0.1 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 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.

Property 0.1.1 *An edge of a weighted terrain $\mathcal{P}$ cannot have a weight more expensive than its adjacent faces.*

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 straight line distance between s and t. 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) $s_1, s_2, ..., s_k$. An approximation $\pi'(s,t)$ of $\pi(s,t)$ has a cost which is denoted by $|\pi'(s,t)|$; it is composed of a sequence of k' adjacent straight line segments $s'_1, s'_2, ..., s'_{k'}$. 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 0.1.2 *(Sharir and Schorr [34]) A Euclidean shortest path $\pi(s,t)$ may cross a face at most once.*

Property 0.1.3 *(Sharir and Schorr [34]) 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 $1 \leq i \leq k-1$. Let e_i be the edge that is shared between faces f_i and f_{i+1}. We refer to the sequence of edges $E = e_1, e_2, ..., e_{k-1}$ as an *edge sequence*. A *planar 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 and furthermore the unfolded faces do not overlap. (We say that two faces overlap if they have a common point in their interior.) Let $F' = f'_1, f'_2, ..., f'_k$ be the sequence of faces obtained by applying a planar unfolding of the faces of F and $E' = e'_1, e'_2, ..., e'_{k-1}$ be the edge sequence of F'. We call F' a *simple sleeve.* An unfolding which does not guarantee planarity may produce 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 $w_{f_i}|s_j|$. An approximation $\Pi'(s,t)$ of a weighted shortest path $\Pi(s,t)$ has cost denoted as $||\Pi'(s,t)||$.

Property 0.1.4 *(Mitchell and Papadimitriou [26]) A weighted shortest path obeys Snell's law [22] of refraction in which the path bends at the edges of $\mathcal{P}$. The amount at which it bends depends on the relative weights of the two faces adjacent to this edge (see Figure 2a)).*

Property 0.1.5 *(Mitchell and Papadimitriou [26]) 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 2b).*

Property 0.1.6 *(Mitchell and Papadimitriou [26]) A weighted shortest path may cross a face $\theta(n)$ times (see Figure 1).*

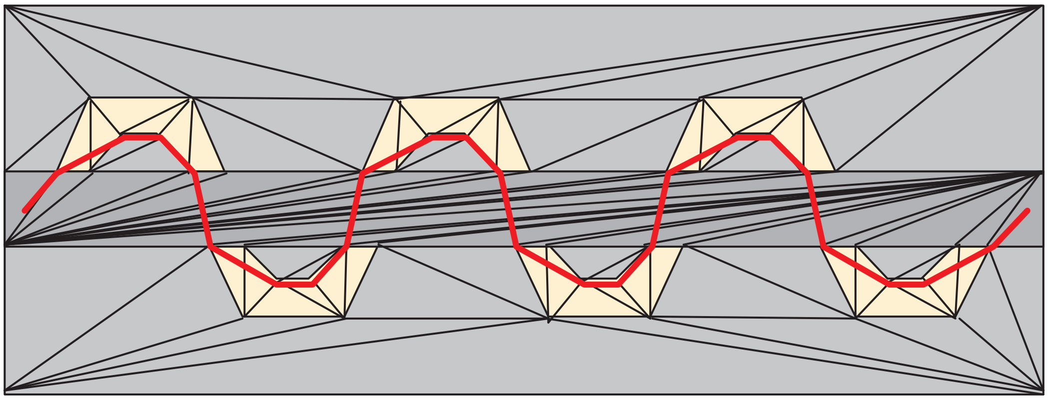

Figure 1: A weighted shortest path that can cross a face $\theta(n)$ times.

From this example, we can see that the inner "ditch" is crossed many times and in fact a poor triangulation of this ditch results in a face being crossed $\theta(n)$ times. In practice however, it is unlikely that such poor triangulations would be used. 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 $\mathcal{P}$.

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: (Dijkstra [11]) *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.*

Algorithm 2: (Fredman and Tarjan [12]) *Given two vertices s and t of G, a modification to Dijkstra's algorithm involving Fibonacci heaps, can be employed to compute a shortest path in G from s to t in $O(V \log V + E)$ time.*

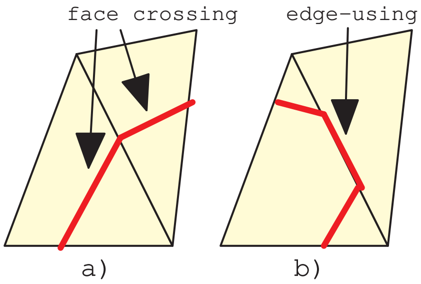

Figure 2: Characteristics of a weighted shortest path. a) adjacent face-crossing segments cause a bend in the path, b) edge-using segments cause a reflection back into the same face.

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.

Observation 0.1.1 follows by straight forward generalization of the planar point location algorithm [31].

Observation 0.1.1 *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.*

Contents

Chapter 1

Approximating Weighted Shortest Paths on Polyhedral Surfaces

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 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. We describe here the design and analysis of several schemes for placing the Steiner points and edges on each face of the polyhedron. The key idea is to transform geometric shortest path problems into graph-theoretic problems so that the following goals are achievable:

1. provable bounds on the approximation factors and the path cost can be established,
2. experimentally verifiable high quality paths can be obtained,
3. existing shortest path graph algorithms can be used,
4. the schemes are easily implementable, and
5. the schemes are appealing to practitioners.

We point out that, in any 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.

In addition to the theoretical contributions, we have implementated all of our schemes, a point-to-point shortest Euclidean path algorithm for sleeves, graph spanners on faces of polyhedra and Chen and Han's algorithm. We performed extensive experiments on triangular irregular networks (TINs) and established excellent performance of the schemes in path quality and run time both being better than the theoretical worst case bounds. We show experimentally that 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 given. 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 [3]. Here also the path accuracy convergences rapidly at a fast running time. We conclude that the schemes presented here are of high value for practitioners from inside and outside the scientific community and for researchers in the above mentioned fields.

The chapter is organized as follows: Section 1.1 describes our approximation algorithms for shortest paths computations. We begin with two simple approaches and then describe improved schemes. Section 1.2 presents the testing procedures, experimental setup and results that were obtained. Section 1.3 discusses an extension to our algorithm to accomodate arbitrary query points. Section 1.4 discusses possible future work in the form of other approximation schemes. We conclude with Section 1.5.

1.1 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 2.

Since this scheme confines the path to traveling on edges of $\mathcal{P}$, the quality of the approximation depends on the given triangulation, which could be bad in the worst case (as is illustrated in Figure 1.1). Assume that the faces of the polyhedron in Figure 1.1 have equal weight. If $\mathcal{P}$ is sufficiently compressed horizontally then $|\Pi(s,t)| \to 0$. Each segment of the shortest path is approximated by an excursion to a vertex and back to another vertex, resulting in an unnecessary traversal of distance $|L|$. The approximate path $\Pi'(s,t)$ with k links has length $k \cdot |L|$ and this could be much larger than $\Pi(s,t)$.

Although this method of approximation can be bad in worst case, we can bound the path cost with respect to parameters of $\mathcal{P}$. We give here a bound on this type of path which depends on θ_{min}.

Property 1.1.1 *Let $\triangle ABC$ be a triangle. Let X be a point on $\overline{AB}$ and Y be a point on $\overline{AC}$ such that $|\overline{AX}| = \delta|\overline{AB}|$ and $|\overline{AY}| = \delta|\overline{AC}|$ for some $0 < \delta < 1$. Then $\overline{XY}$ is parallel to*

[3] At the same time as this work [18], an alternate approach of [23] was proposed.

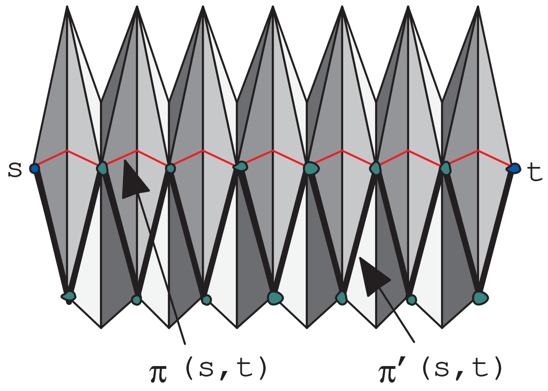

Figure 1.1: The approximated path length $|\Pi'(s,t)|$ is unbounded w.r.t. $|\pi(s,t)|$.

$\overline{BC}$. *Furthermore,* $|\overline{XY}| = \delta|\overline{BC}|$.

Claim 1.1.1 *Let* $\triangle ABC$ *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 1.2 where δ is set to 0.5).

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 1.1.1 $|\overline{DE}| = \delta|\overline{CB}|$. 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 1.1.1 *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* $|\pi'(s,t)| \leq \frac{2}{\sin\theta_{min}}|\pi(s,t)|$.

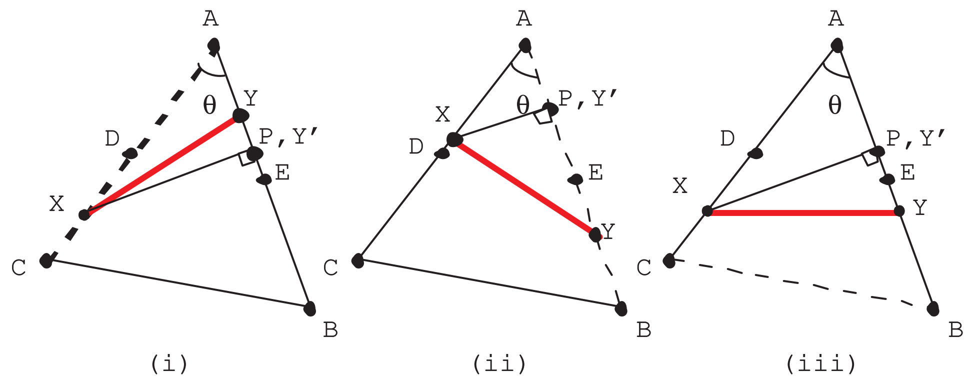

Figure 1.2: The three cases in which an edge of $\triangle ABC$ (shown dashed) has length bounded by a segment $\overline{XY}$ crossing the triangle.

Proof: We will show how $\pi(s,t)$ can be adjusted to form $\pi'(s,t)$ which will travel only on edges of $\mathcal{P}$ and then prove that $\pi'(s,t)$ has the stated bound. Let $s_1 = \overline{sx}$ be the first segment of $\pi(s,t)$ where x lies on some edge edge, say e with vertices v_1 and v_2, of $\mathcal{P}$. Construct s_i' as $\overline{sv_j}$ where v_j is the vertex v_1 or v_2, whichever is closest to x. Next construct s_2' in a similar manner starting now at v_j and going to some other vertex of $\mathcal{P}$ which is determined as an endpoint of the edge that s_2 passes through. Do this for all pairs of adjacent segments of $\pi(s,t)$ to obtain $\pi'(s,t)$. Since all path links were dragged to vertices, the resulting approximation uses only edges of $\mathcal{P}$. In addition, $\pi'(s,t)$ remains connected since we have not "broken" the adjacency of any pair of edges.

Assume that x was at least some distance $\delta'|e|$ from a vertex v. After this dragging stage, x has some distance $\delta''|e|$ from v. If $\delta' > \frac{1}{2}$, then $\delta'' = 1$, otherwise $\delta'' = 0$. By examining the results of Claim 1.1.1 when $\delta = \frac{1}{2}$, it is easily seen that by applying this "dragging" process to X and Y then the resulting segment length is bounded w.r.t. the original length. Hence, by setting $\delta = \frac{1}{2}$, the claim can be used to bound every segment s_i' of $\pi'(s,t)$ such that $|s_i'| \leq (\frac{2}{\sin\theta_{min}})|s_i|$. Since the cost of $\pi(s,t)$ is the sum of the cost of its segments, then $|\pi'(s,t)| \leq (\frac{2}{\sin\theta_{min}})|\pi(s,t)|$. □

Lemma 1.1.2 *A shortest cost path $\Pi(s,t)$ between two vertices s and t of a weighted polyhedral surface $\mathcal{P}$ 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've shown through Lemma 1.1.1 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 0.1.1, s_i' cannot have a weight greater than the weight of its adjacent faces since it would be cheaper to travel "just inside" the face than along the edge. 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 $\frac{2}{\sin\theta_{min}}$ times $|s_i|$. If there are no reflected segments in $\Pi(s,t)$, then the bound of Lemma 1.1.1 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 $\Pi(s,t)$. W.l.o.g., let p_1 lie on $\overline{AC}$, p_2, p_3 lie on $\overline{AB}$ and p_4 lie on either $\overline{AC}$ or $\overline{AB}$. Figure 1.3 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 dragging 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 cases of Figure 1.3A). The points p_1 and p_4 are dragged to vertices A and B, respectively. Since $\overline{AB}$ necessarily has weight cheaper than the face (Property 0.1.1), then the result of Claim 1.1.1 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, $|\overline{p_1p_4}| < |\overline{p_1p_2}| + |\overline{p_2p_3}| + |\overline{p_3p_4}|$ 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 Figure 1.3D) both p_1 and p_4 are dragged 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 1.3C) me merely apply Claim 1.1.1 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 $|\overline{p_3p_4}| < |\pi(p_1p_4)|$, then $\|\overline{AC}\| \leq \frac{2}{\sin \theta_{min}}\|\overline{p_3p_4}\|$. Lastly, Figure 1.3B) depicts the scenario 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 1.1.1 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 $|\overline{AB}|$ w.r.t. $|\overline{p_1p_3}|$ using Claim 1.1.1 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 $\|\Pi'(p_1p_4)\| = \|\overline{AB}\| + \|\overline{BC}\| \leq \frac{2}{\sin \theta_{min}}\|\Pi(p_1p_4)\|$. □

.

In the following theorem we present an upper bound on the approximation quality obtained by using the simple approach.

Theorem 1.1.1 *A shortest cost 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.*

Proof: The proof of the accuracy bound follows from Lemma 1.1.2. The time complexity follows from Algorithm 2. □

.

Although we provided an approximation bound, the worst case can be bad, especially due to the dependency on θ_{min}. In general, the polyhedron may have thin triangles resulting

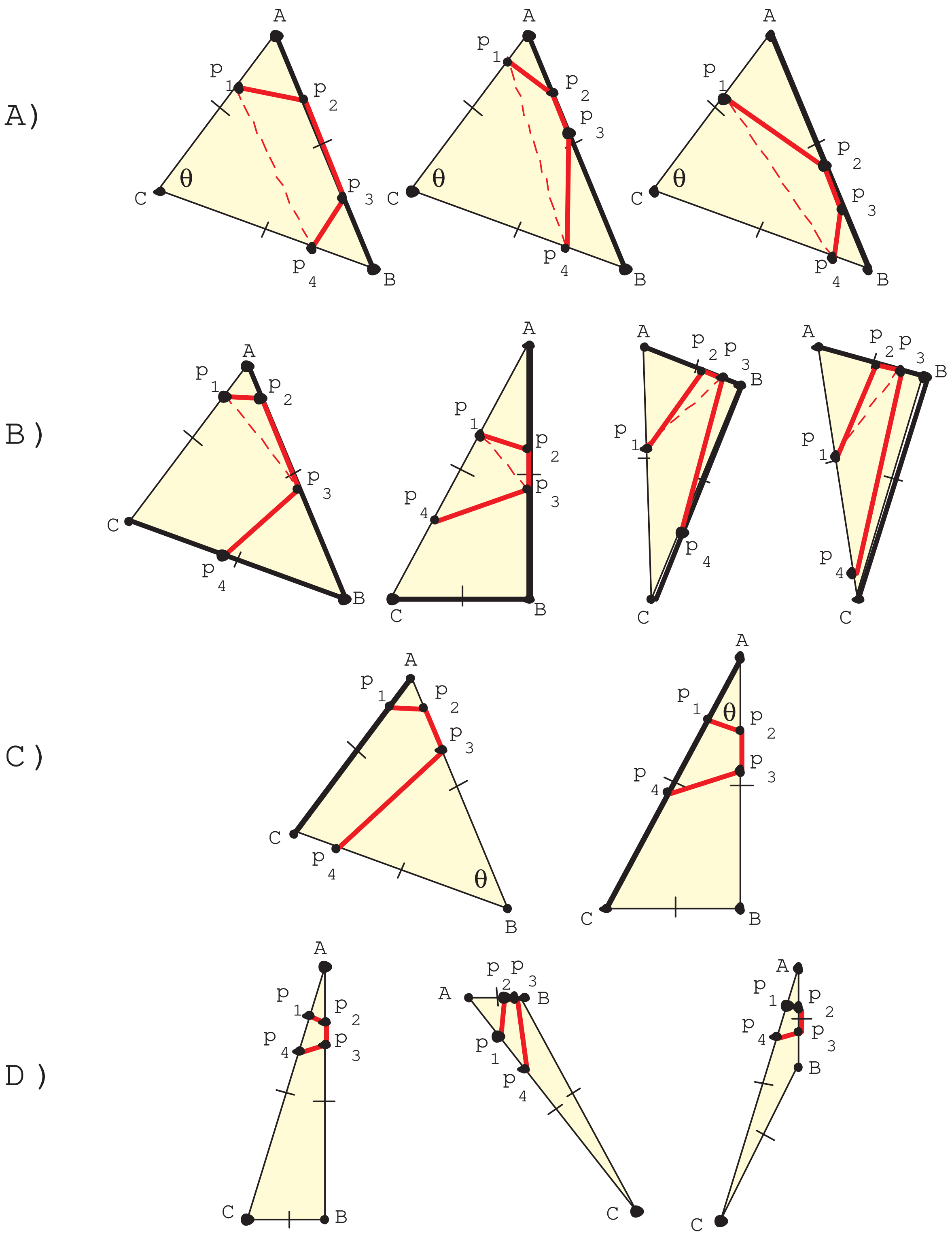

Figure 1.3: 12 cases in which a weighted path can reflect back into a face.

in a small value of θ_{min}. Hence this algorithm may produce a path with unreasonable accuracy. For example, if $\theta_{min} = 5°$ then already the approximation has a worst case bound of roughly 23 times the actual shortest path cost. If $\mathcal{P}$ is a terrain surface (in which vertices are typically sampled from digital elevation data), then the quality of the triangulation can severely effect the resulting approximation. To address this, we investigated a different scheme that does not depend on the geometric parameter of θ_{min}.

1.1.1 Face Decomposition Scheme

Here, we attempt to improve on the simple approach by introducing Steiner points on the polyhedron as well as some additional edges that will allow travel through faces of $\mathcal{P}$. The approach here is to decompose all faces of $\mathcal{P}$ that have edges longer than $\frac{|L|}{k}$ recursively into smaller subfaces. Consider decreasing $|L|$ by a factor of 2. We construct a graph G_i for face f_i as follows. Add Steiner points to the midpoint of each edge of f_i. The vertices of G_i are the vertices of f_i and the Steiner points just added. An edge is added to G_i to connect each vertex to the Steiner points lying on their incident edges of f_i. Also, add edges connecting all three Steiner points. The result is a decomposition of f_i into four smaller triangles called subfaces. We repeat this decomposition recursively on the 4 subfaces until the desired maximal edge length is obtained. Let h denote the number of levels of recursive decomposition, where $h = 0$ implies no decomposition. Figure 1.4 depicts such a construction where $h = 2$. This concludes our construction of G_i. A graph G is then computed by forming the union of all $G_i, 1 \leq i \leq n$. By applying this process to all faces of $\mathcal{P}$, we ensure that $|L|$ has been reduced to $\frac{|L|}{k}$. We can then search G as before, and obtain an approximate shortest path between vertices of $\mathcal{P}$.

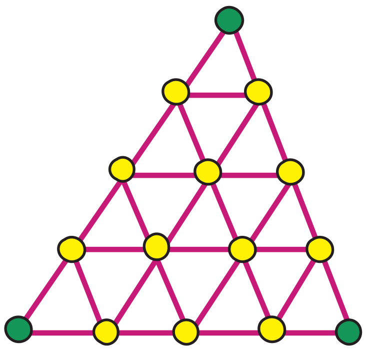

Figure 1.4: Recursively dividing a face into 4 subfaces by joining the midpoints of its edges; $h = 2$.

Property 1.1.2 *G_i is a triangulation of f_i with maximal edge length $\frac{|L|}{2^h}$.*

Property 1.1.3 *Each vertex v of G_i has degree 6.*

Let ϕ_1, ϕ_2, and ϕ_3 be the angles that are parallel to the three edges of f_i make w.r.t. some fixed orientation in the plane defined by f_i.

Property 1.1.4 *Each edge of G_i has orientation ϕ_1, ϕ_2, or ϕ_3.*

Property 1.1.5 *If e and e' are edges of G_i with orientation ϕ_i, then $|e| = |e'|$.*

At first glance, this approximation scheme appears to improve upon the accuracy of the approximation since we allow the path to cross faces of $\mathcal{P}$ instead of restricting it to travel only along edges. However, the portions of the approximated path that pass through faces have an undesirable "zig-zag" characteristic. Since this path does not satisfy Property 0.1.3, this characteristic will result in unnecessary errors in the approximation. In addition, this scheme requires adding Steiner points to each edge in powers of 2 which reduces its scalability. We give here a brief analysis of this face-decomposition scheme.

Define *interior* and *border* vertices to be the vertices of G_i corresponding to Steiner points added inside f_i and on a edge of f_i respectively. Vertices of f_i are considered to be border vertices. Let m designate the number of Steiner points that were added to each edge of $\mathcal{P}$. We can now make the following claims:

Claim 1.1.2 *For any two vertices v_s and v_t of G_i, there exists a shortest path $\pi'(v_s, v_t)$ in G_i such that each segment of $\pi'(v_s, v_t)$ has orientation ϕ_x or $\phi_{(x mod 3)+1}$ where $1 \leq x \leq 3$.*

Proof: Proof is by induction. If $\pi'(v_s, v_t)$ has two segments or less, the claim clearly holds. Let $v_s = v_1, v_2, ..., v_j, ..., v_t$ be the in-order vertices (or Steiner points) of $\pi'(v_s, v_t)$. Assume that the path $\pi'(v_s, v_j)$ contains segments that have orientation ϕ_x and/or ϕ_{x+1}. Consider now, the path $\pi'(v_s, v_{j+1})$. If $\pi'(v_s, v_j)$ used only one orientation, then clearly $\pi'(v_s, v_{j+1})$ will use at most 2 orientations. If $\pi'(v_s, v_j)$ uses both orientations ϕ_x and ϕ_{x+1}, we must show that $\overline{v_j v_{j+1}}$ has orientation ϕ_x or ϕ_{x+1} as well. Let v_a, v_b, v_c, v_d and v_e be the candidate vertices for v_{j+1}. These vertices are the remaining (up to 5) vertices adjacent to v_j. Let v_a and v_e be the two vertices adjacent to both v_{j-1} and v_j (see Figure 1.5). W.l.o.g. assume that $\overline{v_{j-1} v_j}$ has orientation ϕ_x and that $\overline{v_e v_j}$ has orientation ϕ_{x+1}. Now $\pi'(v_s, v_t)$ cannot go from v_j back to v_{j-1}, nor can it go from v_j to v_a (or to v_e) since a shorter path can be made by going directly from v_{j-1} to v_a (or to v_e). Let v_p be the last vertex of $\pi'(v_s, v_j)$ such that $\overline{v_p v_{p+1}}$ has orientation ϕ_{x+1} (again see Figure 1.5). By construction of G_i, there exists a set of edges with orientation ϕ_x forming a straight line from v_p to v_d and so $\pi'(v_s, v_d)$ must use these segments. Therefore, $\overline{v_j v_d}$ cannot be a segment of $\pi'(v_s, v_d)$ and thus $v_{j+1} \neq v_d$. The only remaining candidates for v_{j+1} are v_b and v_c which have orientations ϕ_{x+1} and ϕ_x, respectively. Hence the claim is true by induction. $\square$

Claim 1.1.3 *There exists a shortest path $\pi'(v_a, v_b)$ in G_i between any two border vertices v_a and v_b that does not "bend" at an interior vertex of G_i.*

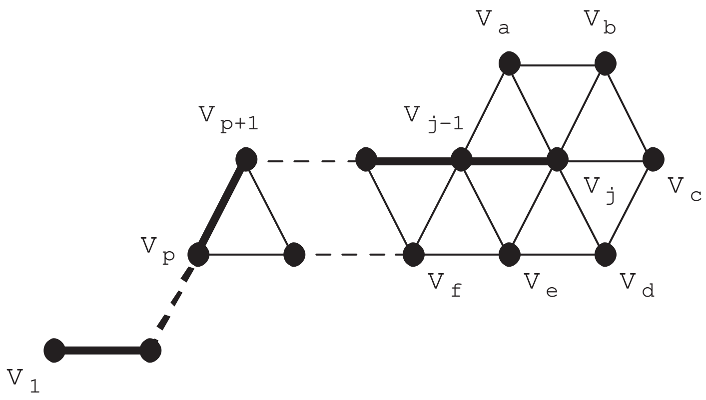

Figure 1.5: The vertices adjacent to v_j.

Proof: From Claim 1.1.2, we know that there exists a path $\pi'(v_s, v_t)$ between any two vertices v_s and v_t of G_i that will use at most two orientations, say ϕ_x and ϕ_{x+1}. Consider the parallelogram with opposite vertices v_s and v_t having a border consisting of edges in G_i with orientations ϕ_x and ϕ_{x+1} (see Figure 1.6 where $v_s = v_a$ and $v_t = v_b$). Using Property 1.1.5, it can be shown that paths along the border of this parallelogram from v_s to v_t have equal length to any path traveling inside the parallelogram. Hence, a 2-piece path can be constructed between v_s and v_t in which one piece consists of segments in G_i with orientation ϕ_x and the other consisting of segments with orientation ϕ_{x+1}. Moreover, for two border vertices v_a and v_b, at least one of these path pieces can be chosen to lie along the border of G_i and this path only "bends" on the border of G_i. $\square$

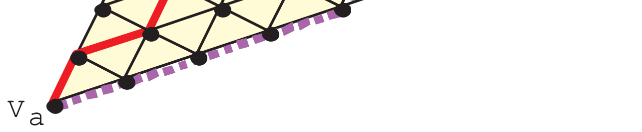

Figure 1.6: There exists a shortest path from v_a to v_b along the borders of their bounding parallelogram.

Claim 1.1.3 implies that this scheme does not make use of the Steiner points placed inside a face. Hence, we've shown already that this face decomposition scheme adds many

Steiner points and edges which do nothing to improve the approximation.

Lemma 1.1.3 *For any two vertices s and t of G there exists a path $\pi'(s,t)$ in G that does not pass through any interior vertex of G.*

Proof: We use proof by induction. If $\pi'(s,t)$ passes through a single face, then the proof is trivial. If $\pi'(s,t)$ passes through two faces (they must be adjacent) we can apply the following argument. Let A and E be two vertices on different faces (i.e. neither vertex is shared with both faces) and let a shortest path from A to E pass through one of the following sets of vertices: BID, BIF, HID or HIF as shown in Figure 1.7.

Let $\delta_1 = \frac{\overline{CD}}{\overline{CE}}$ and $\delta_2 = 1 - \delta_1$. From Property 1.1.2 and from construction of G_i and G_{i+1}, we know that:

$$|\overline{CD}| = \delta_1|\overline{CE}|,\ |\overline{DE}| = \delta_2|\overline{CE}|,\ |\overline{FE}| = \delta_1|\overline{GE}|,\ |\overline{GF}| = \delta_2|\overline{GE}|,$$

$$|\overline{AH}| = \delta_1|\overline{GA}|,\ |\overline{HG}| = \delta_2|\overline{GA}|,\ |\overline{BC}| = \delta_1|\overline{CA}| \text{ and } |\overline{AB}| = \delta_2|\overline{CA}|.$$

In order to show that there is an equally good path from A to E that does not pass through the interior of f_i or f_{i+1}, we must show that one of the following holds:

$$|\overline{BC}| + |\overline{CD}| \leq |\overline{BI}| + |\overline{ID}| \text{ or } |\overline{HG}| + |\overline{GF}| \leq |\overline{HI}| + |\overline{IF}|.$$

We can rewrite this as:

$$\delta_1|\overline{AC}| + \delta_1|\overline{CE}| \leq \delta_1|\overline{AG}| + \delta_1|\overline{GE}| \text{ or } \delta_2|\overline{AG}| + \delta_2|\overline{GE}| \leq \delta_2|\overline{AC}| + \delta_2|\overline{CE}|.$$

Thus we just need to show that,

$$|\overline{AC}| + |\overline{CE}| \leq |\overline{AG}| + |\overline{GE}| \text{ or } |\overline{AG}| + |\overline{GE}| \leq |\overline{AC}| + |\overline{CE}|.$$

At least one of these statements is always true so we've proven the lemma for the 2 face case. Now assume that $\pi'(s,t)$ crosses k faces and does not pass through interior Steiner points. We must then show that any shortest path in G with $k+1$ links will also not pass through interior Steiner points. We prove this by examining three cases as shown in Figure 1.8. In each case, let $\triangle ABC$ and $\triangle BDC$ be the k^{th} and $k^{th}+1$ faces respectively and so $t = D$. W.l.o.g. assume that $\pi'(S,D)$, passes through edges $\overline{AC}$ and $\overline{BC}$ (at one of their endpoints).

Case 1: $\pi'(S,B) = \pi'(S,C) + \overline{BC}$ (see Figure 1.8a).
Assume that $\pi'(S,D)$ does not contain segments $\overline{CD}$ or $\overline{BD}$, then there must exist a point

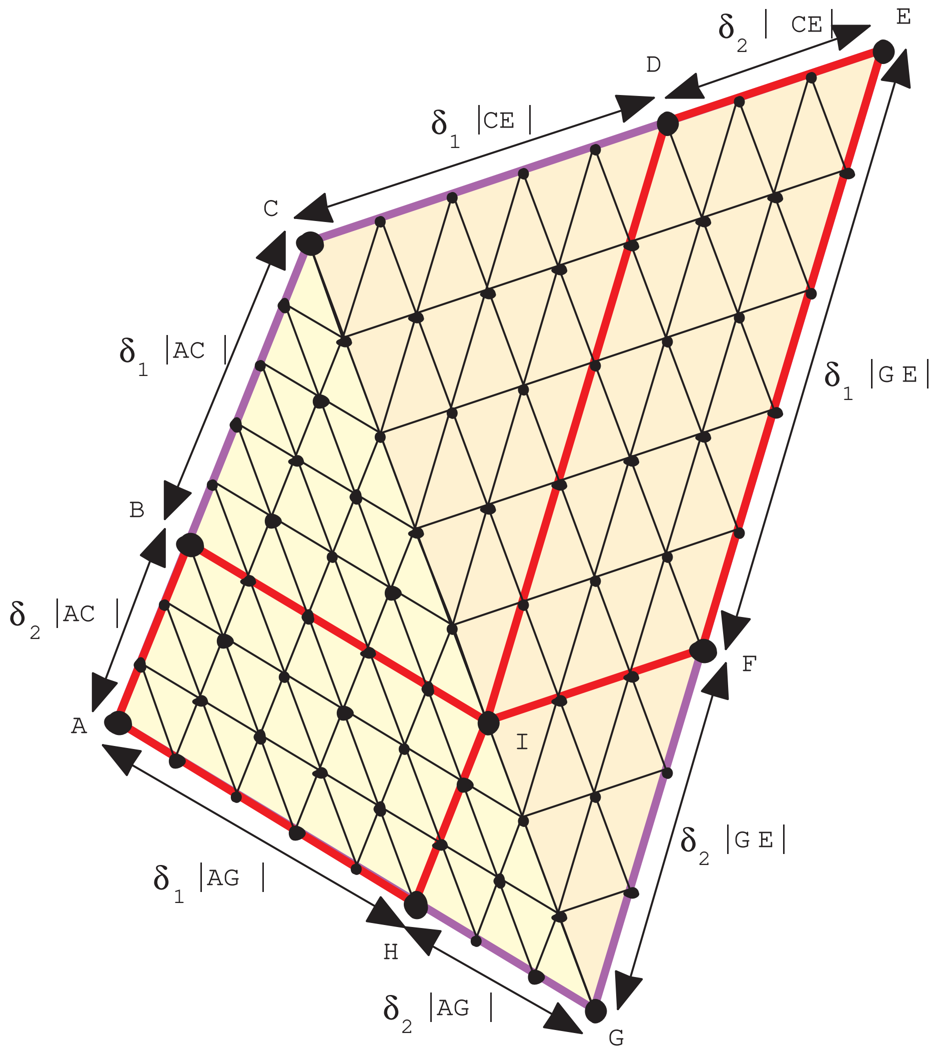

Figure 1.7: Possible shortest paths between opposite vertices of two adjacent faces using the face decomposition scheme.

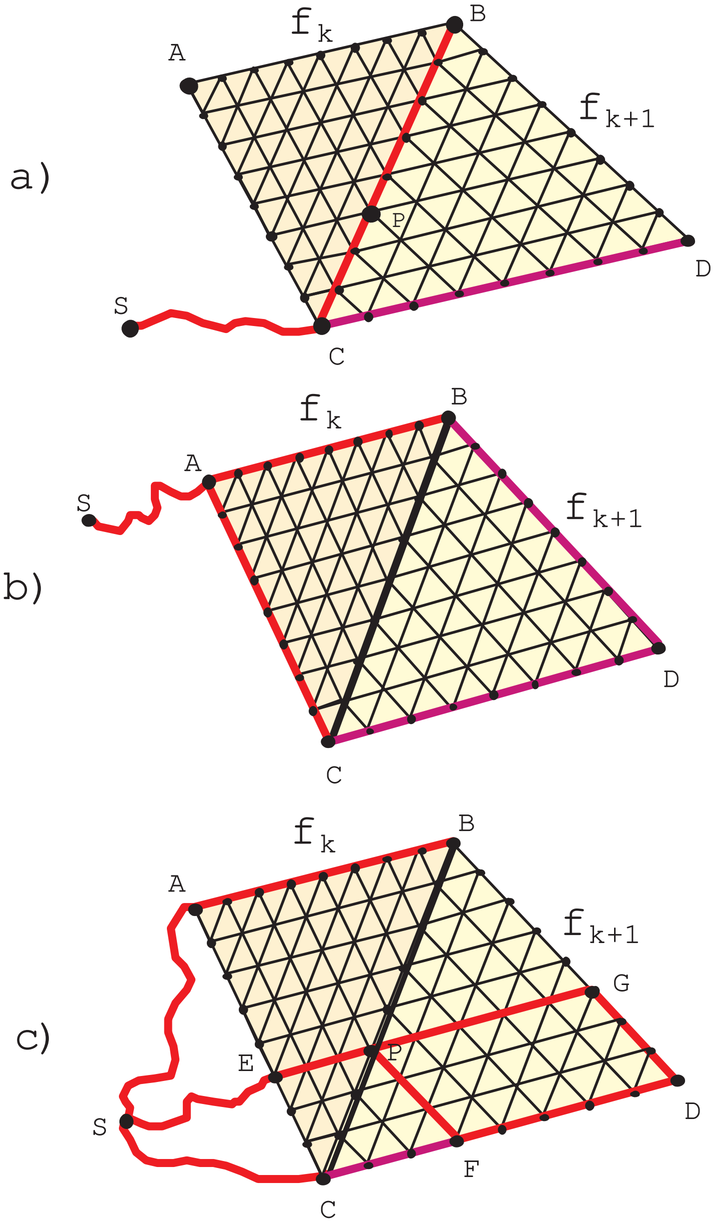

Figure 1.8: The 3 cases for $\pi'(S, D)$.

P on $\overline{BC}$ through which $\pi'(S,D)$ passes. Since $\pi'(S,B) = \pi'(S,C) + \overline{CP} + \overline{PB}$, then we know that $\pi'(S,P)$ must also pass through C, otherwise $\pi'(S,B)$ would not be a shortest path (which contradicts this case assumption). Hence, if $\pi'(S,D)$ passes through P, then $\pi'(S,D) = \pi'(S,C) + \overline{CP} + \pi'(P,D)$. But this path is longer than $\pi'(S,C) + \overline{CD}$ and this results in a contradiction. Hence, $\pi'(S,D)$ must not pass through P.

Case 2: $\pi'(S,B) = \pi'(S,A) + \overline{AB}$ and $\pi'(S,C) = \pi'(S,A) + \overline{AC}$ (see Figure 1.8b).
As in case 1, $\pi'(S,P)$ for any P lying on $\overline{AB}$ or on $\overline{AC}$ must pass through A. Hence $\pi'(S,D)$ must pass through A as well. Since we know that the lemma holds for the simple case of 2 adjacent faces, then either $\pi'(S,D) = \pi'(S,A)+\overline{AB}+\overline{BD}$ or $\pi'(S,D) = \pi'(S,A)+\overline{AC}+\overline{CD}$.

Case 3: $\pi'(S,B) = \pi'(S,A) + \overline{AB}$ and $\pi'(S,C) \neq \pi'(S,A) + \overline{AC}$ (see Figure 1.8c).
As in case 1, we can assume that a shortest path passes through some Steiner point P on $\overline{BC}$. W.l.o.g. we will assume that $\pi'(S,D) = \pi'(S,E) + \overline{EP} + \overline{PF} + \overline{FD}$ where E and F are border nodes lying on $\overline{AC}$ and $\overline{CD}$, respectively. We will show that one of the following is true (hence making a contradiction):

$$|\pi'(S,C)| + |\overline{CD}| \leq |\pi'(S,E)| + |\overline{EP}| + |\overline{PF}| + |\overline{FD}| \tag{1.1}$$

$$|\pi'(S,B)| + |\overline{BD}| \leq |\pi'(S,E)| + |\overline{EP}| + |\overline{PF}| + |\overline{FD}| \tag{1.2}$$

Clearly, $|\pi'(S,E)| \leq |\pi'(S,C)| + |\overline{EC}|$ and we can substitute this into 1.1 to obtain:
$|\pi'(S,E)| - |\overline{EC}| \leq |\pi'(S,C)| \leq |\pi'(S,E)| + |\overline{EP}| + |\overline{PF}| + |\overline{FD}| - |\overline{CD}|$, and so

$|\overline{CD}| - |\overline{FD}| \leq |\overline{EP}| + |\overline{PF}| + |\overline{EC}|$, which implies that $|\overline{CF}| \leq |\overline{CE}| + |\overline{EP}| + |\overline{PF}|$.

But this is always true since $|\overline{CF}|$ is a straight line. Hence, equation 1.1 is true and the lemma holds. Hence, the lemma has been proven by induction. □

With the results of Lemma 1.1.3, we can conclude that the face decomposition scheme provides a poor approximation since adding an arbitrary number of Steiner points and edges to G does not provide any improvement to the approximation for paths between vertices of $\mathcal{P}$. That is, it is no better than the simple approach. This was confirmed experimentally. Moreover, the time complexity is much larger here since the size of G has been increased.

The problem with this scheme lies in the fact that we are placing Steiner points internal to the faces of $\mathcal{P}$, causing the approximated path to bend unnecessarily. We therefore took a different approach in the schemes to follow; namely that of adding Steiner points only along the edges of $\mathcal{P}$.

1.1.2 Improved Approximation Schemes

We describe here two different schemes for placing Steiner points along the edges of $\mathcal{P}$ and describe how they are interconnected on each face to form each G_i using one of two interconnection strategies. We then derive bounds on the portion of the shortest path going through a 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.

Building the Graph

We begin by placing Steiner points evenly along each edge of $\mathcal{P}$ using one of two placement schemes:

- *Fixed* placement - exactly m Steiner points are placed along the edge.
- *Interval* placement - place only enough (up to m) Steiner points such that the distance between adjacent Steiner points on an edge is $\frac{|L|}{m+1}$ (with the exception of the first and last points, whose distance from the vertex may be less than $\frac{|L|}{m+1}$).

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, denoted as V_i. Connect a vertex pair v_a, v_b of G_i to form an edge $v_a v_b$ of G_i if v_a and v_b represent Steiner points that are adjacent on the same edge of f_i. We then provide further interconnection among the vertices of G_i by using one of two connection schemes:

- *Complete* interconnection - 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 represent Steiner points that lie on different edges of f_i.
- *Spanner* interconnection - Let $\mathcal{C}$ be the set of planar cones with apex at all $v_a \in V_i$ and the conical angle $\theta = \frac{\pi}{\rho}$ for an integer constant $\rho > 4$. For each $v_a \in V_i$ perform a radial sweep of the elements of V_i. During this sweep determine the vertex $v_b \in V_i$ that has minimal distance to v_a in each of the ρ cones and add edge $v_a v_b$ to G_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 an example of creating G_i, Figure 1.9 shows how 6 Steiner points ($m = 2$) and 27 edges are added to a face to form the face graph G_i using the fixed placement scheme and complete interconnection strategy. Figure 1.10 shows how edges are formed from a single vertex using the spanner interconnection scheme.

A graph G is computed by forming the union of all face graphs $G_i, 1 \leq i \leq n$. It can be shown that all edges of G lie on the surface of $\mathcal{P}$. Hence, any path in G (i.e., our

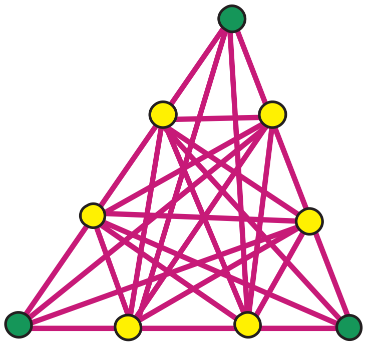

Figure 1.9: Adding Steiner points and edges to a face using the "fixed scheme".

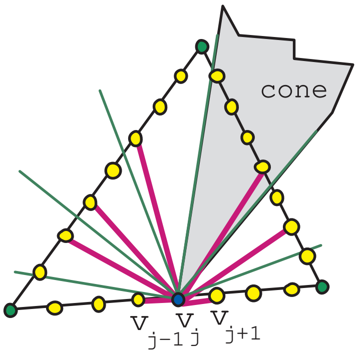

Figure 1.10: The spanner edges added from a vertex v_a with $\theta = 30°$.

approximation) can be transformed to a path on the surface of $\mathcal{P}$. In the sections to follow, we show that there exists a path in G that approximates shortest path $\Pi(s,t)$.

The approximate shortest path $\Pi'(s,t)$ in $\mathcal{P}$ is then computed as before by first determining a shortest path in G using Algorithm 2. The resulting path in G corresponds to a path in $\mathcal{P}$. Now we analyze the cost of an approximate path.

Bounding the Approximation

We begin the analysis by first considering the fixed or interval placement schemes using the complete interconnection strategy.

Claim 1.1.4 *Given a segment s_j crossing face f_i, there exists an edge s'_j in G_i such that* $\|s'_j\| \le w_{f_i} \cdot \|s_j\| + w_{f_i} \cdot \frac{|L|}{m+1}$.

Proof: Each edge in $\mathcal{P}$ is divided into at most $m+1$ intervals which have length at most $\frac{|L|}{m+1}$. Let $s_j = \overline{ab}$, 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 $s'_j \in G_i$ joining them (see Figure 1.11). The triangle inequality ensures that $|s'_j| \leq |\overline{ca}| + |s_j| + |\overline{bd}|$. Since we chose the

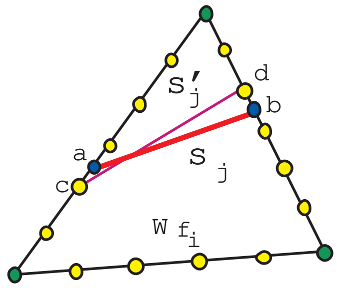

Figure 1.11: A face-crossing segment s_j of a weighted shortest path.

closer interval endpoints, then $|\overline{ca}| \leq \frac{|L|}{2(m+1)}$ and $|\overline{bd}| \leq \frac{|L|}{2(m+1)}$. Hence

$$|s'_j| \leq |s_j| + \frac{|L|}{m+1}. \tag{1.3}$$

Now, multiplying by w_{f_i} we have

$$w_{f_i} \cdot |s'_j| \leq w_{f_i} \cdot |s_j| + w_{f_i} \cdot \frac{|L|}{m+1}. \tag{1.4}$$

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 $\square$

Lemma 1.1.4 *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 1.1.4 that there exists an edge $s'_j \in G$ that approximates s_j. Let $\Pi'(s,t) = \{s'_1, s'_2, \cdots, s'_k\} \in G$. Observe that due to the construction, $\Pi'(s,t)$ is a connected path. By applying the results of Claim 1.1.4 to each segment of $\Pi'(s,t)$ we have: $\sum_{i=1}^{k} \|s'_i\| \leq \sum_{i=1}^{k}(\|s_i\| + w_{f_{s_i}} \cdot \frac{|L|}{m+1})$, where $w_{f_{s_i}}$ denotes the weight of the face that s_i passes through. We can rewrite this as

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

Since $w_{f_{s_i}} \leq W$ for all f_{s_i} by definition, then

$$\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + \frac{|L|}{m+1} \cdot k \cdot W.$$

□

Theorem 1.1.2 *Using the complete interconnection strategy, 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 $\mathcal{P}$ with n faces 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 1.1.4 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 2 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 0.1.6 it follows that $\Pi(s,t)$ may have $\theta(n^2)$ segments. In Lemma 1.1.4, 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. Each edge of $\mathcal{P}$ contributes $O(n^2)$ graph vertices and each face contributes $O(n^4)$ graph edges. Algorithm 2 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.)

□

In our analysis, we made the assumption 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 $|L|$. The interval placement scheme allows less Steiner points per edge while maintaining the same worst-case bounds.

Figure 1.12 shows an example of how Steiner points are added to faces using a) the fixed placement scheme where $m = 7$ and b) the interval placement 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.

We now provide an analysis for the case in which the spanner interconnection strategy is used instead of the complete interconnection strategy. This spanner scheme improves upon the time complexity of the complete scheme; though the approximation achieved is not quite as good. The intuition behind this is that the spanner scheme graph is sparse while the complete scheme graph is much more dense; and so Algorithm 2 will search the sparse graph in less time.

Results of Clarkson [10], ensure that each 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 $|V_i| = O(m)$). Since 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, the resulting graph G has $O(mn)$ vertices and edges.

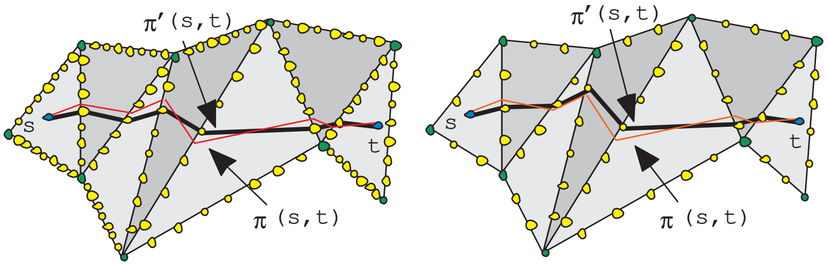

Figure 1.12: The difference in the layout of Steiner points (m=7) for a) the fixed scheme, b) the interval scheme.

Theorem 1.1.3 *Using the spanner interconnection strategy, 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 $\mathcal{P}$ with n faces such that $\|\Pi'(s,t)\| \leq \beta(\|\Pi(s,t)\| + W|L|)$, where $\beta > 1$, 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 $O(n^3 \log n)$ time.*

Proof: In Claim 1.1.4 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 1.1.2, 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 with time complexity 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_a all the cones ($O(1)$ in all) with apex at v_a can be computed in $O(1)$ time. Moreover the closest vertex to v_a 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_a 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 2 and it runs in $O(n^3 \log n)$ time.

□

Fine Tuning the Approximation - An Additional Sleeve Computation

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 in $\mathcal{P}$ according to a preliminary approximated path $\pi'(s,t)$ which is computed as mentioned earlier. 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 [21]). 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 1.13 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.

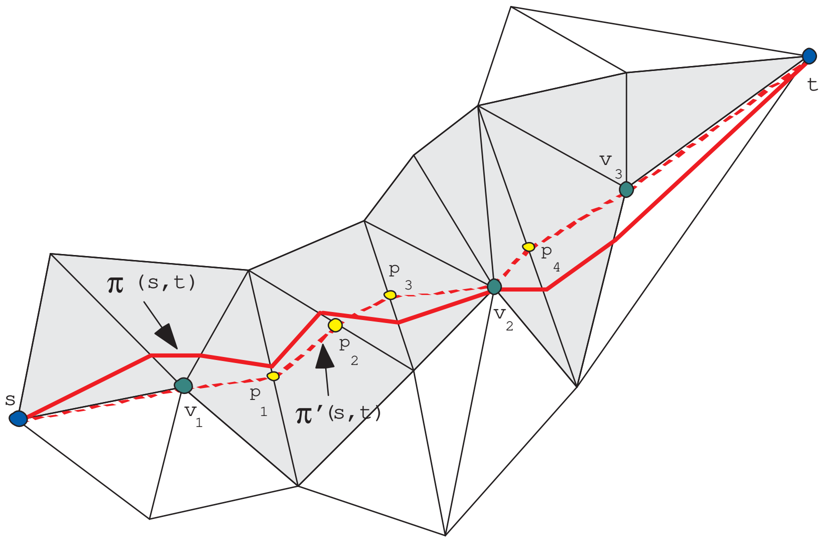

Figure 1.13: Choosing an edge sequence from the path $\pi'(s,t)$.

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 1.13 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 1.2).

Given the unfolded sleeve, we compute the shortest path $\pi_{\mathcal{S}}(s,t)$ in the sleeve from s to t. Lee and Preparata [21] 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 1.14).

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

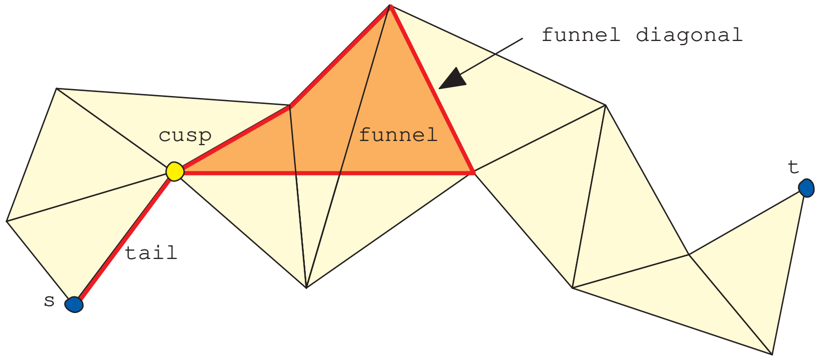

Figure 1.14: The funnel structure.

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 [21] 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 algorithm correctness. We must also show that the resulting path is non-overlapping when projected back onto the surface of $\mathcal{P}$.

Property 1.1.6 *$\pi'(s,t)$ 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 1.1.5 *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 1.1.5 *The projected path $\pi''(s,t)$ is simple.*

Proof: The proof follows from Property 1.1.6, and Claim 1.1.5. □

Section 1.2 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. 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.

1.2 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 [16], 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 1.15. 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 [19]).

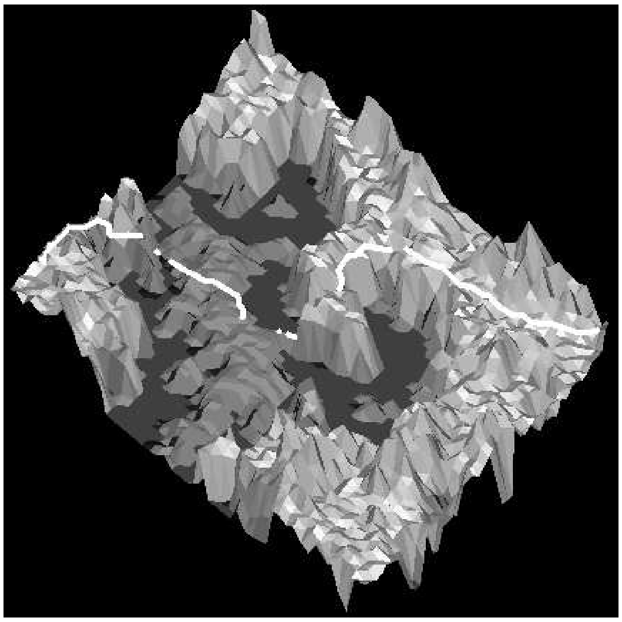

Figure 1.15: A weighted shortest path on a terrain in which traveling on water is expensive.

1.2.1 Implementation and Testing Procedures

In this subsection we discuss the implementation, test data and the testing procedure.

A Variation of Dijkstra's Algorithm

In place of Dijkstra's shortest path algorithm, we implemented a known variation; namely the A* algorithm (see [33]). In this scheme we associate an additional weight to each vertex, namely its Euclidean distance to the destination vertex. Then, during each iteration, we choose the vertex which minimizes the sum of its cost from the source vertex plus the Euclidean distance to the destination vertex, over all candidate vertices.

Numerical Issues

Suri [35] points out that the Chen and Han algorithm [7] based on unfolding is sensitive to numerical problems. This is 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.

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 stabilty issues as required. When using LEDA's data types [24], 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.

1.2.2 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 1.1 shows the attributes of the TINs that we tested. Figure 1.16 depicts screen snapshots of the last 10 TINS of Table 1.1. TINs with stretched heights were created by multiplying their heights by five.

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 1.1: The data used for the experiments showing the TINs and their attributes.

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

Table 1.2: The different approximation schemes.

For the weighted domain, we used the same TINs and set the weight of 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 1.2. 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).

1.2.3 Path Accuracy

We first examine the Euclidean shortest path problem. Figures 1.18 and 1.19 depict the results of these tests for six terrains. The approximated path length rapidly converges to the actual path length (as computed using Chen and Han algorithm [7]). 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 1.17.

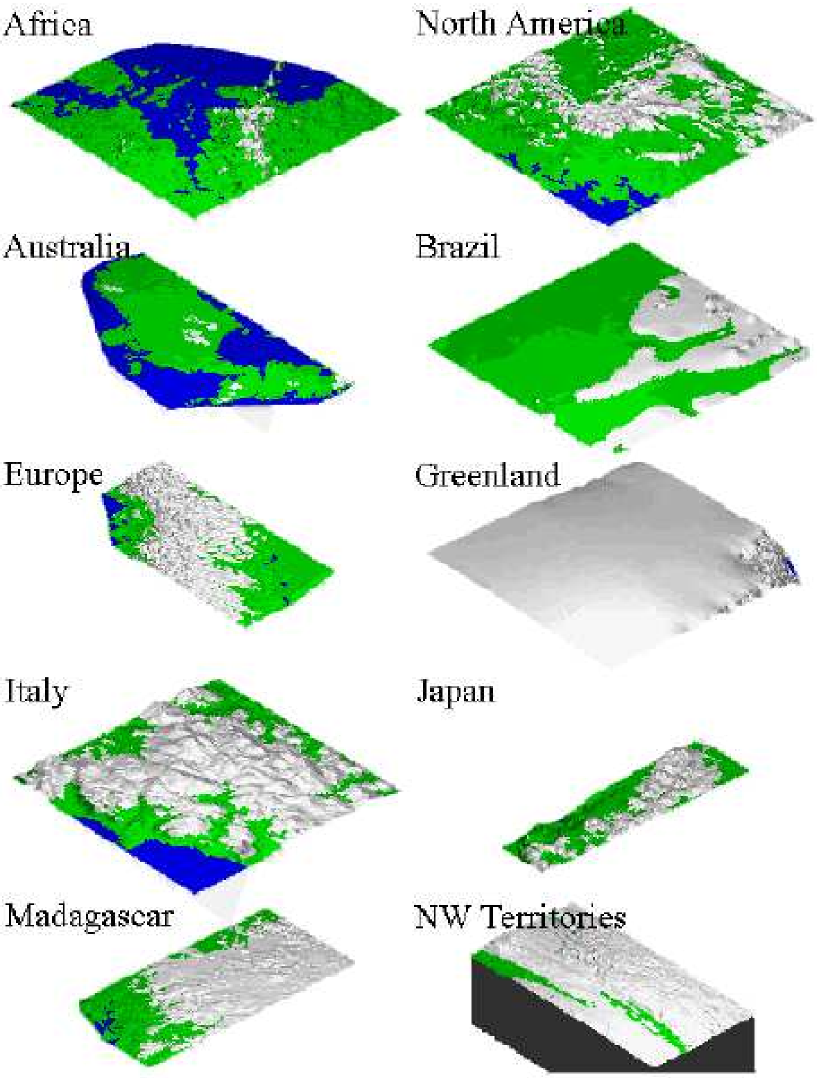

Figure 1.16: Snapshots showing various TINs that were used for testing.

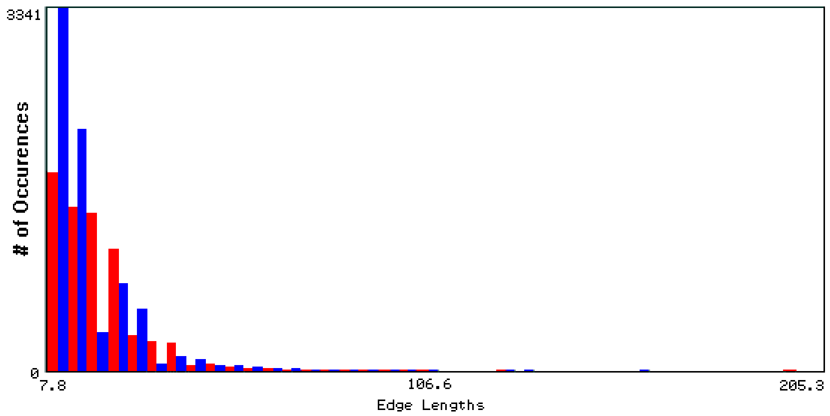

Figure 1.17: Histogram of edge lengths for one of our TINs.

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. Figures 1.18 and 1.19 allow 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.

It is no surprise that the sleeve computation provides a significant improvement on the average path cost. Figure 1.20 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% and 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 as-

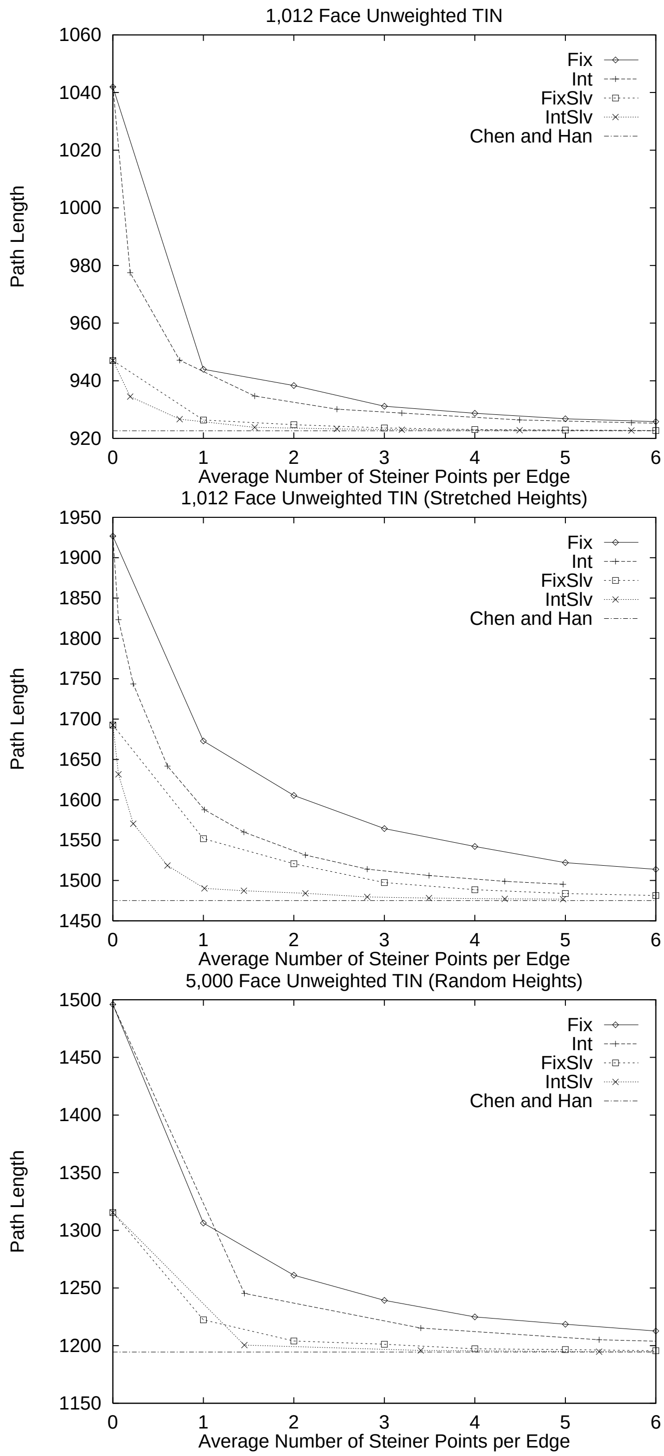

Figure 1.18: Graphs showing average path length for three selected terrains.

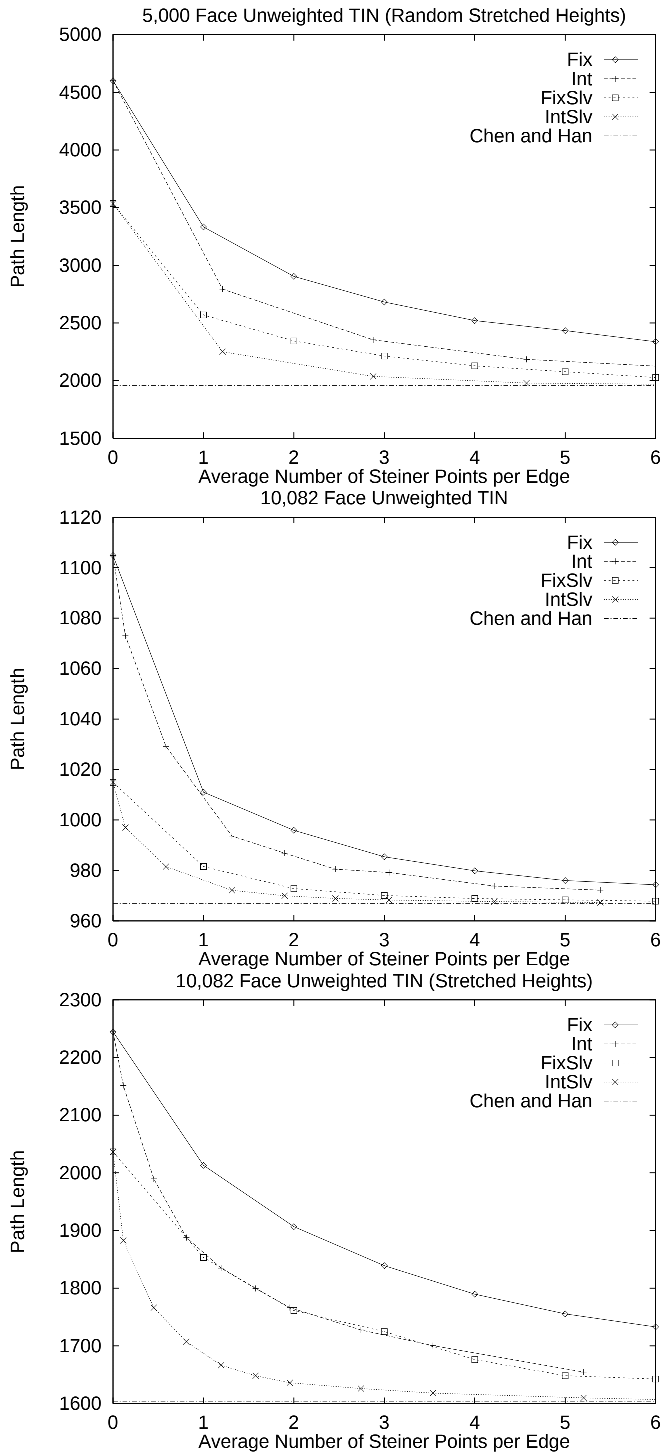

Figure 1.19: Graphs showing average path length for three more selected terrains.

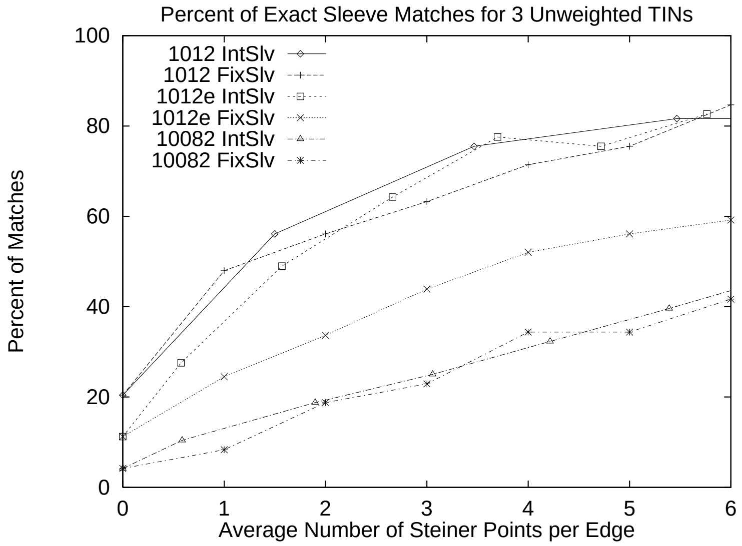

Figure 1.20: Graph showing the percentage of exact edge sequence matches for a 1012 face terrain, a 1012 face stretched terrain and a 10082 face terrain.

sumption 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 1.17). The graphs of Figure 1.21 compare the 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 worst case and the maximum error obtained from the 100 paths tested; whereas the bottom graph depicts the average worst case and the average error obtained. 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 1.1. The results are depicted in Figures 1.22 and 1.23 and they verify that all terrains tested had similar accuracy and convergence behavior. Figure 1.24 shows the typical path accuracy as compared to that of Chen and Han's algorithm 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

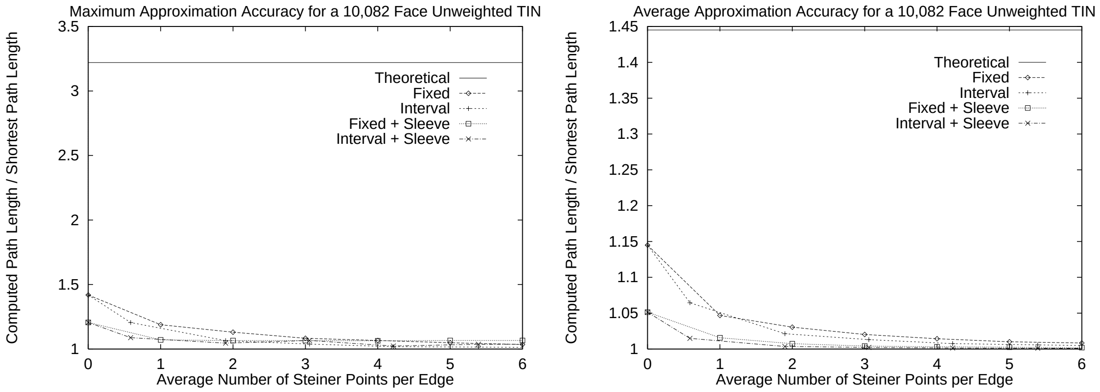

Figure 1.21: Graphs comparing the worst case (theoretical) accuracy with that of the produced accuracy for a 10,082 face terrain using the fixed and interval schemes (top: maximum error; bottom: average error).

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 1.25 depicts the results of computing shortest Euclidean paths on the 10,082 face terrain using the fixed and interval schemes [1]. The graphs show the loss in path accuracy with increasing cone angle. As in the non-spanner schemes, the interval placement scheme converges quicker that the fixed placement scheme.

1.2.4 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 had the same person implement all algorithms and use 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 2. 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 1.26 depicts the actual run-time results of the Euclidean shortest path tests for variations of our approximation schemes on 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). We can see that our algorithms are substantially faster than the one due to Chen and Han[7]. The main reason is that our algorithms do not require any complex data structures, nor do they perform expensive computations (such as 3D rotation 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.

[1] The 100 (random) vertex-pairs considered here are different than those in Figure 1.19

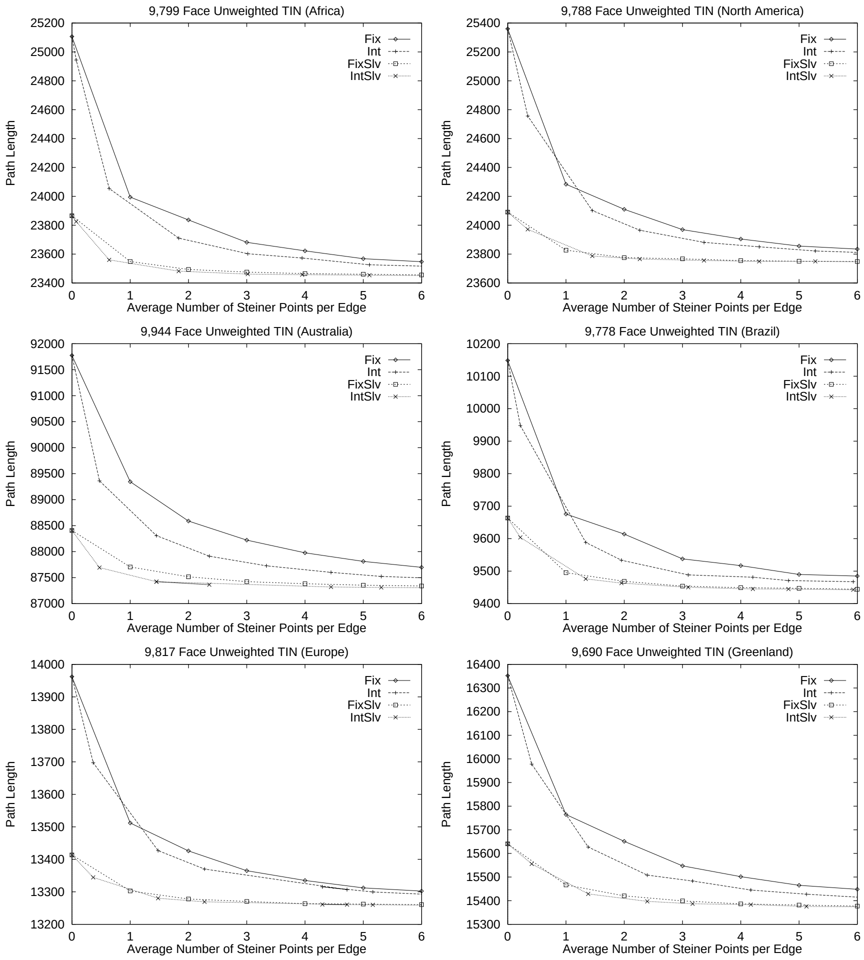

Figure 1.22: Graphs showing average path length for six real-data terrains.

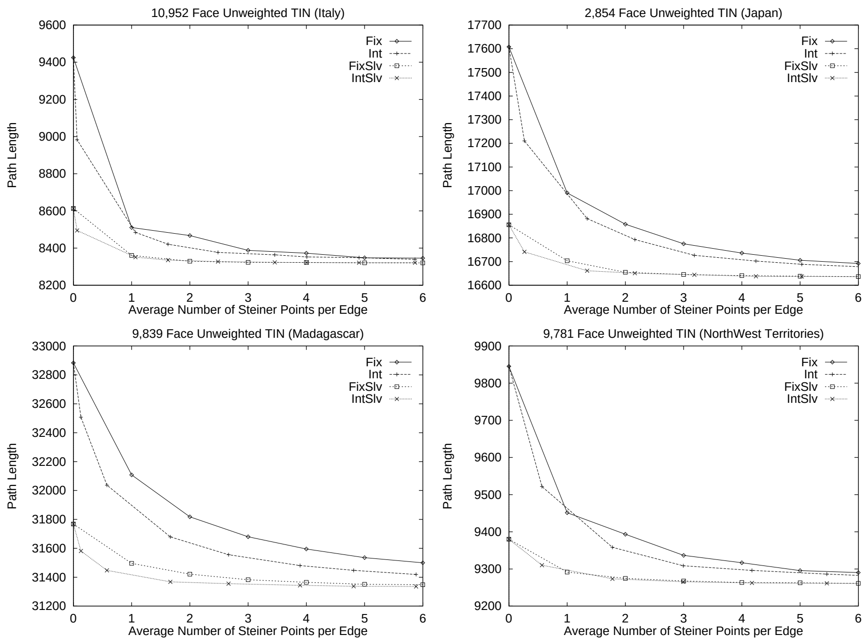

Figure 1.23: Graphs showing average path length for four real-data terrains.

Figure 1.27 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 the 10 additional terrains is shown in Figures 1.28 and 1.29; 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 1.30 shows the running time for the tests corresponding to the graphs of Figure 1.25. 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

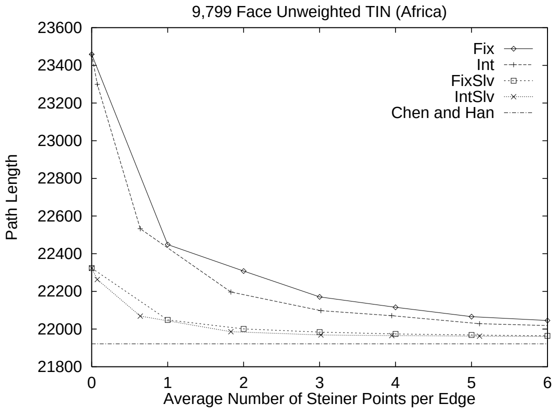

Figure 1.24: Graph comparing average path length between our schemes and that of Chen and Han for the Africa TIN.

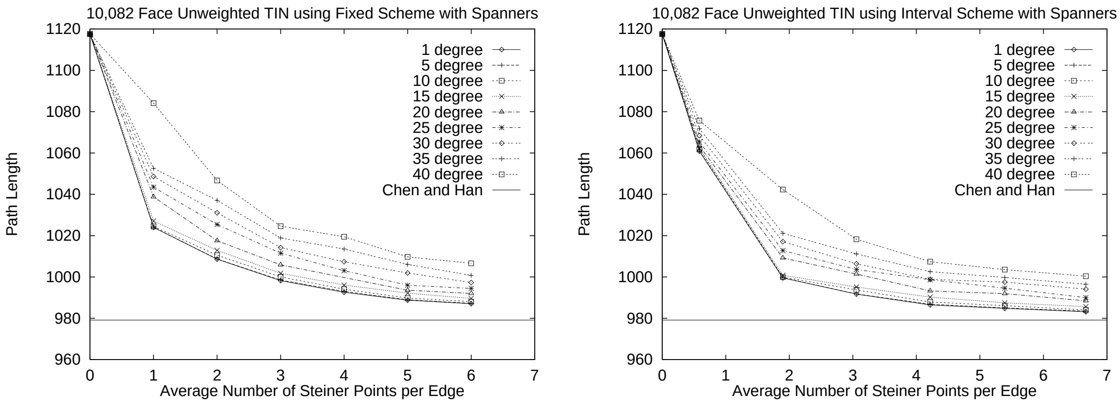

Figure 1.25: Graphs showing path accuracy obtained for a 10,082 face terrain using the fixed and interval schemes for a variety of spanner angles.

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 1.31 help determine the feasibility of the spanner scheme. The graphs show the path accuracy vs. computation time for the same data as Figure 1.25 and Figure 1.30. Here we can see which cone size provides the best path accuracy, when given 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 1.30 illustrates that the 10 and 15 degree spanner results are approximately

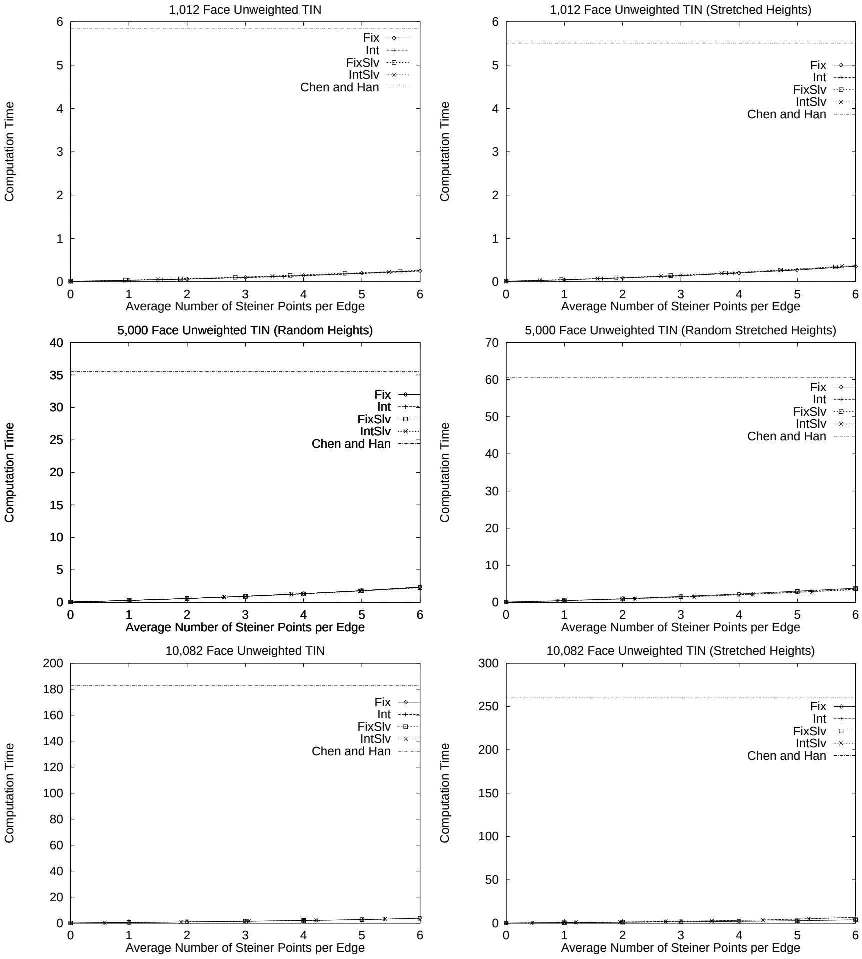

Figure 1.26: Graphs showing average computation time for 6 selected terrains.

0.75 to 1.0 second 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.

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

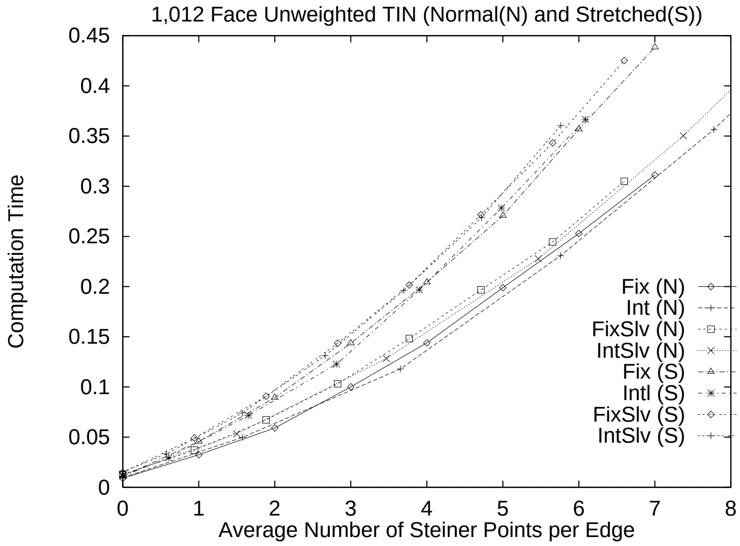

Figure 1.27: Graph showing the typical running time characteristics for the tested terrains using the fixed and interval schemes for normal(N) and stretched(S) terrains.

mation?

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 1.32 shows the accuracy obtained through experimentation on weighted terrains with and without the second stage approximation (using 20 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 1.33. 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 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

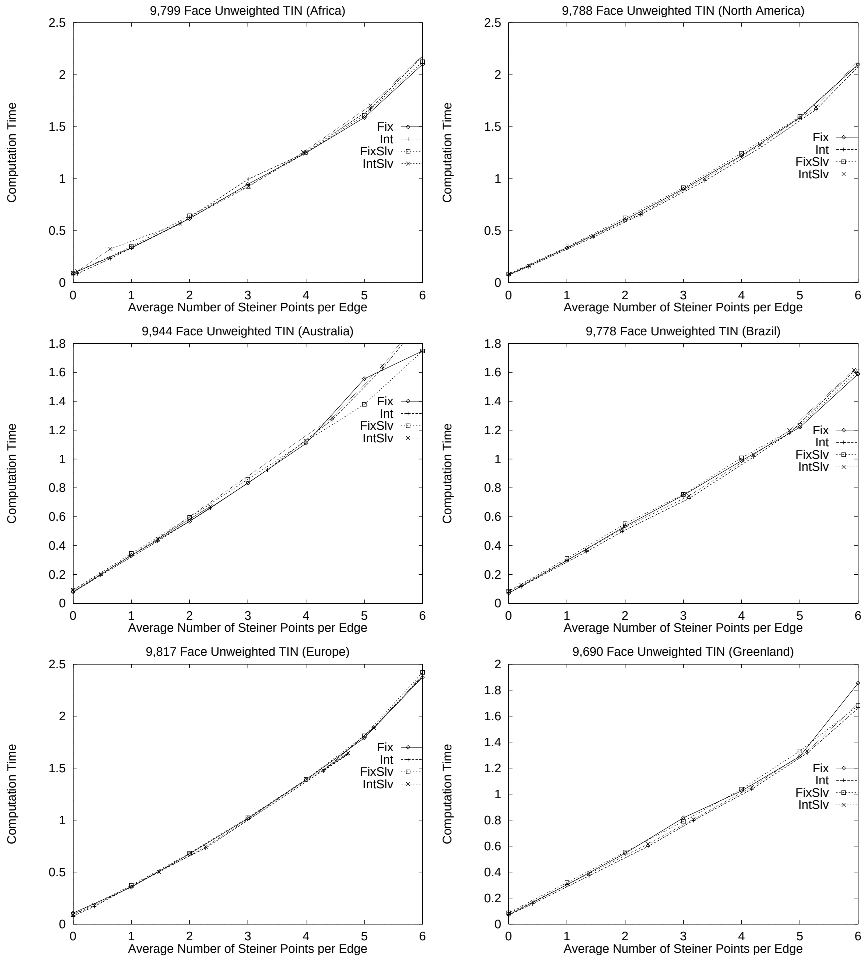

Figure 1.28: Graphs showing average computation time for six real-data terrains.

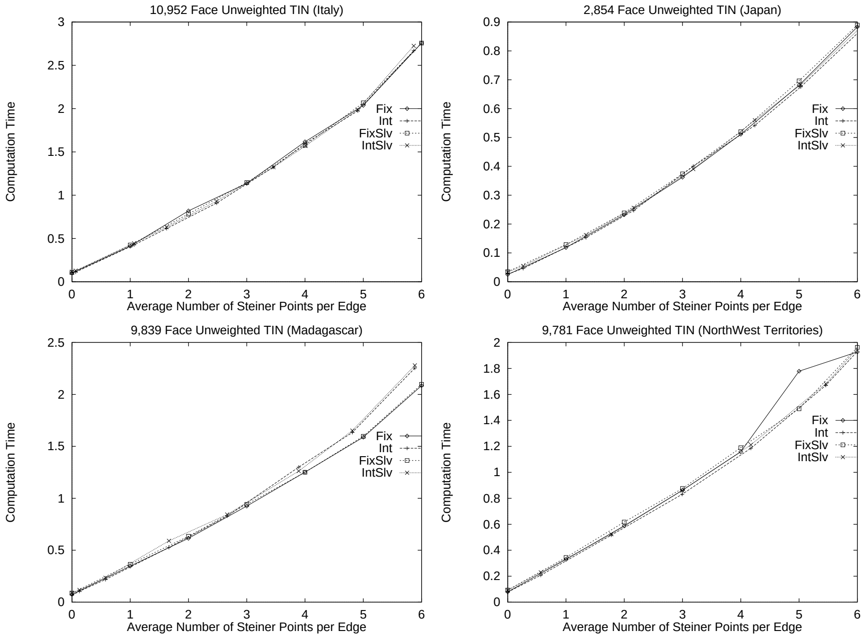

Figure 1.29: Graphs showing average computation time for four real-data terrains.

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 1.34 compares the weighted path costs between terrains with slope weights and random weights. The slope weight tests were re-done here with the new calculation for the second stage approximation. As can be seen by the graphs, there is very little difference between the shape and convergence behavior between the two weight scenarios. The timing for these additional tests was similar to that of the original weighted path tests of Figure 1.33, and is therefore not shown here.

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.

1.3 Extension

In this section, we describe an extension to our algorithms, namely that of computing paths between arbitrary query points on the polyhedral surface.

1.3.1 Shortest Path Queries

We consider two variations of the shortest path query problem: 1) fixed source, and 2) two-point-queries. For each, we describe the preprocessing and query times as well as the

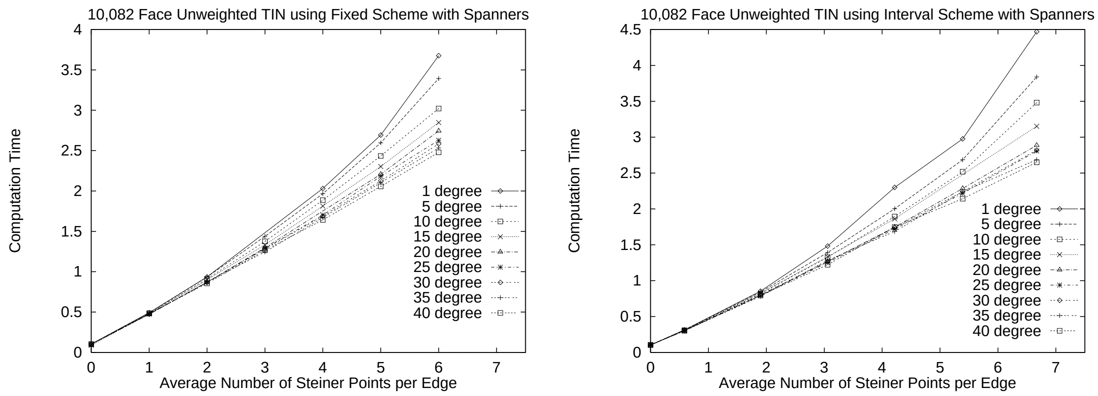

Figure 1.30: Graphs showing computation time obtained for a 10,082 face terrain using the fixed and interval schemes for a variety of spanner angles.

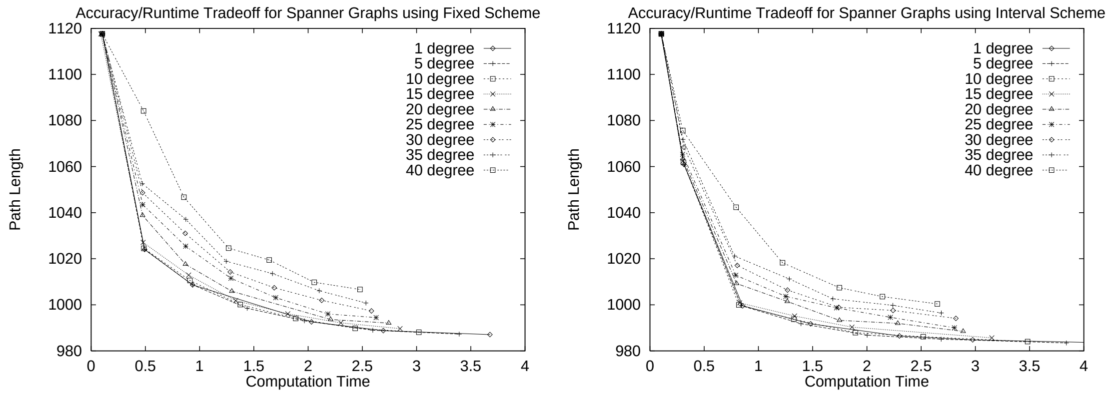

Figure 1.31: Graphs showing path accuracy vs. computation time obtained for a 10,082 face terrain using the fixed and interval schemes for a variety of spanner angles.

accuracy of the approximations.

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 2 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 pre-determined 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

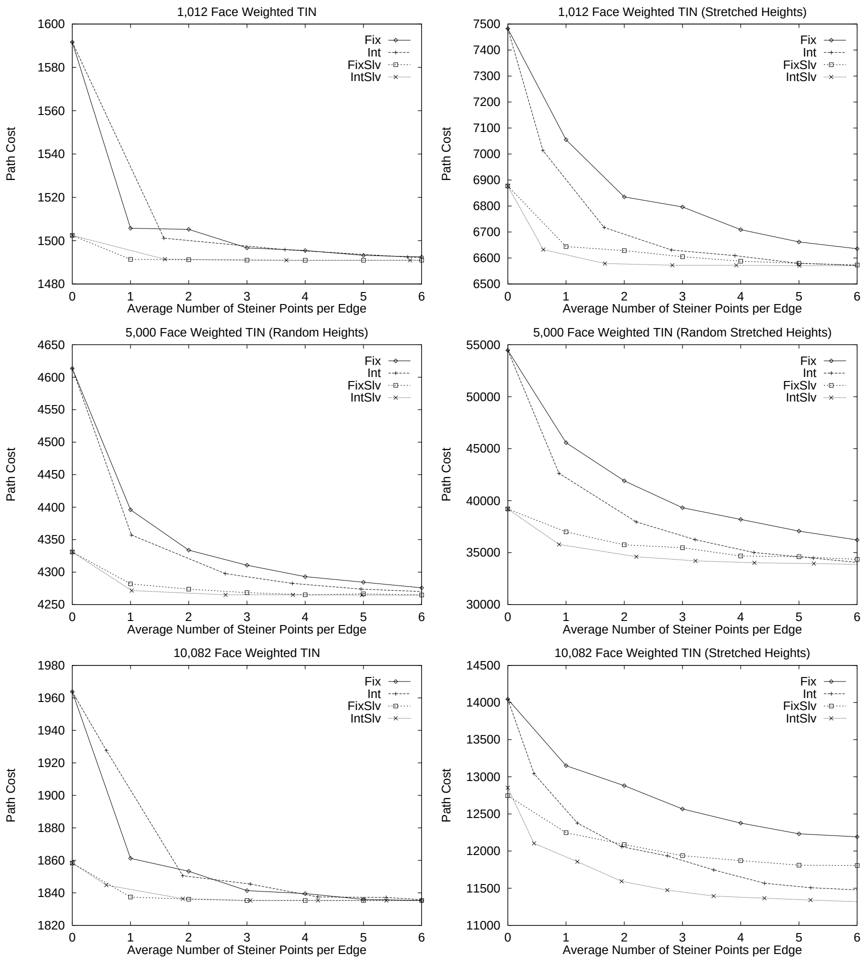

Figure 1.32: Graphs showing average path cost for six selected weighted terrains.

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. Assuming that we choose the first construction above, then we have the following theorem:

Theorem 1.3.1 *Given a polyhedron $\mathcal{P}$ and a source vertex s, we can preprocess $\mathcal{P}$ in $O(nm \log nm + nm^2)$ time such that fixed source queries to a query point t on $\mathcal{P}$, can be answered in $O(m + \log n)$ time with an approximation bound for $\Pi'(s,t)$ satisfying: $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1 + \frac{1}{m+1})W|L|$.*

Proof: The preprocessing time is $O(nm \log nm + nm^2)$. The query time follows from Observation 0.1.1 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 1.1.2 and

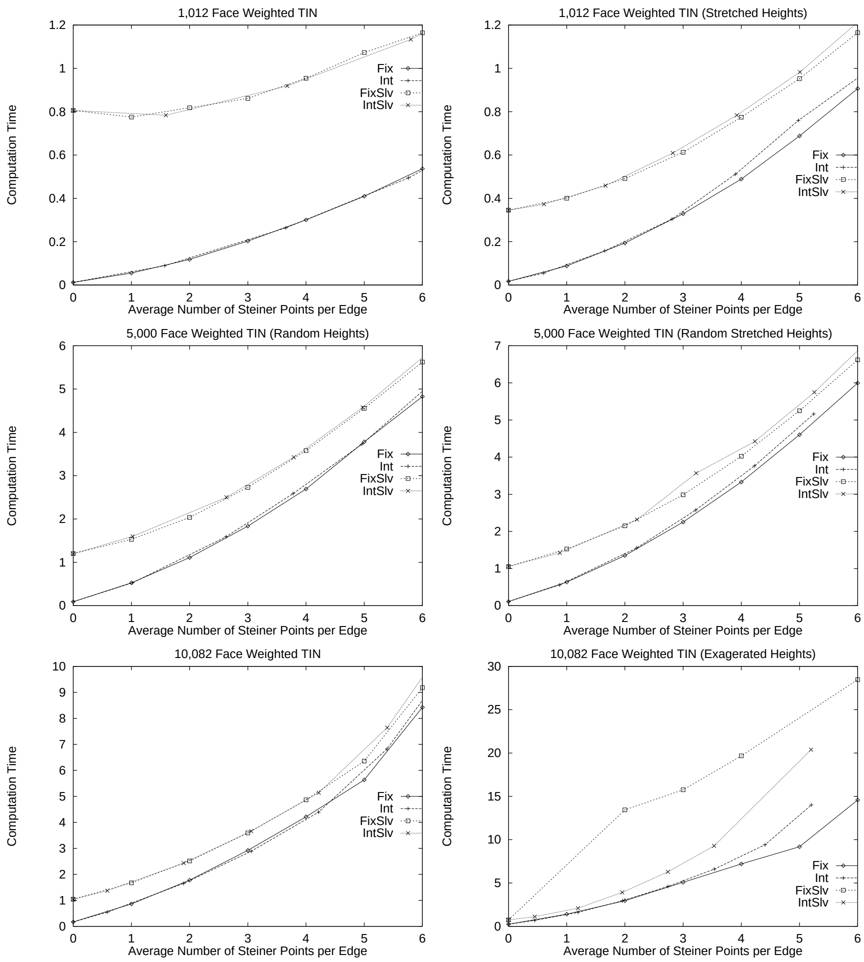

Figure 1.33: Graphs showing average computation time for selected weighted terrains.

triangle inequality we can bound the cost of $\Pi'(s,t)$ as follows:

$$
\begin{aligned}
\|\Pi'(s,t)\| &= \|\Pi'(s,b')\| + \|\overline{b't}\| \\
&\le \|\Pi(s,b')\| + W|L| + \|\overline{b't}\| \\
&\le \|\Pi(s,b)\| + \|\overline{bb'}\| + W|L| + \|\overline{b't}\| \\
&\le \|\Pi(s,b)\| + \|\overline{bb'}\| + W|L| + \|\overline{b'b}\| + \|\overline{bt}\| \\
&= \|\Pi(s,t)\| + 2\|\overline{bb'}\| + W|L| \\
&\le \|\Pi(s,t)\| + 2W\frac{|L|}{2(m+1)} + W|L| \\
&\le \|\Pi(s,t)\| + (1 + \frac{1}{m+1})W|L|
\end{aligned}
$$

$\square$

.

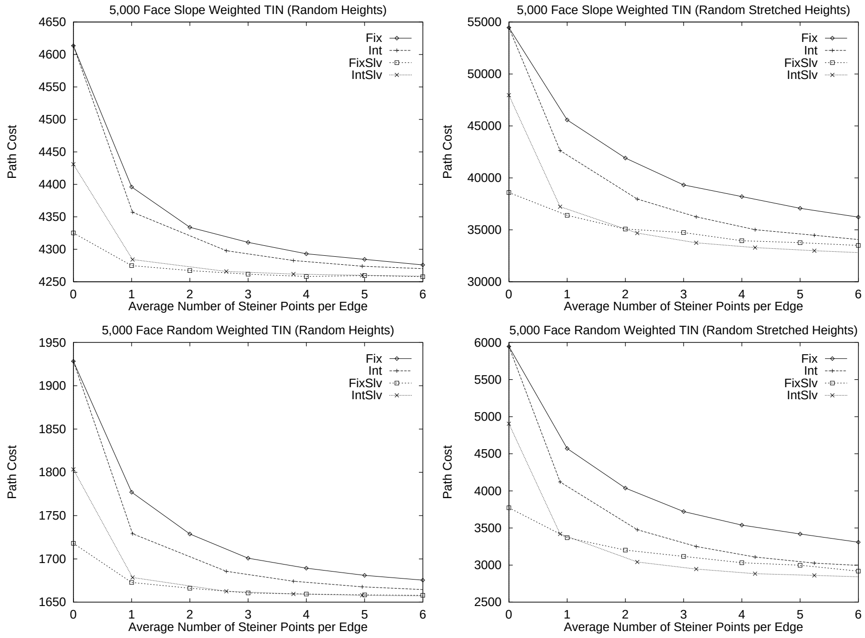

Figure 1.34: Graphs comparing the accuracy of weighted approximations for 2 weight scenarios: slope weights and random weights.

Let us assume that we chose the second construction method as discussed above. For the analysis, we will assume that v_t is the vertex of f_t that is closest to t.

Claim 1.3.1 *Consider the triangular face f_t containing the query point t, and let L be the longest edge among the edges of f_t. The following holds: $|\overline{v_t t}| \leq \frac{|L|}{\sqrt{3}}$.*

Proof: Observe that 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. $\square$

Lemma 1.3.1 $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1 + \frac{1}{\sqrt{3}})W|L|$

Proof: From Theorem 1.1.2 it follows that $\|\Pi'(s, v_t)\| \leq \|\Pi(s, v_t)\| + W|L|$, where v_t is the vertex of face f_t containing t that is closest to t. Applying this to the constructed path $\Pi'(s,t)$ we obtain:

$$\begin{aligned} \|\Pi'(s,t)\| &= \|\Pi'(s,v_t)\| + \|\overline{v_t t}\| \\ &\leq (\|\Pi(s,v_t)\| + W|L|) + \|\overline{v_t t}\| \\ &\leq \|\Pi(s,t)\| + \|\overline{v_t t}\| + W|L| + \|\overline{v_t t}\| \end{aligned}$$

By Claim 1.3.1 we have $|\overline{v_t t}| \leq \frac{|L|}{\sqrt{3}}$ and therefore

$$\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + w_{f_t}\frac{2|L|}{\sqrt{3}} + W|L|$$

$$\leq \|\Pi(s,t)\| + (1+\frac{2}{\sqrt{3}})W|L|$$

□

Theorem 1.3.2 *Given a polyhedron $\mathcal{P}$ and a source vertex s, we can preprocess $\mathcal{P}$ in $O(nm\log nm + nm^2)$ time such that fixed source queries to a query point t on $\mathcal{P}$, can be answered in $O(\log n)$ time with an approximation bound for $\Pi'(s,t)$ satisfying: $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1+\frac{2}{\sqrt{3}})W|L|$.*

Proof: The preprocessing time follows from Algorithm 2 and the query time follows from applying Observation 0.1.1 on n faces. The accuracy of the path follows from Lemma 1.3.1.

□

Corollary 1.3.1 *The preprocessing time in Theorems 1.3.1 and 1.3.2 is reduced to $O(nm\log nm)$ time in case the spanner scheme is used. The path accuracy becomes $\|\Pi'(s,t)\| \leq \beta(\|\Pi(s,t)\| + (1+\frac{1}{m+1})W|L|)$ and $\|\Pi'(s,t)\| \leq \beta(\|\Pi(s,t)\| + (1+\frac{2}{\sqrt{3}})W|L|)$, respectively.*

Two Point Queries

Now consider the case in which both s and t are given as query points. Preprocess the polyhedron by computing the shortest path from each vertex of $\mathcal{P}$ to all Steiner points. This requires $O(n^2m^2\log nm + n^2m^3)$ time when using the non-spanner schemes and $O(n^2m^2\log nm)$ time when using the spanner scheme. 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 $\Pi'(s,t)$ as one of the following paths concatenations:

1. $\overline{sq_s}, \Pi'(q_s,q_t), \overline{q_t t}$
2. $\overline{sv_s}, \Pi'(v_s,q_t), \overline{q_t t}$
3. $\overline{sq_s}, \Pi'(q_s,v_t), \overline{v_t t}$
4. $\overline{sv_s}, \Pi'(v_s,v_t), \overline{v_t t}$

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 of vertices to check the last case can be carried out in constant time. In all cases, an additional $O(\log n)$ time is required to locate f_s and f_t.

For the first case, using a similar proof as for Theorem 1.3.1, it can be shown that the resulting path satisfies $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1 + \frac{2}{m+1})W|L|$. For the second and third case, using similar proofs as for Theorem 1.3.1 and Theorem 1.3.2, it can be shown that $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1 + \frac{2}{\sqrt{3}} + \frac{1}{m+1})W|L|$. For the fourth case, it is also not difficult to show that $\|\Pi'(s,t)\| \leq \|\Pi(s,t)\| + (1 + \frac{4}{\sqrt{3}})W|L|$.

The analysis employing Steiner point schemes is similar. We summarize all results in Table 1.3. In the weighted scenario, m can be $\Omega(n^2)$ and for the unweighted scenario, it has a value of $\Omega(n)$ in the worst case.

Theorem 1.3.3 *Using our schemes, shortest path queries can be answered in the bounds stated in Table 1.3. The query time represents the time required to compute the approximated path cost. The first entry represents the cost in which the query points are vertices of $\mathcal{P}$, the two-point queries assume that s and t are arbitrary. The path accuracy shown is a factor, X, for the upper bound additive cost of $\Pi'(s,t)$. That is, $\|\Pi'(s,t)\| \leq \beta(\|\Pi(s,t)\| + X \cdot W|L|)$ where X is given in the table, β is 1 for the non-spanner case and $\beta > 1$ for the spanner case.*

Query Type	Preprocessing (no spanner)	Preprocessing (spanner)	Query Time	X
Vertex Queries	$O(nm \log nm + nm^2)$	$O(nm \log nm)$	$O(1)$	1
Fixed Source Vertex (1)	$O(nm \log nm + nm^2)$	$O(nm \log nm)$	$O(\log n + m)$	$1 + \frac{1}{m+1}$
Fixed Source Vertex (2)	$O(nm \log nm + nm^2)$	$O(nm \log nm)$	$O(\log n)$	$1 + \frac{2}{\sqrt{3}}$
Two Point Query (1)	$O(n^2m^2 \log nm + n^2m^3)$	$O(n^2m^2 \log nm)$	$O(\log n + m^2)$	$1 + \frac{2}{m+1}$
Two Point Query (2,3)	$O(n^2m^2 \log nm + n^2m^3)$	$O(n^2m^2 \log nm)$	$O(\log n + m)$	$1 + \frac{2}{\sqrt{3}} + \frac{1}{m+1}$
Two Point Query (4)	$O(n^2m^2 \log nm + n^2m^3)$	$O(n^2m^2 \log nm)$	$O(\log n)$	$1 + \frac{2}{\sqrt{3}}$

Table 1.3: Running time and path accuracy for variations of the shortest path problem.

1.4 Other Approximation Schemes

All of the schemes presented here have been based on the discretization of $\mathcal{P}$ by constructing a graph in which Steiner points have been added to the edges of $\mathcal{P}$ and then interconnected. It has also been shown that in one particular scheme (the face decomposition scheme), the addition of Steiner points internal to the faces did not provide an improvement in accuracy and in fact, produced poor approximations. This however, does not imply that the addition of any Steiner points to the interior of each face will result in a poor approximation. In fact, we believe that the problem of determining the optimal placement of these Steiner points, and their connection via edges, is NP hard. Nevertheless, it would be nice to determine some "rules of thumb" that would give some indication as to which schemes would perform better than others.

An even more interesting problem is to determine the optimal placement of these Steiner points and edges such that each G_i is planar. It is not difficult to see that no planar scheme approximation can "beat" the fixed scheme approximation if both schemes have the same number of Steiner points along the edges. Hence given a fixed number of Steiner points on the edges of each face, adding Steiner points inside the face can never improve the overall accuracy of the approximation. However, it is possible that by adding points inside the face, then one can produce a graph on the outer Steiner points with much less edges and better characteristics (such as planarity). The accuracy obtained from such a graph may be close to that of the fixed scheme and the running time would be improved upon. In addition, by using planar graphs on each face, this may allow the application of more efficient search algorithms. Chew [8] for example, showed that there are some planar graphs which nearly approximate the complete graph and allow the edges to be reduced from $O(n^2)$ to $O(n)$. The planar graphs with these properties are variations of Delauney triangulations.

For these reasons, we ran some experiments on two planar schemes as shown in Figure 1.35.

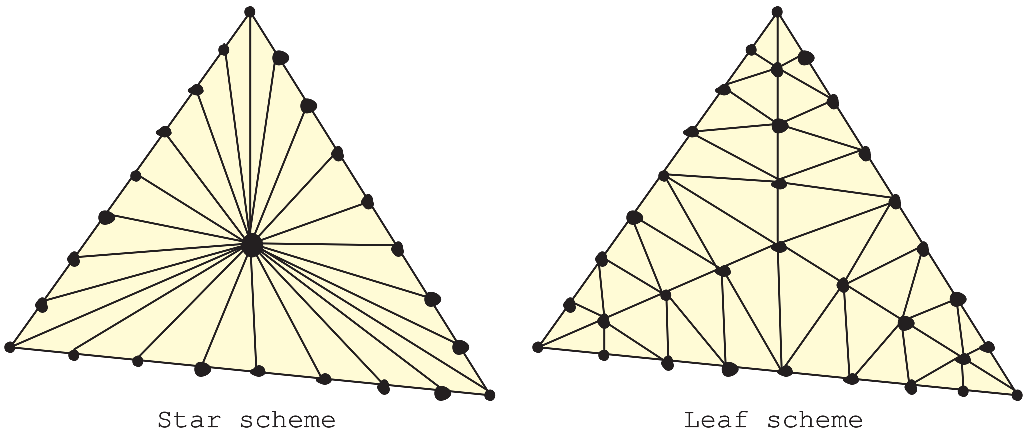

Figure 1.35: The star and leaf approximation schemes.

The *star* approximation scheme was created as an attempt to drastically reduce the number of edges added to each face from $O(m^2)$ to $O(m)$ while ensuring planarity. Here we merely place a Steiner point in the center of each triangle and connect it to the outer edge Steiner points. In essence, the scheme "forces" the approximated path to either pass through a vertex or center of a face. The *leaf* approximation scheme was created as an attempt to improve the star scheme and still allow only $O(m)$ edges per face. Once again, the graph produced on each face is planar. From the results of the fixed scheme, we knew that the biggest improvement on accuracy was the addition of the first Steiner point on each edge of the face. The leaf scheme attempts to keep the connections of the fixed scheme (with exactly one Steiner point per edge), while adding internal vertices to make the graph planar. Then, approximately $\frac{m}{2}$ additional Steiner points are added to 3 of the internal edges and these points are connected to the outer Steiner points so that the resulting graph is a triangulation (although a triangulation is not required by the algorithm).

The graphs of Figure 1.36 compare the accuracy of the paths obtained by the star and leaf schemes with the accuracy of the fixed and interval schemes for our 10,082 face TIN. The graphs confirm that the star and leaf schemes produce worse accuracy than the fixed and interval schemes with and without the additional sleeve computation. The erratic nature of the star scheme indicates that the scheme is susceptible to small changes in Steiner point placement along the edges of the face. More importantly, the star scheme does not improve much as more Steiner points are added per edge. The second graph confirms that the sleeve computation provides a substantial improvement in the accuracy, but still, the resulting path accuracy never reaches that of the fixed and interval schemes. This implies that the schemes were unable to converge to a proper shortest path edge sequence. However, it is observed that the leaf scheme is less sensitive to Steiner point placement and does converge slightly better than the star scheme.

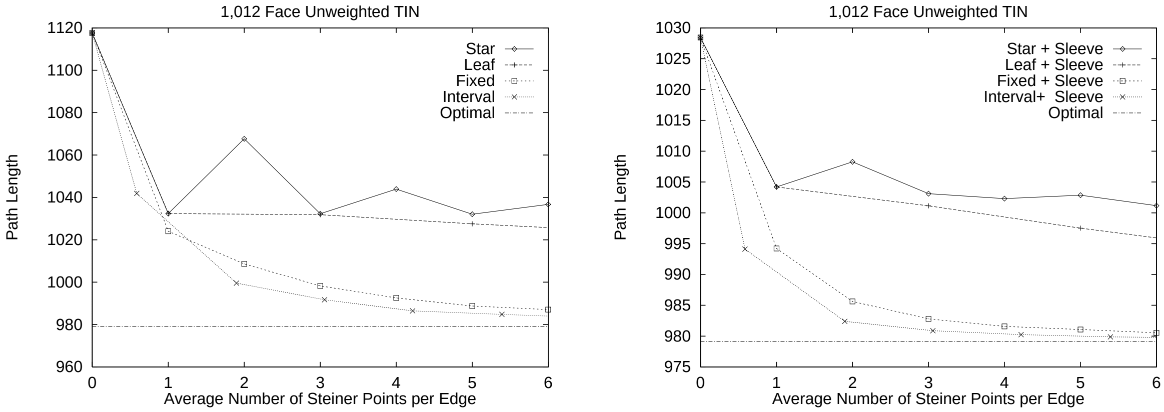

Figure 1.36: Graphs comparing the accuracy of the star and leaf schemes with that of the fixed and interval schemes. The two graphs depict the results with and without the additional sleeve computation.

Getting back to the reason for creating these schemes, we see the running time results for the tests in Figure 1.37. Here we see the linear behaviour of these new schemes as compared to the quadratic behaviour of the fixed and interval schemes. Clearly, the star scheme has the best running time. Since we have only added a few Steiner points per edge, the leaf scheme is shown to have a higher running time than the fixed and interval schemes. If enough points were added, then one would see that the fixed/interval scheme running time would surpass that of the leaf scheme.

Although we've only experimented with two additional schemes, further investigation into other schemes remains as future work. For example, one may like to try schemes such as those depicted in Figure 1.38. The scheme of Figure 1.38a) is similar to the star scheme except that the "ears" of the face are sectioned off and triangulated separately. This may prove more useful than the star scheme since there are less long edges close together. Another possibility is that of perhaps merging the star and face decomposition schemes as shown in Figure 1.38b). This would provide additional edge orientations for the face decomposition scheme (as opposed to its original 6) which would decrease the amount of zig-zagging and actually improve the accuracy of the face decomposition scheme. Figure 1.38c) would provide another method in increasing the orientations of the face decomposition scheme. It represents

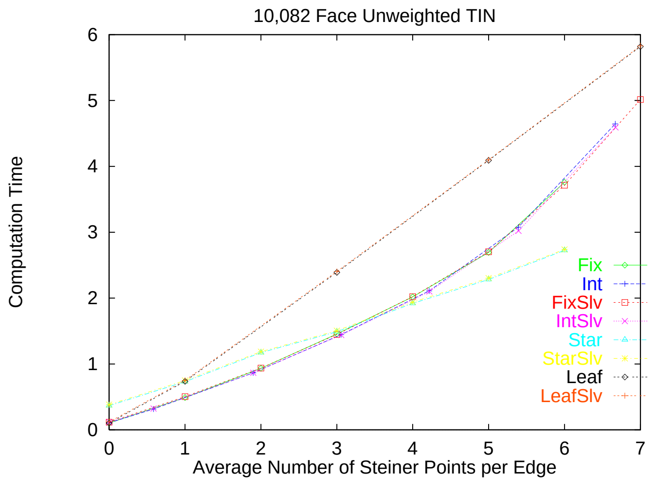

Figure 1.37: Graph comparing the running time of the star and leaf schemes with that of the fixed and interval schemes.

a 12-neighbourhood graph where each Steiner point is connected to its (up to) 12 closest neighbours. The resulting graph allows 12 orientations from each node and hence should improve upon the face decomposition scheme. However, the graph is no longer planar. Finally, Figure 1.38d) represents a layered scheme where a layer of Steiner points is placed inside the face forming a similar but smaller triangle. We add half the number of border Steiner points inside the face to form an inner triangle and connect them using the fixed scheme. The outer and inner triangles are then connected using some kind of triangulation strategy. Although this is described as a two layer approach, the layering can be iterated to allow many layers with a final small triangle (perhaps even a point) in the center of the layers. This particular scheme spawned from some ideas we had regarding the placement of Steiner points based on using the "spring" graph drawing algorithm. The idea was to determine a "good" placement of Steiner points interior to the face. Intuitively, we felt that by applying the spring algorithm in which the Steiner points along the outer edges of the face were fixed and the Steiner points on the inner layer were allowed to move, then we would get an indication as to where the Steiner points should be placed. After applying the spring algorithm on several triangles, we found that the Steiner points near the middle of the inner layer edges seemed to cave inwards slightly. We felt that the effects of the spring algorithm did not provide a significant change in the placement of the inner Steiner points to warrant their usefulness.

1.5 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

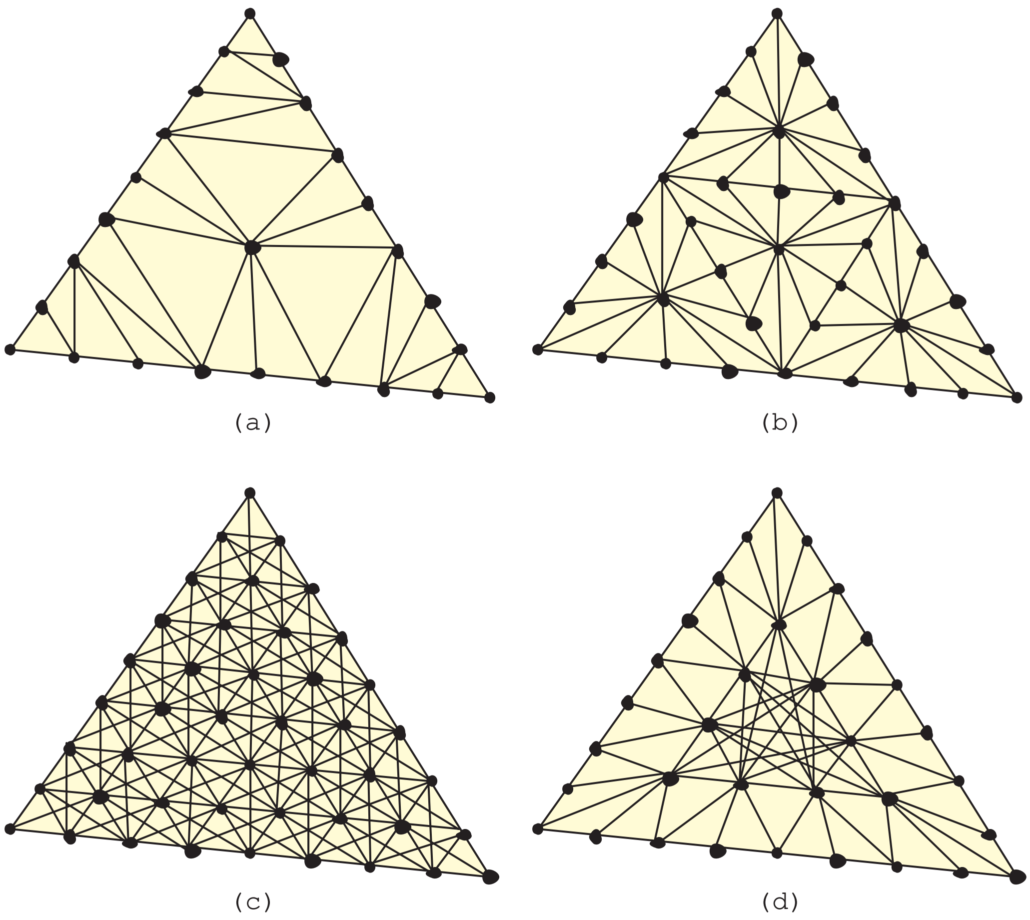

Figure 1.38: a) an ear-clipped star scheme, b) a combined face decomposition and star scheme, c) a 12 neighbourhood scheme and d) a layered scheme.

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 [7] 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 [23]. An improved ϵ-approximation algorithm is described in [4]. Furthermore, the ϵ-approximation scheme for queries is discussed in [3].

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. It remains an open problem as to whether or not it is advantageous to place Steiner points within the interior of the polyhedral faces. We also ran tests on general polyhedral models (based on 3D data) in order to further verify the schemes.

Several 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 [23], 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 [19]. 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.

Bibliography

[1] P.K. Agarwal, S. Har-Peled, M. Sharir, and K.R. Varadarajan, "Approximating Shortest Paths on a Convex Polytope in Three Dimensions", *Technical Report CS-1996-12*, Department of Computer Science, Duke University, Durham, 1996.

[2] P.K. Agarwal and K.R. Varadarajan, "Approximating Shortest Paths on a Polyhedron",*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] L. Aleksandrov, M. Lanthier, A. Maheshwari and J.-R. Sack, "An ϵ-Approximation Algorithm for Weighted Shortest Paths on Polyhedral Surfaces", to appear in *6th Scandinavian Workshop on Algorithm Theory*, Stockholm, Sweden, July 1998.

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

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

[7] J. Chen and Y. Han, "Shortest Paths on a Polyhedron", *Proc. of the 6th ACM Symposium on Computational Geometry*, 1990, pp. 360-369.

[8] L. Paul Chew, "There Are Planar Graphs Almost as Good as the Complete Graph", *Journal of Computer and System Sciences*, 39, 1989, pp. 205-219.

[9] J. Choi, J. Sellen and C.K. Yap, "Approximate Euclidean Shortest Path in 3-Space", *Proc. 10th Annual Symp. on Computational Geometry*, Stony Brook, NY, 1994, pp. 41-48.

[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] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] M. Lanthier, "Approximating Weighted Shortest Paths on Polyhedral Surfaces", *Ph.D. Thesis in progress*, School of Computer Science, Carleton University, Ottawa, Canada, 1998.

[20] 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.

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

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

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

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

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

[30] 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 - Amercian Congress on Survery and Mapping,* pp. 24-31, (1978).

[31] F.P. Preparata and M.I. Shamos, "Computational Geometry: An Introduction", Springer-Verlag, 1985.

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

[33] N.C. Rowe, and R.S. Ross, "Optimal Grid-Free Path Planning Across Arbitrarily Contoured Terrain with Anisotropic Friction and Gravity Effects", *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 5, October 1990, pp. 540-553.

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

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

[36] Paradigm Group Webpage, School of Computer Science, Carleton University, http://www.scs.carleton.ca/˜ gis.