Facility Location on Terrains*
(Extended Abstract)

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

[1] Department of Computer and Information Science, Polytechnic University
Brooklyn, NY 11201-3840, USA
aronov@ziggy.poly.edu
[2] Department of Computer Science, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, Netherlands
{marc,rene}@cs.uu.nl
[3] Department of Computer Science, Duke University
Durham, NC 27708-0129, USA
krv@cs.duke.edu

Abstract. Given a *terrain* defined as a piecewise-linear function with n triangles, and m point *sites* on it, we would like to identify the location on the terrain that minimizes the maximum distance to the sites. The distance is measured as the length of the Euclidean shortest path along the terrain. To simplify the problem somewhat, we extend the terrain to (the surface of) a polyhedron. To compute the optimum placement, we compute the furthest-site Voronoi diagram of the sites on the polyhedron. The diagram has maximum combinatorial complexity $\Theta(mn^2)$, and the algorithm runs in $O(mn^2 \log^2 m(\log m + \log n))$ time.

1 Introduction

Problem statement. A *(polyhedral) terrain* is the graph of a piecewise-linear function defined over a simply-connected subset of the plane. It can be represented by a planar triangulation where each vertex has an associated elevation. The elevation of any point in the interior of an edge (triangle) is obtained by linear interpolation over the two (three) vertices of the edge (resp. triangle). The polyhedral terrain is commonly used as a model for (mountainous) landscapes.

This paper addresses the *facility location problem* for a set of sites on a terrain. More precisely, assume that a set of m point *sites* on a terrain defined by n triangles is given. The distance between two points on the terrain is the minimum length of any path between those points that lies on the terrain. The *facility center* of the sites is the point on the terrain that minimizes the maximum

* B.A. has been partially supported by a Sloan Research Fellowship. 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.

K.-Y. Chwa and O. H. Ibarra (Eds.): ISAAC'98, LNCS 1533, pp. 19–29, 1998.

distance to a site. We assume throughout that $m \leq n$. To avoid complications involving the boundary of the terrain, we extend the terrain to (the surface of) a polyhedron. We allow only polyhedra homeomorphic to a ball, so that their surfaces are homeomorphic to a sphere. We assume that all faces of the polyhedron have been triangulated. This increases the combinatorial complexity of the polyhedron by at most a constant factor.

Previous results. In the Euclidean plane, the facility center, or the center of the *smallest enclosing disc* of a set of m point sites, can be determined in $O(m)$ time. Several algorithms attain this bound. Megiddo [4] gave the first deterministic linear-time algorithm, and a much simpler, linear expected time algorithm was found by Welzl [10].

In his master's thesis, van Trigt [9] gave an algorithm that solves the facility location problem on a polyhedral terrain in $O(m^4 n^3 \log n)$ time, using $O(n^2(m^2 + n))$ space.

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.

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

Mount [6] showed that the *nearest-neighbor* Voronoi diagram of m sites on (the surface of) a polyhedron with n faces with $m \leq n$ 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 do not know of any previous work on furthest-site Voronoi diagrams on a polyhedron.

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 [8], Mitchell, Mount and Papadimitriou [5], and Chen and Han [1]. The best known algorithms [1,5] 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 us to compute the shortest path distance between the source s to any query point p in $O(\log n)$ time.

New results and methods. This paper gives an $O(mn^2 \log^2 m(\log m + \log n))$ time algorithm to compute the furthest-site Voronoi diagram and find the facility center for a set S of m sites on the surface of a polyhedron with n faces. Given the linear-time algorithm for finding the facility center in the plane, this bound may seem disappointing. However, the algorithm for computing the furthest-site Voronoi diagram is near-optimal, as the maximum combinatorial complexity of the diagram is $\Theta(mn^2)$.

2 Geometric Preliminaries

Previous papers on shortest paths on polyhedra [8,5,1,9] use a number of important concepts that we'll need as well. We review them briefly after giving the relevant definitions.

In the remainder of the paper P is (the surface of) a polyhedron. As stated before, we allow only polyhedra homeomorphic to a ball, so that their surfaces are homeomorphic to a sphere. Let S be a set of m point sites on P. Consider first a single site $s \in P$. For any point p on P we consider a shortest path from p to s; note that in general it is not unique. Such a shortest path has a number of properties. First, if it crosses an edge of P properly, then the principle of reflection 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 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. If the path does not contain any vertex, then site s is called the pseudoroot of p.

The *shortest path map (SPM) of* s is defined as the subdivision of P into connected regions where the shortest path to s is unique and has a fixed edge sequence. For non-degenerate 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 are one-dimensional. It is known that the shortest path map 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 Euclidean Voronoi diagram for a set of sites with additive weights. The sites are obtained from the pseudoroots of points on the triangle. The coordinates of the diagram sites are obtained by unfolding the triangles in the edge sequence to the pseudoroot so that they are all co-planar. The weight of a pseudoroot is the distance from the 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 on a hyperbolic arc or a segment there are two shortest paths to s with different pseudoroots.

Given two sites s and t on the polyhedron, the *bisector* $\beta(s,t)$ is the set of all points on the polyhedron whose shortest path to s has length equal to the shortest path to 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. For simplicity, we assume that these degeneracies don't occur.

Fix an integer k, $1 \leq k \leq m-1$. For a point $p \in P$, let $S(p,k) \subset S$ be any set of k sites in S for which the shortest path to p is not longer than the path from p to any site in $S \setminus S(p,k)$. The *order-k Voronoi diagram* of a set S of m sites on P is a planar graph embedded in P that subdivides P into open regions such that for any point p in a region, $S(p,k)$ is unique, and such that for any two points p and q in the same region, $S(p,k) = S(q,k)$. The interior of the boundary between two adjacent regions is an *edge* of the order-k 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 order-k

Voronoi diagram are its *vertices*. We assume that all vertices have degree three; otherwise, a degeneracy is present.

The order-1 Voronoi diagram is known as the (standard) Voronoi diagram or *closest-site Voronoi diagram*, and the order-$(m-1)$ Voronoi diagram is also called the *furthest-site Voronoi diagram*. In this paper, we only deal with the closest and furthest site Voronoi diagrams, and we give a new algorithm for computing the furthest-site Voronoi diagram of a set S of sites on a polyhedron. We denote it by FVD(S), and refer to it more loosely as *the diagram*. For the closest- (furthest-) site diagrams of S, the *region* $\mathcal{R}(s)$ of a site $s \in S$ is the locus of all points that are closer to (further from) s than to (from) all other points in S.

The following facts are crucial for the algorithm below to work and for the analysis to hold. Lemmas 1, 2, and 3 are similar to the lemmas in Leven and Sharir [3]; they are general statements about 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.*

Lemma 2. *Bisector $\beta(s,t)$ is connected, and is homeomorphic to a circle.*

Lemma 3. *For any three distinct sites s, t, u, bisectors $\beta(s,t)$, $\beta(s,u)$ intersect at most twice.*

Any family of simple closed curves (in this case, on a topological sphere) every pair of which crosses at most twice is called a *family of pseudocircles*. Thus for every fixed $s \in S$, bisectors $\{\beta(s,t) : t \neq s\}$ form a family of pseudocircles.

Lemma 4. *Bisector $\beta(s,t)$ consists of $O(n^2)$ straight-line segments and hyperbolic arcs.*

Proof. The Voronoi diagram of $\{s,t\}$ can be constructed by techniques similar to those used by Mitchell *et al.* [5] for the computation of the shortest path map of a single site. By adapting their analysis to the case with two sites, one can show that on each triangle of the polyhedron, $\beta(s,t)$ consists of $O(n)$ *elementary arcs* (straight-line segments and hyperbolic arcs). Summing this over all the triangles gives $O(n^2)$. See the paper by Mount [6]. □

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

Lemma 5. *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. Let $\mathcal{R}_{s>t}$ be the region of points that are further from s than from t, for $s, t \in S$. In this notation $\mathcal{R}(s) = \bigcap_{t \in S, t \neq s} \mathcal{R}_{s>t}$. From Lemma 3 it follows that this intersection is the intersection of a set of pseudo-disks. It follows that for

each $s \in S$, the region $\mathcal{R}(s)$ in FVD(S) is connected. So we have at most one cell (region) for each site in S, and, 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).

Lemma 6. *The maximum total complexity of FVD(S) is $\Theta(mn^2)$.*

Proof. Each edge of FVD(S) is part of some bisector $\beta(s,t)$ for two sites $s,t \in S$. Consequently, the upper bound follows from Lemmas 5 and 4.

As for the lower bound, we describe a construction that shows that FVD(S) for a set S of m point sites on a non-convex polyhedron P with n edges can have complexity $\Omega(mn^2)$. The construction will focus on proving an $\Omega(mn)$-bound for 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 will 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 1 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 semi-circle 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 semi-circle of $\mathcal{C}_1$. Note that any point on the arc of the upper semi-circle $\mathcal{C}_1$ between b_{i-1} and b_i is 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 ray(s_i, b_{i-1}) and ray(s_i, b_i). The portion of the cone γ_i that lies outside $\mathcal{C}_1$ is further away from s_i than from any other site in S_1. Figure 1 shows just 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 the image of s_i from reflecting in ℓ. 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 will specify the location of the obstacles. The important fact is 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 1 (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 few point holes (through which a path can pass). The segment $\overline{ab}$ has an evenly spaced set $h_1, \ldots, h_n$ of point holes. The segment $\overline{a'b'}$ also has an evenly spaced set $h'_1, \ldots, h'_n$ of point holes; the only difference is that these holes are slightly shifted to the left.

We specified all the points and obstacles. Now, we will argue that the line ℓ is intersected by $k = m/2$ edges of FVD(S), each of which crosses ℓ $\Omega(n)$ times. Let us 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) then from any other site in S_1 (resp. S_2), s_i and s'_i are the only interesting sites for ℓ_i. We will now argue that $\beta(s_i, s'_i)$

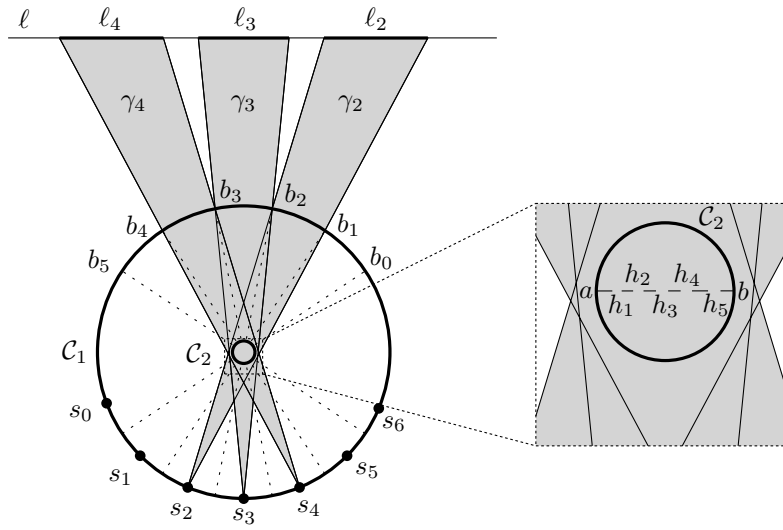

Fig. 1. The configuration of S_1 and the obstacles in $\mathcal{C}_2$ (detail).

crosses ℓ $\Omega(n)$ times. For $1 \leq j \leq n$, 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 as the sequence $p'_{i,1}, p_{i,1}, p'_{i,2}, p_{i,2}, \ldots, p'_{i,n}, p_{i,n}$. This is illustrated in Figure 2 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 Figure 2.

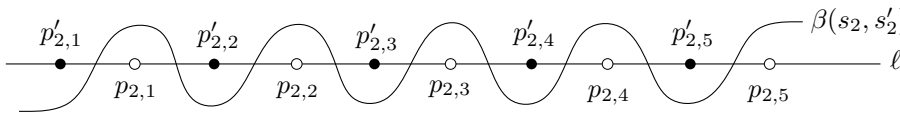

Fig. 2. Detail of $\beta(s_2, s'_2)$.

One gets $\Omega(kn) = \Omega(mn)$ crossings for line ℓ, $\Omega(n)$ for each 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 a 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 can get the $\Omega(mn^2)$ bound for a polyhedron. □

Using standard arguments, and the fact that FVD(S) has maximum total complexity $O(mn^2)$, 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

In this section, we describe our algorithm for computing the furthest-site Voronoi diagram of the given set S of sites on the polyhedron P. Our algorithm uses ideas from the algorithm of Ramos [7] for computing the intersection of unit spheres in three dimensions. We first give an outline of the algorithm, and get into the details in the subsequent subsections.

The algorithm for computing FVD(S) works as follows:

- As a preprocessing step, compute the shortest path map for every site in S.
- Subdivide S into two subsets R (the "red" sites) and B (the "blue" sites) of about equal size.
- Recursively compute FVD(R) and FVD(B).
- Merge FVD(R) and FVD(B) into FVD($R \cup B$) = FVD(S) as follows:
  - Determine the set of sites $R_0 \subset R$ that have a non-empty region in FVD(R), i.e., FVD(R) = FVD(R_0). Observe that the remaining sites in $R \setminus R_0$ don't influence the final diagram. Similarly, compute $B_0 \subset B$.
  - Determine an *low-degree independent set* $M \subset R_0$, which is a subset with the property that the region of a site $s \in M$ has at most 10 neighbors in FVD(R_0), and no two sites $s, s' \in M$ are neighbors in FVD(R_0). (Two sites are said to be *neighbours* if their regions share an edge of the diagram.) Compute $R_1 = R_0 \setminus M$ and FVD(R_1), and repeat this step to generate a Dobkin-Kirkpatrick hierarchy [2] $R_0 \supset R_1 \supset \ldots \supset R_k$ and their furthest-site Voronoi diagrams, such that R_k has only a constant number of sites. Do the same for the blue sites to achieve $B_0 \supset B_1 \supset \ldots \supset B_l$ and their furthest-site Voronoi diagrams.
  - Compute FVD($R_i \cup B_l$) for $0 \leq i \leq k$, exploiting the fact that B_l has only a constant number of sites. Similarly, compute FVD($R_k \cup B_j$) for $0 \leq j \leq l$. This is the *basic merge step*.
  - Compute FVD($R_i \cup B_j$) from FVD($R_i \cup B_{j+1}$) and FVD($R_{i+1} \cup B_j$). This is the *generic merge step*, which when repeated gives FVD($R_0 \cup B_0$) = FVD(S).

3.1 Constructing the Hierarchy for R_0 and B_0.

We describe how to compute the hierarchy $R_0 \supset R_1 \supset \ldots \supset R_k$ and their furthest-site Voronoi diagrams; for B_0, this is done analogously. The computation is similar to the Dobkin-Kirkpatrick hierarchy [2]. Using the fact that furthest-site Voronoi diagrams and their dual graphs are planar graphs, we can show that there is a low-degree independent set $M \subset R_0$ such that M contains a constant fraction of the sites of R_0. In fact, such an independent set can be found in $O(|R_0|)$ time using a greedy algorithm [2].

As for the computation of $\mathrm{FVD}(R_1) = \mathrm{FVD}(R_0 \setminus M)$, we remove the sites in M one at a time from R_0 and update the diagram after the removal of each site. Let s be such a site in M, and let p be a point that lies in the region in $\mathrm{FVD}(R_0)$ of s. After updating the diagram, p must lie in the region of a site s' that is a neighbour of s in $\mathrm{FVD}(R_0)$. So the region of s is divided among its neighbors, of which there are only a constant number, and all diagram edges in that region lie on the bisectors of those neighbors. Computing the bisector for two sites takes $O(n^2 \log n)$ using the techniques from Mitchell, Mount and Papadimitriou[5], and computing the bisector for every pair of neighbors of s takes asymptotically the same time. Given these bisectors, we can easily trace the edges of the diagram in the region of s in $O(n^2 \log n)$ time, using the vertices of that region as starting points.

After all the sites in M have been removed from R_0 and $\mathrm{FVD}(R_1)$ has been constructed, we repeat this procedure to create $\mathrm{FVD}(R_2), \ldots, \mathrm{FVD}(R_k)$. The total number of diagrams we construct this way is $O(\log m)$.

By lemma 6, the size of $\mathrm{FVD}(R_i) = O(|R_i|n^2)$.

Since $\sum_{i=0}^{k} |R_i|$ is a geometric series, the total time for computing all independent sets is $O(m)$. The computation of the bisectors of neighbors and the reconstruction of the diagram after the removal of a single site from a diagram takes $O(n^2 \log n)$ time, and the total number of sites removed is less than m. It follows that the construction of the the hierarchy $R_0 \supset R_1 \supset \ldots \supset R_k$ and their furthest-site Voronoi diagrams takes $O(mn^2 \log n)$ time in total. The total size of all diagrams constructed is $O(mn^2)$.

3.2 The Generic Merge Step

The generic merge step is the computation of $\mathrm{FVD}(R_i \cup B_j)$ from $\mathrm{FVD}(R_i \cup B_{j+1})$ and $\mathrm{FVD}(R_{i+1} \cup B_j)$, which when repeated gives the required $\mathrm{FVD}(R_0 \cup B_0) = \mathrm{FVD}(S)$. First some terminology: we call the sites in R_{i+1} the *old* red sites, and the sites in $R_i \setminus R_{i+1}$ the *new* red sites. Similarly, the sites in B_{j+1} are the old blue sites, and the sites in $B_j \setminus B_{j+1}$ are the new blue sites. Now consider any vertex v of $\mathrm{FVD}(R_i \cup B_j)$. The important fact is that not all three Voronoi regions incident to that vertex correspond to new sites; there must be at least one old red or blue site whose face is incident to v, because new red (blue) regions form an independent set in $\mathrm{FVD}(R_i)$ (resp. $\mathrm{FVD}(B_j)$). So to determine all the vertices of $\mathrm{FVD}(R_i \cup B_j)$, it suffices to compute the regions in $\mathrm{FVD}(R_i \cup B_j)$ of all old red and blue sites.

Consider an old red site r. The region of r in $\mathrm{FVD}(R_i \cup B_{j+1})$ contains all points that are further from r than from any other site in $R_i \cup B_{j+1}$, and the region of r in $\mathrm{FVD}(R_{i+1} \cup B_j)$ contains all points that are further from r than from any other site in $R_{i+1} \cup B_j$. The region of r in $\mathrm{FVD}(R_i \cup B_j)$ is therefore the intersection of its regions in $\mathrm{FVD}(R_i \cup B_{j+1})$ and $\mathrm{FVD}(R_{i+1} \cup B_j)$. We can compute this intersection for each face of the polyhedron separately by a line-sweep of the regions of r in $\mathrm{FVD}(R_i \cup B_{j+1})$ and $\mathrm{FVD}(R_{i+1} \cup B_j)$. The time needed for computing the vertices of $\mathrm{FVD}(R_i \cup B_j)$ is therefore bounded by $O(C \log C)$, where $C = \max(n^2|R_i \cup B_{j+1}|, n^2|R_{i+1} \cup B_j|, n^2|R_i \cup B_j|)$, which in turn equals $n^2(|R_i| + |B_j|)$. Hence, computing the vertices of $\mathrm{FVD}(R_i \cup B_j)$ takes $O(n^2(|R_i| + |B_j|) \log(n^2(|R_i| + |B_j|))) = O(n^2(\log n + \log m)(|R_i| + |B_j|))$.

The edges of $\mathrm{FVD}(R_i \cup B_j)$ are either edges incident to the faces of old red or blue sites (which we already computed), or edges between the faces of two new sites of the same color (these edges are sub-edges of edges in $\mathrm{FVD}(R_i)$ or $\mathrm{FVD}(B_j)$, and can easily be found), or they are edges between the faces of a new red and a new blue site. For the latter category of edges we already have the incident vertices computed, and we can trace the edges after computing the bisector of the new red and new blue site. The total number of bisectors we have to compute and trace is bounded by $|R_i \cup B_j|$, so this takes $O(n^2 \log n(|R_i|+|B_j|))$ time. We conclude that computing $\mathrm{FVD}(R_i \cup B_j)$ from $\mathrm{FVD}(R_i \cup B_{j+1})$ and $\mathrm{FVD}(R_{i+1} \cup B_j)$ takes $O(n^2(\log n + \log m)(|R_i| + |B_j|))$ time.

Summing this over all $0 \leq i \leq k$, $0 \leq j \leq l$ gives

$$O(n^2(\log n + \log m)\sum_{i=0}^{k}\sum_{j=0}^{l}(|R_i| + |B_j|)) \tag{1}$$

We have

$$\sum_{i=0}^{k}\sum_{j=0}^{l}|B_j| = O(\sum_{i=0}^{k}|B_0|) = O(k|B_0|) = O(m \log m), \tag{2}$$

and similarly $\sum_{i=0}^{k}\sum_{j=0}^{l}(|R_i|)$ is $O(m \log m)$. It follows that the total time spent in all the iterations of the generic merge step is $O(mn^2 \log m(\log m + \log n))$.

3.3 The Basic Merge Step

In the basic merge step, we compute $\mathrm{FVD}(R_i \cup B_l)$ for $0 \leq i \leq k$, and $\mathrm{FVD}(R_k \cup B_j)$ for $0 \leq j \leq l$. We will exploit the fact that B_l and R_k contain only a constant number of sites. We will only describe the computation of $\mathrm{FVD}(R_i \cup B_l)$ for a fixed i; all other diagrams are computed similarly.

1. For each site $r \in R_i$ and $b \in B_l$, we compute the region of r in $\mathrm{FVD}(\{r, b\})$. To do this, we compute the *closest-site* Voronoi diagram for sites r and b using the $O(n^2 \log n)$ algorithm of Mount [6]. The region of r in $\mathrm{FVD}(\{r, b\})$ is clearly the region of b in the closest-site diagram. The total time for all pairs r and b is $O(|R_i|n^2 \log n)$, since there are only $O(|R_i|)$ pairs.
2. Next, we compute the region of each site $r \in R_i$ in $\mathrm{FVD}(r \cup B_l)$; to do this, we intersect the regions of r in $\mathrm{FVD}(\{r, b\})$ over all $b \in B_l$. Since B_l has only a constant number of sites, the intersection can be accomplished in $O(n^2 \log n)$ time for a single red site $r \in R_i$ by iterating the intersection procedure in the generic merge step. The time taken for all the sites in R_i is $O(|R_i|n^2 \log n)$.
3. Next, we compute the region of each site $r \in R_i$ in $\mathrm{FVD}(R_i \cup B_l)$ by intersecting its regions in $\mathrm{FVD}(r \cup B_l)$ and $\mathrm{FVD}(R_i)$. The intersection procedure is similar to the one in the generic merge step, and it can be shown that its running time for all sites in R_i is $O(|R_i|n^2 \log n)$.
4. To complete the computation of $\mathrm{FVD}(R_i \cup B_l)$, it remains to compute the regions of the blue sites. Note that all the vertices of $\mathrm{FVD}(R_i \cup B_l)$ are known at this stage; these are either vertices on the boundary of the red regions, or

they are the vertices of $\mathrm{FVD}(B_l)$ that have 'survived' after the computation of the red sites. The edges of $\mathrm{FVD}(R_i \cup B_l)$ are now traced out just as in the generic merge step; in fact, the situation is much simpler here. The time taken in this step is $O(|R_i|n^2 \log n)$.

Putting everything together, the time complexity of computing $\mathrm{FVD}(R_i \cup B_l)$ is $O(|R_i|n^2 \log n)$. Hence, the time needed for computing all the diagrams in the basic merge step is $O(mn^2 \log n)$.

3.4 Total Running Time and Memory Requirements

The time for merging $\mathrm{FVD}(R)$ and $\mathrm{FVD}(B)$ into $\mathrm{FVD}(R \cup B)$ is dominated by the generic merge step, which requires $O(mn^2 \log m(\log m + \log n))$ time; the total running time satisfies the recurrence

$$T(m) = T(\lfloor m/2 \rfloor) + T(\lceil m/2 \rceil) + O(mn^2 \log m(\log m + \log n)) \qquad (3)$$

which solves to $T(m) = O(mn^2 \log^2 m(\log m + \log n))$.

The memory requirements of the algorithm are linear in the size of all diagrams that are constructed in the process, which is $O(mn^2 \log m)$.

4 Discussion and Conclusions

We have shown that the furthest-site Voronoi diagram of a set S of m sites on the surface of a polyhedron P has complexity $\Theta(mn^2)$, and we have given an algorithm for computing the diagram in $O(mn^2 \log^2 m(\log m + \log n))$ time. 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.

The merge step in our divide-and-conquer approach for the computation of $\mathrm{FVD}(S)$ is quite complicated, and it would be pleasant to find a simpler method. Merging the recursively computed diagrams by sweeping seems natural, but the number of intersections of edges of both diagrams can be superlinear (in m), while only a linear number of them can end up as a vertex of the resulting diagram.

It would be a challenge to find an output-sensitive algorithm, i.e., an algorithm that takes time proportional to the number edges/vertices 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 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 if the facility location problem can be solved without constructing the furthest-site Voronoi diagram. Recall that the facility location problem in the plane can be solved using techniques related to fixed-dimensional linear programming [4,10].

References

1. J. Chen and Y. Han. Shortest paths on a polyhedron. *Internat. J. Comput. Geom. Appl.*, 6:127–144, 1996. 20
2. D. P. Dobkin and D. G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *J. Algorithms*, 6:381–392, 1985. 25
3. D. Leven and Micha 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. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987. 22
4. N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12:759–776, 1983. 20, 28
5. Joseph S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16:647–668, 1987. 20, 22, 26
6. D. M. Mount. Voronoi diagrams on the surface of a polyhedron. Technical Report 1496, Department of Computer Science, University of Maryland, 1985. 20, 22, 27
7. E. Ramos. Intersection of unit-balls and diameter of a point set in R^3. *Computat. Geom. Theory Appl.*, 6:in press, 1996. 25
8. Micha Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, 15:193–215, 1986. 20
9. M.J. van Trigt. Proximity problems on polyhedral terrains. MSc. thesis, Dept. Comput. Sci., Utrecht University, 1995. INF/SCR-95-18. 20
10. Emo Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, volume 555 of *Lecture Notes Comput. Sci.*, pages 359–370. Springer-Verlag, 1991. 20, 28