Some constrained minimax and maximin location problems

Ferran Hurtado[1] Vera Sacristán[1] Godfried Toussaint[2]

Abstract

In this paper we consider constrained versions of the Euclidean minimax facility location problem. We provide an $O(n+m)$ time algorithm for the problem of constructing the minimum enclosing circle of a set of n points with center constrained to satisfy m linear constraints. As a corollary, we obtain a linear time algorithm for the problem when the center is constrained to lie in an m-vertex convex polygon, which improves the best known solution of $O((n+m)\log(n+m))$ time.

We also consider some constrained versions of the maximin problem, namely an obnoxious facility location problem in which we are given a set of n linear constraints, each representing a halfplane where some population may live, and the goal is to locate a point such that the minimum distance to the inhabited region is maximized. We provide optimal $\Theta(n)$ time algorithms for this problem in the plane, as well as on the sphere.

[1]Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Pau Gargallo 5, 08028 Barcelona, Spain. e-mail: hurtado,vera@ma2.upc.es. Partially supported by Proyecto DGES-SEUID PB96-0005-C02-02.

[2]School of Computer Science, McGill University, 3480 University Street, Montréal, Canada H3A2A7. e-mail: godfried@cs.mcgill.ca.

1 Introduction

In a classical facility location problem [10] we are given a set of n points C in the plane representing n customers, plants to be serviced, schools, markets, distribution sites or any other locations, depending on the context in which the problem is embedded, and it is desired to determine the location X (find another point in the plane) where a facility (service, transmitter, dispatcher, etc.) should be located so as to minimize the Euclidean distance from X to its furthest customer. Such a *minimax* criterion is particularly useful in locating emergency facilities, such as police stations, fire-fighting stations and hospitals where it is desired to minimize the worst-case response time. This problem has an elegant and succinct geometrical interpretation: find the smallest circle that encloses a given set of n points. The center of this circle is precisely the location of X. This problem is known in the literature under various names such as the *minimum spanning circle* or the *Euclidean 1-center problem*. It has a long history and was posed originally in 1857 by Sylvester [25]. This geometric setting together with the fact that the smallest enclosing circle of C is determined by either a pair or a triplet of points in C immediately suggests a naive, brute-force, method for obtaining a solution: (1) for every pair of points determine its diametral circle, (2) for every three points determine the circle they uniquely define, (3) for every circle thus formed determine if no other points lie outside it and (4) out of all such "full" circles select the smallest encountered. This algorithm has a time complexity of $O(n^4)$. An improved adaptive algorithm for this problem was proposed by Bass and Schubert in 1967 [2] which runs in $O(h^4 + n \log n)$ time, where h is the number of extreme points of the *convex hull* of C. In 1972 Elzinga and Hearn [9] proposed a more efficient algorithm that runs in $O(n^2)$ time (includes a linear test which might be performed a linear number of times).

Shamos [23], Shamos and Hoey [24] and Preparata [20] were the first to discover $O(n \log n)$ time algorithms, a considerable improvement over the $O(n^2)$ solution of Elzinga and Hearn [9]. The algorithms in [23] and [24] have a step in which they compute the diameter of the set with an invalid diameter-algorithm and a counter-example to this diameter algorithm is given by Bhattacharya and Toussaint [4]. In spite of this default it is shown in [4] that the minimal spanning circle algorithm proposed in [23] always yields the correct solution, and two alternate $O(n \log n)$ time algorithms are also given there. Independently, Lee [15] proposed a similar $O(n \log n)$ time algorithm. See also the survey by Lee and Wu [16]. Finally, Megiddo [17] found an optimal $O(n)$ algorithm for solving this problem. His solution, based on the *prune-and-search technique*, has been extended in [8, 18] to solve optimally the generalized weighted version of the problem in any fixed dimension. Several other generalizations of the problem have been considered as well where non-Euclidean metrics are used and where the facility is not a point but either a line, a polygonal path or a polygonal region. For a survey of these generalizations and more recent results the reader is referred to [21, 22].

While much has been done on such unconstrained versions of the classical problem, little has been done for the case when additional constraints are present. Megiddo studied the case in which the center of the smallest enclosing circle is forced to lie on a straight line, as a fundamental step in his solution to the unconstrained problem. Some work has been done for the 2-dimensional problem where the center is constrained to lie in a given simple or convex polygon [6].

In this paper we consider constrained versions of the problem. We provide an $O(n+m)$

time algorithm for the problem of the minimum enclosing circle with its center constrained to satisfy m linear constraints. As a corollary, we obtain a linear time algorithm for the problem when the center is constrained to lie in an m-vertex convex polygon, which improves the best known solution of $O((n+m)\log(n+m))$ time [6]. As a byproduct of our technique, we also show that the smallest circle enclosing n points with the constraint that the circle must pass through a given point or that the circle must be tangent to a given line can be solved in $\Theta(n)$ time.

We also consider some versions of the *maximin* problem, that can be considered as a dual of the previous one, namely an obnoxious facility location problem, in which we are given a set of linear constraints, each one of them representing a halfplane where some population may live, and the goal is to locate a point such that the minimum distance to the inhabited region is maximized. Such a maximin criterion es particularly useful in locating obnoxious facilities, such as nuclear plants, chemical factories, and waste disposal centres.

Geometrically, the problem consists of finding the largest circle enclosed in the convex polygon implicitly given by the intersection of the halfplanes defined by a set of linear constraints. The center of this circle is the location of the obnoxious optimal facility. We provide an $O(n)$ time algorithm for this problem in the plane, as well as on the sphere.

2 Locating Minimax Facilities

In this section we study some two-dimensional constrained versions of the minimax facility location problem. We provide an optimal linear time algorithm that locates the minimum enclosing circle of a set of points, with center constrained to satisfy a set of linear constraints. This result can be applied in particular when the center is constrained to lie in a convex polygon. Finally, we indicate how a similar technique can also be applied to optimally solve other related problems, such as finding the minimum spanning circle with the constraint that the circle must pass through a given point or that the circle must be tangent to a given line.

The algorithm we present is simple and direct. It follows a prune-and-search strategy that will be used again in the other algorithms of this section. This underlying common scheme is inspired by Megiddo's algorithm for finding the minimum spanning center without constraints in [17]. The strategy of Megiddo in [17, 18] was independently studied and improved by Dyer in [7, 8] and has been applied with success by Bhattacharya et al. in [3, 14] to obtain optimal algorithms for the intersection radius of sets of lines, segments and convex polygons. The basic idea of our algorithm is the following:

1. Solve the problem in one less dimension (in this case, with the constraint of having the center of the minimum spanning circle lying on a given line), determine whether or not the solution to the original problem lies on the line and, if it does not, determine which side (halfplane) of the line it belongs to.

2. Apply this result recursively to reduce the size of the original problem.

Let us see first the details of step 1.

Lemma 1 *Given a line s, by solving the problem restricted to s it is possible to find the center of the minimum spanning circle of a set of n points in the plane, constrained to* *belong to the intersection of a set of m halfplanes, if it lies on s, or to decide on which side of s it lies, in optimal $\Theta(n+m)$ time.*

Proof: In the first place, we want to decide whether or not the center of the minimum circle that contains a given set of n points $\{p_1, \ldots, p_n\}$ while satisfying a set of m linear constraints, $a_i x + b_i y + c_i \geq 0$, $i = 1, \ldots, m$, lies on s. If it does, we need to find it. If it does not, we need to know on which side of s it lies. We can use the following algorithm:

1. Detect if the line s and the constraint polygon intersect, and determine on which side of s the polygon lies in case they do not intersect:
   (a) Determine both the intersection of all upper and all lower halfplanes (left and right, if s is vertical) with boundary line parallel to s. Each such intersection is either a halfplane or the whole plane (when the set of intersected halfplanes is empty). If their intersection with s is empty, then the constraint polygon is itself empty. If s lies in only one of these intersections, the side of s containing the constraint region is determined. In s lies in both intersections, proceed to the next step.
   (b) Intersect the line s with all the upper halfplanes with respect to the direction of s. The result will be a ray whose origin will be determined by one or at most two halfplanes, depending on whether s intersects the associated polygonal region in an edge or a vertex (notice that in a degenerate case with redundant constraints, the vertex may be determined by more than two halfplanes, but only two of them may be relevant, and they can be easily detected). Proceed the same way with the lower halfplanes.
   (c) If the two rays are disjoint, the intersection of s and the constraint polygon is again empty. The intersection of the (at most) four halfplanes that determine the two rays indicates on which side of s the constraint polygon lies (in some cases, it may indicate even the emptiness of the polygon). If the two rays have a common segment, call it s', and proceed to the next step.

2. Solve the s'-constrained problem, that is, start by finding the minimum spanning circle of $p_1, \ldots, p_n$ with center on s. If it belongs to s', it is the s'-optimum. Otherwise, the s'-optimum is located at the segment's endpoint closest to it (this holds by convexity).

3. Let us call o the s'-constrained solution. In any case, the oracle finds the point p_i that determines the maximum distance to o. There are two possibilities:
   (a) If o is the s-restricted solution, then p_i determines the halfplane where the solution to the original problem lies (Figure 1).
   (b) If o is the endpoint of the segment s' which lies closer to the s-restricted solution, then the halfplane where the solution to the original problem lies is found in the following way: Consider the edge of the polygon through o, and orthogonally project p_i onto it (or its prolongation). The projection point indicates the direction that minimizes the distance to p_i, and hence indicates which side of s allows to improve the radius of the solution circle (Figure 2).

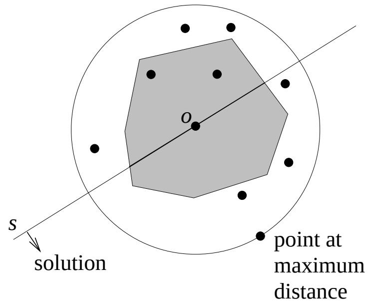

Figure 1: How to determine the solution halfplane when the s-restricted solution belongs to the polygon.

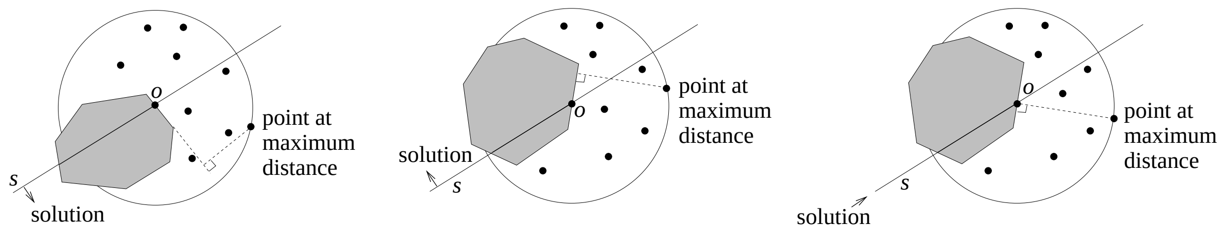

Figure 2: How to determine the solution halfplane when the s-restricted solution does not belong to the polygon.

Two remarks must still be made at this point:

- It may happen that the edges through the point o are two (when o is a vertex of the polygon). In this case, both of them must be considered in the previous explanation.
- It may also happen that the points at maximum distance from o are more than one. If they all indicate the same halfplane as the solution one, this is the solution halfplane. If at least two of them indicate opposite solution halfplanes, then the optimim point o just found is the global optimum.

This algorithm takes $O(n+m)$ time. Clearly step 1 can be solved in $O(m)$ time. In step 2, the center of the minimum spanning circle constrained to lie on a line can be found in $O(n)$ time using Megiddo's algorithm in [17]. Finally, for step 3, $O(n)$ time suffices, as it consists essentially in determining the maximum distance from the segment constrained optimum to the set of points. □

Let us see now how the original problem can be solved.

Theorem 2 *The minimum spanning circle of a set of n points in the plane, with center constrained to satisfy a set of m linear restrictions, can be found in optimal $\Theta(n+m)$ time.*

Proof: We want to solve the problem of finding the minimum circle that contains a given set of n points $\{p_1, \ldots, p_n\}$ and whose center satisfies a set of m linear constraints, $a_i x + b_i y + c_i \geq 0$, $i = 1, \ldots, m$. From Lemma 1 we know that we can use as an oracle a linear time algorithm that can decide on which side of a given line the solution to the problem lies. We can then solve the problem using the following algorithm:

1. Take the n points of the initial set, $p_1, \ldots, p_n$, and pair them up. Consider the orthogonal bisectors of the segments they form. Refer to Figure 3.
   (a) Form the set B_1 of the x-coordinates of all the vertical bisectors.
   (b) Consider the angle α that each of the remaining bisectors forms with the horizontal positive halfline, $-\pi/2 \leq \alpha < \pi/2$, and compute the median value of all these angles. Then take the direction obtained as horizontal (by a coordinate change that does not modify the vertical direction).
   (c) Pair each negative slope bisector with one of positive slope, such that each pair of bisectors intersects in a point c_i and at most one bisector is left unpaired (use horizontal bisectors if necessary).

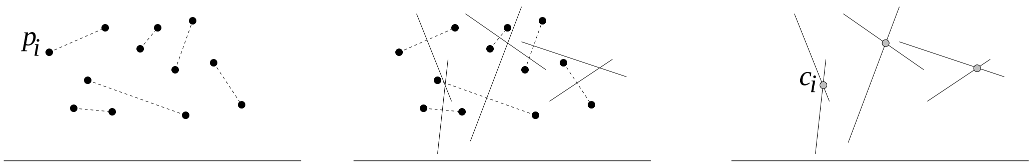

Figure 3: How to obtain the points c_i from the p_i.

   (d) Form the set B_2 of the y-coordinates of all the remaining horizontal bisectors.
   (e) Consider the set C_1 of the x-coordinates of all the intersection points c_i, and the set C_2 of their y-coordinates.

2. Consider the halfplanes determined by the linear constraints and pair them up. Always pair an upper halfplane (that is, a halfplane $a_i x + b_i y + c_i \geq 0$ having $b_i > 0$) with another non parallel upper halfplane, and a lower halfplane (that is, a halfplane $a_i x + b_i y + c_i \geq 0$ having $b_i < 0$) with another non parallel lower halfplane. When two upper or two lower halfplanes are parallel, one of them is redundant and can be discarded. Halfplanes whose equation is $a_i x + c_i \geq 0$ (vertical halfplanes) can be classified as upper or lower halfplanes by considering the sign of their a_i coefficient. This process will pair up all the halfplanes, except possibly at most two of them (an upper one and a lower one).

3. Consider the intersection points d_i of all the pairs of linear restrictions. Form the set D_1 of the x-coordinates of all the intersection points d_i, and the set D_2 of their y-coordinates.

4. Compute the median value y_m of all the y-coordinates $y \in B_2 \cup C_2 \cup D_2$. Apply the oracle of Lemma 1 to the horizontal line $y = y_m$, determining whether the center of the solution circle lies on, above or below the line. If it lies on the line, the oracle will find it, and we are done. Else, suppose that the subroutine has determined that the solution to the general problem lies below the line $y = y_m$.

5. At that point, we can discard a point p_i for each $y \in B_2$ belonging to the halfplane opposite to the solution (Figure 4). Take any $y \in B_2$ (i.e. corresponding to a horizontal bisector) with $y \geq y_m$. One of the two points p_i that define the bisector must be redundant, namely the lower one, as it is certainly placed closer than the other to the center of the solution circle, and hence cannot determine it.

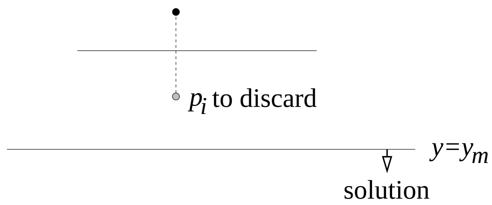

Figure 4: How to discard a point p_i for each horizontal bisector lying in the upper halfplane.

6. Replace C_1 (respectively D_1) by its subset C_1' (D_1') containing the x-coordinates of all the points c_i (d_i) whose y-coordinates $y \in C_2$ (D_2) belong to the halfplane opposite to the solution (in the example, the upper one).

7. Now compute the median value x_m of the x-coordinates $x \in B_1 \cup C_1' \cup D_1'$, and apply the subroutine to the vertical line $x = x_m$. This determines if the center of the solution point lies on the line, in which case we are done. If it does not, the oracle determines on which of the four quadrants the center of the solution circle lies. Suppose that it lies in the lower right quadrant with respect to the lines $y = y_m$ and $x = x_m$.

8. Then, it will be possible to discard a point p_i for each $x \in B_1 \cup C_1'$ that belongs to the halfplane opposite to the solution, as well as a linear restriction $a_i x + b_i y + c_i \geq 0$ for each $x \in D_1'$ that belongs to the same halfplane. Let us consider this discarding process in more detail.

   (a) Each $x \in B_1$ corresponds to a vertical bisector. If $x \leq x_m$, then one of the two points p_i that define the bisector is redundant, namely the right one, for it is certainly placed closer than the other to the center of the solution circle and hence cannot determine it (Figure 5).

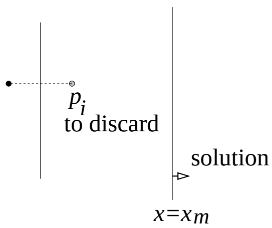

Figure 5: How to discard a point p_i for each vertical bisector lying in the left halfplane.

   (b) Each $x \in C_1'$ with $x \leq x_m$ corresponds to a point c_i lying in the upper left quadrant. So, the positive slope bisector through c_i does not intersect the solution quadrant (the lower right one). This means that one of the two points p_i that determine this bisector (in this case, the lower right one) is closer than the other to the center of the solution circle, and cannot determine it (see Figure 6).

   (c) Each $x \in D_1'$ with $x \leq x_m$ corresponds to a point d_i lying in the upper left quadrant. There are three possible situations for the pair of constraints that intersect at d_i:

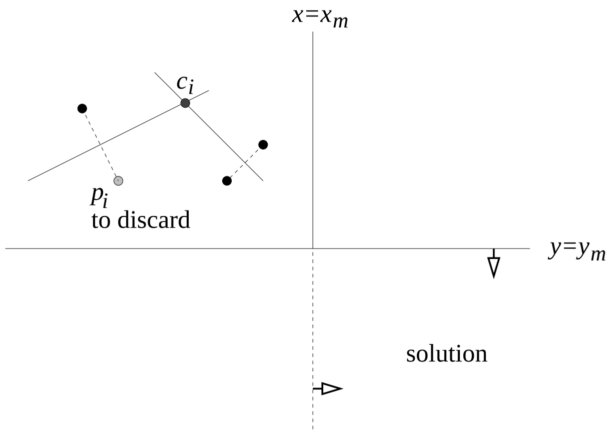

Figure 6: How to discard a point p_i for each c_i lying in the upper left quadrant.

- The lines determining the two halfplanes have non negative slope: in this case, both constraints are irrelevant for the solution of the problem and can be discarded (Figure 7), for they cannot intersect the quadrant that contains the solution point.

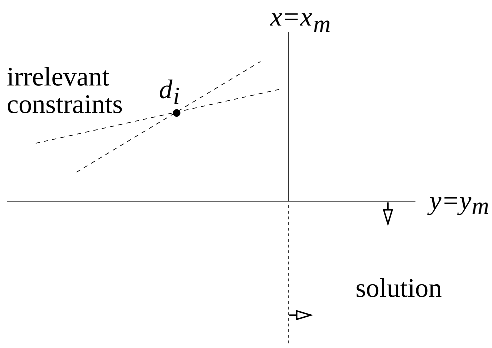

Figure 7: Discarding two non negative slope constraints.

- One of the lines determining the two halfplanes has non negative slope: in this case, the associated constraint is irrelevant and can be discarded for the same reason as above (Figure 8).

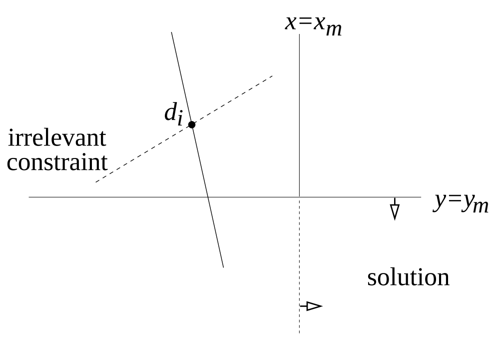

Figure 8: Discarding one non negative slope constraint.

- Both lines have negative slope. Then, two cases must be considered (Figure 9):

– If both are lower halfplanes, then the line having greater slope corresponds to an irrelevant constraint to the problem, and can be discarded.
– If both are upper halfplanes, then the line having smaller slope corresponds to an irrelevant constraint to the problem, and can be discarded.
– The case in which one is a lower halfplane and the other is an upper halfplane has been eliminated when pairing the halfplanes.

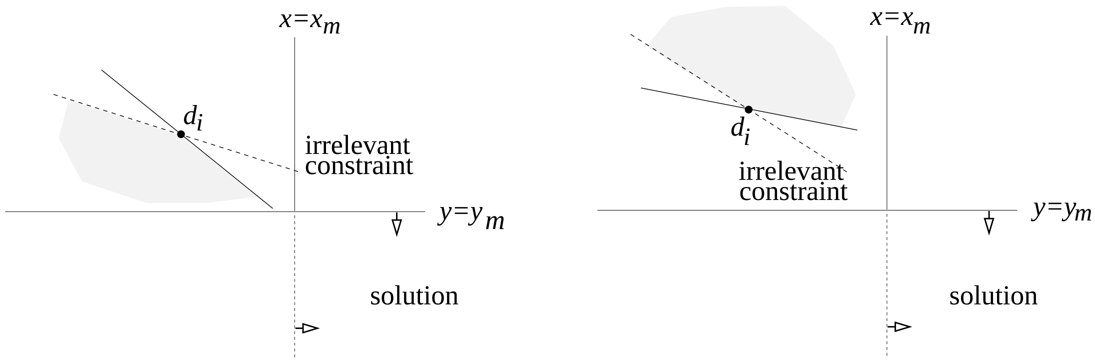

Figure 9: Discarding one negative slope constraint.

We have seen that each iteration of the algorithm allows us to discard one point p_i for each horizontal bisector lying in the upper halfplane, another one for each vertical bisector lying in the left halfplane, and another one for each intersection point c_i lying in the upper left quadrant. As for the linear restrictions, the algorithm discards at least one for each intersection point d_i lying in the upper left quadrant. We will prove that in this process we discard at least a fixed fraction of the input elements, either points or constraints, before applying the algorithm again to the reduced input.

Let b_1, b_2, c_1, c_2, d_1 and d_2 represent, respectively, the cardinality of the sets B_1, B_2, C_1, C_2, D_1 and D_2. Let $\overline{c}_1$ and $\overline{d}_1$ represent, respectively, the cardinality of the sets C_1' and D_1'. Consider h_2 to be the number of $y \in B_2$ that lie in the upper halfplane, h_1 to be the number of $x \in B_1$ that lie in the left halfplane, s_1 to be the number of $x \in C_1'$ that lie in the upper left quadrant, and r_1 to be the number of $x \in D_1'$ that lie in the upper left quadrant. With this notation, the number of points p_i that are eliminated from the input is

$$h_2 + h_1 + s_1 + r_1.$$

Since y_m and x_m are median values, the following inequalities hold:

$$h_2 + \overline{c}_1 + \overline{d}_1 \geq \tfrac{1}{2}(b_2 + c_2 + d_2), \tag{1}$$
$$h_1 + s_1 + r_1 \geq \tfrac{1}{2}(b_1 + \overline{c}_1 + \overline{d}_1); \tag{2}$$

together with the equalities:

$$\begin{aligned} c_1 &= c_2, \\ d_1 &= d_2, \\ n + m &= 2b_1 + 2b_2 + 4c_2 + 2d_2. \end{aligned} \tag{3}$$

The above relations yield a lower bound for the number of discarded elements from the input:

$$\begin{aligned} h_2 + h_1 + s_1 + r_1 &\overset{(2)}{\geq} h_2 + \tfrac{1}{2}(b_1 + \overline{c}_1 + \overline{d}_1) \overset{(1)}{\geq} \\ &\geq h_2 + \tfrac{1}{2}b_1 + \tfrac{1}{4}(b_2 + c_2 + d_2) - \tfrac{1}{2}h_2 \overset{(3)}{=} \\ &= \tfrac{1}{2}h_2 + \tfrac{1}{2}b_1 + \tfrac{1}{4}b_2 + \tfrac{1}{4}d_2 + \tfrac{1}{16}(n + m - 2b_1 - 2b_2 - 2d_2) = \\ &= \tfrac{1}{2}h_2 + \tfrac{3}{8}b_1 + \tfrac{1}{8}b_2 + \tfrac{1}{8}d_2 + \tfrac{1}{16}(n+m) \geq \tfrac{1}{16}(n+m). \end{aligned}$$

Hence, at least $\frac{1}{16}$ of the initial elements of the input is discarded. Since the oracle runs in linear time (by Lemma 1), the cost of each application is linear in the size of the input, because it only requires computing the median [5, 12]. So, the total complexity of the algorithm is

$$T(n+m) \leq O(n+m) + O(\tfrac{15}{16}(n+m)) + O((\tfrac{15}{16})^2(n+m)) + O((\tfrac{15}{16})^3(n+m)) + \cdots = O(n+m).$$

□

Applying Theorem 2 to the constraints determined by the ordered edges of a polygon we obtain the following Corollary, which improves the best known solution of $O((n+m)\log(n+m))$ time [6]. In fact, a direct and simple solution is described in [13], but is omitted here due to the fact that Theorem 2 is a more general result whose proof is based on the same technique.

Corollary 3 *Let P be a convex m-gon given by its vertices, ordered as they appear on the boundary. The minimum spanning circle of a set of n points in the plane, with center constrained to lie in P, can be found in $\Theta(n+m)$ time.*

The technique that we have used to optimally solve the problem of finding the minimum enclosing circle of a set of points with its center constrained to satisfy a set of linear restrictions, can be applied to a family of problems. In each case, an oracle must be found that decides in linear time on which side of a given line the solution to the problem lies. When such an oracle exists, a main algorithm following the scheme of the one proposed in the proof of Theorem 2 gives the optimal solution. We propose two examples.

Theorem 4 *The minimum spanning circle of a set n of points $p_1, \ldots, p_n$ in the plane, constrained to be anchored to a fixed point q, can be found in optimal $\Theta(n)$ time.*

Proof: First notice that the anchor point q must be external to the convex hull of the set of points p_i (or, at most, it can be placed on a vertex), for the problem to have a solution. This situation can be detected in linear time by an algorithm of [17], by checking if q is a vertex of the convex hull of $\{q, p_1, \ldots, p_n\}$. At the same time, the two neighbors of q in the hull can be obtained. Call them p_1 and p_2.

Therefore, let us suppose that the anchor point q lies in the exterior of the convex hull (or on one of its vertices). The main algorithm is a simplification of the one proposed in the proof of Theorem 2, for the input of the problem does not contain linear restrictions, but only points.

As for the oracle, consider the farthest point Voronoi diagram of $\{q, p_1, \ldots, p_n\}$ [19]. The center of the minimum spanning circle that we are looking for is located on the boundary of the Voronoi region of q. Given any line s, we can solve the s-constrained problem by computing the intersection of s with the orthogonal bisectors of all the segments qp_i, obtaining the intersection of s with the farthest point Voronoi region of q. Notice, also, that the Voronoi region of q is an unbounded polygonal region delimited by two halflines, r_1 and r_2, that are the orthogonal bisectors of qp_1 and qp_2.

- If the intersection of s with the Voronoi region of q is empty, then the constrained problem has no solution. The intersection of s with r_1 and r_2 (the two halflines in the boundary of the region) allows us to decide on which side of s the region lies and, with it, on which side of s the solution lies.

- If s intersects the Voronoi region in a non empty segment, the solution to the constrained problem is the closest point of the segment to point q. This point is used to decide on which side of s the solution of the general problem lies, in a similar way to that in the proof of Lemma 1. □

Theorem 5 *The minimum spanning circle of a set of n points $p_1, \ldots, p_n$ in the plane, constrained to be tangent to a given line l, can be found in optimal $\Theta(n)$ time.*

Proof: When the enclosing circle is to be tangent to a fixed line l, the problem is very similar to the previous one, the only difference being that in Theorem 4 the Voronoi region of the anchor point q is determined by the orthogonal bisectors of the segment lines qp_i, whereas here the Voronoi region of the tangent line l is determined by parabolas having the line l as directrix and the points p_i as foci. □

3 Locating Maximin Facilities

In this section, we study two versions of a maximin facility location problem. Geometrically, finding the point that maximizes the minimum distance to a given set of points in the plane is equivalent to finding the center of the maximum circle enclosed in the complement of that set. We provide an optimal linear time algorithm that computes the maximum circle such that all its points satisfy a given set of linear constraints, that is the maximum circle enclosed in a given intersection of halfplanes. This result was known [1] for the particular case in which the circle is constrained to lie in a convex polygon given by the ordered list of its edges, as they appear on the boundary of the polygon. The solution relies heavily on this order, while the algorithm that we present here applies to any unordered and possibly redundant implicit description of the polygon as an intersection of halfplanes. The possibility of directly approaching the problem through linear programming was mentioned in [26]. We also provide an optimal linear time algorithm that solves the equivalent problem on a sphere: find the maximum spherical cap enclosed in a convex polygon defined on a sphere as the intersection of spherical halfspaces. The problem is solved by reducing it to a dual minimax problem on the halfsphere.

Theorem 6 *The maximum circle enclosed in the intersection of a set of n halfplanes can be found in optimal $\Theta(n)$ time.*

Proof: Consider the plane π, where the problem is posed, as imbedded in a three-dimensional space. Each of the n halfplanes H_i is delimited by a line h_i. For each H_i, consider the plane π_i through h_i that forms with π an angle of $45°$ so that the lower

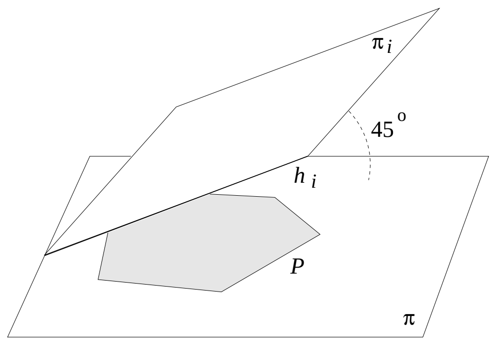

Figure 10: How to construct the polyhedral region.

halfspace determined by π_i contains H_i (Figure 10). The intersection of all these lower halfspaces is a polyhedral region that satisfies the following property: for each point $x \in H_i$, the distance from x to the line h_i equals the vertical distance from x to π_i. Hence, the center of the maximum circle enclosed in the polygon $P = \cap_{i=1}^{n} H_i$ is located in the orthogonal projection onto π of the polyhedral region's highest point. This point can be found in optimal $\Theta(n)$ time by a three-dimensional linear programming algorithm. $\square$

Let us consider now the equivalent problem on a sphere: no matter if we consider the Euclidean three-dimensional distance or the geodesic distance on the sphere, the goal is to find the largest spherical cap that satisfies a set of linear restrictions on the sphere, that is, that is enclosed in the intersection of a set of halfspaces determined by halfplanes containing the center of the sphere.

This problem can be reduced to a minimax problem by a polarity transformation. Consider the center of the sphere to be the origin. Each halfspace through the center of the sphere can be associated to the point on the sphere which is the intersection with the external normal ray to the halfspace through the origin (see Figure 11).

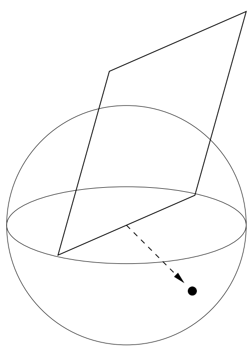

Figure 11: The polarity associates a point to each plane through the origin.

Each spherical cap C determines a circular halfcone D with vertex in the origin, which is also determined by the set of all the planes that are tangent to it. The set of all the normal rays associated as above to these planes define a new halfcone D^*, and hence a cap C^* that we will call polar to C. Clearly, D and D^* are coaxial circular halfcones, and their respective aperture angles α and α^* add up to π. The open halfspaces through the origin that enclose D are exactly those whose external normal ray belongs to D^*.

Let us see now how the problem can be solved using this polarity.

Theorem 7 *The maximum spherical cap enclosed in a convex polygon defined on a halfsphere as the intersection of n spherical halfspaces can be found in optimal $\Theta(n)$ time.*

Proof: Consider the planes determined by the edges e_i of the polygon and the center of the sphere. Associate to each of these planes its polar point p_i. Consider any spherical cap C on the halfsphere and its polar cap C^*. The following property holds: C is enclosed in the polygon $P = \{e_1, \ldots, e_n\}$ if, and only if, C^* encloses the set of points $P^* = \{p_1, \ldots, p_n\}$ (see Figure 12).

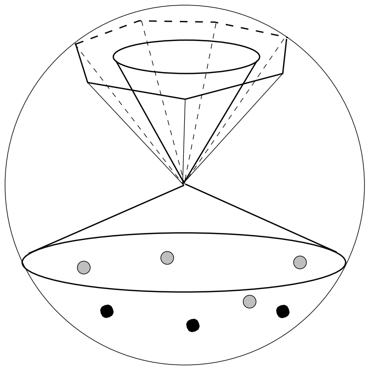

Figure 12: The cap is enclosed in the polygon if, and only if, its polar contains the points.

The relation between the aperture angles of the associated halfcones D and D^* affords a reduction of the problem of finding the maximum cap C enclosed in the polygon P to the problem of finding the minimum cap C^* enclosing the set of points P^*. Notice that this reduction does not require the polygon to be given explicitly, but only as an intersection of (possibly redundant) spherical halfspaces.

As for the minimax problem, it is known that the smallest cap enclosing a set of n points on an open halfsphere can be found in $\Theta(n)$ time [11]. □

It is worth noticing that the polar points to be covered by a minimum cap all lie in a halfsphere, for it can be proved that the problem of covering a set of n points on a sphere with a minimum spherical cap has complexity $\Omega(n \log n)$ in the general case [13].

4 Conclusions

In this paper, we have used the *prune-and-search* technique to solve the minimax location problem consisting in locating a point x such as to minimize the maximum distance to n given points in the case in which the solution point is constrained to satisfy a set of m linear restrictions, giving rise to an optimal $\Theta(n+m)$ algorithm. As a corollary, we have proved that the complexity of the problem, when the solution point x is constrained to lie in an m-vertex convex polygon, is also $\Theta(n+m)$. We have also applied the technique with success to optimally solve in $\Theta(n)$ time some other constrained minimax location problems, such as finding the smallest circle enclosing n points, whose boundary is anchored to a given point or tangent to a given line.

We have also considered maximin problems, in which the goal is to locate a point x such as to maximize the minimum distance to n given halfplanes. An optimal $\Theta(n)$

solution to this problem can be found by reducing it to a linear programming problem in three dimensions, i.e. to a minimax problem in one more dimension. The analogous problem on the sphere can also be solved in optimal $\Theta(n)$ time by a polarity that reduces it to a minimax problem.

As a conclusion, we have obtained optimal results for some realistic constrained versions of both minimax and maximin location problems, which illustrates the power of the *prune-and-search* technique.

Acknowledgments

Thanks go to the anonymous referees for their valuable comments.

References

[1] A. Aggarwal, Leonidas J. Guibas, J. Saxe, and P. Shor. A linear time algorithm for computing the Voronoi diagram of a convex polygon. In *Proc. 19th Annu. ACM Sympos. Theory Comput.*, pages 39–45, 1987.

[2] L. J. Bass and S. R. Schubert. On finding the disc of minimum radius containing a given set of points. *Math. Comput.*, 12:712–714, 1967.

[3] B. K. Bhattacharya, S. Jadhav, A. Mukhopadhyay, and J.-M. Robert. Optimal algorithms for some smallest intersection radius problems. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 81–88, 1991.

[4] B. K. Bhattacharya and G. T. Toussaint. On geometric algorithms that use the furthest-point Voronoi diagram. In G. T. Toussaint, editor, *Computational Geometry*, pages 43–61. North-Holland, Amsterdam, Netherlands, 1985.

[5] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *J. Comput. and Syst. Sci.*, 7:448–461, 1973.

[6] P. Bose and G. Toussaint. Computing the constrained euclidean, geodesic and link center of a simple polygon with applications. In *Proc. of Pacific Graphics International*, pages 102–112, 1996.

[7] M. E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comput.*, 13:31–45, 1984.

[8] M. E. Dyer. On a multidimensional search technique and its application to the Euclidean one-centre problem. *SIAM J. Comput.*, 15:725–738, 1986.

[9] J. Elzinga and D. W. Hearn. Geometrical solutions to some minimax location problems. *Transp. Sci.*, 6:379–394, 1972.

[10] R. L. Francis and J. A. White. *Facility Layout and Location.* Prentice Hall, Englewood Cliffs, NJ, 1974.

[11] F. Gómez, S. Ramaswami, and G. Toussaint. On removing non-degeneracy assumptions in computational geometry. In *Algorithms and Complexity (Proc. CIAC' 97)*, *Lecture Notes Comput. Sci.*, Springer-Verlag, 1997.

[12] C. A. R. Hoare. Algorithm 63 (partition) and algorithm 65 (find). *Communications of the ACM*, 4(7):321–322, 1961.

[13] F. Hurtado, V. Sacristán, and G. Toussaint. Facility location problems with constraints. Technical Report MA2-IR-96-0010, U. Politècnica de Catalunya, 1997.

[14] S. Jadhav, A. Mukhopadhyay, and B. Bhattacharya. An optimal algorithm for the intersection radius of a set of convex polygons. *J. Algorithms*, 20:244–267, 1996.

[15] D. T. Lee. Furthest neighbor Voronoi diagrams and applications. Report 80-11-FC-04, Dept. Elect. Engrg. Comput. Sci., Northwestern Univ., Evanston, IL, 1980.

[16] D. T. Lee and Y. F. Wu. Geometric complexity of some location problems. *Algorithmica*, 1:193–211, 1986.

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

[18] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31:114–127, 1984.

[19] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Chichester, UK, 1992.

[20] F. Preparata. Minimum spanning circle. Technical report, Univ. Illinois, Urbana, IL, 1977. in 'Steps into Computational Geometry'.

[21] J.-M. Robert and G. Toussaint. Computational geometry and facility location. Technical Report SOCS 90.20, McGill Univ., Montreal, PQ, 1990.

[22] J.-M. Robert and G. Toussaint. Linear approximation of simple objects. *Comput. Geom. Theory Appl.*, 4:27–52, 1994.

[23] M. I. Shamos. *Computational Geometry.* Ph.D. thesis, Dept. Comput. Sci., Yale Univ., New Haven, CT, 1978.

[24] M. I. Shamos and D. Hoey. Closest-point problems. In *Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 151–162, 1975.

[25] J. J. Sylvester. A question in the geometry of situation. *Quarterly Journal of Mathematics*, 1:79, 1857.

[26] E. Welzl. LP with small d – algorithms and applications. Manuscript, 1994.