Discrete Comput Geom 30:357–372 (2003)
DOI: 10.1007/s00454-003-2769-0

Discrete & Computational
Geometry

Facility Location on a Polyhedral Surface*

Boris Aronov,[1] Marc van Kreveld,[2] René van Oostrum,[2] and Kasturi Varadarajan[3]

[1]Department of Computer and Information Science, Polytechnic University,
Brooklyn, NY 11201-3840, USA
aronov@ziggy.poly.edu

[2]Institute of Information and Computing Sciences, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{marc,rene}@cs.uu.nl

[3]Department of Computer Science, University of Iowa,
IA 52240, USA
kvaradar@cs.uiowa.edu

Abstract. Given an orientable genus-0 polyhedral surface defined by n triangles, and a set of m point *sites* on it, we would like to identify its 1-center, i.e., the location on the surface that minimizes the maximum distance to the sites. The distance is measured as the length of the Euclidean shortest path along the surface. To compute the 1-center, we compute the furthest-site Voronoi diagram of the sites on the polyhedral surface. We show that the diagram has maximum combinatorial complexity $\Theta(mn^2)$, and present an algorithm that computes the diagram in $O(mn^2 \log m \log n)$ expected time. The 1-center can then be identified in time linear in the size of the diagram.

* B.A. has been partially supported by a Sloan Research Fellowship and by NSF Grants CCR-99-72568 and ITR CCR-00-81964. M.v.K. and R.v.O. have been partially supported by the ESPRIT IV LTR Project No. 21957 (CGAL). K.V. has been supported by National Science Foundation Grant CCR-93-01259, by an Army Research Office MURI Grant DAAH04-96-1-0013, by a Sloan fellowship, by an NYI award, by matching funds from Xerox Corporation, and by a grant from the U.S.–Israeli Binational Science Foundation. Part of the work was carried out while B.A. was visiting Utrecht University.

1. Introduction

1.1. *Problem Statement*

In this paper a *polyhedral surface* is the boundary of a not necessarily convex polyhedron of genus 0.[1] For simplicity we assume hereafter that the surface has been triangulated.

This paper addresses the 1-*center problem* for a set of m point *sites* on a (triangulated) polyhedral surface P of 0 genus with n triangles. The *distance* between two points on P is the minimum length of any path between those points that lies on P. The *facility center*, or 1-*center*, of the sites is the point of P that minimizes the maximum distance to a site. We assume throughout that $m \leq n$.

Our algorithm constructs the furthest-site Voronoi diagram of the point sites on P. The location for the facility center can then be identified by traversing the edges and vertices of the furthest-site Voronoi diagram.

1.2. *Previous Work and New Results*

In the Euclidean plane the facility center, or the center of the *smallest enclosing disk* of a set of m point sites, can be determined in $O(m)$ time. Several algorithms attain this bound. Megiddo [9] gave the first deterministic linear-time algorithm, and a much simpler, randomized, linear-expected-time algorithm was found by Welzl [17].

There is a close connection between the facility center and the furthest-site Voronoi diagram of the sites. Namely, the facility center must lie at a vertex or on an edge of this diagram. In the plane, with Euclidean distance, the furthest-site Voronoi diagram has cells only for the sites on the convex hull of the set of sites, and all cells are unbounded.

On a polyhedral surface some of the properties of furthest-site Voronoi diagrams in the plane no longer hold. For instance, a bisector of two sites on P is generically a closed curve consisting of as many as $\Theta(n^2)$ straight-line segments and/or hyperbolic arcs, in the worst case. It may also contain two-dimensional portions of the surface of the polyhedron.

Mount [11] showed that the *closest-site* Voronoi diagram of $m \leq n$ sites on a polyhedral surface with n faces has complexity $\Theta(n^2)$ in the worst case; he also gave an algorithm that computes the diagram in $O(n^2 \log n)$ time. We are not aware of any previous work on furthest-site Voronoi diagrams on polyhedral surfaces.

The problem of computing the shortest path between two points along the surface of a polyhedron has received considerable attention; see the papers by Sharir and Schorr [15], Mitchell et al. [10], and Chen and Han [4]. The best known algorithms [4], [10] compute the shortest path between two given points, the source s and destination t, in roughly $O(n^2)$ time. In fact, these algorithms compute a data structure that allows one to compute the shortest path distance from the source s to any query point p in $O(\log n)$ time. The algorithm of Mitchell et al. [10] is a continuous version of Dijkstra's algorithm for finding

[1] In fact, our argument goes through unmodified for an arbitrary orientable triangulated polyhedral 2-manifold without boundary, of genus 0. No embedding in 3-space is assumed or required.

shortest paths in a graph [6], while Chen and Han [4] solve the problem by determining shortest paths in an *unfolding* of the polyhedron; see also [3].

In his master's thesis, van Trigt [16] gave an algorithm that solves the 1-center problem on a polyhedral terrain in $O(m^4 n^3 \log n)$ time, using $O(n^2(m^2 + n))$ space.

This paper gives an $O(mn^2 \log m \log n)$ expected-time randomized algorithm to compute the furthest-site Voronoi diagram and thus find the facility center for a set S of m sites on a genus-0 polyhedral surface with n faces. The algorithm for computing the furthest-site Voronoi diagram is near-optimal, as the maximum combinatorial complexity of the diagram is $\Theta(mn^2)$. A preliminary version of this paper gives a more complicated deterministic algorithm for computing the diagram in $O(mn^2 \log^2 m \log n)$ time [1], using ideas from the algorithm of Ramos [14] for computing the intersection of unit spheres in three dimensions. It is conceivable that, as in the case of the plane, there is an algorithm for computing the facility center that is faster than any algorithm for computing the furthest-site Voronoi diagram. However, the fast algorithms for computing the facility center in the plane do not generalize to the case of a polyhedral surface.

2. Complexity of the Furthest-Site Voronoi Diagram

Previous papers on shortest paths on polyhedra [4], [10], [15], [16] use a number of concepts that we will need as well. We review them briefly after giving the relevant definitions.

In the remainder of this paper, P is the surface of a polyhedron. As stated before, we only allow polyhedra homeomorphic to a ball, so that their surfaces are homeomorphic to a sphere. For two points p and p' on P, we define the *distance* $\mathrm{d}(p, p')$ to be the length of the shortest path from p to p' along P. Let S be a set of $m \leq n$ point sites on P. Consider first a single site $s \in S$. For any point p on P we consider a shortest path from p to s; note that in general such a path need not be unique. Such a shortest path has a number of properties. First, if it crosses an edge of P properly, then a principle of refraction holds. This means that if the two incident triangles were pivoted about their common edge to become co-planar, then the shortest path would cross the edge as a straight-line segment. This principle is called *unfolding*. For any vertex on the polyhedron, we define its *total angle* as the sum of the interior angles at that vertex in each of the triangles incident to it. The shortest path cannot contain any vertex for which the total angle is less than 2π, except possibly at the source p and the target s.

Any shortest path crosses a sequence of triangles, edges, and possibly vertices. If two shortest paths on the polyhedron cross the same sequence (in the same order), we say that these paths have the same *edge sequence*. If a shortest path from p to s contains a vertex of the polyhedron, the vertex reached first from p is called the *pseudoroot* of p. Otherwise, site s is the pseudoroot of p.

The *shortest-path map* (*SPM*) *of* s is defined as the subdivision of P into maximal path-connected regions where the shortest path to s is unique and has a fixed edge sequence. For generic placements of s, the closures of the regions cover P, so the portion of P outside any region, where more than one shortest path to s exists, consists of one-dimensional portions. When a vertex of the polyhedron has more than one shortest path

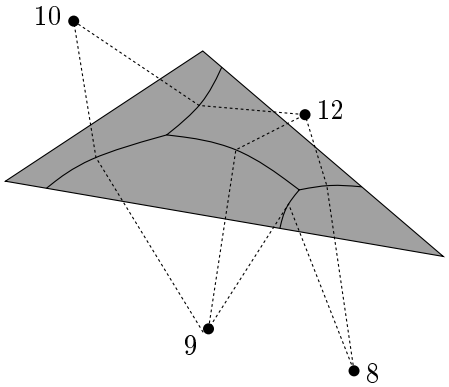


Fig. 1. The SPM of a site s, restricted to a triangle, is the Euclidean Voronoi diagram for a set of pseudosites (shown as dots) with additive weights (the labels near the dots). The weight of a pseudosite is the shortest-path distance from the corresponding pseudoroot to s.

to s, the complement of the regions of the SPM may have two-dimensional portions. We exclude such situations as degenerate.

It is known that the SPM of a site has complexity $O(n^2)$; this bound is tight in the worst case. The SPM restricted to a triangle is actually the planar Euclidean Voronoi diagram for a set of pseudosites with additive weights (see Fig. 1). The pseudosites are obtained from the pseudoroots by unfolding the triangles in the edge sequence to the pseudoroot so that they are all co-planar. The weight of a pseudosite is the distance from the corresponding pseudoroot to the site s. It follows that the boundaries of regions in the SPM within a triangle consist of straight-line segments and/or hyperbolic arcs. For any point in the relative interior of such an arc and/or a segment there are two shortest paths to s with different pseudoroots. Vertices of the SPM are connected to s by three or more shortest paths.

Given two sites s and t on P, the *bisector* $\beta(s,t)$ is the set of points p on the polyhedron equidistant from s and t. The bisector consists of straight-line segments, hyperbolic arcs, and may even contain two-dimensional regions. Such regions occur only when two sites have exactly the same distance to some vertex of P, a situation that we exclude as degenerate.

We catalog here the list of degeneracies that are assumed not to occur in the remainder of the paper. Our first assumption is that any vertex v of P and any site $s \in S$ are connected by a unique shortest path. We also assume that no vertex of the polyhedron is equidistant to two or more sites. Finally, for any real $d \geq 0$ and any point p on the polyhedron, we require that the number of distinct shortest paths of length d from p to all sites of S not exceed three. This implies in particular that no point on the polyhedron is equally distant to four or more sites. It also implies that for any three sites s, t, and u, the bisectors $\beta(s,t)$ and $\beta(s,u)$ do not intersect "tangentially"; we do not elaborate this implication here.

The *closest-site Voronoi diagram* of a set S of m sites on P, denoted by $\mathrm{V}(S)$, is a planar graph embedded in P that subdivides P into maximal open regions associated with the sites in S, with the property that a point $p \in P$ lies in the region of a site

$s \in S$ if and only if $d(p, s) < d(p, s')$ for each $s' \in S$ with $s' \neq s$. The interior of the boundary between two adjacent regions is an *edge* of the Voronoi diagram; it is easy to see that each edge lies on a bisector of two sites in S. The non-empty intersections of the closures of three or more regions of the Voronoi diagram are its *vertices*. We assume that all vertices have degree three; otherwise, a degeneracy is present.

The *furthest-site Voronoi diagram* $FVD(S)$ of a set S of m sites on P is a similar subdivision of P into maximal open regions. The difference is that a point $p \in P$ lies in the region of a site $s \in S$ if and only if $d(p, s) > d(p, s')$ for each $s' \in S$ with $s' \neq s$. In this paper we give a new (and to our knowledge the first known) algorithm for computing $FVD(S)$. We use the notation $\mathcal{R}(s)$ to denote the region of $s \in S$ in the Voronoi diagram. Whether this is the region of s in $V(S)$ or $FVD(S)$ should be clear from context.

The following facts are crucial for the algorithm below to work and for the analysis to hold. Some of them are similar to the lemmas in the paper of Leven and Sharir [8]; they apply to a large class of metrics and hold under very general conditions.

Lemma 1. *In the closest-site Voronoi diagram of a set S of sites on P, the region $\mathcal{R}(s)$ of a site $s \in S$ is path-connected.*

Proof. Let p be a point in $\mathcal{R}(s)$, let $\pi(p, s)$ be a shortest path from p to s, and let p' be an arbitrary point on $\pi(p, s)$. The subpaths $\pi(p, p'), \pi(p', s) \subset \pi(p, s)$ are also shortest paths, and $d(p, s) = d(p, p') + d(p', s)$. It follows that $d(p', s) < d(p', t)$ for any $t \in S \backslash \{s\}$; otherwise, there would be a path from p to t via p' no longer than $d(p, s)$, contradicting the fact that p is closer to s than to t. Hence, any point p' on $\pi(p, s)$ lies in $\mathcal{R}(s)$, and any two points p and q in $\mathcal{R}(s)$ are connected via s by a path that lies completely in $\mathcal{R}(s)$. □

Lemma 2. *Under the assumptions of non-degeneracy, in the closest-site Voronoi diagram of a set S of sites on P, $\beta(s, t)$ lies in the intersection of closures of $\mathcal{R}(s)$ and $\mathcal{R}(t)$.*

Proof. Let p be any point on $\beta(s, t)$. Consider any point $p' \neq p$ on the shortest path $\pi(p, s) = \pi(p, p') \cup \pi(p', s)$. Arguing as in the proof of Lemma 1, we conclude that $d(p', t) < d(p', s)$ is impossible, as it would imply $d(p, t) < d(p, s)$, contradicting $p \in \mathcal{R}(s)$. Thus either $p' \in \mathcal{R}(s)$ and p lies in the closure $\mathcal{R}(s)$ as claimed, or we must have $d(p', t) = d(p', s)$. The latter would imply that $\pi(p, p') \cup \pi(p', t)$ is a shortest path from p to t and that two shortest paths from p, one to s and one to t, overlap in their initial segment. This is only possible if p has the same pseudoroot with respect to s and to t, which would have to be a vertex of P equidistant from two sites $s, t \in S$, which we explicitly excluded in our non-degeneracy assumption. Hence p' and thus $\beta(s, t)$ lie in the closure of $\mathcal{R}(s)$ and, by symmetric reasoning, of $\mathcal{R}(t)$. □

Lemma 3. *Bisector $\beta(s, t)$ is connected and homeomorphic to a circle.*

Proof. Consider the closest-site Voronoi diagram of $\{s, t\}$. The closures of $\mathcal{R}(s)$ and $\mathcal{R}(t)$ in this Voronoi diagram cover the entire P and both $\mathcal{R}(s)$ and $\mathcal{R}(t)$ are path-connected by Lemma 1. Since P is homeomorphic to a sphere, $\beta(s, t)$, which is the

common boundary of $\mathcal{R}(s)$ and $\mathcal{R}(t)$ by Lemma 2, must be connected and homeomorphic to a circle. □

Lemma 4. *For any three distinct sites s, t, and u, bisectors $\beta(s,t)$ and $\beta(s,u)$ intersect at most twice.*

Proof. Consider $\mathrm{V}(\{s,t,u\})$. At an intersection χ of $\beta(s,t)$ and $\beta(s,u)$, we have $\mathrm{d}(\chi,s) = \mathrm{d}(\chi,t) = \mathrm{d}(\chi,u)$. Therefore, χ also lies on the third bisector $\beta(t,u)$ and thus is a vertex of $\mathrm{V}(\{s,t,u\})$, incident to $\mathcal{R}(s)$, $\mathcal{R}(t)$, and $\mathcal{R}(u)$.

Now suppose for the sake of contradiction that the bisectors $\beta(s,t)$ and $\beta(s,u)$ (and consequently $\beta(t,u)$) intersect in at least three distinct points χ_1, χ_2, and χ_3. Connect each of s, t, u to each of χ_1, χ_2, χ_3 by a shortest path. No two of the paths sharing an endpoint cross, though they may overlap. (Two paths starting from the same site do not cross because shortest paths from a single site cannot cross. Two paths from a point of intersection, for instance the paths from χ_1 to s and χ_1 to t, cannot cross because the paths lie in different regions except for the common point χ_1. These properties follow readily from the local optimality of shortest paths and the non-degeneracy assumptions.)

Consider a pair of paths not sharing an endpoint, say $\pi(s,\chi_1)$ and $\pi(t,\chi_2)$. The former is contained in $\mathcal{R}(s) \cup \{\chi_1\}$ (by the proof of Lemma 2), the latter in $\mathcal{R}(t) \cup \{\chi_2\}$. In particular, the two paths are disjoint.

To summarize, the six points and the nine interconnecting paths form a non-crossing embedding of $K_{3,3}$, the 3×3 complete bipartite graph, on the topological sphere P—a contradiction. □

Any family of simple closed curves (in this case on a topological sphere) of which every two cross at most twice is called a *family of pseudocircles*. Thus for every fixed $s \in S$, the bisectors $\{\beta(s,t) : t \in S \backslash \{s\}\}$ form a family of pseudocircles. Every bisector partitions the surface of the polyhedron into two path-connected two-dimensional regions, or *pseudodisks*. We call the region that contains s the *interior* (with respect to s) of a pseudocircle; the region not containing s is called the *exterior*.

Lemma 5. *Let $\mathcal{B}$ be a set of pseudocircles on a homology sphere P. If the common interior of the pseudocircles in $\mathcal{B}$ is non-empty, then their common exterior is simply connected.*

Proof. If the closures of interiors of pseudocircles in $\mathcal{B}$ cover P, the claim holds vacuously. Consider the family $\mathcal{B}'$ of closed pseudodisks, each corresponding to the exterior of a pseudocircle in $\mathcal{B}$. We are interested in $X := \bigcap \mathcal{B}'$, where we may assume $X \neq \emptyset$. By a theorem of Debrunner [7, Theorem 4.8, page 403], since P is a homology sphere, to show that X is simply connected (and thus path-connected), it is sufficient to check that every set in $\mathcal{B}'$ is simply connected, every two sets meet in a path-connected set, and every three in a non-empty set. $\mathcal{B}'$ satisfies these requirements because every set in $\mathcal{B}'$, being a pseudodisk, is simply connected; the intersection of two sets in $\mathcal{B}'$, being an intersection of two pseudodisks, is path connected; the intersection of any three sets in $\mathcal{B}'$ is non-empty since $X \neq \emptyset$. □

Lemma 6. *Bisector $\beta(s, t)$ consists of $O(n^2)$ straight-line segments and hyperbolic arcs.*

Proof. The claim follows directly from the fact that the Voronoi diagram of m sites on a polyhedron with n faces with $m \leq n$ has complexity $\Theta(n^2)$ in the worst case; see the paper by Mount [11]. □

Since the edges of the closest- and furthest-site Voronoi diagram lie on the bisectors of pairs of sites from S, each edge also consists of $O(n^2)$ line segments and hyperbolic arcs. To simplify our exposition, the intersections between two adjacent segments or arcs on a Voronoi edge are referred to as *breakpoints*, as opposed to the *vertices* of the diagram that we defined above. We consider the point where a bisector crosses an edge of P also to be a breakpoint.

Lemma 7. *The furthest-site Voronoi diagram* FVD(S) *of a set S of m sites on a polyhedron has $O(m)$ cells, vertices, and edges.*

Proof. $\mathcal{R}(s) = \bigcap_{t \in S, t \neq s} \{x : d(x, s) > d(x, t)\}$. By Lemmas 4 and 5, this intersection is the common exterior of a set of pseudodisks that all contain s and thus is simply connected. So FVD(S) is a graph drawn on P in such a manner that the resulting "spherical" map contains at most one simply connected region for each site of S, and each vertex of the diagram has degree at least three. By Euler's relation for planar graphs, the number of vertices and edges of FVD(S) is also $O(m)$. □

We define the *total complexity* of FVD(S) to be the sum of the number of vertices and breakpoints in FVD(S).

Theorem 1. *The maximum total complexity of the furthest-site Voronoi diagram* FVD(S) *of a set S of $m \leq n$ point sites on an orientable genus-0 polyhedral surface P consisting of n triangles is $\Theta(mn^2)$.*

Proof. Each edge of FVD(S) is a connected portion of some bisector $\beta(s, t)$ for two sites $s, t \in S$. Consequently, the upper bound follows from Lemmas 6 and 7.

As for the lower bound, we describe a construction that shows that FVD(S) for a set S of $m \leq n$ point sites on a non-convex polyhedron P with $O(n)$ edges can have total complexity $\Omega(mn^2)$. The construction will focus on proving an $\Omega(mn)$-bound for the number of breakpoints on a single edge of P. It is described for point sites in the plane with obstacles. This can then be "lifted" to a non-convex polyhedron.

First we describe the location of the sites, then the obstacles. Assume that $|S|$ is even; we split S into S_1 and S_2 with $k = m/2$ points each. Figure 2 shows the configuration of the sites $S_1 = \{s_1, \ldots, s_k\}$ (in the figure, $k = 5$). For ease of description, we also specify two additional points s_0 and s_{k+1}; these are *not* sites. The sites $s_1, \ldots, s_k \in S_1$ and the points s_0 and s_{k+1} are placed equally spaced on the lower semicircle of a circle $\mathcal{C}_1$. For $1 \leq i \leq k+1$, let b_{i-1} be the point where the bisector $\beta(s_{i-1}, s_i)$ meets the upper semicircle of $\mathcal{C}_1$. Note that any point on the (shorter) arc of $\mathcal{C}_1$ between b_{i-1} and b_i is

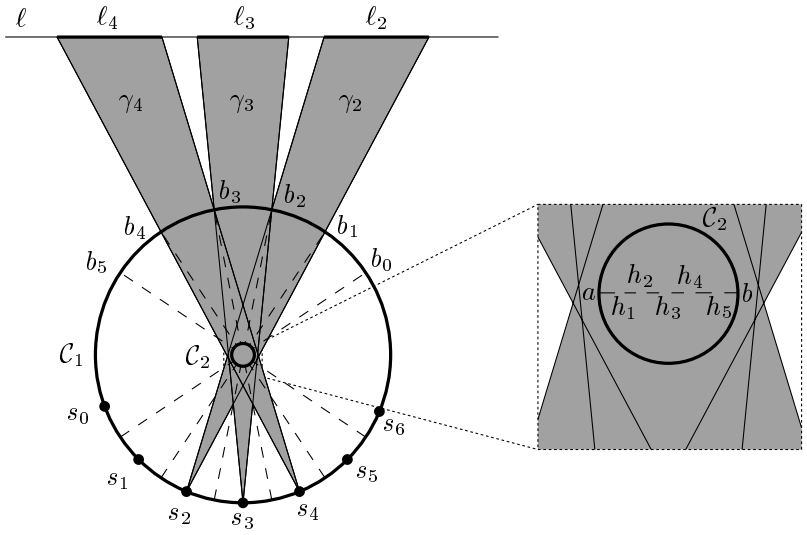


Fig. 2. The configuration of S_1 and the obstacles in $\mathcal{C}_2$ (detail).

further away from s_i than from any other site in S_1. Let γ_i denote the cone originating at site s_i that is bounded by the rays $\text{ray}(s_i, b_{i-1})$ and $\text{ray}(s_i, b_i)$. The portion of the cone γ_i outside $\mathcal{C}_1$ is further away from s_i than from any other site in S_1. Figure 2 only shows the cones γ_2, γ_3, and γ_4.

Let ℓ be a horizontal line lying some distance above the circle $\mathcal{C}_1$. The second set of sites $S_2 = \{s'_1, \ldots, s'_k\}$ is obtained by reflecting the set S_1 through ℓ. That is, s'_i is such that ℓ is the bisector of s_i and s'_i. The points in S_2 lie on a circle $\mathcal{C}'_1$ which is the reflection of $\mathcal{C}_1$. The cone γ'_i is defined analogously and is the reflection of γ_i. Let ℓ_i be the intersection of cone γ_i and ℓ. Note that ℓ_i is also the intersection of γ'_i and ℓ.

We have specified the point sites. Now we specify the location of the obstacles. It is important that the cones $\gamma_i, \ldots, \gamma_k$ have a common intersection around the center of circle $\mathcal{C}_1$. Let $\mathcal{C}_2$ be a small circle lying within this common intersection, and let the segment $\overline{ab}$ be the horizontal diameter of $\mathcal{C}_2$. Figure 2 (detail) shows the circle $\mathcal{C}_2$ and the segment $\overline{ab}$. Let $\overline{a'b'}$ be the reflection of $\overline{ab}$ through ℓ. Our obstacle set will be the segments $\overline{ab}$ and $\overline{a'b'}$ minus a number of narrow holes (through which a path can pass). The segment $\overline{ab}$ has an evenly spaced set $h_1, \ldots, h_n$ of narrow holes. The segment $\overline{a'b'}$ also has an evenly spaced set $h'_1, \ldots, h'_n$ of narrow holes; the only difference is that these holes are slightly shifted to the left.

We specified all the points and obstacles. Now, we argue that the line ℓ is intersected by $k = m/2$ edges of $\text{FVD}(S)$; the number of times that each of these edges crosses ℓ is $\Omega(n)$. We focus on the portion ℓ_i of the line ℓ. Since any point in ℓ_i is further away from s_i (resp. s'_i) than from any other site in S_1 (resp. S_2), s_i and s'_i are the only relevant sites for $\text{FVD}(S)$ near ℓ_i. We now argue that the number of times that $\beta(s_i, s'_i)$ crosses ℓ is $\Omega(n)$. For $1 \le j \le n$, let $p_{i,j}$ (resp. $p'_{i,j}$) be the point of intersection of the line through s_i (resp. s'_i) and h_j (resp. h'_j) and the line ℓ. Because of the horizontal shift of the holes in $\overline{a'b'}$, the points occur interleaved on ℓ_i in the sequence $p'_{i,1}, p_{i,1}, p'_{i,2}, p_{i,2}, \ldots, p'_{i,n}, p_{i,n}$.

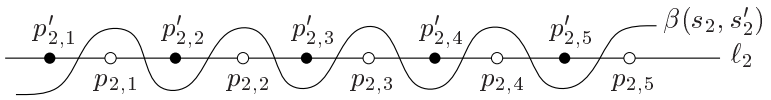


Fig. 3. Detail of $\beta(s_2, s'_2)$.

This is illustrated in Fig. 3 for ℓ_2. For $1 \leq j \leq n$, since s_i can "see" $p_{i,j}$ whereas s'_i cannot, there is a neighborhood around $p_{i,j}$ that is closer to s_i than to s'_i. By symmetric reasoning, there is a neighborhood around $p'_{i,j}$ that is closer to s'_i than to s_i. It follows that the bisector $\beta(s_i, s'_i)$ must cross ℓ_i between $p'_{i,j}$ and $p_{i,j}$, and also between $p_{i,j}$ and $p'_{i,j+1}$. Thus, $\beta(s_i, s'_i)$ crosses ℓ_i $\Omega(n)$ times, as illustrated in Fig. 3.

One gets $\Omega(kn) = \Omega(mn)$ crossings for line ℓ, namely, $\Omega(n)$ per ℓ_i. The pattern can be repeated on n lines parallel to ℓ and sufficiently close to ℓ. This gives $\Omega(mn)$ crossings for each of the n lines. The sites and the obstacles can be perturbed to lie in general position without affecting the lower bound complexity. By treating the lines as edges on a polyhedron, and raising vertical cylinders with the obstacles as bases, we obtain the claimed $\Omega(mn^2)$ bound for the total complexity of FVD(S) on a polyhedron. □

The facility center of S can be found by traversing the edges of FVD(S), and determining for each edge the point that minimizes the distance to the two sites of the regions on both sides of the edge. For a single edge, this can be done in $O(n^2)$ time by examining each elementary arc or line segment contributing to the edge. The minimum of these distances over all edges determines the location of the facility center. Since FVD(S) has $O(m)$ edges, we obtain the following.

Corollary 1. *Given* FVD(S), *the facility center of S can be computed in $O(mn^2)$ time.*

3. Computing the Furthest-Site Voronoi Diagram

Let S be a set of m sites on an orientable genus-0 triangulated polyhedral surface P with n triangles. We describe a randomized incremental algorithm, following the framework of Clarkson and Shor [5], to construct FVD(S). We pick a random permutation $\langle s_1, \ldots, s_m \rangle$ of the sites. Put $S_i = \{s_1, \ldots, s_i\}$. The algorithm consists of m stages; at the end of the ith stage, FVD(S_i) is computed. In the $(i+1)$th stage, the diagram is updated with the addition of the site s_{i+1}. We first describe the primitives that the algorithm needs and the data structure it maintains.

3.1. *Primitives*

Our algorithm invokes a number of primitive operations; each of these can be implemented in $O(n^2 \log n)$ time. Given these primitives, the overall running time of our algorithm can be described in terms of the number of invocations of the primitive operations, which will be a function depending only on m, the number of sites, and not on n,

the complexity of the surface. One primitive needed by our algorithm is the following: given a connected section γ of $\beta(s,t)$, $s,t \in S$, and a third site u, determine the at most two portions of γ that are further from u than from s and t. This primitive can be implemented by computing the FVD of the three sites, which can be done in $O(n^2 \log n)$ time. Another primitive is to compute the distance between two sites. Some other related primitives are also needed, but all of them can be implemented in $O(n^2 \log n)$ time by computing the furthest- or nearest-neighbor Voronoi diagram of two or three sites.

The computation of the nearest-neighbor Voronoi diagram of two or three sites can be done in $O(n^2 \log n)$ using the algorithms of Mitchell et al. [10] and Mount [11]. To compute the furthest-neighbor diagram $\mathrm{FVD}(\{s,t\})$ of two sites s and t, we simply compute their nearest-neighbor diagram. The region of s (resp. t) in $\mathrm{FVD}(\{s,t\})$ is clearly the region of t (resp. s) in the nearest-neighbor diagram. Thus $\mathrm{FVD}(\{s,t\})$ can be computed in $O(n^2 \log n)$ time. To compute the furthest-neighbor diagram $\mathrm{FVD}(\{s,t,u\})$ of three sites s, t, and u, we proceed as follows: We first compute the region of s in $\mathrm{FVD}(\{s,t\})$ and the region of s in $\mathrm{FVD}(\{s,u\})$. The region of s in $\mathrm{FVD}(\{s,t,u\})$ is clearly the intersection of these two regions. The key step in computing this intersection is to compute the intersection points of the bisectors $\beta(s,t)$ and $\beta(s,u)$. We compute the intersection points on each triangle of the polyhedron P separately (by using a standard sweepline algorithm [13]). Since each bisector consists of $O(n^2)$ line segments and hyperbolic arcs and the number of intersection points is bounded by a constant, the total time for computing all the intersection points is $O(n^2 \log n)$. (Alternatively, the vertices of $\mathrm{FVD}(\{s,t,u\})$ can be computed as the vertices of $\mathrm{V}(\{s,t,u\})$.) Once the intersection points are available, it is easy to obtain the region of s; we omit the straightforward details. In a similar manner, we compute the regions of t and u, and "glue" the three regions together to obtain $\mathrm{FVD}(\{s,t,u\})$. Thus the furthest-neighbor diagram of three sites can be computed in $O(n^2 \log n)$ time.

3.2. *Data Structure*

Given a set of sites $S' \subseteq S$ and a site $s \in S \backslash S'$, we say that a point p on the polyhedron *conflicts* with site s with respect to S' if the distance $\mathrm{d}(p,s)$ from p to s is strictly greater than the distance to any site in S'. Let $C(S',s)$ denote the points of the polyhedron that conflict with s with respect to S', and let $\mathcal{R}_{S''}(s)$ denote the region of s in $\mathrm{FVD}(S'')$ for $S'' \subseteq S$ and $s \in S''$. Observe that $C(S',s)$ is simply $\mathcal{R}_{S' \cup \{s\}}(s)$. Let $\alpha(S',s)$ denote the portion of $\mathrm{FVD}(S')$ that is contained in $C(S',s)$ (see Fig. 4); recall that the furthest-site Voronoi diagram is defined to be a planar graph embedded in P. The following lemma is easily proved using the fact that each region in $\mathrm{FVD}(S')$ and in $\mathrm{FVD}(S' \cup \{s\})$ is topologically a disk.

Lemma 8. *Let $S' \subseteq S$ be a subset of the sites and let $s \in S \backslash S'$ be a site not in S'. Then $\alpha(S',s)$ is connected; it is empty if and only if $\mathcal{R}_{S' \cup \{s\}}(s) = \emptyset$.*

The algorithm represents $\mathrm{FVD}(S_i)$ using standard techniques. (The coarse structure of the diagram is represented using techniques for representing planar Voronoi diagrams;

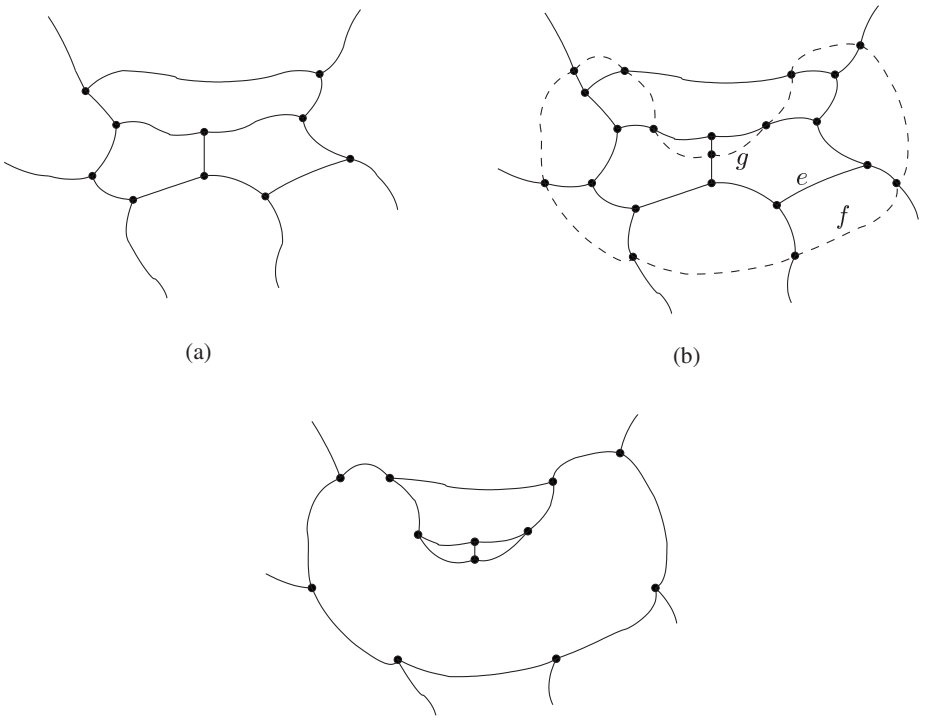


Fig. 4. (a) A portion of FVD(S_i); (b) for a site $s_{i+1} \notin S_i$, the region inside the dashed curve is $C(S_i, s_{i+1})$; the portion of FVD(S_i) inside this region is $\alpha(S_i, s_{i+1})$; assuming s_{i+1} is the site being added in the current step, the edges f and g are two edges of the new diagram; they are the horizon edges corresponding to edge e in FVD(S_i); (c) the updated diagram FVD(S_{i+1}).

the fine structure, the elementary arcs and line segments constituting each edge, does not need to be stored explicitly but can be re-computed when needed from the information about the edge.) The algorithm maintains, for each site s that is yet to be added, a pointer ptr(s) to a single edge e of the diagram FVD(S_i) such that $e \cap \alpha(S_i, s) \neq \emptyset$; we say that the edge e *conflicts* with site s with respect to FVD(S_i). If $\alpha(S_i, s) = \emptyset$, ptr($s$) is set to null. The algorithm also maintains, for each edge e of FVD(S_i), a list of all the sites $s \in S \backslash S_i$ such that ptr(s) points to e. Finally, it maintains a list of all the sites $s \in S \backslash S_i$ such that ptr(s) is not null. Notice that if a site $s \in S \backslash S_i$ is to own a region in FVD($S_i \cup \{s\}$), it must conflict with some edge of FVD(S_i), by Lemma 8.

3.3. *Initialization*

The algorithm starts off by computing the FVD of the two sites $\{s_1, s_2\}$. It also initializes the conflict pointer ptr(s) for every other site appropriately. It is straightforward to check that all this can be done using $O(m)$ primitive operations.

3.4. *The Incremental Step*

The incremental step starts with $\mathrm{FVD}(S_i)$ and calculates $\mathrm{FVD}(S_{i+1})$, where $S_{i+1} = S_i \cup \{s_{i+1}\}$. In order to describe how our algorithm computes $\mathrm{FVD}(S_{i+1})$ and updates the conflict pointers, we need to make some preliminary observations. Suppose that $\alpha = \alpha(S_i, s_{i+1})$ is neither empty nor the entire $\mathrm{FVD}(S_i)$. Consider a region $\mathcal{R}_{S_i}(s_j)$ (of a site $s_j \in S_i$), a proper subset of which is contained in $C(S_i, s_{i+1})$. The intersection $\mathcal{R}_{S_i}(s_j) \cap C(S_i, s_{i+1})$ can have several connected components. However, the boundary of each such component is topologically a circle and consists of a connected portion γ of the boundary of $\mathcal{R}_{S_i}(s_j)$ plus a single edge e of the new diagram $\mathrm{FVD}(S_{i+1})$ that is incident to $\mathcal{R}_{S_{i+1}}(s_{i+1})$ and $\mathcal{R}_{S_{i+1}}(s_j)$. For each point p on γ, we call the edge e the "horizon edge" of p with respect to $\mathcal{R}_{S_i}(s_j)$. Note that a point in α that is not a vertex has up to two horizon edges, one with respect to each region it is incident to. Also note that for an edge e' of $\mathrm{FVD}(S_i)$, $e' \cap \alpha$ can be empty, be equal to e', or consist of up to two connected portions. (The existence of three or more connected components would contradict Lemma 4.) All the points on any one connected portion of $e' \cap \alpha$ have the same horizon edges, and we call these the horizon edges of the portion. Thus the edge e' can have up to four horizon edges associated with it.

Updates During the Incremental Step. We now describe how $\mathrm{FVD}(S_{i+1})$ is obtained from $\mathrm{FVD}(S_i)$ and how the pointers are updated.

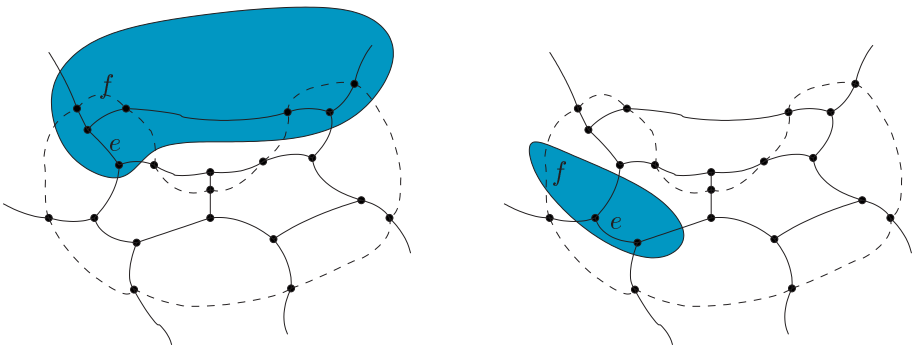
If $\mathrm{ptr}(s_{i+1})$ is null (α is empty) the diagram remains the same, so we do nothing; $\mathrm{FVD}(S_{i+1}) = \mathrm{FVD}(S_i)$ and no pointers need to be changed. If $\mathrm{ptr}(s_{i+1})$ points to an edge, we use that edge as a starting point to do a depth-first search of $\mathrm{FVD}(S_i)$ to determine the portion α of the edge skeleton that lies in $C(S_i, s_{i+1})$; recall that α is connected (Lemma 8). The number of primitive operations needed to compute α is proportional to the number of edges in $\mathrm{FVD}(S_i)$ that contribute to α.

We first treat the special case where α is the entire edge skeleton of $\mathrm{FVD}(S_i)$. Notice that in this case $\mathrm{FVD}(S_{i+1})$ is $\mathrm{FVD}(\{s_j, s_{i+1}\})$, where s_j is the (unique) site in S_i that is furthest from s_{i+1}. We determine s_j by computing the distance of s_{i+1} from each site that has a region in $\mathrm{FVD}(S_i)$ and setting s_j to be the site that maximizes this distance. We then compute $\mathrm{FVD}(\{q, p\})$. The number of primitive operations is proportional to the number of sites having a region in $\mathrm{FVD}(S_i)$. To update the conflict lists, we look at each site s in $S \backslash S_{i+1}$ such that $\mathrm{ptr}(s)$ is not null. We check if there is a point on $\beta(s_j, s_{i+1})$ further from s than from s_j and s_{i+1}; if so, we set $\mathrm{ptr}(s)$ to the corresponding edge, otherwise we set it to null. The number of primitive operations needed is proportional to the number of such sites s.

We now consider the case where α is neither empty nor the entire $\mathrm{FVD}(S_i)$. In this case an edge e of $\mathrm{FVD}(S_i)$ that contributes to α can be of one of two types: (a) if $e \cap \alpha = e$, e completely disappears from the diagram after s_{i+1} is added, or (b) if $e \cap \alpha \neq e$, one or two portions of e will appear in the new diagram. For accounting purposes, we say that in either case the edge e is "deleted" on adding s_{i+1}. (Thus case (b) involves deleting an edge from the current diagram and creating one or two new edges in the new diagram.) Once α is available, we can traverse it and update the diagram using standard techniques: this involves identifying the new vertices and the newly created edges (the ones bounding the region of s_{i+1} in $\mathrm{FVD}(S_{i+1})$ and the portions of those edges of $\mathrm{FVD}(S_i)$ that contributed

to α). In the process we can also find the horizon edges, if any, for each edge of FVD(S_i) that contributes to α. Since these details are quite standard, we omit them here. It is easy to see that the diagram can be updated using a number of primitive calls that is proportional to the number of edges deleted plus the number of edges created.

We now describe how we update the pointers ptr(s) for each $s \in S \backslash S_{i+1}$. We will do this only if the edge e pointed to by ptr(s) gets deleted (otherwise no update is necessary). Notice that e contributes to α. If $e \cap \alpha = e$, we compute a witness point p on e that conflicts with s (with respect to S_i). If $e \cap \alpha \neq e$, either (a) one section of e is an edge e' in the new diagram; we set ptr(s) to point to e' if e' conflicts with s; or (b) two sections of e become edges e' and e'' in the new diagram; if either of these edges conflicts with s, we set ptr(s) to point to the appropriate edge. If we have not yet reset ptr(s), we compute a point p on $e \cap \alpha$ that conflicts with s (with respect to S_i); we know that such a point must exist because e conflicts with s (with respect to S_i). Consider the portion $\alpha \cap \alpha(S_i, s)$. We do a depth-first-search to compute λ, the connected component of this portion that contains p. If the closure of λ contains a point on the boundary of the new region $\mathcal{R}_{S_{i+1}}(s_{i+1})$, we are done, because we have discovered a vertex of the new diagram, and this vertex lies in $\alpha(S_{i+1}, s)$. (We are using here the non-degeneracy assumption that no point is equally close to four sites.) We update ptr(s) to point to either of the two new edges incident to this vertex. Otherwise (if the closure of λ does not contain a vertex of the new diagram), we look at each edge e' of FVD(S_i) that contributes to λ and check if any of the four horizon edges of e' conflict with s (with respect to S_{i+1}); if any of them does, we reset ptr(s) to point to that edge. If after all this we have not found an edge that conflicts with s, we set ptr(s) to null. It is easy to see that the number of primitive calls needed to implement this step (given the updated diagram and the horizon edges) is proportional to the complexity of λ, which is exactly the number of deleted edges that conflicts with s. See Fig. 5 for an illustration of this step. We now argue the correctness of the above scheme for updating ptr(s).



(a) (b)

Fig. 5. Both (a) and (b) show a portion of FVD(S_i), i.e., $p \in \alpha \cap \alpha(S_i, s)$; the region inside the dashed curve is $C(S_i, s_{i+1})$; the shaded region is $C(S_i, s)$. Before s_{i+1} is added, ptr(s) points to e. After s_{i+1} is added, it gets updated to the edge f of FVD(S_{i+1}). In (a) the edge f intersects the closure of λ. In (b) f is a horizon edge of some edge contributing to λ.

Lemma 9. *Let p be a point on α that conflicts with $s \in S \setminus S_{i+1}$ with respect to S_i. Let λ be the connected component of $\alpha \cap \alpha(S_i, s)$ that contains p. If $\alpha(S_{i+1}, s)$ is non-empty, it either contains a point in the closure of λ or it intersects a horizon edge for some edge of* FVD(S_i) *that contributes to λ.*

Proof. We first consider the case when $\alpha(S_{i+1}, s)$ contains a point q that is also on $\alpha(S_i, s)$. Since α's intersection with FVD(S_{i+1}) is empty, q does not lie in α and therefore not in λ. Since $\alpha(S_i, s)$ is connected (by Lemma 8), there is a path π from p to q that stays within $\alpha(S_i, s)$. Since $p \in \lambda$, $q \notin \lambda$, there is a point r where the path π first leaves λ. The point r is in the closure of α (and also in the closure of γ). Therefore its distance to s_{i+1} is equal to its distance to the furthest site in S_i. Since r lies in $\alpha(S_i, s)$, it follows that it lies in $\alpha(S_{i+1}, s)$.

We now consider the case when $\alpha(S_{i+1}, s)$ does not contain any point that is also on $\alpha(S_i, s)$. Observe that this means that $\alpha(S_i, s) \subseteq \alpha$, and therefore $\lambda = \alpha(S_i, s)$. We may assume $\alpha(S_{i+1}, s)$ contains a point q lying on edge f, where f is one of the edges bounding $\mathcal{R}_{S_{i+1}}(s_{i+1}) = C(S_i, s_{i+1})$. Since $q \in \alpha(S_{i+1}, s)$, q belongs to $C(S_i, s)$. Since $p, q \in C(S_i, s)$ and $C(S_i, s)$ is path-connected, there is a path π from p to q that stays within $C(S_i, s)$. Since $p \in C(S_i, s_{i+1})$ and $q \notin C(S_i, s_{i+1})$, there is a point q' where that path first leaves $C(S_i, s_{i+1})$. The point q' lies on the boundary of $C(S_i, s_{i+1})$, so its distance to s_{i+1} equals its distance to the furthest site in S_i. This implies that $q' \in C(S_{i+1}, s)$. Notice that q' is in the interior of some edge f' that bounds $C(S_i, s_{i+1})$ in FVD(S_{i+1}). Let p' be the last point on the subpath of π from p to q' that lies on an edge of FVD(S_i). (There is such a point because p lies on an edge of FVD(S_i).) We assume that e' is the edge on which p' lies. Since $p' \in \alpha(S_i, s)$, p' lies in λ, and so e' contributes to λ. It is easy to see that f' is a horizon edge corresponding to e'. □

3.5. *Running time*

When the point s_{i+1} is added, the number of primitive calls needed to update the diagram is proportional to the number of edges of FVD(S_i) that get deleted. The time needed to update the conflict pointers is proportional to the sum, over all sites yet to be added, of the number of deleted edges from FVD(S_i) that conflicted with the site. This is exactly what is needed to show that the expected number of primitives used by the entire algorithm is $O(m \log m)$. We refer the reader to the paper by Clarkson and Shor [5] or the book by Mulmuley [12].[2]

Theorem 2. *The furthest-site Voronoi diagram* FVD(S) *of m sites on an orientable genus-0 polyhedral surface composed of n triangles can be computed in $O(mn^2 \log m \log n)$ time.*

[2] To apply the framework of randomized incremental construction developed in these works, we have to be more precise in defining the identity of an edge and in saying what it means for an edge to be in the diagram of a set of sites. It is easy to verify that the standard way of doing this ensures the validity of the claims made about the number of primitive calls needed to update the diagram and the conflict pointers.

4. Conclusions and Further Research

We have shown that the furthest-site Voronoi diagram of a set S of m sites on the surface of a genus-0 polyhedron P with n triangles has maximum complexity $\Theta(mn^2)$, and we have given an algorithm for computing the diagram in $O(mn^2 \log m \log n)$ randomized expected time. A preliminary version of this paper gives a more complicated deterministic algorithm for computing the diagram in $O(mn^2 \log^2 m \log n)$ time [1]. Once the diagram has been computed, the facility center, which is the point on P that minimizes the maximum distance to a site in S, can be found in $O(mn^2)$ time by traversing the edges of the diagram. We have explained in [2] how the problems of locating the facility center on a terrain can be reduced to that on a polyhedral surface.

It would be a challenge to find an output-sensitive algorithm, i.e., an algorithm that takes time proportional to the number of elementary arcs and segments in the diagram plus the number of their intersections with the edges of P. Even more ambitious would be the computation of the diagram without explicitly representing all intersections of furthest-site Voronoi edges and edges of the polyhedron.

Another interesting issue is approximation: find (in $o(mn^2)$ time) a point with the property that the distance to the furthest site is at most $(1 + \varepsilon)$ times the radius of the smallest "enclosing circle."

Finally, it is worth investigating whether the 1-center problem can be solved without constructing the furthest-site Voronoi diagram. Recall that the 1-center problem in the plane can be solved using techniques related to fixed-dimensional linear programming [9], [17]. The same question on a polyhedral surface does not fit into the framework of so-called *LP-type* problems. It is easy to construct an example with five sites on (a polyhedral surface approximating) the unit sphere that demonstrates this.

Acknowledgments

We thank an anonymous referee for suggesting investigating the use of the framework of Clarkson and Shor [5] instead of our original (more complicated) deterministic algorithm [1] for computing the furthest-site Voronoi diagram of the sites on the polyhedron.

References

1. B. Aronov, M. van Kreveld, R. van Oostrum, and K. Varadarajan. Facility location on terrains. In *Algorithms and Computation*, *Proceedings of the* 9*th International Symposium* (*ISAAC* '98), volume 1533 of Lecture Notes in Computer Science, pages 19–28. Springer-Verlag, Berlin, 1998.
2. B. Aronov, M. van Kreveld, R. van Oostrum, and K. Varadarajan. Facility location on terrains. Technical Report UU-CS-2001-56, Institute of Information and Computing Sciences, Utrecht University, 2001, `http://archive.cs.uu.nl/pub/RUU/CS/techreps/CS-2001/2001-56.pdf`. Full version of [1].
3. B. Aronov and J. O'Rourke. Nonoverlap of the star unfolding. *Discrete Comput. Geom.*, 8:219–250, 1992.
4. J. Chen and Y. Han. Shortest paths on a polyhedron. *Internat. J. Comput. Geom. Appl.*, 6:127–144, 1996.
5. K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

6. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math*., 1:269–271, 1959.
7. J. Eckhoff. Helly, Radon and Carathéodory type theorems. In *Handbook of Convex Geometry*, pages 389–448. North-Holland, Amsterdam, 1993.
8. D. Leven and M. Sharir. Intersection and proximity problems and Voronoi diagrams. In J. T. Schwartz and C.-K. Yap, editors, *Advances in Robotics* 1: *Algorithmic and Geometric Aspects of Robotics*, pages 187–228. Erlbaum, Hillsdale, NJ, 1987.
9. N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput*., 12:759–776, 1983.
10. J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput*., 16:647–668, 1987.
11. D. M. Mount. Voronoi diagrams on the surface of a polyhedron. Technical Report 1496, Department of Computer Science, University of Maryland, 1985.
12. K. Mulmuley. *Computational Geometry*: *An Introduction Through Randomized Algorithms*. Prentice-Hall, Englewood Cliffs, NJ, 1993.
13. F. P. Preparata and M. I. Shamos. *Computational Geometry*: *An Introduction*. Springer-Verlag, New York, 1990.
14. E. Ramos. Intersection of unit-balls and diameter of a point set in R^3. *Comput. Geom. Theory Appl*., 8:57–65, 1997.
15. M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput*., 15:193–215, 1986.
16. M. J. van Trigt. Proximity problems on polyhedral terrains. Master's Thesis, Department of Computer Science, Utrecht University, 1995. INF/SCR-95-18.
17. E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, volume 555 of Lecture Notes in Computer Science, pages 359–370. Springer-Verlag, Berlin, 1991.

Received July 28, 2000, *and in revised form March* 13, 2003. *Online publication August* 6, 2003.