Facility Location Constrained to a Polygonal Domain

Prosenjit Bose[1] and Qingda Wang[2]

[1] School of Computer Science, Carleton University, Ottawa, Ontario, Canada. E-mail: jit@scs.carleton.ca. Research supported by NSERC and MITACS.
[2] Platform Computing, Toronto, Ontario, Canada. E-mail: qiwang@platform.com

Abstract. We develop efficient algorithms for locating an obnoxious facility in a simple polygonal region and for locating a desirable facility in a simple polygonal region. Many realistic facility location problems require the facilities to be constrained to lie in a simple polygonal region. Given a set S of m demand points and a simple polygon R of n vertices, we first show how to compute the location of an obnoxious facility constrained to lie in R, in $O((m+n)\log m + m\log n)$ time. We then show how to compute the location of a desirable facility constrained to lie in R, also in $O((m+n)\log m + m\log n)$ time. Both running times are an improvement over the known algorithms in the literature. Finally, our results generalize to the setting where the facility is constrained to lie within a set of simple polygons as opposed to a single polygon at a slight increase in complexity.

1 Introduction

Facility location is a classical problem of operations research that has also been examined in the computational geometry community. In classical facility location problems, we try to locate *desirable* facilities, such as hospitals, supermarkets, post-offices and warehouses, such that the distance function between the facilities and the demand sites (e.g., customers) is minimized. In *obnoxious* facility location problems, we try to locate obnoxious facilities, such as nuclear power plants, garbage dump sites, mega-airports, and chemical plants, such that the distance function between the facilities and the demand points is maximized. Notice that for the obnoxious facility location problem to be meaningful and practical, some constraints on the location of the facility should be specified, e.g., forcing it to lie in some bounded region.

Most facility location problems dealt with in the literature assumed convex feasible regions (including rectangular regions). However, this assumption is usually not very realistic since the feasible regions are often not convex due to geographical reasons and zoning laws. Therefore in this paper we study facility location problems under the assumption of simple polygonal feasible region. We have included all of the details that are deemed essential, however, due to space constraints, some details may be omitted in this extended abstract. Full details can be found in [Wan01].

In Section 2 and 3 we present improved algorithms for locating an obnoxious facility and a desirable facility in a simple polygonal region, respectively, under Euclidean distance. Both our algorithms run in subquadratic time. Section 4 concludes the paper.

2 Locating an Obnoxious Facility in a Simple Polygon

2.1 Related Work

The problem of locating a single obnoxious facility for a set of demand points is also known as the largest empty circle problem. Toussaint[Tou83] was the first to have studied the obnoxious facility location problem constrained to a simple polygonal region. Given n demand points, he proposed an $O(n \log n)$ algorithm if the facility is constrained to lie in a convex n-gon, and an $O(n \log n + k \log n)$ algorithm if the facility is constrained to lie in a simple n-gon P, where k denotes the number of intersections occurring between edges of P and edges of the Voronoi diagram of the demand points. Note k could be $\Omega(n^2)$.

In this section we improve Toussaint's algorithm. Our algorithm runs in $O(n \log n)$ time even if the constraining polygon is not convex, eliminating the dependence of the running time on k.

2.2 Preliminaries

The problem is stated as the following:

Problem 1 *Given a set $S = s_1, s_2, ..., s_m$ of m demand points and an n vertex simple polygon R, locate a single obnoxious facility within R such that the smallest Euclidean distance between each demand point and the facility is maximized.*

It is assumed that the demand points are in general position, i.e. no three points are collinear and no four points are co-circular.

Toussaint [Tou83] proved the following lemma:

Lemma 1. *The location of the single obnoxious facility under the constraints stipulated in Problem 1 lies on one of the following points that yields the largest empty circle:*

1. *A vertex of the Voronoi diagram of S (denoted as $VD(S)$) contained in R.*
2. *A vertex of the polygon R.*
3. *A proper intersection point of $VD(S)$ and the boundary of R.*

The quadratic running time in Toussaint's algorithm occurs in computing the third part of the above. We refine the characterization of the location of the optimal solution allowing us to circumvent this problem.

For convenience, i.e. to avoid problems caused by unbounded edges of $VD(S)$, let B be a large rectangle that contains S, R and all vertices of $VD(S)$. Denote by $BVD(S)$ the graph induced by $B \cap VD(S)$. Given $VD(S)$, $BVD(S)$ can be computed in linear time.

Lemma 2. *If the location of the single obnoxious facility under the constraints stipulated in problem 1 lies on a proper intersection point c between an edge e of $BVD(S)$ and the boundary of R, then among all intersection points between e and the boundary of R, c is the closest to one of the two endpoints of e.*

Proof. Let p and q be the two endpoints of e. For the sake of contradiction, suppose both $[cp]$ and $[cq]$ contains at least one intersection point between $BVD(S)$ and the boundary of R, denoted by x and y, respectively. Let $V(b)$ be one of the two Voronoi cells incident on e. We have either $\angle bcp$ or $\angle bcq$ is $\geq \pi/2$. Without loss of generality suppose $\angle bcp \geq \pi/2$. Now the circle centered at x with radius $d(x, b)$ is empty. However, $d(x, b) > d(c, b)$. Hence, we have a contradiction.

■

Lemma 2 indicates that we need not consider all the proper intersection points between $VD(S)$ and R. Rather, for all intersection points occurring between a given edge $e = [pq]$ of $BVD(S)$ and the boundary of R, we need consider at most two points, i.e. the points closest to p and q. The intersection points with this property can be found efficiently by ray shooting queries over R [HS95] as follows. For each vertex v of $BVD(S)$ and each edge e incident on v, we shoot a ray emanating from v and containing e. Let p be the hit point of the boundary of R by the ray. If p is on e, it is the intersection point of e and R that is the closest to v, a candidate with the property mentioned in Lemma 2.

We can preprocess R in $O(n)$ time such that a ray shooting query over R can be answered in $O(\log n)$ time [HS95]. Using this technique allows us to avoid the expensive computation of all intersection points between $VD(S)$ and R. To summarize, the following lemma characterizes the desired location of the obnoxious facility.

Lemma 3. *The location of the single obnoxious facility under the constraints stipulated in Problem 1 lies on one of the following points that yields the largest empty circle:*

1. *A vertex of $VD(S)$ contained in R;*
2. *A vertex of the polygon R;*
3. *A point of the boundary of R hit by a ray emanating from a vertex v of $BVD(S)$ and containing an edge of $BVD(S)$ incident on v.*

2.3 The Algorithm

Based on Lemma 3, we outline the following algorithm to compute the location of the obnoxious facility constrained to a simply polygon.

Algorithm 1 *Single obnoxious facility location (S, R)*

Input: A set of demand points $S = s_1, s_2, ..., s_m$, and a simple polygon R of n vertices.

Output: Point(s) for the location of a single obnoxious facility within R that maximizes the minimum distance between the facility and the demand points in S.

1. Compute $VD(S)$ and $BVD(S)$. Preprocess R for point inclusion test using the algorithms of Kirkpatrick [Kir83] or Edelsbrunner, Guibas and Stolfi [EGS86]. Compute the set of vertices of $VD(S)$ contained in R using the point location structure of R, and compute their largest empty circles.
2. Preprocess $VD(S)$ for point inclusion test using the algorithms of [EGS86] or [Kir83]. For each vertex v of R, determine which Voronoi cell it is in using the point location structure of $VD(S)$, and compute its largest empty circle.
3. Preprocess R in $O(n)$ time for ray shooting query over R [HS95]. For each vertex v of $BVD(S)$ and each edge e of $BVD(S)$ incident on v, perform a ray shooting query over R, with a ray emanating from v and containing e [HS95]. If the hit point of the boundary of R by the ray is on e, compute its largest empty circle.
4. Select and output the largest empty circles found in steps 1, 2 and 3.

Theorem 1. *Given a set of demand points $S = s_1, s_2, ..., s_m$, and an n-vertex simple polygon R, we can compute the location of a single obnoxious facility constrained to lie in R, under Euclidean distance, in time $O((m+n)\log m + m\log n)$.*

Proof. The correctness of the algorithm follows from Lemma 3.

Let us analyze the complexity of the algorithm:

1. In step 1, $VD(S)$ can be computed in $O(m \log m)$ time [PS85] and $BVD(S)$ can then be computed in $O(m)$ time. Preprocess of R for point inclusion test can be done in $O(n)$ time ([Kir83] or [EGS86]). There are $O(m)$ vertices in $VD(S)$, each of which costs $O(\log n)$ time to decide if it is contained in R. Thus step 1 can be done in total $O(m\log m + m \log n)$ time.
2. In step 2, preprocessing of $VD(S)$ for point inclusion test can be done in $O(m)$ time ([Kir83] or [EGS86]). Each of the n vertices of R costs $O(\log m)$ time to locate the Voronoi cell it is in. Therefore step 2 can be done in total $O(m + n\log m)$ time.
3. In step 3, preprocessing of R for ray shooting query can be done in $O(n)$ time and each ray shooting query takes $O(\log n)$ time [HS95]. Since the number of edges in $BVD(S)$ is $O(m)$, and we perform at most two ray shooting queries for each edge, the total time spent in step 3 is $O(n + m\log n)$.
4. Step 4 can be done in $O(m + n)$.

Therefore, the total complexity of the algorithm is $O((m+n)\log m + m\log n)$.

■

By reducing to the MAX-GAP problem, the largest empty circle problem has a known lower bound of $\Omega(m \log m)$ [PS85]. Thus Algorithm 1 runs in optimal $\Theta(m\log m)$ time when m is $\Omega(n)$, and $\Theta(n)$ time when m is $O(1)$. It remains an open problem whether the algorithm is optimal when m is $o(n)$ and $\omega(1)$.

Algorithm 1 can be extended to the case where the constraining region R is a set of r disjoint simple polygons $\{R_1, R_2, \ldots, R_r\}$ with a total of n vertices.

Hershberger and Suri [HS95] show how to compute a data structure in $O(n\sqrt{r} + n \log n + r^{3/2} \log r)$ time and $O(n)$ space to answer a ray shooting query in $O(\sqrt{r} \log n)$ time. This leads to the following corollary:

Corollary 1. *Given a set of demand points $S = s_1, s_2, ..., s_m$, and a region R consisting of r disjoint simple polygons with a total of n vertices, we can compute the location of a single obnoxious facility constrained to lie in R, under Euclidean distance, in time $O((m + n) \log m + (m\sqrt{r} + n) \log n + n\sqrt{r} + r^{3/2} \log r)$.*

3 Euclidean Center Constrained to a Simple Polygon

3.1 Related Work

The problem of locating a single desirable facility for a set of demand points is also known as the Euclidean center problem. The problem of computing the unconstrained Euclidean center has a rich history, which is summarized in [PS85].

Bose and Toussaint[BT00] solved the problem of computing the Euclidean center constrained to a simple polygon. Their algorithm finds the Euclidean center of a set of m points constrained to lie in a simple polygon P of n vertices in $O((m+n) \log(m+n))$ time when P is convex, and $O((m+n) \log(m+n)+k)$ time when P is not convex, where k denotes the number of intersections occurring between edges of P and edges of the farthest point Voronoi diagram of the demand points. Notice k could be $\Omega(mn)$.

Extending the technique of Megiddo [Meg83], Hurtado, Sacristan and Toussaint [HST00] obtained an $O(n + m)$ time algorithm for the problem when the Euclidean center is constrained to lie in a convex polygon, reducing by a logarithmic factor the running time of the algorithm presented in [BT00].

It can be shown that the Euclidean center problem constrained to a simple polygon is equivalent to the problem in the presence of polygonal obstacles. D. Halperin, M. Sharir and K. Goldberg studied the Euclidean 2-center problem in the presence of polygonal obstacles and obtained efficient algorithms [HSG00]. In the paper they mentioned that D. Halperin and C. Linhart have solved the 1-center problem with obstacles in $O((m+n) \log(mn))$ time [HL99]. However, their algorithm with the above mentioned time bound solved the decision problem only, and an approximation approach based on binary search is used. Thus the algorithm is not exact.

In this section we present an efficient algorithm that is based on Bose and Toussaint's approach [BT00] to solve the Euclidean center problem constrained to a simple polygon. Our algorithm runs in $O((m + n) \log m + m \log n)$ time even if the constraining polygon is not convex, eliminating the dependency of the running time on k. Although the algorithms in [HL99] has the advantage that it can be generalized to solve Euclidean 2-center problem, the merits of our algorithm compared to [HL99] are the following: (i) it is exact; (ii) it is more efficient, considering the approximation scheme in [HL99] adds a multiplicative logarithmic factor to the running time. (iii) it is simpler to implement, since it involves well-known structures only.

3.2 Preliminaries

The problem is stated as the following:
Problem 2 *Given a set $S = s_1, s_2, ..., s_m$ of m demand points and an n vertex simple polygon region R, locate a single desirable facility within R such that the largest Euclidean distance between each demand point and the facility is minimized.*

It is assumed that the demand points are in general position, i.e. no three points are collinear and no four points are co-circular.

Bose and Toussaint's algorithm [BT00] makes use of the farthest point Voronoi diagram of the set S, denoted as $FPVD(S)$. Given a point $x \in E^2$, we let $\phi(x)$ denote the farthest neighbors of x in S, that is, the set of points in S such that $d(x, \phi(x)) = max_{y \in S} d(x, y)$ where d is the Euclidean distance. The $FPVD(S)$ partitions the plane into unbounded convex cells, $V(s_i)$, such that for any point $p \in V(s_i), s_i \in \phi(p)$.

The following lemma is adopted from [BT00] characterizing the location of the Euclidean center:

Lemma 4. *The Euclidean center under the constraints stipulated in Problem 2 is either the midpoint of the diameter of S provided that the diametrical circle contains the set S and the midpoint is contained in R, or lies on one of the following points that yields the smallest enclosing circle:*

1. *A vertex of $FPVD(S)$ contained in R;*
2. *A vertex of the polygon R;*
3. *A proper intersection point of $FPVD(S)$ and the boundary of R;*
4. *A point x on an edge e of R with the property that $\forall y \in e$, if $\phi(y) = \phi(x)$ then $d(y, \phi(x)) \geq d(y, \phi(x))$.*

The quadratic runtime occurs in computing part 3 and part 4 of the above. In the following we are going to prove Lemmas 5 to 8, which will allow us to improve Bose and Toussaint's algorithm, reducing the quadratic runtime, by improving the characterization of the Euclidean center.

Improving Part 3 of Lemma 4 A similar approach as in Section 2.2 is used to improve part 3 of Lemma 4.

Lemma 5. *If the Euclidean center c under the constraints stipulated in Problem 2 lies on a proper intersection point c between an edge e of $FPVD(S)$ and the boundary of R, and if c is not the midpoint of $DIAM(S)$, then among all intersection points between e and the boundary of R, c is closest to an endpoint of e.*

Proof. See Figure 1. Let $V(a)$ and $V(b)$ be the two cells incident on e. The edge e must lie on the bisector of line segment $[ab]$, for the points on e are equidistant from a and b. The points a, b, and c must form a triangle because otherwise $[ab]$

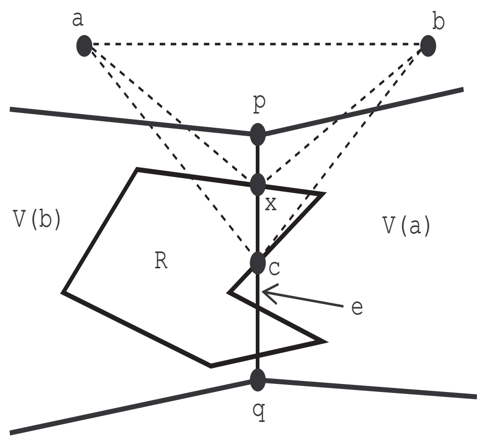

Fig. 1. Illustration of proof for Lemma 5

would be the diameter. Thus neither $[ca]$ nor $[cb]$ is perpendicular to e. If e is unbounded, let p be the unique endpoint of e. Otherwise, let p be the endpoint of e that is closer to a (and b). Then we have $\angle acp < \pi/2$. For the sake of contradiction, suppose c is not the closest to p among all intersection points between e and the boundary of R. Let x be the closest to p. Now the circle centered at x with radius $d(x, a)$ encloses the set S. However, $d(x, a) < d(c, a)$. Hence, we have a contradiction.

■

Thus, for all intersection points occurring between a given edge e of $FPVD(S)$ and the boundary of R, we need only consider the points closest to an endpoint of e, of which there are at most two.

Similar as in Section 2.2, the intersection points with this property can be found efficiently by a ray shooting query over R. Next we develop another technique to efficiently compute part 4 of Lemma 4.

Improving Part 4 of Lemma 4

Lemma 6. *If the Euclidean center x under the constraints stipulated in Problem 2 lies on an edge e of R and has the following properties:*

1. $\forall y \in e$, *if* $\phi(y) = \phi(x)$ *then* $d(y, \phi(x)) \geq d(y, \phi(x))$;
2. x *is not the midpoint of* $DIAM(S)$;
3. x *is not a vertex of* R;
4. x *is not an intersection point between the boundary of* R *and edges of* $FPVD(S)$;

then the line segment $[\phi(x), x]$ is perpendicular to e, and $\phi(x)$ is the unique farthest point to the line containing e among all points in S.

Proof. Let $a = \phi(x)$, or equivalently, $x \in V(a)$. Let $[pq]$ be the portion of e that lies in $V(a)$. Notice then p and q are either vertices of R or intersection points of e and $V(a)$, and $\forall y \in e$, $\phi(y) = \phi(x)$ if and only if $y \in [pq]$.

To prove the first part of the lemma, for the sake of contradiction suppose $[ax]$ is not perpendicular to e, then either $\angle axp$ or $\angle axq$ is $< \pi/2$. Without loss of generality suppose $\angle axp < \pi/2$. Let y be the projection point of a on e, if it exists, otherwise let $y = p$. Now the circle centered at y with radius $d(y, a)$ encloses the set S. However, $d(y, a) < d(x, a)$. Hence, we have a contradiction.

Let l denote the line containing e. To prove the second part (see Figure 2), for the sake of contradiction suppose there exists a point b of S, such that the distance between b and l is no less than that between a and l. Let y be the projection point of b on l. Given $d(y, b) \geq d(x, a)$, in the following we will prove $d(x, b) > d(x, a)$, which contradicts that $x \in V(a)$.

First, if $d(y, b) > d(x, a)$, or $d(y, b) = d(x, a)$ and $x \neq y$, then obviously $d(x, b) > d(x, a)$. Secondly, we show it is impossible that $d(y, b) = d(x, a)$ and $x = y$. In this case b is the mirror image of a with respect to l and x is the midpoint of $[ab]$. Since both a and b are the farthest points of x, x lies on the common edge of $V(a)$ and $V(b)$. In other words, $[ab]$ crosses and is perpendicular to an edge of $FPVD(S)$. Thus $[ab]$ is $DIAM(S)$ and x is its midpoint. However, we know x is not a midpoint of $DIAM(S)$. Hence we have a contradiction.

■

Lemma 6 indicates that for each edge e of R, there is at most one point that can be qualified as a candidate for the constrained Euclidean center with the property mentioned in the lemma, despite the fact that e may span many cells of $FPVD(S)$. When there are two points in S that are both the farthest to l, there is no point on e that can be qualified.

To find the point in S that is the farthest to a line, we first observe the following: the point in a given set of points S that is the farthest to a given line l is a vertex of the convex hull of S, denoted as $CH(S)$ (Proof omitted). Let h be the number of edges of $CH(S)$. In the following we show that by preprocessing $CH(S)$ in $O(h)$ time, we can find the farthest point of S to a given query line in $O(\log h)$ time.

Let $ang(e)$ be the angle of an edge e of $CH(S)$, measured as the angle between a right oriented horizontal ray x and e. The edges are directed in such a way that a walk along the directed edges constitutes a counterclockwise traversal of the boundary of $CH(S)$ (Figure 3). We store $ang(e)$ for each edge e of $CH(S)$ in sorted order in a set A. This can be done in $O(h)$ time given that edges in $CH(S)$ are in sorted order.

Given a line l, to answer the query of the farthest point in $CH(S)$, we first compute the angle of l for both directions of l, denoted as $ang(l_\alpha)$ and $ang(l_\beta)$. For each of $ang(l_k)$ ($k = \alpha$ or β), we do a binary search in A to find out the two

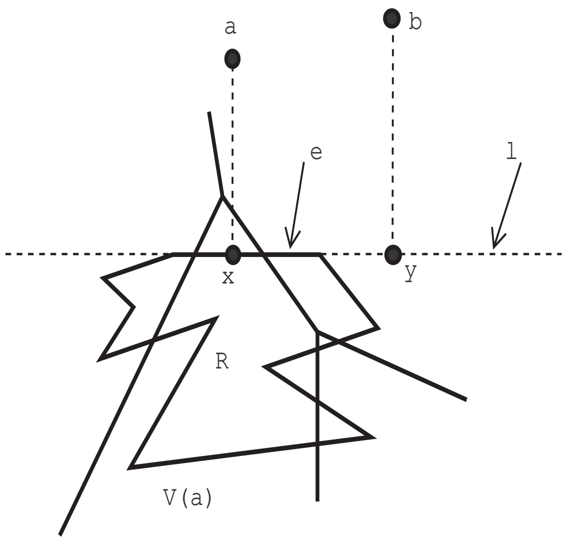

Fig. 2. Illustration of proof for Lemma 6

angels in A in between which $ang(l_k)$ falls. The two angles correspond to two neighboring edges in $CH(S)$, whose common vertex is then obtained. Denote by u and v the vertices obtained corresponding to $ang(l_\alpha)$ and $ang(l_\beta)$, respectively (Figure 3). Compute their respective distance to l. The vertex corresponding to the larger distance is then returned. If the two distances happen to be equal, by Lemma 6 we need not return a vertex. Moreover, during the binary search, if $ang(l_k)$ ($k = \alpha$ or β) happens to be equal to one of the angles in A, which means there are two vertices in $CH(S)$ on the same side of l that are both the farthest to l, then again we need not return a vertex corresponding to $ang(l_k)$ according to Lemma 6.

The query time is bounded by the binary search time, which is $O(\log h)$. Thus to compute Part 4 of Lemma 4, after preprocessing $CH(S)$ in $O(h)$ time, for each edge e of R, in $O(\log h)$ time we can find the unique point $p \in S$ that is the farthest from the line containing e. We then compute the projection point of p on e. Given $h = O(m)$, we have

Lemma 7. *Given $CH(S)$, part 4 of Lemma 4 can be computed in $O(m+n\log m)$ time.*

Characterization of the Constrained Euclidean Center Lemma 4, 5 and 6 lead to the following lemma characterizing the location of the Euclidean center constrained to a simple polygon.

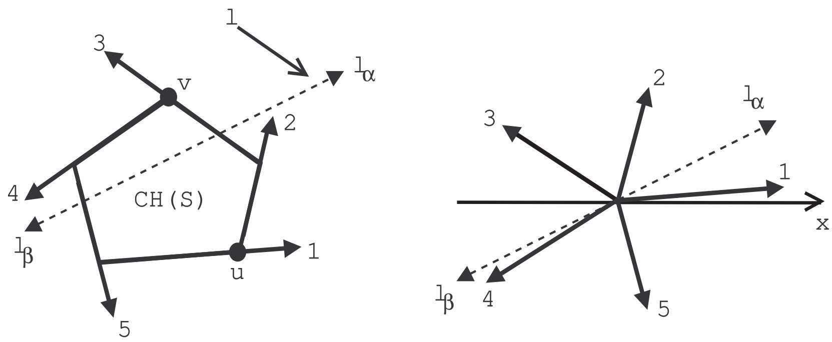

Fig. 3. Finding the farthest point to a given line l on the convex hull of S.

Lemma 8. *The Euclidean center under the constraints stipulated in Problem 2 is either the midpoint of $DIAM(S)$ provided that the diametrical circle contains the set S and the midpoint is contained in R, or lies on one of the following points that yields the smallest enclosing circle:*

1. *A vertex of $FPVD(S)$ contained in R;*
2. *A vertex of R;*
3. *A point of the boundary of R hit by a ray emanating from a vertex v of $FPVD(S)$ and containing an edge of $FPVD(S)$ incident on v;*
4. *A projection point on an edge e of R of a point in S that is the farthest to e.*

3.3 The Algorithm

Based on Lemma 8, we outline the following algorithm to compute the Euclidean center constrained to a simply polygon.
Algorithm 2 *Euclidean Center (S, R)*
Input: A set of demand points $S = s_1, s_2, ..., s_m$, and a simple polygon R of n vertices.

Output: Euclidean center of S constrained to lie in R.

1. Compute $CH(S)$ and $DIAM(S)$. Compute the circle C having $DIAM(S)$ as diameter. Preprocess R for point inclusion test using the algorithms of Kirkpatrick [Kir83] or Edelsbrunner, Guibas and Stolfi [EGS86]. If the midpoint of $DIAM(S)$ is contained in R and all points of S are contained in C then exit with the midpoint of $DIAM(S)$ and radius of C.
2. Compute $FPVD(S)$. Compute the set of vertices of $FPVD(S)$ contained in R using the point location structure of R, and compute their smallest enclosing circles.

3. Preprocess $FPVD(S)$ for point inclusion test using the algorithms of [EGS86] or [Kir83]. For each vertex v of R, determine which Voronoi cell it is in using the point location structure of $FPVD(S)$, and compute its smallest enclosing circle.
4. Preprocess R in $O(n)$ time for ray shooting query over R [HS95]. For each vertex v of $FPVD(S)$ and each edge e of $FPVD(S)$ incident on v, perform a ray shooting query over R, with a ray emanating from v and containing e [HS95]. If the point of the boundary of R hit by the ray is on e, compute its smallest enclosing circle.
5. Store in A in sorted order the set of angles of each edge of $CH(S)$. For each edge e of R, find the unique farthest point p of $CH(S)$ to e, if it exists, and compute the projection point of p on e, using the techniques of Lemma 7. Compute the smallest enclosing circle of the projection point.
6. Select and output the smallest enclosing circles found in steps 2-5.

Theorem 2. *Given a set of demand points $S = s_1, s_2, ..., s_m$, and an n-vertex simple polygon R, we can compute the Euclidean center of S constrained to lie in R in time $O((m+n)\log m + m \log n)$.*

Proof. The correctness of the algorithm follows from Lemma 8. The analysis is similar to that given in Theorem 1 and will not be duplicated here. The full proof can be found in [Wan01]

■

When m is $O(1)$, Algorithm 2 runs in optimal $\Theta(n)$ time. It remain open problems whether Algorithm 2 is optimal when m is $\omega(1)$ or whether there exists linear time solution to the problem.

As with the case of obnoxious facility location, the placement of single desirable facility can be generalized.

Corollary 2. *Given a set of demand points $S = s_1, s_2, ..., s_m$, and a region R consisting of r disjoint simple polygons with a total of n vertices, we can compute the location of a single desirable facility constrained to lie in R, under Euclidean distance, in time $O((m+n)\log m + (m\sqrt{r} + n)\log n + n\sqrt{r} + r^{3/2}\log r)$.*

4 Conclusion

We developed an efficient algorithm for locating a single obnoxious facility in a simple polygonal region. We also presented an efficient algorithm for locating a single desirable facility in a simply polygonal region. The algorithms make use of the Voronoi diagram or the farthest-point Voronoi diagram. They are based on and are improvements of the algorithms in [HST00] and [BT00]. Given a set S of m demand points and a simple polygon R of n vertices, we have shown how to compute the location of either an obnoxious or a desirable facility to lie in R, in $O((m+n)\log m + m \log n)$ time. The previous best known algorithms solving the above problems both had worst case quadratic running times.

References

[BT00] P. Bose and G. T. Toussaint. Computing the constrained euclidean, geodesic and link centers of a simple polygon with applications. *Studies of Location Analysis, Special Issue on Computational Geometry*, 15:37–66, 2000.

[EGS86] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15:317–340, 1986.

[HL99] D. Halperin and C. Linhart. The smallest enclosing disks with obstacles. Manuscript, Dep. of Comp. Science, Tel-Aviv University, Israel, 1999.

[HS95] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *Journal of Algorithms*, 18(3):403–431, 1995.

[HSG00] D. Halperin, M. Sharir, and K. Goldberg. The 2-center problem with obstacles. In *Proc. of the 16th ACM Symp. on Comp. Geom.*, pages 80–90, 2000.

[HST00] F. Hurtado, V. Sacristn, and G. Toussaint. Constrained facility location. *Studies of Location Analysis, Special Issue on Computational Geometry*, 15:17–35, 2000.

[Kir83] D. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12(1):28–35, 1983.

[Meg83] N. Megiddo. Linear-time algorithms for linear programming in r3 and related problems. *SIAM J. Comput.*, 12:759–776, 1983.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, New York, 1985.

[Tou83] G. T. Toussaint. Computing largest empty circles with location constraints. *Int. J. of Comp. and Information Science*, 12(5):347–358, 1983.

[Wan01] Q. Wang. *Facility Location Constrained to a Simple Polygon*. Master's thesis, Carleton University, 2001.