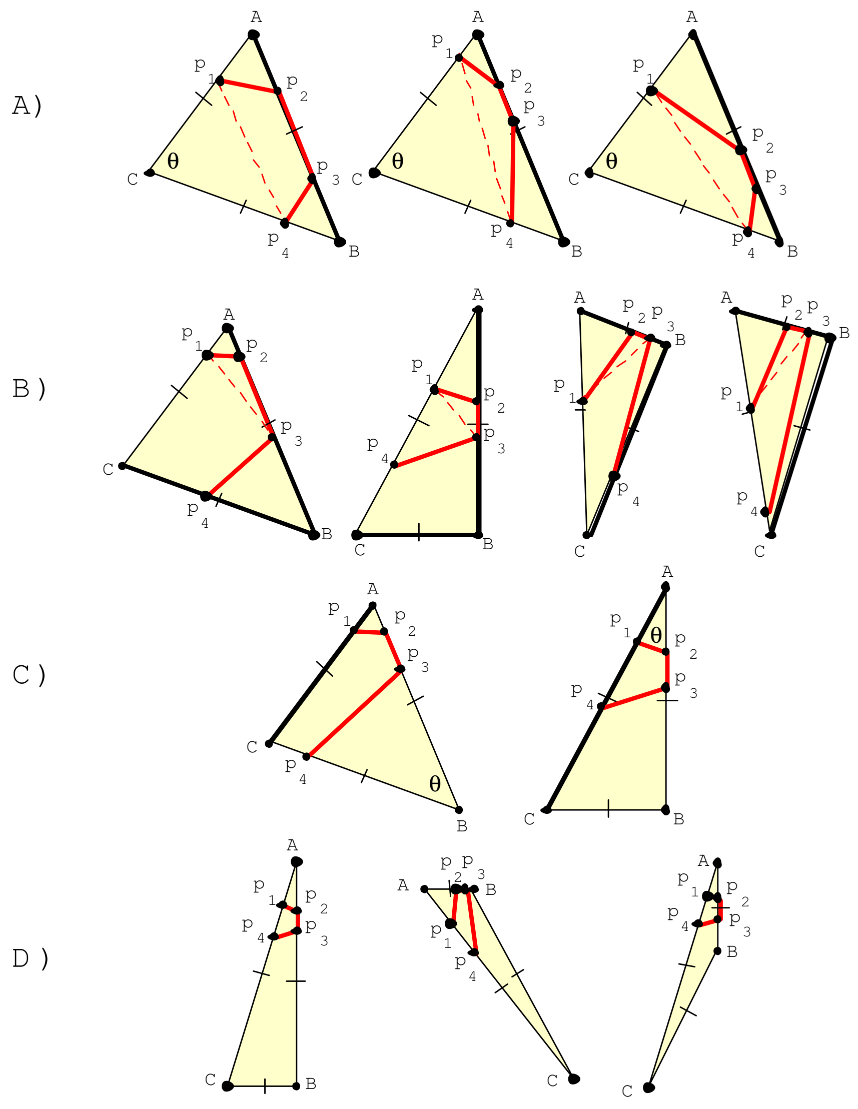


FIGURE 25. 12 cases in which a weighted path can reflect back into a face.

let p_1 lie on $\overline{AC}$, p_2, p_3 lie on $\overline{AB}$ and p_4 lie on either $\overline{AC}$ or $\overline{BC}$. Figure 25 shows many examples of such a set of points. In addition, we will assume that $|\overline{Ap_1}| < |\overline{Cp_1}|$. (we can apply a similar proof when $|\overline{Ap_1}| \geq |\overline{Cp_1}|$. Also, a similar proof can be constructed for when p_2 and p_3 lie on $\overline{CB}$ while p_4 lies on $\overline{AB}$). There are 12 different types of placements of p_2, p_3 and p_4 which differ with respect to which vertex the points are closest to. We will prove that the subpath $\Pi'(p_1, p_4)$ produced, by sliding these 4 points to the closest vertex of the edge on which they lie, will be bounded such that $\|\Pi'(p_1, p_4)\| \leq (\frac{2}{\sin\theta_{min}})\|\Pi(p_1, p_4)\|$.

Consider the three cases of Figure 25A) in which $\Pi'(p_1, p_4) = \overline{AB}$. Since $\overline{AB}$ necessarily has weight cheaper than the face (Property 7.1), then the result of Claim 7.1 proves that $|\overline{AB}| \leq \frac{2}{\sin\theta}|\overline{p_1p_4}| \leq \frac{2}{\sin\theta_{min}}|\overline{p_1p_4}|$. By triangle inequality, $|\overline{p_1p_4}| \leq |\overline{p_1p_2}| + |\overline{p_2p_3}| + |\overline{p_3p_4}|$ and hence $|\overline{AB}| \leq \frac{2}{\sin\theta_{min}}|\Pi(p_1, p_4)|$. Since $|\overline{AB}|$ has weight which is no more than any portion of $\Pi(p_1, p_4)$ then $\|\Pi'(p_1, p_4)\| = \|\overline{AB}\| \leq \frac{2}{\sin\theta_{min}}\|\Pi(p_1, p_4)\|$. In the cases of Figure 25D) both p_1 and p_4 are slid to A and so we can assume that this portion of the path never "leaves" A and hence has a cost of zero. Thus, $\|\Pi'(p_1, p_4)\| < \|\Pi(p_1, p_4)\|$. In Figure 25C) me apply Claim 7.1 to show that $|\overline{AC}| \leq \frac{2}{\sin\theta}|\overline{p_3p_4}|$, where θ is as shown. Since $\overline{p_3p_4}$ is internal to the face and $\|\overline{p_3p_4}\| < \|\Pi(p_1, p_4)\|$, then $\|\overline{AC}\| \leq \frac{2}{\sin\theta_{min}}\|\overline{p_3p_4}\|$. Lastly, the cases in Figure 25B) depict the scenarios in which the approximated path has two segments. In each case, $|\overline{BC}| \leq \frac{2}{\sin\theta_{min}}|\overline{p_3p_4}|$ through Claim 7.1 and since $\overline{p_3p_4}$ is internal to the face then, $\|\overline{BC}\| \leq \frac{2}{\sin\theta_{min}}\|\overline{p_3p_4}\|$. As for $|\overline{AB}|$, we make use of the fact that $|\overline{p_1p_3}| < |\overline{p_1p_2}| + |\overline{p_2p_3}|$, and bound $|\overline{AB}|$ with respect to $|\overline{p_1p_3}|$ using Claim 7.1 again. Once again, the cheapest weight is along edge $|\overline{AB}|$ and so $\|\overline{AB}\| \leq \frac{2}{\sin\theta_{min}}\|p_1p_3\|$. Hence $\|\Pi'(p_1, p_4)\| = \|\overline{AB}\| + \|\overline{BC}\| \leq \frac{2}{\sin\theta_{min}}\|\Pi(p_1, p_4)\|$. □