ELSEVIER

Computational Geometry 10 (1998) 31–46

Computational Geometry Theory and Applications

Practical methods for approximating shortest paths on a convex polytope in $\mathbb{R}^3$ [☆]

John Hershberger [a,*], Subhash Suri [b]

[a] *Mentor Graphics, 1001 Ridder Park Drive, San Jose, CA 95131, USA*

[b] *Department of Computer Science, Washington University, St. Louis, MO 63130, USA*

Communicated by J.-R. Sack; submitted 3 February 1996; accepted 20 December 1996

Abstract

We propose an extremely simple approximation scheme for computing shortest paths on the surface of a convex polytope in three dimensions. Given a convex polytope P with n vertices and two points p, q on its surface, let $d_P(p,q)$ denote the shortest path distance between p and q on the surface of P. Our algorithm produces a path of length at most $2d_P(p,q)$ in time $\mathrm{O}(n)$. Extending this result, we can also compute an approximation of the shortest path tree rooted at an arbitrary point $x \in P$ in time $\mathrm{O}(n \log n)$. In the approximate tree, the distance between a vertex $v \in P$ and x is at most $cd_P(x,v)$, where $c = 2.38(1+\varepsilon)$ for any fixed $\varepsilon > 0$. The best algorithms for computing an exact shortest path on a convex polytope take $\Omega(n^2)$ time in the worst case; in addition, they are too complicated to be suitable in practice. We can also get a weak approximation result in the general case of k disjoint convex polyhedra: in $\mathrm{O}(n)$ time our algorithm gives a path of length at most $2k$ times the optimal. © 1998 Elsevier Science B.V.

Keywords: Shortest path; Approximation; Convex polyhedron; Shortest path tree

1. Introduction

Shortest paths are an important topic of research in computational geometry, in part due to their natural applications in robotics and motion planning. One of the best-studied problems in this area is to compute Euclidean shortest paths in an environment containing polyhedral obstacles. In two dimensions, an algorithm with time complexity $\mathrm{O}(n^2 \log n)$ has been known since the late seventies [12,17], based on the notion of a "visibility graph"; n is the total number of vertices in all the obstacles. Despite numerous attempts to improve the worst-case time complexity, the problem was not solved until very recently, when Hershberger and Suri [10] obtained an optimal $\mathrm{O}(n \log n)$ time algorithm.

[☆] A preliminary version of this paper appeared in the Proceedings of the 6th Annual ACM–SIAM Symposium on Discrete Algorithms, 1995.

[*] Corresponding author.

PII S0925-7721(97)00004-7

The three-dimensional version of the shortest path problem turns out to be much more difficult. The problem is no longer even "discrete" because a shortest path may turn at interior points of polyhedral edges, and thus has a continuum of possible "corners". In fact, the problem has two separate sources of difficulty: the *combinatorial component*, which requires computing the sequence of polyhedral edges touching a shortest path, and the *algebraic component*, which requires computing the actual contact points given the sequence of edges touching a shortest path. In 1986, Bajaj [3] showed the difficulty of the algebraic subproblem by proving that, in the worst case, the contact points are defined by high-degree irreducible polynomials. A little later, Canny and Reif [4] proved that the combinatorial subproblem is NP-hard, and thus even computing the edge-sequence of a shortest path is intractably difficult.

It is therefore natural to consider approximation algorithms for the three-dimensional shortest path problem. The first general result in this direction is due to Papadimitriou [15], who gave an $O(n^4(L + \log(n/\varepsilon))^2/\varepsilon^2)$ time algorithm for computing a path of length $(1 + \varepsilon)$ times the optimal; L is the number of bits of precision in the model of computation. Another algorithm, due to Clarkson [7], computes a $(1+\varepsilon)$-optimal path in roughly $O(n^2 \mathrm{polylog}\, n/\varepsilon^4)$ time. Recently, Choi et al. [6] tightened the analysis of Papadimitriou's algorithm, removing several gaps and inconsistencies in the process.

In this paper, we address the problem of finding a shortest path on the surface of a convex polytope in three dimensions. Despite its apparent simplicity, the shortest path problem for convex polytopes turns out to be highly nontrivial and rich in mathematical lineage. Shortest paths on a polyhedron (convex or nonconvex) possess a pleasing property called *unfolding*, which allows one to compute an optimal path in polynomial time. As a result, this special case of the shortest path problem has received considerable attention in computational geometry. (Within mathematics as well, the unfolding of convex polyhedra is a rich topic with a distinguished history – see, for instance, [1].)

Our main result is a very simple and practical algorithm for computing an approximation to the shortest path. Formally, let P denote a convex polytope with n vertices in $\mathbb{R}^3$, and let $d_P(p, q)$ denote the length of a shortest path on the surface of P between the points p and q. The quantity $d_P(p, q)$ is also called the *geodesic distance* between p and q. We propose an $O(n)$ time algorithm for computing a path of length at most $2d_P(p, q)$ for any two points p and q on the surface of P. Extending this result, we design an $O(n \log n)$ time algorithm to approximate the geodesic distances between a fixed source point x and all the vertices of P. The latter can be viewed as an approximation of the *geodesic tree* of P rooted at x.

We regard simplicity and efficiency as the main virtues of our algorithms. Our algorithm computes a shortest path between two points in $O(n)$ time, contrasting with the $\Theta(n^2)$ time required by all known algorithms for computing an exact shortest path [5,13,14]. Many applications (e.g., interactive) simply cannot afford quadratic time for computing an exact shortest path. Having a fast and simple algorithm that returns a good path, with a provable worst case guarantee, is not only quite adequate, but can be desirable in these applications. Likewise, a fast approximation algorithm may be used as a filter for an exact algorithm: source/destination pairs may be selected from among several candidates based on their approximate shortest path lengths, with exact distances computed only for pairs that meet certain length criteria. While the worst-case approximation ratio of our algorithm is two, we suspect that except for pathological instances, the actual ratio is much better, so the penalty of approximation is quite low. Additionally, in cases where the input polytope is already an approximation, a slower but exact algorithm ultimately doesn't do much good either.

Most exact algorithms (and our approximation algorithm as well) rely on the technique of "unfolding" to compute a shortest path. Unfolding, while simple and elegant in concept, can be numerically expensive because it's essentially a rotation in 3-space. Exact algorithms can perform chains of unfolding involving as many as $\Theta(n)$ faces, leading to numerical instability. By contrast, our algorithm unfolds chains of at most *three* faces, which may significantly limit potential numerical problems.

This paper has five sections. In Section 2 we show that the bounding box, a common practical device, is a bad approximation tool for shortest paths in the worst case. Sections 3 and 4 describe our algorithms for approximating a single path and a shortest path tree, respectively. We finish with some closing remarks in Section 5.

2. Where bounding boxes fail

The *bounding box* is a valuable concept that is also used widely in practice. Given a set of points $S \in \mathbb{R}^3$, its bounding box is the smallest axis-parallel rectangular solid containing S. Because a bounding box has only a constant number of vertices, edges and faces, many practical algorithms use it as a first-order approximation for S. Particularly in robotics and motion planning, the use of bounding boxes is often unavoidable because of the immense combinatorial complexity of the "configuration space". So, for instance, a typical algorithm for intersection-detection between polyhedra uses bounding boxes to quickly winnow out large numbers of pairs that do not intersect.

Given the widespread use of bounding boxes, it is only natural to ask if they are also good for approximating shortest paths on convex polytopes. Formally, let P denote a convex polytope in $\mathbb{R}^3$ with n vertices. Let B denote the bounding box of P. Given two points p, q on the surface of P, we want to know if the shortest path between p and q on the surface of B is a close approximation of the shortest path on P. We need a (trivial) modification of B to ensure that p, q also lie on B. Let H_p and H_q denote two planes containing p and q, respectively, and supporting P on one side. Let H_p^+ and H_q^+ denote the halfspaces defined by these planes containing the polytope P. Let

$$B(p,q) = B \cap H_p^+ \cap H_q^+.$$

We call $B(p,q)$ the *bounding box of P for the pair* (p,q). Observe that $B(p,q)$ has at most two more faces than B and, of course, $P \subseteq B(p,q)$. Let $d_B(p,q)$ denote the shortest path distance between p and q on the surface of $B(p,q)$. The question we want to address is this: what is the least upper bound on the ratio $d_B(p,q)/d_P(p,q)$ in the worst case?

It can easily be shown that in two dimensions (shortest paths on a convex polygon) the ratio $d_B(p,q)/d_P(p,q)$ is bounded by $\sqrt{2}$. Unfortunately, the bounding box stops being a useful approximation tool for shortest paths in three dimensions. The following lemma shows that the approximation ratio is unbounded.

Lemma 2.1. *For any $\varepsilon > 0$, there exist a convex polytope $P \in \mathbb{R}^3$ and two points $p, q \in P$ such that*

$$\frac{d_P(p,q)}{d_B(p,q)} < \varepsilon.$$

Proof. The construction is shown in Fig. 1. The polyhedron P is a thin, rectangular tube whose cross section is a square with side length $\varepsilon/2$. P is oriented at 45° from the origin so that its major axis

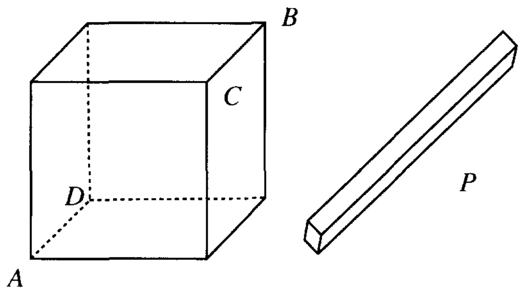

Fig. 1. A bad example for approximation via the bounding box.

coincides with one of the major diagonals AB of the unit cube. The source and destination points p and q lie on the faces opposite vertices C and D, respectively, so that $d_P(p,q) \leqslant \varepsilon$.

The bounding box $B(p,q)$ is obtained by slicing the cube with planes parallel to the top and bottom faces of P. It is easily checked that the worst-case length of a shortest path from p to q on $B(p,q)$ is greater than 1. The ratio $d_P(p,q)/d_B(p,q)$ is therefore at most ε, which proves the lemma. □

One may try to salvage the bounding-box idea by considering more carefully constructed "bounding boxes". A more clever choice of a bounding box, for instance, may be the following: project P on each of the three axis planes; compute a smallest enclosing rectangle (not necessarily axis-parallel) for the projection in each plane; let B be the common intersection of three prisms erected on these rectangles. This modified box solves the case in Fig. 1. Another alternative is simply to choose a smallest (volume-wise) rectangular solid containing P, without regard to coordinate axes; a caveat to this last suggestion is that computing such a box itself is a nontrivial problem. By replacing the long, thin rectangular polytope P of Fig. 1 by a disc-shaped polytope resembling a "thickened" regular n-gon, we can show that any constant-size approximating polyhedron whose shape is *independent* of the *position* of source and destination points on their faces fails to give a good bound on the approximation in the worst case.

3. Approximating a shortest path

3.1. Definitions and conventions

Let P be a convex polytope with n vertices in $\mathbb{R}^3$. We are interested in computing shortest paths on the surface of P. Unless otherwise stated, the notation $x \in P$ means a point x lying on the surface of P. Given two points $p, q \in P$, a shortest path between them on the surface of P is denoted $\pi_P(p,q)$, and its length is defined to be the *shortest path distance* of p and q on P, denoted $d_P(p,q)$.

A key geometric concept in our approximation algorithm is the notion of a *wedge* defined by two or three planes. In order to introduce the idea of a wedge, we first need a few more definitions. By convention, for any plane H not properly intersecting P, the positive halfspace H^+ defined by H is the one *containing* P. Furthermore, the normal η for a plane H is assumed to be directed into the positive halfspace H^+.

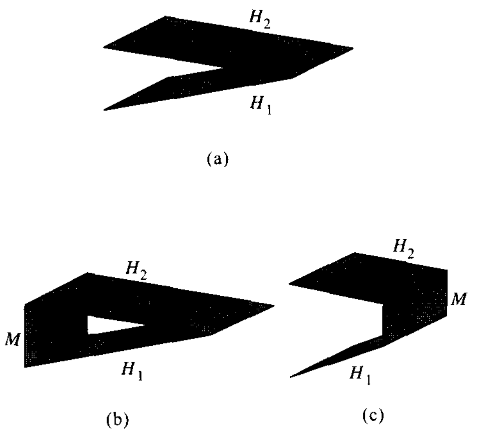

Fig. 2. (a) A 2-plane wedge, (b) and (c) two possible 3-plane wedges.

Let H_1, H_2 be two planes, each containing P in its positive halfspace. The *dihedral angle* formed by H_1 and H_2 in their *positive quadrant* is denoted $\alpha(H_1, H_2)$. A *2-plane wedge* $W(H_1, H_2)$ is the two-dimensional surface of the (unbounded) convex polyhedron $H_1^+ \cap H_2^+$; this convex polyhedron is the positive quadrant formed by H_1, H_2. A *3-plane wedge* $W(H_1, M, H_2)$ is the surface of the convex polyhedron $H_1^+ \cap M^+ \cap H_2^+$ where the plane M satisfies the additional condition $\alpha(H_1, M) = \alpha(M, H_2)$; that is, M makes equal dihedral angles with H_1 and H_2. Fig. 2 illustrates these wedges.

3.2. Horizon edges and planes

Let f_p and f_q denote the faces of P containing the source p and the destination q, respectively; if either point lies on more than one face, we arbitrarily pick one. Let H_p and H_q be the planes defined by the faces f_p and f_q, and let η_p, η_q be the corresponding unit normals. (H_p, H_q are the affine spans of the faces f_p and f_q, respectively.) Let $\hat{\imath}_z$ be the unit vector corresponding to the positive z-axis. We rotate the coordinate system so that

$$\eta_p \cdot \hat{\imath}_z = -\eta_q \cdot \hat{\imath}_z;$$

i.e., the normals η_p and η_q make complementary angles with the vertical axis. (This also means that H_p and H_q make equal dihedral angles with any plane parallel to $\hat{\imath}_z$.) Our 3-plane wedges will be formed by H_p, H_q, and a plane parallel to $\hat{\imath}_z$. In the following, we describe a minimal family of these latter planes.

In order to simplify our discussion, we assume that no face of P is vertical; this condition is easily enforced, if necessary, by a symbolic perturbation (rotation) of the coordinate system. Thus, the face-normals for all the faces of P have nonzero dot product with $\hat{\imath}_z$. Call a face f positive if $\eta_f \cdot \hat{\imath}_z > 0$, and negative otherwise. Let $\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$ be the set of edges forming the boundary between the positive and negative faces of P. We call these edges the *horizon* edges of P. Let H_i be a vertical plane passing through the edge e_i, and define $\mathcal{H} = \{H_1, H_2, \ldots, H_k\}$, where $k \leqslant n$. We call elements of $\mathcal{H}$ the *horizon planes* of P. (Observe that the polyhedron $\bigcap_{i=1}^{k} H_i^+$ is a vertical prism whose intersection with a horizontal plane is the vertical shadow of P.) Each of the planes H_i also satisfies $\alpha(H_p, H_i) = \alpha(H_i, H_q)$, and therefore determines a valid 3-plane wedge with H_p and H_q.

In the following subsection, we describe our approximation algorithm for computing a shortest path on P. The algorithm computes a path on a wedge, either a 2-plane or a 3-plane wedge, in the exterior of P. In Section 3.5, we describe a procedure for mapping this path onto the surface of P without increasing its length.

3.3. *The algorithm*

P is a convex polytope in $\mathbb{R}^3$, and we want to approximate the shortest path distance between two points $p, q \in P$.

```
ALGORITHM SHORTESTPATH

1. If the dihedral angle $\alpha(H_p, H_q) \geqslant \pi/3$, then compute the shortest path distance
   $d_0(p,q)$ between $p$ and $q$ on the 2-plane wedge $W(H_p, H_q)$. Output $d_0(p,q)$ and
   stop.
2. For each plane $H_i$ in the horizon family $\{H_1, H_2, \ldots, H_k\}$ do
      Compute the shortest path distance $d_i(p,q)$ between $p$ and $q$ on the 3-plane
      wedge $W(H_p, H_i, H_q)$.
3. Output $\min_{1 \leqslant i \leqslant k} d_i(p,q)$ and stop.
```

3.4. *Proof of correctness*

We begin with an elementary geometric lemma, which forms a crucial part of our proof.

Lemma 3.1. *Let ℓ_1 and ℓ_2 be two rays originating from a point o, making an angle θ at o, where $\theta < \pi$, and let $p \in \ell_1$ and $q \in \ell_2$ be two arbitrary points. Then we have the following bound:*

$$\frac{\overline{op} + \overline{oq}}{\overline{pq}} \leqslant \frac{1}{\sin(\theta/2)}.$$

Equality holds if and only if $\overline{op} = \overline{oq}$.

Proof. If $\overline{op} = \overline{oq}$, then clearly $\overline{op} + \overline{oq} = \overline{pq}/\sin(\theta/2)$. Let us therefore assume, without loss of generality, that $\overline{oq} \neq \overline{op}$, and that the horizontal line ox bisects the angle $\angle poq$. Let s denote the point where pq intersects ox. See Fig. 3 for illustration. We have the following inequalities:

$$\frac{\overline{op} + \overline{oq}}{\overline{pq}} = \frac{\overline{op} + \overline{oq}}{\overline{ps} + \overline{sq}} \leqslant \max\left(\frac{\overline{op}}{\overline{ps}}, \frac{\overline{oq}}{\overline{sq}}\right). \tag{1}$$

Assume, without loss of generality, that $\overline{op}/\overline{ps} \leqslant \overline{oq}/\overline{qs}$. Let r denote the reflection of q in the line ox; thus, $\overline{qs} = \overline{rs}$.

$$\frac{1}{\sin(\theta/2)} = \frac{\overline{or} + \overline{oq}}{\overline{rq}} > \frac{\overline{or} + \overline{oq}}{\overline{rs} + \overline{sq}} = \frac{\overline{oq}}{\overline{sq}} \geqslant \frac{\overline{op} + \overline{oq}}{\overline{pq}},$$

where the last inequality follows from (1). This completes the proof. □

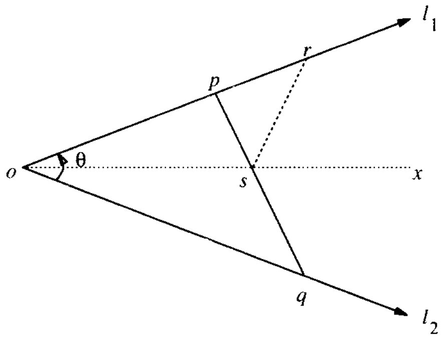

Fig. 3. Illustration for the proof of Lemma 3.1.

The following lemma shows that the shortest path distance $d_O(p,q)$ computed in Step 1 of the algorithm gives the desired approximation.

Lemma 3.2. $d_O(p,q)/d_P(p,q) \leqslant 1/\sin(\alpha/2)$, *where* $\alpha = \alpha(H_p, H_q)$.

Proof. The proof depends on two facts: first, the shortest path $\pi_O(p,q)$ computed on the wedge $W(H_p, H_q)$ makes an angle $\theta \geqslant \alpha$ at the boundary line $\ell = H_p \cap H_q$, and second, $d_P(p,q) \geqslant \overline{pq}$. The second fact is trivial; we prove the first as follows. Let p' and q', respectively, be the points on the line ℓ such that pp' and qq' are perpendicular to ℓ, and let o be the point where the shortest path $\pi_O(p,q)$ touches the line ℓ. Then, unfolding and the triangle inequality imply that o lies in the closed interval $[p', q']$ on ℓ. Elementary geometry shows that $\angle poq \geqslant \alpha$, for every point $o \in [p', q']$. The lemma now follows from applying the bound in Lemma 3.1 on the triangle Δpoq. □

Since Step 1 of the algorithm ShortestPath applies only when $\alpha(H_p, H_q) \geqslant \pi/3$, the preceding lemma gives the bound $d_O(p,q)/d_P(p,q) \leqslant 2$. The next two lemmas concern the approximation via 3-plane wedges.

Lemma 3.3. *The dihedral angle formed between a horizon plane $H_i \in \mathcal{H}$ and the source-destination planes H_p, H_q always satisfies the following bounds:*

$$\frac{\pi - \alpha(H_p, H_q)}{2} \leqslant \alpha(H_p, H_i) \leqslant \frac{\pi + \alpha(H_p, H_q)}{2}.$$

Proof. The vector dot product is invariant under a rigid motion of the space, and so we may choose the initial rotation that makes $(\eta_p \cdot \hat{\imath}_z) = -(\eta_q \cdot \hat{\imath}_z)$ in such a way that vectors η_p and η_q lie in the YZ-plane, in symmetrical position with respect to the Y-axis. (The points η_p, η_q, and the origin O define a unique plane, which we may take to be the YZ-plane. Now, take the line bisecting the angle between η_p and η_q as the Y-axis, and take the orthogonal line that passes through O in the YZ-plane as the Z-axis.) In this coordinate system, let β be the angle between η_p and $\hat{\imath}_z$. Then observe that $\eta_p \cdot \hat{\imath}_z = \cos\beta$, and $\alpha(H_p, H_q) = 2\beta$. In rectangular coordinates, the normals η_p and η_q may be written as $\eta_p = (0, \sin\beta, \cos\beta)$, and $\eta_q = (0, \sin\beta, -\cos\beta)$. If η_i denotes the unit normal for the horizon plane H_i, then we may write $\eta_i = (\cos\theta, \sin\theta, 0)$, for some $0 \leqslant \theta \leqslant 2\pi$. The dot-product of η_i with η_p, η_q gives

$$\eta_p \cdot \eta_i = \eta_q \cdot \eta_i = \sin\beta \, \sin\theta.$$

As θ varies in $[0, 2\pi]$, this dot product varies in the range $[-\sin\beta, \sin\beta]$. Since $\eta_p \cdot \eta_i = -\cos(\alpha(H_p, H_i))$, the dihedral angle $\alpha(H_p, H_i)$ is bounded between $\cos^{-1}(\sin\beta) = \pi/2 - \beta$ and $\cos^{-1}(-\sin\beta) = \pi/2 + \beta$. This yields the desired result, since $\beta = (1/2)\alpha(H_p, H_q)$. □

Notice that since $\alpha(H_p, H_i) = \alpha(H_q, H_i)$, the same bounds also hold for $\alpha(H_q, H_i)$.

Lemma 3.4. *The condition $\alpha(H_p, H_q) < \pi/3$ implies that*

$$\min_{1 \leqslant i \leqslant k} \frac{d_i(p,q)}{d_P(p,q)} \leqslant 2.$$

Proof. Consider a shortest path $\pi_P(p,q)$. It must cross at least one horizon edge: every path joining a point on a positive face to a point on a negative face crosses the horizon. Let r_i be a point of crossing, and let e_i be the edge of P containing r_i. See Fig. 4. We focus on the 3-plane wedge path $\pi_i(p,q)$ computed by our approximation algorithm in Step 2. Let this path be (p, x_1, x_2, q), where x_1 and x_2 are the two interior vertices of $\pi_i(p,q)$; observe that $x_1 \in H_p \cap H_i$ and $x_2 \in H_q \cap H_i$.

Join r_i to some interior point r' on the segment x_1x_2, and assume that r'' is a point for which the function $f(r) = \overline{pr} + \overline{qr}$ is minimized as r varies along the segment r_ir'. Let $\pi_O(p, r'')$ and $\pi_O(q, r'')$, respectively, denote the shortest paths on the 2-plane wedges $W(H_p, H_i)$ and $W(H_i, H_q)$. By Lemma 3.3, the dihedral angle in each of these wedges is at least $\pi/3$, since $\alpha(H_p, H_q) < \pi/3$. Lemma 3.1 therefore implies the following bound:

$$\max\left(\frac{d_O(p,r'')}{\overline{pr''}}, \frac{d_O(q,r'')}{\overline{qr''}}\right) \leqslant \frac{1}{\sin(\pi/6)} = 2. \tag{2}$$

On the other hand, elementary algebra shows that

$$\max\left(\frac{d_O(p,r'')}{\overline{pr''}}, \frac{d_O(q,r'')}{\overline{qr''}}\right) \geqslant \frac{d_O(p,r'') + d_O(q,r'')}{\overline{pr''} + \overline{qr''}} \geqslant \frac{d_i(p,q)}{\overline{pr_i} + \overline{r_iq}} \geqslant \frac{d_i(p,q)}{d_P(p,q)}. \tag{3}$$

The lemma follows from inequalities (2) and (3). □

In conclusion, we have the following lemma.

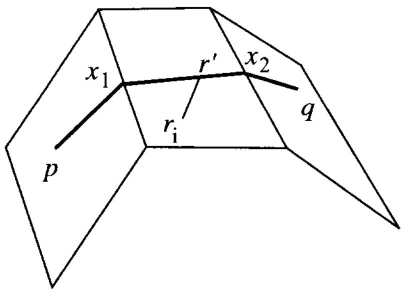

Fig. 4. Proof of Lemma 3.4 (the figure is rotated 90°).

Lemma 3.5. *For any pair of points $p, q \in P$, the algorithm ShortestPath correctly computes a path in the exterior of P of length at most $2d_P(p,q)$.*

3.5. Mapping approximate paths onto P

The algorithm ShortestPath computes paths on a 2-plane or a 3-plane wedge in the exterior of P. We show here how to convert such a path into a path lying on the surface of P *without increasing its length.*

First, consider the 2-plane wedge $W(H_p, H_q)$ and a shortest path $d_0(p,q)$ on it. This path touches the line $\ell = H_p \cap H_q$ at a point r. The three points p, q, r determine a unique plane H that intersects P in a convex polygon $C(p,r,q)$. This polygon gives two paths between p and q on P, and we take the shorter one of these as our approximation path.

In the case of a 3-plane wedge $W(H_p, H_i, H_q)$, we apply this procedure twice to project the path $\pi_i(p,q)$ onto P. Let the path $\pi_i(p,q)$ be (p, x_1, x_2, q), where $x_1 \in H_p \cap H_i$ and $x_2 \in H_i \cap H_q$. Suppose for the moment that the segment x_1x_2 touches P at a point y. In this case we project the 2-plane paths (p, x_1, y) and (y, x_2, q) onto P independently, then concatenate the two projected paths at y to get the final path. If x_1x_2 does not touch P, then we replace (p, x_1, x_2, q) by a shorter 3-plane path that does touch P, and then project that path onto P. Let x_1' be the point on the segment px_1 such that the segment $x_1'x_2$ is tangent to the surface of P at a point y. We can compute x_1' by finding a tangent from x_2 to the polygon $C(p, x_1, x_2)$ in the plane determined by p, x_1, and x_2. By the triangle inequality, the path (p, x_1', x_2, q) is shorter than (p, x_1, x_2, q). We project the 2-plane paths (p, x_1', y) and (y, x_2, q) onto P independently, then concatenate them at y to get the final path, as in the first case.

The following lemma shows that our method of projecting a path from a wedge to P does not increase the length of the path.

Lemma 3.6. *Let Δabc be a triangle in the plane, and let $R = (b, r_1, r_2, \ldots, r_k, c)$ be a convex polygonal chain contained in Δabc. Then the total length of the chain R does not exceed $\overline{ab} + \overline{ac}$.*

Proof. We use induction on the number of internal vertices in the chain R. The base case ($k = 0$) follows trivially from the triangle inequality: $\overline{bc} \leqslant \overline{ab} + \overline{ac}$. Inductively assume that the lemma holds for $k - 1$ internal vertices, for $k \geqslant 1$. Let r_i, where $1 \leqslant i \leqslant k$, be an extreme vertex of R in the direction perpendicular to bc, and let b_ic_i be the maximal chord passing through r_i and parallel to bc. See Fig. 5.

Then, by induction, we have the following inequalities:

$$\overline{b_ir_i} + \overline{r_ic_i} \leqslant \overline{ab_i} + \overline{ac_i},$$

$$\overline{br_1} + \sum_{j=1}^{i-1} \overline{r_jr_{j+1}} \leqslant \overline{bb_i} + \overline{b_ir_i}, \quad \text{and}$$

$$\sum_{j=i}^{k-1} \overline{r_jr_{j+1}} + \overline{r_kc} \leqslant \overline{r_ic_i} + \overline{c_ic}.$$

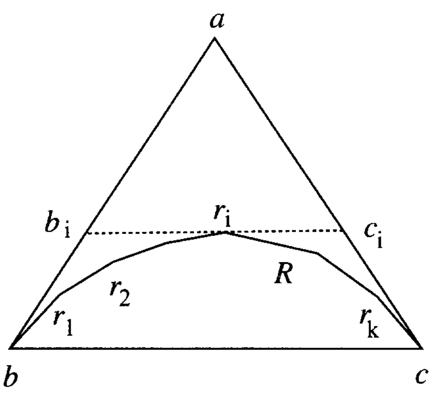

Fig. 5. Illustration for Lemma 3.6.

Combining the three inequalities above, we get our result:

$$\overline{br_1} + \sum_{j=1}^{k-1} \overline{r_j r_{j+1}} + \overline{r_k c} \leqslant \overline{ab} + \overline{ac}.$$

This completes the proof. □

3.6. Time complexity

Shortest paths on a convex polyhedron obey what is known as the "unfolding rule": if we unfold the sequence of faces through which a shortest path passes onto a plane, then the shortest path becomes a straight line. The following lemma states this fact more formally; a proof may be found in [17].

Lemma 3.7 (Unfolding rule). *Let P be a polyhedron in 3-space, and let p, q be two points on the surface of P. Let f_p, f_1, $f_2, \ldots, f_k$, f_q be the sequence of faces of P crossed by the shortest path $\pi_P(p,q)$. If we unfold the faces $f_1, \ldots, f_k$, f_q in sequence until they all become coplanar with f_p, then the shortest path $\pi_P(p,q)$ unfolds to a straight line.*

Using the "unfolding rule", we can compute a shortest path on a 2-plane or a 3-plane wedge in constant time; these wedges are convex polyhedra with a constant number of faces. The algorithm ShortestPath computes one shortest path on a 2-plane wedge, and one shortest path on each of the $O(n)$ 3-plane wedges. The computation necessary to determine the horizon edges and the horizon planes is clearly $O(n)$. Using the method in Section 3.5, the approximate path computed by the algorithm ShortestPath can be mapped to a path on P in $O(n)$ time. We thus have the following theorem.

Theorem 3.8. *Given a convex polytope P of n vertices and two points $p, q \in P$, one can compute in $O(n)$ time a path of length at most $2d_P(p,q)$ joining p and q on the surface of P.*

4. Approximating a shortest path tree

In this section, we show how to approximate shortest path distances to all vertices of P from a fixed point $p \in P$ in $O(n \log n)$ time.

4.1. Northern and southern hemispheres

Without loss of generality, we assume that the face containing p, namely f_p, is horizontal, with P lying below it. Thus, the face normal η_p is directed toward the negative z-axis. We choose a threshold angle $\omega < \pi$, and define the *southern hemisphere* of P, denoted $S(\omega)$, to be the set of faces s satisfying

$$\eta_p \cdot \eta_s \leqslant \cos(\pi - \omega).$$

In other words, for each face s in the southern hemisphere $S(\omega)$, we have $\alpha(H_p, H_s) \leqslant \omega$, whereas each face in the northern hemisphere $N(\omega)$ forms a dihedral angle greater than ω with H_p. Fig. 6 shows an example in two dimensions. We leave the value of ω undetermined for now and optimize it at the end.

Lemma 4.1. *For every vertex $v \in P$ lying in the northern hemisphere $N(\omega)$ or on the boundary between northern and southern hemispheres, the shortest path distance computed on the 2-plane wedge $W(H_p, H_v)$ is at most $d_P(p, v)/\sin(\omega/2)$.*

Proof. The claim follows immediately from Lemma 3.2. □

Since a shortest path on a 2-plane wedge can be computed in constant time, the approximate distances to all northern-hemisphere vertices can be determined in $O(n)$ time. The remainder of this section deals with the southern-hemisphere vertices.

4.2. Subdividing the horizon

The boundary between the southern and northern hemispheres is a polygonal chain consisting of vertices and edges of P. Let $s_1, s_2, \ldots, s_k$ denote the counterclockwise sequence of vertices forming

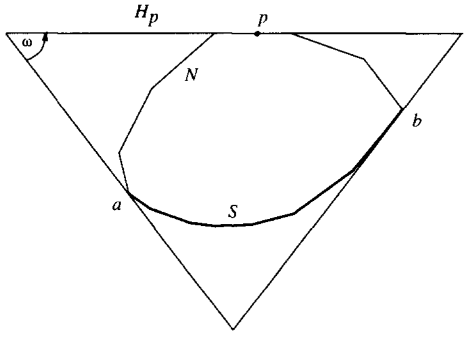

Fig. 6. A two-dimensional illustration of hemispheres. The bottom part of the polygon between the vertices a and b, drawn in thick ink, constitutes the southern hemisphere $S(\omega)$; the top part is the northern hemisphere.

this boundary chain; thus, the sequence of edges along the boundary chain is $s_1s_2, \ldots, s_{k-1}s_k, s_ks_1$. Let H_i, $i = 1, 2, \ldots, k$, denote the (unique) plane that (1) contains s_is_{i+1}, (2) is tangent to P, and (3) satisfies $\alpha(H_p, H_i) = \omega$. (These planes are similar to the horizon planes of Section 3.)

Our algorithm requires that the angles $\angle s_ips_{i+1}$ be small. In order to ensure that, we subdivide the boundary chain by adding extra points on the segments s_is_{i+1}, as explained below. Let $\varepsilon > 0$ be an arbitrarily small but positive constant; this constant controls the quality of our approximation. Consider a triangle Δs_ips_{i+1}, and let $\theta_i = \angle s_ips_{i+1}$. If the condition

$$\sqrt{\frac{1+\sin\theta_i}{1-\sin\theta_i}} < 1+\varepsilon, \tag{4}$$

does not hold, then we subdivide the edge s_is_{i+1} by adding a new vertex t_i at its midpoint. Otherwise, we leave the segment s_is_{i+1} alone. This process is repeated until all the edges in the boundary chain satisfy the ε-condition of inequality (4).

Since ε is a fixed constant, it follows easily that an edge s_is_{i+1} of the original boundary chain can be subdivided at most a constant number of times. So, let $t_1, t_2, \ldots, t_m$, where $m = O(n)$, be the final sequence of vertices along the boundary chain separating the northern and southern hemispheres. We can approximate shortest path distances to all vertices of the boundary chain using the 2-plane wedges $W(H_p, H_i)$. The dihedral angle bound $\omega = \alpha(H_p, H_i)$ ensures that the approximation is within $1/\sin(\omega/2)$ of the true shortest path distance. (Notice that for a boundary chain vertex t that is also a vertex of the polytope, there are two possible wedges, determined by the two edges incident to t. In this case, we may choose the shorter approximation, although our proof does not use this fact. All other vertices of the boundary chain lie in the interior of polytope edges, and we have a single 2-plane wedge approximation for them.) Let $w(t_i)$ denote the approximate distance between p and t_i. By Lemma 3.2, this approximation satisfies $w(t_i) \leqslant d_P(p, t_i)/\sin(\omega/2)$. We are now ready to describe our algorithm for approximating distances between p and all the vertices in the southern hemisphere.

```
ALGORITHM SOUTHERNHEMIDIST

1. Determine the north and south hemispheres, $N(\omega)$ and $S(\omega)$, of $P$.
2. Subdivide the boundary chain between $N(\omega)$ and $S(\omega)$, so that each wedge obeys
   the $\varepsilon$ condition in (4).
3. Using 2-plane wedge unfoldings, compute approximate distances from $p$ to all
   vertices of the subdivided boundary chain $T = \{t_1, t_2, \ldots, t_m\}$. Let $w(t_i)$ be the
   approximate distance from $p$ to $t_i$.
4. Vertically project onto the plane $z = 0$ all the boundary vertices $t_i$, $i = 1, 2, \ldots, m$,
   as well as the vertices of the southern hemisphere $S(\omega)$. Let $t'$ denote the projec-
   tion of a point $t$.
5. Compute the (additive) weighted Voronoi diagram of the set $T' = \{t'_1, t'_2, \ldots, t'_m\}$,
   using $w(t_i)$ as the weight of $t'_i$. Preprocess the diagram for point location queries.
6. For each vertex $v \in S(\omega)$, locate its projected image $v'$ in the Voronoi diagram.
7. Let $t'_i \in T'$ be the (weighted) nearest neighbor of $v'$. Compute the shortest path
   between $v$ and $t_i$ on the 2-plane wedge $W(H_v, H_i)$. Let $d_O(v, t_i)$ be this distance.
   Output $w(t_i) + d_O(v, t_i)$ as the approximation of the distance $d_P(p, v)$.
```

4.3. Analysis of the algorithm SouthernHemiDist

We prove two purely geometric lemmas: the first one shows that, for any two points $u, v \in S(\omega)$, $\overline{uv} \leqslant \overline{u'v'}/\cos\omega$, where u', v' are the projections of u, v on a horizontal plane; the second one bounds the increase in path-length when a path is constrained to go through one of the boundary-chain vertices.

Lemma 4.2. *Let $u, v \in P$ be two points in the southern hemisphere $S(\omega)$, and let u', v' be their projections on the plane $z = 0$. Then $\overline{uv} \leqslant \overline{u'v'}/\cos\omega$.*

Proof. Without loss of generality, assume that $u_z \leqslant v_z$. By vertical translation, we may also assume that $u_z = 0$, and thus $u = u'$. We claim that the angle $\angle vu'v' \leqslant \omega$. The lemma then follows readily by considering the triangle $\Delta vu'v'$: $\overline{u'v'}/\overline{uv} = \cos(\angle vu'v')$.

The bound on the angle $\angle vu'v'$ follows from the observation that the plane H_v that is tangent at v makes a dihedral angle of at most ω with the horizontal plane H_p, and u lies above H_v. □

Lemma 4.3. *Let Δaob be a triangle with $\overline{oa} \leqslant \overline{ob}$ and $\angle aob = \theta$. Then we have*

$$\frac{\overline{oa} + \overline{ab}}{\overline{ob}} \leqslant \sqrt{\frac{1 + \sin\theta}{1 - \sin\theta}}\,.$$

Proof. Let α denote the angle $\angle oab$. See Fig. 7 for illustration. The Law of Sines gives the following equality:

$$\frac{\overline{oa} + \overline{ab}}{\overline{ob}} = \frac{\sin(\pi - \theta - \alpha) + \sin\theta}{\sin\alpha}.$$

To maximize this quantity, we differentiate with respect to α, obtaining

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}\left(\frac{\sin(\theta + \alpha) + \sin\theta}{\sin\alpha}\right) = \frac{-\sin\theta(1 + \cos\alpha)}{(\sin\alpha)^2},$$

which is always non-positive. Thus, the ratio $(\overline{oa} + \overline{ab})/(\overline{ob})$ diminishes as α increases, and it is maximized at the smallest legal value of α. The condition $\overline{oa} \leqslant \overline{ob}$ forces that $\alpha \geqslant \pi/2 - \theta$, which in turn gives

$$\frac{\overline{oa} + \overline{ab}}{\overline{ob}} \leqslant \frac{\sin(\pi/2) + \sin\theta}{\sin(\pi/2 - \theta)} = \frac{1 + \sin\theta}{\cos\theta} = \sqrt{\frac{1 + \sin\theta}{1 - \sin\theta}} \leqslant 1 + \varepsilon.$$

This completes the proof. □

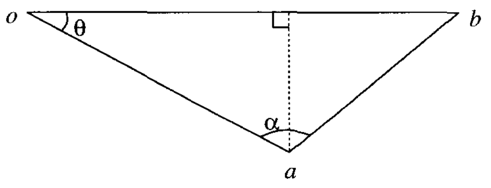

Fig. 7. Illustration for Lemma 4.3.

We are now ready to prove our approximation result for the algorithm SouthernHemiDist. Consider an arbitrary vertex v in the southern hemisphere $S(\omega)$, and its shortest path to p, namely $\pi_P(p,v)$. This path crosses the boundary chain; let r be a point of crossing, and suppose that t_i and t_{i+1} are the vertices in the boundary chain adjacent to r. Then, the following bounds clearly hold:

$$d_P(p,v) = d_P(p,r) + d_P(r,v) \geqslant \overline{pr} + \overline{rv}. \tag{5}$$

By unfolding along the edge t_it_{i+1}, let us make the triangles Δt_ipt_{i+1} and Δt_ivt_{i+1} coplanar, and consider the plane quadrilateral (p,t_i,v,t_{i+1}). If

$$(\overline{pr} + \overline{rv}) \geqslant \min\left\{ \left(\overline{pt_i} + \overline{vt_i}\right), \left(\overline{pt_{i+1}} + \overline{vt_{i+1}}\right) \right\},$$

then assume, without loss of generality, that

$$\overline{pr} + \overline{rv} \geqslant \overline{pt_i} + \overline{vt_i}. \tag{6}$$

Otherwise, elementary geometry shows that the quadrilateral (p,t_i,v,t_{i+1}) is necessarily convex. In this case, we apply Lemma 4.3 and inequality (4) to the triangle Δpt_iv. Since $\angle t_ipt_{i+1}$ satisfies the ε-condition and $\angle t_ipv < \angle t_ipt_{i+1}$, we have

$$\overline{pr} + \overline{rv} \geqslant \overline{pv} \geqslant \frac{\overline{pt_i} + \overline{vt_i}}{1+\varepsilon} \tag{7}$$

in the unfolded plane. Finally, recalling that v' and t_i' denote the projections of v and t_i onto the plane $z=0$, we have the trivial inequality $\overline{vt_i} \geqslant \overline{v't_i'}$. By combining (5)–(7), we get the following bound for $d_P(p,v)$:

$$d_P(p,v) \geqslant \frac{\overline{pt_i} + \overline{v't_i'}}{1+\varepsilon}. \tag{8}$$

Since $\alpha(H_p,H_i) = \omega$, Lemma 3.2 implies the following bound for the approximate shortest path distance $w(t_i) = d_O(p,t_i)$ between p and t_i computed in Step 3:

$$\overline{pt_i} \geqslant d_O(p,t_i)\sin(\omega/2). \tag{9}$$

Since v and t_i are both in the southern hemisphere, it is easily seen that $\alpha(H_v,H_i) \geqslant \pi - 2\omega$. Hence the approximate shortest path distance $d_O(v,t_i)$, computed on the 2-plane wedge $W(H_v,H_i)$, satisfies the following bounds:

$$\frac{\overline{v't_i'}}{\cos\omega} \geqslant \overline{vt_i} \geqslant d_O(v,t_i)\sin(\pi/2-\omega) = d_O(v,t_i)\cos\omega. \tag{10}$$

Plugging (9) and (10) into (8), we get

$$d_P(p,v) \geqslant \frac{d_O(p,t_i)\sin(\omega/2) + d_O(v,t_i)(\cos\omega)^2}{1+\varepsilon}.$$

By setting $\sin(\omega/2) = (\cos\omega)^2$, we get $\omega \approx 49.62°$, corresponding to $\sin(\omega/2) \approx 0.41964$. This gives the desired ratio for the approximation path $\pi_O(p,t_i) \cup \pi_O(t_i,v)$:

$$\frac{d_O(p,t_i) + d_O(t_i,v)}{d_P(p,v)} \leqslant 2.38(1+\varepsilon).$$

Since the algorithm SouthernHemiDist picks the minimum $d_O(p,t_i) + \overline{v't_i'}$ over all choices of t_i, it follows that the approximate path is at most $2.38(1+\varepsilon)$ times the length of the shortest path.

The running time of the algorithm is bounded by $O(n \log n)$: the dominant steps are 5 and 6, in which an additive-weight, planar Voronoi diagram is computed and $O(n)$ point location queries are performed. (A good reference on Voronoi diagrams is the survey paper by Aurenhammer [2]. The point location algorithms are described in [8,11,16].) We have proved the following theorem.

Theorem 4.4. *Given a convex polytope P of n vertices, a source point $p \in P$, and a fixed constant $\varepsilon > 0$, we can compute in $O(n \log n)$ time approximate distances between p and all vertices of P. The approximation factor is no worse than $2.38(1+\varepsilon)$.*

5. Related problems and concluding remarks

Our algorithms can be used to get a very simple, though weak, approximation of a shortest path in the general case of multiple polyhedra as well. Specifically, using Theorem 3.8, we can get a path of length at most $2k$ times the optimal if there are k (disjoint) convex polyhedra. The algorithm runs in $O(n)$ time, where n is the total number of vertices in all the polyhedra. This is an order of magnitude faster than the general approximation algorithms of Choi et al. and Clarkson, although our approximation bound is much worse than the $(1+\varepsilon)$ factor achieved by [6,7].

In summary, we have presented fast, simple, and quite practical methods for approximating shortest paths on a convex polyhedron. We believe that the algorithm ShortestPath is at least as easy to understand and implement as any of the known exact algorithms, and easier than most. However, we have not implemented it ourselves, so this is simply a value judgment. Since this paper originally appeared, Har-Peled et al. [9] have extended our work and obtained a $(1+\varepsilon)$-approximation algorithm. Our work has also opened up the possibility that a similarly simple and efficient algorithm might be possible for nonconvex polyhedra, though none has been found to date.

We believe that the actual ratio achieved by our algorithm ShortestPath is better than 2, but we have not been able to prove this; the claim concerns the actual path reported on P, not the wedge-path that is used for the purpose of analysis. The same holds for the shortest path tree approximation. An open problem is to either tighten the analysis and improve the approximation ratio, or provide a lower-bound example for which the performance of our approximation algorithm is close to the claimed bound.

Several other open problems are suggested by our work. Is it possible to achieve a ratio of $(1+\varepsilon)$, for any fixed $\varepsilon > 0$, for a single shortest path using roughly linear time? Is it possible to improve substantially the ratio of $2k$ in the general case of k convex polytopes, without significantly sacrificing the running time or the simplicity of our approach?

Acknowledgements

We thank Pankaj Agarwal and Boris Aronov for some discussions on the subject of this paper. They have independently derived results similar to Theorem 3.8 of our paper.

References

[1] P.K. Agarwal, B. Aronov, J. O'Rourke, C. Schevon, Star unfolding of a polytope with applications, Technical Report 031, Dept. Comput. Sci., Smith College, Northampton, MA, July 1993.
[2] F. Aurenhammer, Voronoi diagrams: A survey of a fundamental geometric data structure, ACM Comput. Surv. 23 (1991) 345–405.
[3] C. Bajaj, The algebraic complexity of shortest paths in polyhedral spaces, in: Proc. 23rd Allerton Conf. Commun. Control Comput., 1985, pp. 510–517.
[4] J. Canny, J.H. Reif, New lower bound techniques for robot motion planning problems, in: Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci., 1987, pp. 49–60.
[5] J. Chen, Y. Han, Shortest paths on a polyhedron, in: Proc. 6th Annu. ACM Sympos. Comput. Geom., 1990, pp. 360–369.
[6] J. Choi, J. Sellen, C.K. Yap, Approximate Euclidean shortest path in 3-space, in: Proc. 10th Annu. ACM Sympos. Comput. Geom., 1994, pp. 41–48.
[7] K.L. Clarkson, Approximation algorithms for shortest path motion planning, in: Proc. 19th Annu. ACM Sympos. Theory Comput., 1987, pp. 56–65.
[8] H. Edelsbrunner, L.J. Guibas, J. Stolfi, Optimal point location in a monotone subdivision, SIAM J. Comput. 15 (1986) 317–340.
[9] S. Har-Peled, M. Sharir, K.R. Varadarajan, Approximate shortest paths on a convex polytope in three dimensions, in: Proc. 12th Annu. ACM Sympos. Comput. Geom., 1996, pp. 329–338.
[10] J. Hershberger, S. Suri, Efficient computation of Euclidean shortest paths in the plane, in: Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci., 1993, pp. 508–517.
[11] D.G. Kirkpatrick, Optimal search in planar subdivisions, SIAM J. Comput. 12 (1983) 28–35.
[12] T. Lozano-Pérez, M.A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Comm. ACM 22 (1979) 560–570.
[13] J.S.B. Mitchell, D.M. Mount, C.H. Papadimitriou, The discrete geodesic problem, SIAM J. Comput. 16 (1987) 647–668.
[14] D.M. Mount, On finding shortest paths on convex polyhedra, Technical Report 1495, Department of Computer Science, University of Maryland, MD, 1985.
[15] C.H. Papadimitriou, An algorithm for shortest-path motion in three dimensions, Inform. Process. Lett. 20 (1985) 259–263.
[16] F.P. Preparata, M.I. Shamos, Computational Geometry: An Introduction, Springer, New York, 1985.
[17] M. Sharir, A. Schorr, On shortest paths in polyhedral spaces, SIAM J. Comput. 15 (1986) 193–215.