Prunning can solve from facility location to visibility problems

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

Abstract

We present several results concerning minimax facility location, geometric problems with convex polygons defined by intersection of halfplanes, and optimization visibility problems, both in two and three dimensions. These problems, in spite of the fact that they appear to be of very different kind, are optimally solved in linear time by using a prune-and-search strategy that appears to be really powerful.

1 Introduction

In his solution to the linear programming problem [13, 14], Megiddo proposed a linear time algorithm that was independently developed and improved by Dyer [6, 7], and whose scheme was proved to be useful also in solving the euclidean 1-center problem.

In the linear programming problem, a linear objective function is given, together with a set of linear restrictions for the solution point. Megiddo's algorithm proceeds by recursively eliminating a fixed fraction of the restrictions at each step, until the size of the input is small enough to be solved in a straight way. At each step, the constraints that are redundant and can be discarded are determined by an oracle, a linear time algorithm that, essentially, solves the problem in one less dimension.

In the minimum spanning center problem, we look for a point that minimizes the maximum distance to a set of given points. Again, the algorithm that solves this problem proceeds by recursively eliminating a fixed fraction of the input points at each step, and the discarding process is ruled by an oracle that solves the problem in one less dimension.

Our work shows that this prune-and-search strategy can be applied with success in many different situations, obtaining linear time algorithms that optimally solve several kind of problems.

First of all, linear restrictions and sets of points can be handled together in a combined algorithm that solves a very natural minimax facility location problem with constraints:

Problem 1 *Find the minimum spanning circle of a set of n points in the plane, with center constrained to satisfy a set of m linear restrictions.*

While much had be done on the unconstrained version of the classic 1-center problem [15, 16], little had 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 had been done for the 2-dimensional problem where the center is constrained to lie in a given simple or convex polygon [5]. In our work, we have solved several other constrained versions in two and three dimensions [11]. As an example of a three-dimensional result, consider the following:

Problem 2 *Find the smallest spherical cap enclosing a set of n points on a halfsphere.*

or its dual:

Problem 3 *Find the largest spherical cap contained in a spherical convex polygon defined by n spherical halfspaces.*

[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 Projecte UPC PR9410.

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

But the fact that Megiddo used this technique to solve a facility location problem does not mean that only these problems can be solved in that way. We have also obtained good results for a large range of geometrical problems related to polygons given as an intersection of (possibly redundant) halfplanes. Such as the followings:

Problem 4 *Find the minimum distance from a given point to a polygon defined as an intersection of halfplanes.*

Problem 5 *Find the largest circle enclosed in an intersection of halfspaces.*

Problem 6 *Find the four common tangents of two disjoint polygons defined by intersection of halfplanes.*

Notice that all these polygonal problems could be solved by reconstructing the polygons (i.e. by obtaining the ordered list of their vertices) from the halfplanes information. But this would give rise to $\Omega(n \log n)$ time algorithms, while ours have complexity $\Theta(n)$. Several more examples of this kind of problems, as well as the details of the algorithms and proofs can be found in [17].

Finally, we have also solved some optimization visibility problems with a similar technique. Given a viewpoint v and a bi- or three-dimensional object O, the vision angle of O from v is defined as the angle of the vision cone determined by O, with apex in v. Bose et al. and Hurtado solved in [4, 9, 10] several problems concerning the optimization of the vision angle of an object, or through given obstacles, when the viewpoint is allowed to be placed in a given trajectory or region of the plane. We have studied similar problems in two and three dimensions, when the objects to be seen, the obstacles to the vision, as well as the feasibility region for the viewpoint, are given as a set of linear restrictions. Optimal algorithms can be obtained to solve, for example, the following problems (further examples can be found in [12, 17]):

Problem 7 *Find the point of the plane that sees a given segment with maximum aperture angle and satisfies a given set of linear constraints.*

Problem 8 *Find the point of the plane that, while satisfying a given set of linear restrictions, is able to see with maximum vision angle through two polygonal obstacles defined by intersection of halfplanes.*

Problem 9 *Compute the three-dimensional vision angle of a polyhedron from a given external viewpoint.*

We have studied several problems concerning three apparently different topics: minimax facility location, geometry of polygons, and visibility optimization. All of them can be optimally solved using similar strategies.

2 The technique

We present a simple and direct algorithm as an example of how the technique works. It follows a prune-and-search strategy inspired in Megiddo's algorithm for finding the minimum spanning center without constraints (and for solving the linear programming problem) in [13]. The strategy of Megiddo in [13, 14] was independently studied and improved by Dyer in [6, 7] and has been applied with success by Bhattacharya et al. in [1, 2] to obtain optimal algorithms for the intersection radius of sets of lines, segments and convex polygons. The basic idea of the algorithm is the following:

1. Solve the problem in one less dimension (that is, in Megiddo's case, with the constraint of having the center of the minimum spanning circle lying on a given line), if such a solution exists. In any case, determine which side (halfplane) of the line the solution to the original problem belongs to.
2. Applying these results, recursively reduce the size of the original problem.

Theorem 1 *The minimum spanning circle of a set of n points in the plane, with center constrained to satisfy a set of m linear constraints $a_i x + b_i y + c_i \geq 0$, $i = 1, \ldots, m$, can be found in $\Theta(n + m)$ time.*

Proof: Suppose we have an oracle, to be described later, which is able to decide in linear time, given a line s, in which side of s lies the center of the solution circle. The following algorithm then solves the problem:

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 so obtained.

(a) Form the set V 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 obtained direction as horizontal (by a coordinates change) and form the set H of all the horizontal bisectors.

(c) Finally, pair each negative slope bisector with a positive slope one. In this way, each pair of non vertical and non horizontal bisectors will intersect in a point c_i (Figure 1).

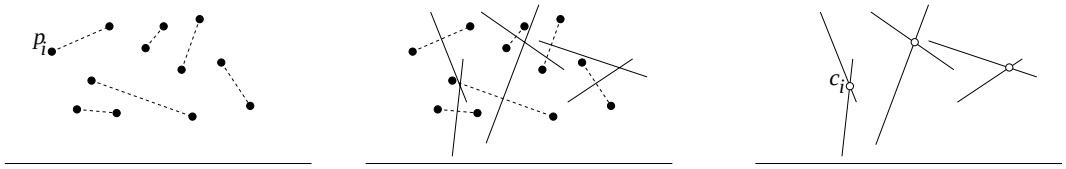

Figure 1: How to obtain the points c_i from the p_i.

2. Consider the halfplanes determined by the linear constraints and pair them up. Always pair an upper halfplane with another non parallel upper hafplane, and a lower halfplane with another non parallel lower halfplane (when two upper or two lower halfplanes are parallel, one of them is redundant and can be discarded). Then consider the intersection points d_i of the pairs of lines just formed.

3. Compute the median value y_m of the y-coordinates of all the points c_i and d_i, together with the horizontal bisectors, and apply the oracle to the line $y = y_m$, determining whether the solution to the problem lies on, above ot below the line. If the solution lies on the line $y = y_m$, we are done. Else, the solution lies in one of the two open halfplanes.

4. This allows to discard a point p_i for each horizontal bisector belonging to the halfplane opposite to the solution. Suppose that the subroutine determines that the solution lies below the line $y = y_m$. Then for each horizontal bisector lying above $y = y_m$, 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 can not determine it (Figure 2).

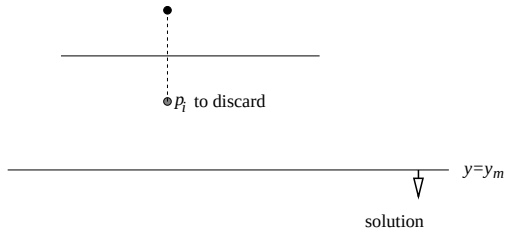

Figure 2: How to discard a point p_i for each horizontal bisector lying in the upper halfplane.

5. Then compute the median value x_m of the x-coordinates of all the vertical bisectors together with all the points c_i and d_i belonging to the halfplane opposite to the solution (the upper one). Apply now the oracle to the line $x = x_m$. If the solution lies on the line, we are done. Else, suppose that the oracle detects that the solution lies to the right of the line $x = x_m$. This determines the quadrant that contains the center of the solution circle (in our example, the lower right one).

6. At that point, it is possible to discard a point p_i for each vertical bisector lying in the left halfplane (opposite to the solution) and a point p_i for each intersection point c_i belonging to the upper left quadrant (opposite to the solution), together with a constraint for each point d_i belonging to the same upper left quadrant.

   (a) The discarding process of points p_i corresponding to vertical bisectors is analogous to that in point 4: for each vertical bisector lying to the left of the halfplane that contains the solution, one of the two points p_i that define the bisector is redundant, namely the right one, for it is certainly

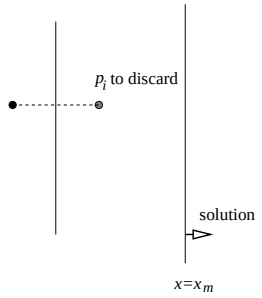

Figure 3: How to discard a point p_i for each vertical bisector lying in the left halfplane.

placed closer than the other to the center of the solution circle and hence can not determine it (Figure 3).

(b) Each point c_i belonging to the upper left quadrant (the opposite to the solution one) is the intersection of two bisectors. The positive slope one does not intersect the solution quadrant (the lower right one). This means that one of the two points p_i that determined this bisector (the lower right one) is closer that the other to the center of the solution circle, and cannot determine it (see Figure 4).

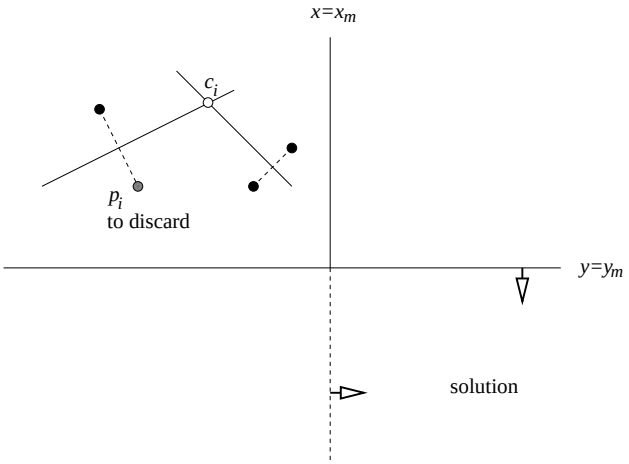

Figure 4: How to discard a point p_i for each c_i lying in the upper left quadrant.

(c) Given a pair of constraints whose intersection point d_i belong to the upper left quadrant, there are three possible situations:

- The lines determining the two halfplanes have positive slope: in that case, both constraints are irrelevant for the solution of the problem and can be discarded (Figure 5), for they can not intersect the quadrant that contains the solution point.

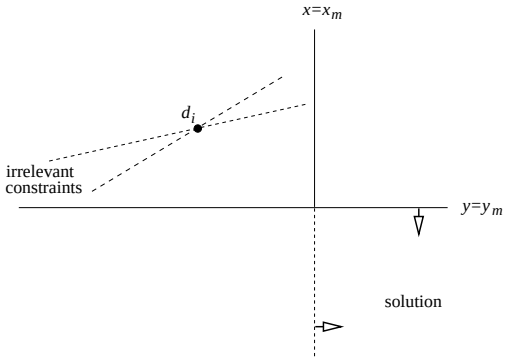

Figure 5: Discarding two positive slope constraints.

- One of the lines determining the two halfplanes has positive slope: in this case, the associated constraint is irrelevant and can be discarded for the same reason as above (Figure 6).

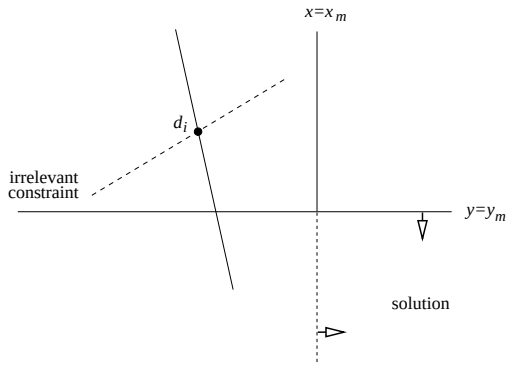

Figure 6: Discarding one positive slope constraint.

- Both lines have negative slope. Then, two cases must be considered (Figure 7):
  - If both of them are lower halfplanes, then the line having greater slope corresponds to an irrelevant constraint to the problem, and can be discarded.
  - If both of them are upper halfplanes, then the line having smaller slope corresponds to an irrelevant constraint to the problem, and can be discarded.

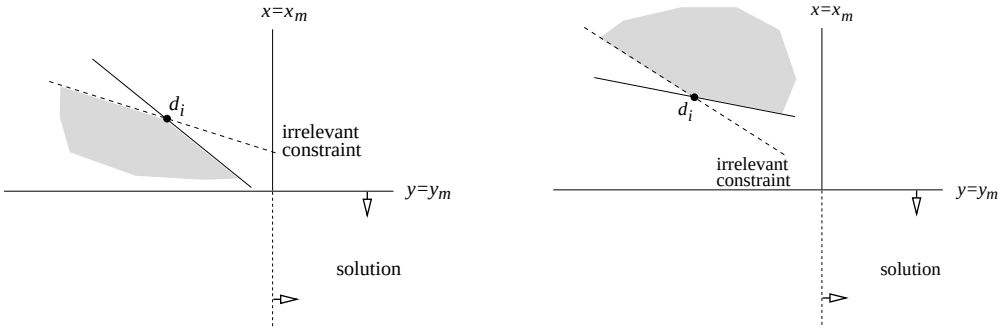

Figure 7: Discarding one negative slope constraint.

  - The case in which one of them is a lower halfplane and the other one is an upper halfplane has been eliminated when pairing them.

In this way, we discard a fixed fraction of the input elements, either points or constraints, before applying again the algorithm to the reduced input. More precisely, one point p_i is discarded for each horizontal bisector lying in the upper halfplane, one is discarded for each vertical bisector lying in the left halfplane, and one for each intersection point c_i lying in the upper left quadrant, while one constraint is discarded for each pair of parallel restrictions, and one for each intersection point d_i lying in the upper left quadrant.

Let h, v and c respectively be the number of horizontal bisectors, vertical bisectors and intersection points c_i, obtained from the n points p_i. The following equality holds:

$$n = 2h + 2v + 4c.$$

Consider now the m linear restrictions, and let p and d respectively be the number of parallel pairs of restrictions and the number of intersection points d_i. The following equality holds:

$$m = 2(p + d).$$

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

$$h_a + c_a + d_a \geq \tfrac{1}{2}(h + c + d),$$
$$v_l + c_{al} + d_{al} \geq \tfrac{1}{2}(v + c_a + d_a);$$

where k_a (k_l) represents the number of elements of kind k that lie *above (left)* to the line $y = y_m$ ($x = x_m$) and k_{al} represents the number of elements of kind k that lie above-and-left, that is, in the upper left quadrant. With this notation, the number of input elements, either points or restrictions, that are eliminated from the input is

$$p + h_a + v_l + c_{al} + d_{al}.$$

A simple computation gives a lower bound for the number of discarded elements:

$$\begin{aligned} p + h_a + v_l + c_{al} + d_{al} &\geq p + h_a + \tfrac{1}{2}(v + c_a + d_a) \\ &\geq p + \tfrac{1}{2}v + \tfrac{1}{2}(h_a + c_a + d_a) \\ &\geq p + \tfrac{1}{2}v + \tfrac{1}{4}(h + c + d) \\ &\geq \tfrac{1}{4}(p + d) + \tfrac{1}{4}(h + v + c) \\ &\geq \tfrac{1}{8}m + \tfrac{1}{16}n \\ &\geq \tfrac{1}{16}(n + m). \end{aligned}$$

Hence, at least a fixed fraction of $\frac{1}{16}$ of the initial points is discarded. Supposing that the oracle runs in linear time, the cost of each application is linear on the size of the input, because it only requires to compute some median values [3, 8]. So, the total complexity of the algorithm is

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

We will now explain how to obtain a linear time algorithm to be used as an oracle. For each line s, the goal is to find in linear time the point of s that minimizes the radius of the minimum circle that covers $p_1, \ldots, p_n$ and satisfies the linear restrictions, when there exists such a point, and to decide in any case to which side of s lies the center of the circle which is the solution of the general problem. This is the procedure:

1. Detect if the line s and the constraint polygon intersect, and obtain the information of which side of the line lies the polygon in case they do not intersect:
   (a) Start intersecting line s with all the halfplanes determined by lines parallel to s. Notice that, in fact, only two of these halfplanes can be relevant in determining the constraint polygon. If the intersection of s and these two halfplanes is empty, the constraint polygon itself is empty. If only one of the two intersections is empty, it is clearly determined which side of s lies the constraint polygon. If none of the two intersections is empty, proceed to next step.
   (b) Intersect 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 if s intersects the associated polygonal region in an edge or in a vertex. Proceed the same way with the lower halfplanes.
   (c) If the two rays are disjoint, the intersection of s and the constraint polygon is empty. The intersection of the at most four halfplanes that determine the two rays shows which side of s lies the constraint polygon (in some cases, it can even show the emptiness of the polygon).
   (d) If the two rays have a common segment, then the line s intersects the constraint polygon, and the intersection segment is found.
2. In the line s and the constraint polygon do intersect, then the s-constrained problem must be solved, that is, the point of s that is the center of the minimum spanning circle of the set of points must be found.
3. Compute the solution constrained to the intersection segment of s and the constraint polygon, that can be either the s-constrained solution, if it belongs to the segment, either one of the two endpoints of the segment.

   In any case, find the point of the set that determines the maximum distance to the optimum over the segment. This point determines the halfplane where the solution to the original problem lies (Figure 8). When the points at maximum distance are several, the problem can be solved in a similar way: either they all lie in the same halfplane (the solution halfplane), either there is at least one in each halfplane, and in that case the optimum point just found is the global optimum. Only in some degenerate cases

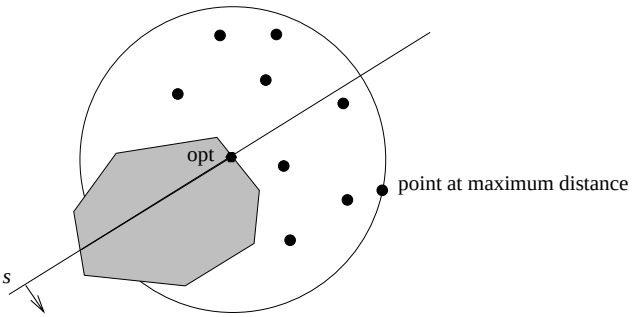

Figure 8: Detecting which side of s lies the solution.

it will happen that the segment line and the point that gives the maximum distance are aligned. When this happens, the optimum over the segment will be located at an endpoint. The way of determining the halfplane that contains the solution consists in considering the edge of the polygon trough that endpoint and determine which of its two sides allows to minimize the maximum distance.

This algorithm takes $O(n+m)$ time: step 1 can clearly 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 [13]. Finally, step 3 only requires $O(n)$ time, as it essentially consists in determining the maximum distance from the segment constrained optimum to the set of points. □

References

[1] B. K. Bhattacharya, S. Jadhav, A. Mukhopadhayay, J. M. Robert, Optimal algorithms for some smallest intersection radius problems, *Proc. of the Seventh Annual ACM Symp. on Computational Geometry*, 1991, pp. 81-88.

[2] B. K. Bhattacharya and G. T. Toussaint, On geometric algorithms that use the furthest-point Voronoi diagram, in *Computational Geometry*, ed., G. T. Toussaint, North-Holland, 1985, pp. 43-61.

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

[4] P. Bose, F. Hurtado, E. Omaa, J. Snoeyink, G. Toussaint, Some aperture-angle optimization problems, *Proc. 7th Canad. Conf. Comput. Geom.*, 1995, pp. 73-78, submitted for publication to *Algorithmica*.

[5] P. Bose, G. Toussaint, Computing the Constrained Euclidean, Geodesic and Link Center of a Simple Polygon with Applications, *Proc. of Pacific Graphics International*, Pohang, South Korea, 1996, pp. 102-112.

[6] M. E. Dyer, Linear time algorithms for two- and three-variable linear programs, *SIAM J. Comput.*, Vol.13, No.1, 1984, pp.31-45.

[7] M. E. Dyer, On a multidimensional search technique and its application to the euclidean one-center problem, *SIAM J. Comput.*, Vol.15, No.3, 1986, pp.725-738.

[8] C. A. R. Hoare, Algorithm 63 (partition) and algorithm 65 (find), *Communications of the ACM*, Vol.4, No.7, 1961, pp. 321-322.

[9] F. Hurtado, Looking through a window, *Proc. 5th Canad. Conf. Comput. Geom.*, 1993, pp. 234-239.

[10] F. Hurtado, Problemas geomtricos de visibilidad, Ph. D. thesis, U. Politcnica de Catalunya, 1993.

[11] F. Hurtado, V. Sacristán, G. Toussaint, Constrained minimax facility location, Tech. Report MA2-IR-96-0010, U. Politècnica de Catalunya, October 1996, Submitted for publication to *Studies in locational analysis*.

[12] F. Hurtado, V. Sacristán, G. Toussaint, Ángulo de apertura en la visin en dos y tres dimensiones, *Actas de los VII Ecuentros de Geometra Computacional*, pp. 15-28, 1997.

[13] N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM J. Comput.*, Vol.12, No.4, November 1983, pp. 759-776.

[14] N. Megiddo, Linear Programming in Linear Time When the Dimension Is Fixed, *Journal ACM*, Vol.31, No.1, January 1984, pp. 114-127.

[15] J.-M. Robert and G. T. Toussaint, Linear approximation of simple objects, *Computational Geometry: Theory and Applications*, Vol. 4, 1994, pp. 27-52.

[16] J. M. Robert and G. T. Toussaint, Computational geometry and facility location, *Proc. International Conference on Operations Research and Management Science*, Manila, The Philippines, Dec. 11-15, 1990, pp. B-1 to B-19.

[17] V. Sacristán, Optimizacin geomtrica y aplicaciones en visibilidad, Ph. D. thesis, U. Politcnica de Catalunya, May 1997.