Facility Location on Terrains*

Boris Aronov† Marc van Kreveld‡ René van Oostrum‡ Kasturi Varadarajan§

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 $O(mn^2 \log^2 m \log n)$ time.

1 Introduction

1.1 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). Polyhedral terrains are commonly used to model (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 over a bounded rectangle and consisting of 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 distance to a site. We assume throughout that $m \leq n$.

To be able to utilize the extensive previous work on shortest paths on polyhedra, we show how to transform the terrain to (the surface of) a polyhedron such that for any two points p and q on the original terrain, any path between p and q that leaves the original terrain cannot be a shortest path. All facets of the resulting polyhedron are triangles, and the total number of them is linear in the number of triangles of the original terrain. The polyhedron is homeomorphic to a ball, so that its surface is homeomorphic to a sphere. The transformation to a polyhedron can be applied to terrains that are defined over a rectangle and to unbounded terrains.

Our algorithm constructs the furthest-site Voronoi diagram of the point sites on the surface of the polyhedron obtained from the terrain. The location for the facility center can then be found by traversing the edges and vertices of the furthest-site Voronoi diagram.

*B.A. has been partially supported by a Sloan Research Fellowship and by NSF Grant CCR–99-72568. 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.

†Department of Computer and Information Science, Polytechnic University, Brooklyn, NY 11201-3840, USA. Email: aronov@ziggy.poly.edu

‡Institute of Information & Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, Netherlands. Email: {marc,rene}@cs.uu.nl

§Department of Computer Science, University of Iowa, IA 52240, USA. Email: kvaradar@cs.uiowa.edu

1.2 Previous work and new 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 [7] gave the first deterministic linear-time algorithm, and a much simpler, randomized, linear–expected-time algorithm was found by Welzl [15].

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 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 [9] showed that the *closest-site* 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 [12], Mitchell, Mount and Papadimitriou [8], and Chen and Han [2]. The best known algorithms [2, 8] 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 between the source s to any query point p in $O(\log n)$ time. The algorithm of Mitchell et al. [8] is a continuous version of Dijkstra's algorithm for finding shortest paths in a graph [4], while Chen and Han [2] solve the problem by determining shortest paths in an *unfolding* of the polyhedron; see also [1].

In his master's thesis, van Trigt [14] 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.

This paper gives an $O(mn^2 \log^2 m \log n)$ 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 Extending a terrain to a polyhedron

In many practical situations, a terrain is defined over a rectangle, i.e., it is the graph of a piecewise-linear function defined over $[x_{\text{left}}, x_{\text{right}}] \times [y_{\text{bottom}}, y_{\text{top}}]$. To avoid complications involving the boundary of the terrain, and to be able to use results on shortest paths and Voronoi diagrams on polyhedra, we extend the terrain to the surface of a polyhedron. A similar transformation can be applied to unbounded terrains.

Any terrain defined over a rectangle $[x_{\text{left}}, x_{\text{right}}] \times [y_{\text{bottom}}, y_{\text{top}}]$ and consisting of n triangles can be extended with $O(n)$ additional triangles to the surface of a polyhedron that is homeomorphic to a sphere, such that for any two points p, q on the original terrain, any path from p to q that leaves the original terrain cannot be a shortest path on the polyhedron. The construction is as follows:

The polyhedron will be shaped somewhat like a box, with the original terrain 'on top' (see Figure 1). Let d be an upper bound on the length of the shortest path between two points on the original terrain. Such an upper bound can easily be found in $O(n)$ time, by summing the lengths of the longest sides of all triangles. First, we extend the domain of the terrain to $[x_{\text{left}} - d, x_{\text{right}} + d] \times [y_{\text{bottom}} - d, y_{\text{top}} + d]$. For each vertex $v = (x_v, y_v, z_v)$ on the original boundary of the terrain, we add a vertex $v' = (x'_v, y'_v, z'_v)$ on the new boundary, with $z'_v = z_v$, and with x'_v and y'_v chosen such that the Euclidean distance between v and v' is minimized. Next, we add an edge (v, v'). Note that each of the new edges is horizontal (normal to the z-axis), and in the projection onto the (x,y)-plane it is perpendicular to the original and new boundary. The new vertices on the new boundary of the terrain are connected with edges along this new boundary. Note

that these edges on the new boundary are not all horizontal, unless the edges on the original boundary are horizontal. Next, the resulting new "rectangles" are triangulated by adding a diagonal edge. So far, we have constructed the top of the polyhedron; an example is shown in Figure 1.

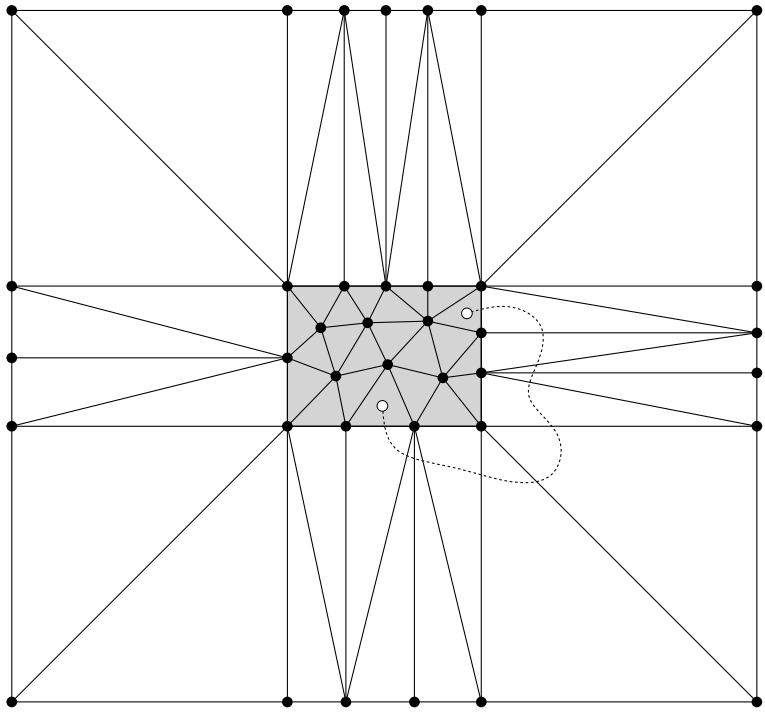

Figure 1: Extending a terrain (shaded) to the top of a 'box-like' polyhedron.

Let z_{low} be the z-value of the lowest vertex in the terrain. The 'bottom' of the polyhedron is the rectangle $([x_{\text{left}} - d, x_{\text{right}} + d] \times [y_{\text{bottom}} - d, y_{\text{top}} + d], z_{\text{low}} - 1)$.

From the vertices on the boundary of the top of the polyhedron, we start edges parallel to the z-axis, and ending at the boundary of the bottom rectangle. The resulting vertical rectangles are triangulated by adding diagonal edges. Finally, we place a vertex in the interior of the bottom rectangle, and connect it with edges to all vertices on the boundary of the bottom rectangle. The resulting polyhedron is highly degenerate, but our algorithm is not influenced by these degeneracies.

Because of the dimensions of the top of the polyhedron, no shortest path from p to q, both on the original terrain, can leave the top of the polyhedron. For any path from p to q that stays on the top of the polyhedron, the maximal sub-paths that lie outside the original terrain can be replaced by shorter paths along the boundary of the original terrain. Therefore, any shortest path on the polyhedron between p and q will lie completely on the original terrain.

For unbounded terrains, a similar transformation can be applied by limiting the domain of the terrain to a rectangle that encloses all m point sites (in the projection onto the (x,y)-plane). The size of this rectangle should be large enough to guarantee that no shortest path between two point sites leaves the rectangle. Next, we convert the resulting terrain to a polyhedron as before.

3 The complexity of the furthest-site Voronoi diagram on a polyhedron

Previous papers on shortest paths on polyhedra [12, 8, 2, 14] use a number of important 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 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 such a path need not to 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. 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 path-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, consists of one-dimensional pieces. When two pseudoroots have the same distance to s, the complement of the regions of the SPM may have two-dimensional parts.

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 planar Euclidean Voronoi diagram for a set of pseudo-sites with additive weights (see Figure 2). The pseudo-sites 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 pseudo-site is the shortest-path 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 on a hyperbolic arc or a segment there are two shortest paths to s with different pseudoroots.

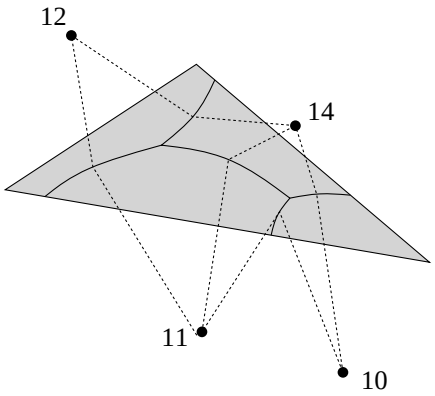

Figure 2: The SPM of a site s, restricted to a triangle, is the Euclidean Voronoi diagram for a set of pseudo-sites with additive weights. The weight of a pseudo-site is the shortest-path distance from the pseudo-site to s.

Given two sites s and t on the polyhedron, the *bisector* $\beta(s,t)$ is the set of those 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. For simplicity, we assume that these degeneracies do not occur.

The *closest-site Voronoi diagram* of a set S of m sites on P, denoted by $\mathrm{VD}(S)$, 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 $\mathrm{d}(p,s) < \mathrm{d}(p,s')$ 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* 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 $\mathrm{d}(p,s) > \mathrm{d}(p,s')$ for each $s' \in S$ with $s' \neq s$. In this paper, we give a new algorithm for computing the furthest-site Voronoi diagram of a set S of sites on a polyhedron. We will use the notation $R(s)$ to denote the region of $s \in S$ in the Voronoi diagram. Whether this is the region of s in $\mathrm{VD}(S)$ or $\mathrm{FVD}(S)$ should be clear from context.

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 the paper of Leven and Sharir [6]; 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 $R(s)$ of a site $s \in S$ is path-connected.*

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

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

Proof: Consider the closest-site Voronoi diagram of $\{s,t\}$. The closures of $R(s)$ and $R(t)$ in this Voronoi diagram cover the whole surface of the polyhedron, and, by the previous lemma, both $R(s)$ and $R(t)$ are path-connected. Since P is homeomorphic to a sphere, $\beta(s,t)$, which is the common boundary of $R(s)$ and $R(t)$, must be connected and homeomorphic to a circle. □

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

Proof:

Consider the closest-site Voronoi diagram of $\{s,t,u\}$. 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 the Voronoi diagram of $\{s,t,u\}$, incident to $R(s)$, $R(t)$, and $R(u)$.

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. It is always possible to pick the paths in such a manner that no two of the paths sharing an endpoint cross, though they may overlap (if the paths crossed, they could be replaced by new paths that share the initial portion from the common endpoint to the point of crossing).

Consider a pair of the paths not sharing an endpoint, say $\pi(s,\chi_1)$ and $\pi(t,\chi_2)$. The former is contained in the closure of $R(s)$, the latter in the closure of $R(t)$, and their interesection lies in $\beta(s,t)$. In particular, the two paths cannot cross.

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 $\{\beta(s,t) : t \neq s\}$ form a set 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 4 *Let B be a set of pseudocircles on the surface of a simple polyhedron P. If the common interior of the pseudocircles in B is non-empty, then their common exterior is path-connected.*

Proof: Suppose for the sake of contradiction that the common interior of the pseudocircles in B is non-empty, and that their common exterior is not path-connected. Let $B' \subseteq B$ be a minimal subset of pseudocircles such that their common exterior consists of at least two path-connected regions R_1, R_2. Observe that both R_1 and R_2 must be incident to all pseudocircles in B'. Otherwise, the removal of a pseudocircle not incident to, say, R_1 would leave R_1 unchanged and can only enlarge R_2, but it cannot join the two regions, which contradicts the minimality of B'. Also observe that all pseudocircles in B' must intersect: a pseudocircle that lies completely in the interior of another one is not incident to R_1 and R_2, and can be removed without affecting the two regions.

Let p_1 (p_2) be a point in the interior of R_1 (R_2, respectively). Since p_1 and p_2 lie in different components of the common exterior of the pseudocircles in B', there exists a closed path in the union of their interiors that separates p_1 and p_2. Indeed, the situation must be as depicted in Figure 3: each pseudocircle can intersect at most two other pseudocircles. Otherwise, the incidence graph of the pseudocircles would contain a chord, so we could drop at least one of the pseudocircles and still find a closed path in the union of the interiors of the remaining ones that separates p_1 and p_2. On the other hand, the incidence graph of the pseudocircles in B' must be a complete graph, since every two of them intersect.

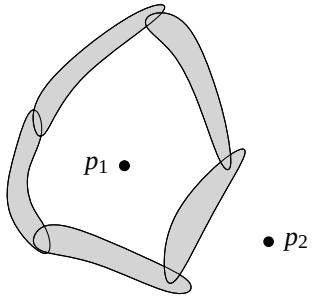

Figure 3: Pseudocircles separating p_1 from p_2.

It follows that the number of pseudocircles in B' is at most three, and inspection of all topologically different arrangements of up to three pseudocircles shows that in none of these arrangements the pseudocircles have a common exterior of two or more path-connected regions if their common interior is non-empty. □

Lemma 5 *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 [9]. □

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 before. We consider the point where a bisector crosses an edge of P also to be a breakpoint.

Lemma 6 *The furthest-site Voronoi diagram* $\mathrm{FVD}(S)$ *of a set S of m sites on a polyhedron has $O(m)$ cells, vertices, and edges.*

Proof: Let $R_{s>t}$ be the region of points that are further away from s than from t, for $s,t \in S$. In this notation $R(s) = \bigcap_{t \in S, t \neq s} R_{s>t}$. By Lemmas 3 and 4, this intersection is the common exterior of a set of pseudo-disks that all contain s and thus is path-connected. So we have at most one cell (region) for each site in 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 $\mathrm{FVD}(S)$ is also $O(m)$. □

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

Lemma 7 *The maximum total complexity of* $\mathrm{FVD}(S)$ *is* $\Theta(mn^2)$.

Proof: Each edge of $\mathrm{FVD}(S)$ 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 5.

As for the lower bound, we describe a construction that shows that $\mathrm{FVD}(S)$ 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 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 4 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 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 C_1. Note that any point on the (shorter) arc of 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 $\mathrm{ray}(s_i, b_{i-1})$ and $\mathrm{ray}(s_i, b_i)$. The portion of the cone γ_i outside C_1 is further away from s_i than from any other site in S_1. Figure 4 only shows the cones γ_2, γ_3 and γ_4.

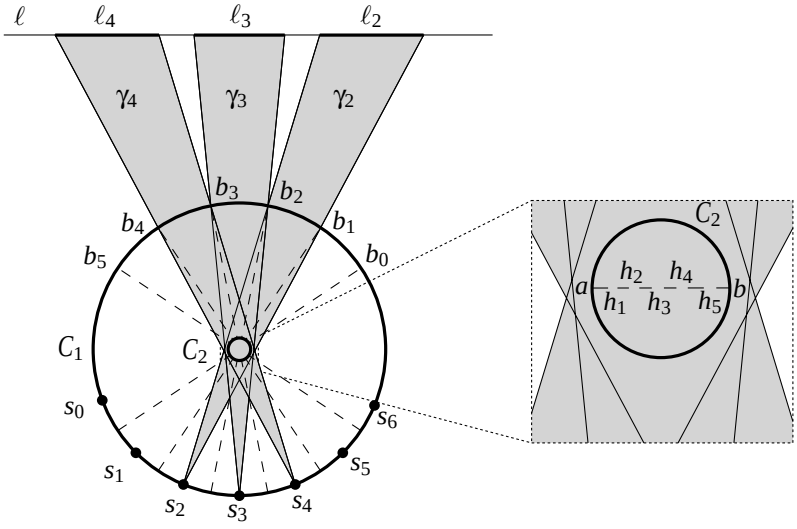

Figure 4: The configuration of S_1 and the obstacles in C_2 (detail).

Let ℓ be a horizontal line lying some distance above the circle 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 C'_1 which is the reflection of 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 C_1. Let C_2 be a small circle lying within this common intersection, and let the segment $\overline{ab}$ be the horizontal diameter of C_2. Figure 4 (detail) shows the circle 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 will argue that the line ℓ is intersected by $k = m/2$ edges of $\mathrm{FVD}(S)$, 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) than from any other site in S_1 (resp. S_2), s_i and s'_i are

the only relevant sites for FVD(S) near ℓ_i. We will now argue that $\beta(s_i, s_i')$ 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 in 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 5 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 5.

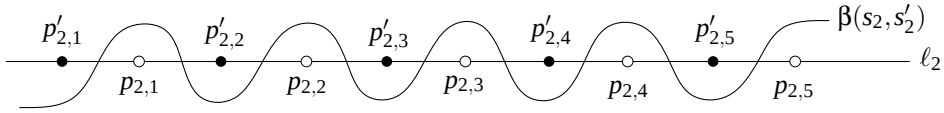

Figure 5: Detail of $\beta(s_2, s_2')$.

One gets $\Omega(kn) = \Omega(mn)$ crossings for line ℓ, $\Omega(n)$ 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 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 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 elementary arc or line segment of each edge the point that maximizes the distance to the two sites of the regions on both sides of the edge. These distances can be computed in $O(1)$ time, if FVD(S) is appropriately labeled, and the maximum of all these distances determines the location of the facility center. Since 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.*

4 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 m sites on the surface of polyhedron P, consisting of n triangles. Our algorithm uses ideas from the algorithm of Ramos [11] 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:

- If $|S| = 1$, then FVD(S) has no vertices and edges, and only a single cell that is the whole surface of the polyhedron. If $|S| = 2$, compute the closest-site Voronoi diagram (which is equivalent to the furthest-site Voronoi diagram) with the algorithm of Mount [9] in $O(n^2 \log n)$ time. The diagram has no vertices, one edge (the bisector of the two sites, which is a closed curve, homeomorphic to a circle), and two regions. The bisector of the two sites consists of $O(n^2)$ line segments and hyperbolic arcs. For each of these segments and arcs we maintain the two pseudoroots that are closest to the two sites in S respectively, and the distances of these pseudoroots to the two sites. We need this information to determine the facility center after FVD(S) has been computed.

- Otherwise, if $|S| \geq 3$, subdivide S into two subsets R (the *red* sites) and B (the *blue* sites) of about equal size, i.e., $|R| = \lfloor |S|/2 \rfloor$, and $|B| = \lceil |S|/2 \rceil$.

- 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 $R_0 \subseteq R$ that have a non-empty region in $\mathrm{FVD}(R)$, i.e., such that $\mathrm{FVD}(R) = \mathrm{FVD}(R_0)$. Observe that the remaining sites in $R \setminus R_0$ do not influence the final diagram and can be discarded. Similarly, compute $B_0 \subseteq B$.

– Determine a *low-degree independent set* $R'_0 \subset R_0$, which is a subset with the property that the region of a site $s \in R'_0$ has at most 9 neighbors in $\mathrm{FVD}(R_0)$, and no two sites $s, s' \in R'_0$ are neighbors in $\mathrm{FVD}(R_0)$. (Two sites are said to be *neighbors* if their regions share an edge of the diagram.) Compute $R_1 = R_0 \setminus R'_0$ and $\mathrm{FVD}(R_1)$, and repeat this step to generate a Dobkin-Kirkpatrick hierarchy [5] $R_0 \supset R_1 \supset \ldots \supset R_k$ 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 obtain $B_0 \supset B_1 \supset \ldots \supset B_l$ and their furthest-site Voronoi diagrams. See Section 4.2 for details.

– Compute $\mathrm{FVD}(R_i \cup B_l)$ for $0 \leq i \leq k$, exploiting the fact that B_l has only a constant number of sites. Similarly, compute $\mathrm{FVD}(R_k \cup B_j)$ for $0 \leq j \leq l$. This is the *basic merge step*. See Section 4.3 for details.

– Compute $\mathrm{FVD}(R_i \cup B_j)$ from $\mathrm{FVD}(R_i \cup B_{j+1})$ and $\mathrm{FVD}(R_{i+1} \cup B_j)$. This is the *generic merge step*, which when repeated gives $\mathrm{FVD}(R_0 \cup B_0) = \mathrm{FVD}(S)$. See Section 4.4 for details.

During the construction of $\mathrm{FVD}(S)$, we create closest- and furthest-site Voronoi diagrams of subsets of S as intermediate structures. We maintain $\mathrm{FVD}(S)$ and these intermediate structures as doubly connected edge lists [3, 10], to be able to efficiently determine and preserve topological relations between Voronoi regions, edges, and vertices.

4.1 Edge tracing

Several stages of the algorithm for constructing $\mathrm{FVD}(S)$ involve the computation of new Voronoi cells of $\mathrm{FVD}(S')$ for $S' \subseteq S$, or the modification of existing Voronoi cells. A basic step is the generation of Voronoi edges or parts of Voronoi edges. Recall that the edges of $\mathrm{FVD}(S')$ lie on the bisectors of sites in S', and consist of $O(n^2)$ hyperbolic arcs and/or straight line segments each. To generate an edge e that is incident to the regions of $s_i, s_j \in S'$, we need to know a starting point of the edge (i.e., the location of one of the vertices of $\mathrm{FVD}(S')$ incident to e), and an endpoint (i.e., the location of the other vertex incident to e). We calculate the bisector $\beta(s_i, s_j)$ in $O(n^2 \log n)$ time using the algorithm of Mitchell et al. [8]. We store it as a doubly linked list of hyperbolic arcs and straight line segments. Next, we traverse $\beta(s_i, s_j)$, until we reach the starting point of e. From that point on, we output the hyperbolic arcs and straight line segments of which e consists, until we reach the endpoint of e. Traversing $\beta(s_i, s_j)$ takes $O(n^2)$ time, and testing whether we have reached the starting point or the endpoint of e can be done in $O(1)$ time for each elementary hyperbolic arc or straight line segment. Hence, the total time needed to generate an edge of $\mathrm{FVD}(S')$ is $O(n^2 \log n)$. The amount of memory needed is bounded by the size of the shortest path maps of s_i and s_j, and of $\beta(s_i, s_j)$, which is $O(n^2)$. These results are summarized in the following lemma:

Lemma 8 *Given a set of sites S on a polyhedron P with n triangles and the two vertices incident to an edge e of $\mathrm{FVD}(S')$ for $S' \subseteq S$, e itself can be computed in $O(n^2 \log n)$ time using $O(n^2)$ memory.*

Suppose that we have generated all the edges of the region of $s_i \in S'$, including e, the common edge of the regions of s_i and s_j. Later on in the algorithm, we may have to generate the edges of the region of s_j, including e. Rather than computing the bisector of s_i and s_j again, bisector computations are cached, so any bisector is computed no more than once, and no edge is generated more than once.

In some cases we need a variation on the edge tracing procedure. As before, we compute a bisector of two sites (or retrieve it from the cache, if it has been computed before), traverse it until we find the starting point of the edge that is to be generated, and output hyperbolic arcs and straight line segments from that moment on. The difference is that we don't have a single endpoint at which we stop the tracing, but a constant number of candidate endpoints, and we stop when we have reached the first of these. Testing whether we have reached any of the candidate endpoints takes $O(1)$ time per hyperbolic arc or straight line segment, and this edge tracing variation also requires $O(n^2 \log n)$ time and $O(n^2)$ memory per Voronoi edge.

4.2 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 construction [5].

Let G_0 be the dual graph of $\mathrm{FVD}(R_0)$, i.e., $G_0 = (R_0, E_0)$, with $(s_i, s_j) \in E_0$ for $s_i, s_j \in R_0$ if the regions of s_i and s_j share an edge in $\mathrm{FVD}(R_0)$. Note that $\mathrm{FVD}(R_0)$ and G_0 are planar graphs. An *independent set* of vertices in a graph $G = (V,E)$ is a set $V' \subset V$ such that there is no edge (v_i, v_j) in E for any $v_i, v_j \in V'$. Snoeyink and van Kreveld [13] showed that for any planar graph $G = (V,E)$ with n vertices, an independent set V' of vertices of degree at most 9 with $|V'| \geq |V|/6$ can be found in $O(n)$ time. We apply this to G_0 to find an independent set $R'_0 \subset R_0$ in $O(|R_0|)$ time. A site $s \in R'_0$ has at most 9 neighbors in $\mathrm{FVD}(R_0)$, no two sites $s, s' \in R'_0$ are neighbors in $\mathrm{FVD}(R_0)$, and $|R'_0| \geq |R_0|/6$.

To compute $\mathrm{FVD}(R_1) = \mathrm{FVD}(R_0 \setminus R'_0)$, we remove the sites in R'_0 one at a time from R_0 and update the diagram after the removal of each site. Let s_0 be such a site in R'_0, and let p be a point that lies in the region in $\mathrm{FVD}(R_0)$ of s_0. After updating the diagram, p must lie in the region of a site s' that is a neighbor of s_0 in $\mathrm{FVD}(R_0)$, since furthest-site Voronoi regions are connected. So the region of s_0 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 (see Figure 6).

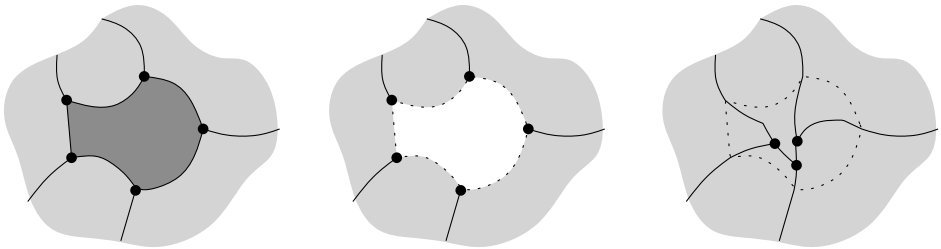

Figure 6: Removing a site s_0, and dividing its region among its neighbors.

Let $v_1, \ldots, v_k$ be the at most 9 vertices of the region of s_0 in clockwise order. Let $e_1, \ldots, e_k$ be the edges of $R(s_0)$, with e_i incident to v_i and v_{i+1} for $1 \leq i < k$, and with e_k incident to v_k and v_1. Finally, let s_i be the neighbor of s_0 whose region is incident to e_i for $1 \leq i \leq k$. See Figure 7.

We will describe how to reconstruct $R(s_1)$ after the removal of s_0; for $R(s_2), \ldots, R(s_k)$ this is similar. Edge e_1 is no longer an edge of $R(s_1)$, and it is removed. Vertices v_1 and v_2 are also removed; the edges of $R(s_1)$ incident to these vertices (e'_1 and e'_2) have to be extended into the region of s_0. Recall that e'_1 lies on the bisector $\beta(s_1, s_k)$ of s_1 and s_k, and e'_2 lies on $\beta(s_1, s_2)$. We extend e'_1 by tracing $\beta(s_1, s_k)$ as described in Section 4.1, starting at the location of v_1. At some point p on the bisector, the distance between p and s_1 and the distance between p and s_k equals the distance between p and some other neighbor s_i of s_0. At this point p we reach a new vertex of $R(s_1)$. Note that this vertex is also a vertex of the closest-site Voronoi diagram of s_1, s_k and s_i. We have finished the reconstruction of e'_1 at this point, record the new vertex of $R(s_1)$, and proceed with the next edge of $R(s_1)$ by tracing $\beta(s_1, s_i)$. This is repeated until we finally trace $\beta(s_1, s_2)$ and end up at the location of v_2, which concludes the reconstruction of $R(s_1)$.

To determine whether we have reached a vertex during the edge tracing, we precompute all the points p that are equidistant to three neighbors of s_0. For a fixed triple of neighbors of s_0, these points p are the $O(1)$ vertices of the closest-site Voronoi diagram of the triple, which can be computed in $O(n^2 \log n)$ time again with the techniques from Mitchell et al. [8] and Mount [9]. Computing the Voronoi diagrams for all triples of the at most 9 neighbors of s_0 takes asymptotically the same amount of time. We trace a constant number of edges, and for each hyperbolic arc or straight-line segment on these edges we determine in constant time (by testing the $O(1)$ precomputed vertices of the Voronoi diagrams of triples of neighbors of s_0) whether we have reached the endpoint of the edge that we are generating. It follows that the removal of a single site and the reconstruction involved takes $O(n^2 \log n)$ time.

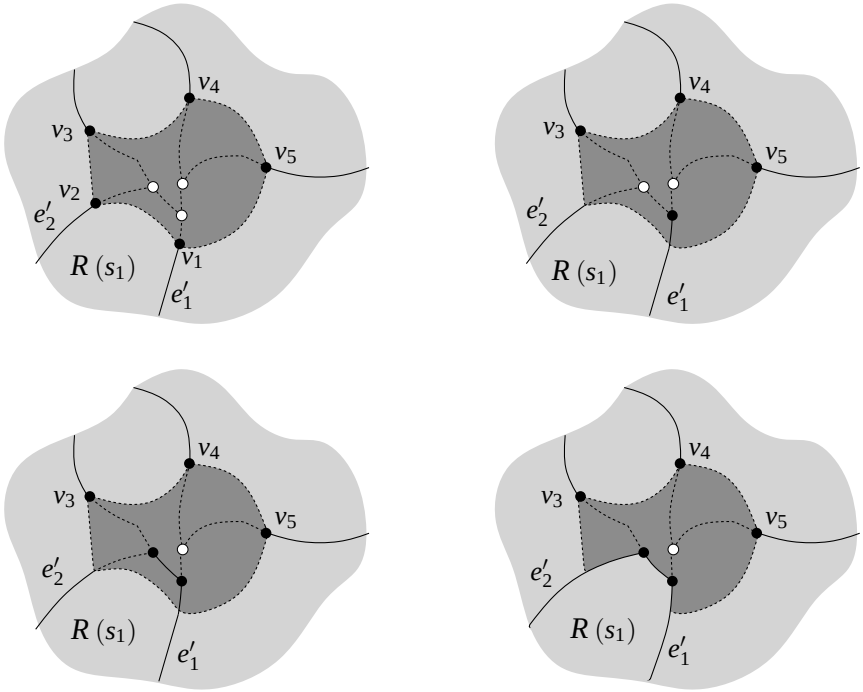

Figure 7: The reconstruction of $R(s_1)$ after the removal of s_0.

After all the sites in R'_0 have been removed from R_0 and $\mathrm{FVD}(R_1)$ has been constructed, we recursively repeat the procedure of removing an independent set of sites to create $\mathrm{FVD}(R_2),\ldots,\mathrm{FVD}(R_k)$. The total number of diagrams we construct this way is $O(\log m)$.

Since $\sum_{i=0}^{k}|R_i|$ is a geometric series, the total time for computing all independent sets is $O(m)$. The the reconstruction of the diagram after the removal of a single site 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 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. By Lemma 7, the size of $\mathrm{FVD}(R_i) = O(|R_i|n^2)$. Therefore, the total size of all bisectors and diagrams constructed is $O(mn^2)$.

4.3 The basic merge step

In the basic merge step, we compute $\mathrm{FVD}(R_i \cup B_l)$ for $0 \le i \le k$, and $\mathrm{FVD}(R_k \cup B_j)$ for $0 \le j \le l$. 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.

- 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 Mitchell et al. [8, 9]. 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.

- Next, we compute the region of each site $r \in R_i$ in $\mathrm{FVD}(\{r\} \cup B_l)$ by successively intersecting the regions of r in $\mathrm{FVD}(\{r,b\})$ over all $b \in B_l$ with a line-sweep. We do this separately for each triangle of P met by an edge of either of the two regions to be intersected. The total number of intersection computations for a single site $r \in R_i$ is $|B_l| - 1$, which is bounded by a constant. Since B_l has only a constant number of sites, r has a constant number of neighbors in $\mathrm{FVD}(\{r\} \cup B')$ for any $B' \subseteq B_l$, and the complexity of the region of r in any of these diagrams is $O(n^2)$. This means that the intersections can be computed in $O(n^2\log n)$ time for a single red site $r \in R_i$. The time taken for all the sites in R_i is $O(|R_i|n^2\log n)$.

- 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 $\mathrm{FVD}(R_i)$ with a line-sweep as above. Since the complexity of the region of r in $\mathrm{FVD}(R_i \cup B_l)$ is $O(n^2)$ and the complexity of the region of r in $\mathrm{FVD}(R_i)$ is $O(N_r \cdot n^2)$, where N_r is the number of neighbors of r in $\mathrm{FVD}(R_i)$, the time needed to compute the region of a single site r in $\mathrm{FVD}(R_i \cup B_l)$ is $O(n^2 \log n \cdot N_r \log N_r)$. Summing this over all $r \in R_i$ gives

$$O(n^2 \log n \sum_{r \in R_i} N_r \log N_r)\,.$$

  We have that $N_r \leq m$ for all $r \in R_i$, and $\sum_{r \in R_i} N_r = |R_i|$. The time needed to compute the region of each site $r \in R_i$ in $\mathrm{FVD}(R_i \cup B_l)$ is therefore bounded by $O(|R_i| n^2 \log m \log n)$.

- To complete the computation of $\mathrm{FVD}(R_i \cup B_l)$, it remains to compute the regions of the blue sites. We have computed the regions of all red sites, and carefully maintained topological relations between each region of a red site and its neighbors. Using the topological information, we do a depth-first traversal of the regions in $\mathrm{FVD}(R_i \cup B_l)$ (that is partially constructed), starting in the region of a red site. When a region of a blue site is visited for the first time, it has to be constructed. At least one of its edges has already been computed, namely, the edge incident to its predecessor in the traversal. Zero or more of the other edges of the region are incident to red regions; these have also been computed already. We still have to trace these edges, namely to find the starting points for the edges that have not been computed yet. These are *blue edges*, the edges between two blue regions. All blue edges are sub-edges of the edges of $\mathrm{FVD}(B_l)$, of which there are only a constant number. The endpoints of these edges are either vertices of $\mathrm{FVD}(B_l)$, or they are vertices of a red region. The latter vertices can be determined in $O(|R_i| n^2)$ time by traversing the edges of all regions of red sites, and reporting the vertices that are incident to blue edges. The total number of endpoints of blue edges is bounded by $O(|B_j|)$, which is a constant, so we can exploit the techniques described in Section 4.1 to trace the blue edges in $O(n^2 \log n)$ time per edge. The total time for computing the regions of the blue sites is $O(|R_i| n^2 \log n)$.

Putting everything together, computing $\mathrm{FVD}(R_i \cup B_l)$ takes $O(|R_i| n^2 \log m \log n)$ time, and computing $\mathrm{FVD}(R_k \cup B_j)$ takes $O(|B_j| n^2 \log m \log n)$ time. Since both $\sum_{i=0}^{k} |R_i|$ and $\sum_{i=0}^{l} |B_j|$ are bounded by $O(m)$, the time needed for computing all the diagrams in the basic merge step is $O(mn^2 \log m \log n)$. The amount of memory needed in the basic merge step is linear in the complexity of all bisectors and diagrams that we computed, which is $O(mn^2)$.

4.4 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 eventually 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 is at most $n^2(|R_i| + |B_j|)$. Hence, computing the vertices of $\mathrm{FVD}(R_i \cup B_j)$ takes $O(n^2(|R_i| + |B_j|) \log(n^2(|R_i| + |B_j|))) = O(n^2(|R_i| + |B_j|) \log n)$ (recall that $m < n$).

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 traced), 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 $O(n^2 \log n(|R_i| + |B_j|))$ time.

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

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

We have

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

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

4.5 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 $O(mn^2 \log m \log n)$ time; the total running time satisfies the recurrence

$$\begin{aligned} T(1) &= O(1) \\ T(2) &= O(n^2 \log n) \\ T(m) &= T(\lfloor m/2 \rfloor) + T(\lceil m/2 \rceil) + O(mn^2 \log m \log n) \end{aligned}$$

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

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

Theorem 1 *The complexity of the furthest-site Voronoi diagram of m sites on the surface of a polyhedron with n triangles has complexity $\Theta(mn^2)$. The diagram can be computed in $O(mn^2 \log^2 m \log n)$ time, using $O(mn^2 \log m)$ memory.*

5 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 polyhedron P with n triangles has maximum complexity $\Theta(mn^2)$, and we have given an algorithm for computing the diagram in $O(mn^2 \log^2 m \log n)$ 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 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 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 [7, 15].

References

[1] B. Aronov and J. O'Rourke. Nonoverlap of the star unfolding. *Discrete Comput. Geom.*, 8:219–250, 1992.

[2] J. Chen and Y. Han. Shortest paths on a polyhedron. *Internat. J. Comput. Geom. Appl.*, 6:127–144, 1996.

[3] M. de Berg, M. van Kreveld, M. Overmars, and O .Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[5] D. P. Dobkin and D. G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *J. Algorithms*, 6:381–392, 1985.

[6] 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. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[7] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12:759–776, 1983.

[8] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16:647–668, 1987.

[9] D. M. Mount. Voronoi diagrams on the surface of a polyhedron. Technical Report 1496, Department of Computer Science, University of Maryland, 1985.

[10] D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, 7:217–236, 1978.

[11] E. Ramos. Intersection of unit-balls and diameter of a point set in R^3. *Computat. Geom. Theory Appl.*, 8:57–65, 1997.

[12] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, 15:193–215, 1986.

[13] J. Snoeyink and M. van Kreveld. Linear-time reconstruction of Delaunay triangulations with applications. In *Algorithms – ESA'97, Proc. 5th Ann. European Symp.*, volume 1284 of *Lecture Notes Comput. Sci.*, pages 459–471. Springer-Verlag, 1997.

[14] M. J. van Trigt. Proximity problems on polyhedral terrains. Master's thesis, Dept. of Computer Science, Utrecht University, 1995. INF/SCR-95-18.

[15] 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 Comput. Sci.*, pages 359–370. Springer-Verlag, 1991.