Obnoxious Facility Location: Complete Service with Minimal Harm*

Boaz Ben-Moshe, Matthew J. Katz, Michael Segal

Department of Mathematics and Computer Science
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

May 9, 1999

Abstract

We present efficient algorithms for several instances of the following facility location problem. (Facilities and demand sites are represented as points in the plane.) Place k obnoxious facilities, with respect to n given demand sites and m given regions, where the goal is to *maximize* the minimal distance between a demand site and a facility, under the constraint that each of the regions must contain at least one facility.

We also present an efficient solution to the following planar problem that arises as a subproblem. Given n transmitters, each of range r, construct a compact data structure that supports coverage queries, i.e., determine whether a query polygonal/rectangular region is fully covered by the transmitters.

1 Introduction

We consider several instances of the following generally-stated problem, which has several applications in, e.g., urban, industrial and military task planning.

Placing Obnoxious Facilities: Let P be a set of n points in the plane (called *demand* points), and let $\mathcal{R}$ be a set of m, $m \leq n$, regions in the plane (called *neighborhoods*). Let k be a positive integer (k is the number of facilities, e.g., garbage dumps, to be placed). Find k sites $c_1, \ldots, c_k$ for the k facilities, such that (i) $C = \{c_1, \ldots, c_k\}$ is a *piercing set* for $\mathcal{R}$, that is, each of the neighborhoods in $\mathcal{R}$ is served by at least one facility that is located in

*Work by M. Katz has been supported by the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities.

the neighborhood. (ii) The minimal distance between a demand point in P and a site in C is maximized.

This problem belongs to a class of problems that deal with the location of facilities, both desirable and undesirable, under various conditions. This class of problems occupies researches in operations research, especially in the field of *location science*. Some of the more geometric problems have also been treated in the computational geometry literature. In a typical facility location problem, we need to find a location for some facility, with respect to a given set of demand sites. Both the demand sites and the facility are represented as points in the plane. The chosen location should satisfy a given set of conditions, e.g., minimize the maximal distance to a demand site (known as the 1-center problem). Our problem is somewhat more complex (though definitely realistic) than most of the related problems. There are several facilities, and the desired locations must satisfy both a piercing condition and a distance optimization condition.

Most of the facility location problems described in the literature are concerned with finding locations for "desirable" facilities. The goal there is to *minimize* a distance function between the facilities (e.g., supermarkets) and the demand sites (e.g., customers). However, just as important are the problems dealing with the location of "undesirable" or obnoxious facilities (e.g., garbage dumps, dangerous chemical factories or nuclear power plants). In these problems, instead of, e.g., minimizing the maximal distance between the facility and a demand point, we want to maximize the minimal distance between the facility and a demand point. Notice that if the domain of possible locations for the facility is the entire plane, then the problem becomes impractical and not interesting. Therefore, some constraints on the location of the facility should be specified, e.g, forcing it to lie in some bounded region R.

In this paper we assume that the regions in $\mathcal{R}$ are unit axis-parallel squares (actually, translated copies of some axis-parallel rectangle). We consider both the L_∞ case and the L_2 case. In the L_∞ case (resp. L_2 case), we seek the maximal value d^* for which there exist k locations such that (i) none of the locations lies in the interior of a square of edge length $2d^*$ (resp. in a disk of radius d^*) centered at a demand point, and (ii) for each of the squares $R \in \mathcal{R}$, at least one of the k locations lies in R (in other words, the k locations consist of a k-piercing set for the set of squares $\mathcal{R}$). We present efficient solutions for $k = 1, 2$, both

under the L_∞ metric and under the Euclidean metric. Our solutions consist of solutions to the corresponding decision problems to which we apply either the sorted matrices technique of Frederickson and Johnson [10] or the parametric searching technique of Megiddo [14] to obtain the maximal value d^*. For $k \geq 3$ we show two examples which in some sense imply that there is not much hope for a subquadratic solution for $k = 3$ or for any other value of k greater than 3. In addition, we also present a solution to the weighted version of the simplest problem ($k = 1$ under L_∞), and present a lower bound for its corresponding decision problem.

Consider for example the decision version of the problem in which we need to place two facilities, under the Euclidean metric. In this problem, we need to consider all possible solutions to the 2-piercing problem for $\mathcal{R}$. For each such solution p_1, p_2, we must check whether both p_1 and p_2 are not covered by the n disks of radius d centered at the points of P. By adapting a lemma of Katz et al. [11], and employing, in a sophisticated way, a technique due to Sharir [18] that resembles searching in monotone matrices, we transform the problem into the following *reception problem* (with some additional issues that require attention): Given m transmitters, each of range d (e.g., the transmitters of some communication system), construct a compact data structure that supports coverage queries, i.e., determine whether a query rectangular region is fully covered by the transmitters. In other words, preprocess a set of m congruent disks, so that, given a query rectangle R, one can quickly determine whether R is fully contained in the union of the disks. We present a simple, though non-trivial, solution to this problem, and to the problem where the query regions are constant-size polygons instead of rectangles. We are not aware of any previous solution to these problem. Our solution uses the Voronoi diagram of the centers of the disks and data structures for orthogonal (alternatively, simplex) range searching, and vertical (alternatively, general) ray-shooting among line segments. The construction time is nearly linear for both rectangular queries and polygonal queries, the space required is linear, and the query cost is $O(\log n)$ for rectangular queries and roughly $O(n^{1/2})$ for polygonal queries.

The problem in which the n demand points lie in a simple polygon R with at most n vertices, and one needs to place a single facility in R, such that the minimal Euclidean distance between a demand point and the facility is maximized, was solved in $O(n \log n)$

time by Bhattacharya and Elgindy [4]. If the polygon R is a rectangle, and each of the demand points p_i is assigned a weight w_i, so that the distance between p_i and a point $q \in R$ is w_i times the L_∞ distance between them, then one can apply the $O(n \log^4 n)$ solution of Follert et al. [9]. The latter bound was improved recently to $O(n \log^2 n)$ by Bespamyatnikh et al. [3], who actually consider a slightly more general problem. In their problem each demand point p_i is assigned two weights w_i^x and w_i^y, and the distance between p_i and q is $w_i^x|p_i^x - q^x| + w_i^y|p_i^y - q^y|$. Brimberg and Mehrez [5] solve the following problem: Find k locations in the rectangle R (for k facilities), such that (i) the distance between any two locations is at least some given value d, and (ii) the minimal distance between a demand point and a facility is at least some given value r. Katz et al. [12] present improved solutions to this problem.

The paper is organized as follows. In Section 2 we present our solution to the reception (sub)problem. Section 3 deals with our main problem (placing obnoxious facilities), where we distinguish between the L_∞ case and the L_2 case.

2 The Reception Problem

We consider the following problem: Given n transmitters, each of range r, construct a compact data structure that supports coverage queries, i.e., determine whether a query (rectangular or polygonal) region is fully covered by the transmitters. In other words, preprocess a set of n congruent disks for coverage queries, i.e., determine whether a query region R is fully contained in the union of the disks.

We first present a solution for polygonal region queries, that is, we assume the query regions are simple polygons with at most c vertices, for some constant c. Then we show how the bounds for the preprocessing time and query cost can be improved if the query regions are axis-parallel rectangles.

2.1 Polygonal region queries

In this subsection we deal with the following problem: Given a set D of n unit disks in the plane, construct a compact data structure, so that, given a query polygon Q, one can quickly

determine whether Q is fully covered by the disks in D. Let P denote the set consisting of the center points of the disks in D. The main components of our data structure are (i) the Voronoi diagram VD of P and a corresponding point location data structure, (ii) a data structure for simplex range searching over a subset of the vertices of VD, and (iii) a data structure for ray shooting over a set of portions of edges of VD.

The preprocessing phase. The preprocessing phase consists of the following three steps:

1. Construct the Voronoi diagram VD of P, and the corresponding point location data structure, both in $O(n \log n)$ time [2].

2. Compute the set V' of all vertices of VD that are not covered by disks in D, by checking, for each vertex v of VD, whether the distance between v and its corresponding Voronoi sites is greater or equal to 1. Construct in $O(n^{1+\varepsilon})$ time a linear-size data structure for simplex range searching queries over V' [13].

3. Compute the set E' of the portions of the edges of VD that are not covered by disks in D. According to the claim below, the size of E' is only $O(n)$, and it can be computed in $O(n)$ time. Construct in $O(n^{1+\varepsilon})$ time a linear-size data structure for ray shooting over E' [1].

Claim 1 *The number of edge portions in E' is $O(n)$, and E' can be computed from VD in $O(n)$ time.*

Proof. Let e be an edge of VD, and let $p \in P$ be one of the two corresponding Voronoi sites. Notice that if a point on e is covered by one or more of the disks in D, then this point is necessarily covered by the disk centered at p. Thus, in order to determine the portions of e that are not covered by D, it is enough to consider the disk centered at p, ignoring all other disks. The number of such portions is clearly at most two. ∎

Answering queries. A query with a polygon Q is treated as follows. We first partition Q into a constant number of triangles. (Recall that Q has at most c vertices, for some constant c.) For each of the triangles Δ, we perform a range searching query using the simplex range searching data structure. If the answer obtained to one (or more) of these queries is positive,

i.e., one (or more) of these triangles contains points of V', then we conclude that Q is not fully covered by the disks in D, return "NO" and stop. Otherwise, we proceed as follows. For each edge $e = \overline{ab}$ of Q:

- Find the cell C_a (resp. C_b) of VD containing the endpoint a (resp. b) of e, using the point location data structure.
- Calculate the distance d_a (resp. d_b) between a (resp. b) and the point of P defining C_a (resp. C_b). If $d_a \geq 1$ (resp. $d_b \geq 1$), return "NO" and stop.
- If a and b do not lie in the same cell of VD (i.e., if $C_a \neq C_b$), then perform a ray shooting query with the ray emanating from a and containing e, using the ray shooting data structure. If the answer obtained is positive and the hitting point is on e, return "NO" and stop.

If we have reached this point, we may conclude that Q is fully covered by the disks in D and return "YES".

The algorithm for answering a query is quite simple, however, it is not obvious that it is correct. In the following theorem we prove that it is indeed correct, that is, it returns "YES" if the query polygon Q is fully covered by the disks in D, and "NO" otherwise.

Theorem 2 *The algorithm is correct; it returns "YES" if the query polygon Q is fully covered by the disks in D and "NO" otherwise.*

Proof. Consider a point ψ in Q, that lies in the Voronoi cell of p. ψ is covered by D if and only if it is covered by the disk centered at p. Moreover, the disk centered at p is "responsible" for covering the region $Q \cap C_p$, where C_p is the Voronoi cell of p. It is therefore enough to verify that the point in $Q \cap C_p$ that is the furthest from p, is at distance less than 1 from p. This point, however, is either a vertex of the boundary of C_p, or an intersection point between an edge of the boundary of C_p and the boundary of Q, or a vertex of Q. The range searching queries performed in the first part of the algorithm take care of points of the first kind; by the answers obtained, we immediately know whether there exist vertices of VD that lie in Q and are not covered. The second part of the algorithm takes care of

points of the two other kinds, by checking whether the boundary of Q is fully covered by D. Let e be an edge of the boundary of Q. We first verify that both its endpoints are covered. If both endpoints lie in the same Voronoi cell and they are both covered, then clearly the entire edge is covered. However, if the endpoints lie in different Voronoi cells, it is possible that portions of the interior of e are not covered. However, this can happen if and only if e intersects a segment of E', and this will be detected by the ray shooting query performed for e. ∎

Concerning the complexity of the above algorithm, the preprocessing time is $O(n^{1+\varepsilon})$, which is the time required to construct the range searching and ray shooting data structures [1, 13], the space complexity is $O(n)$, and the query cost, determined by the range searching and ray shooting queries, is $O(n^{\frac{1}{2}+\varepsilon})$. We thus obtain:

Theorem 3 *Let D be a set of n unit disks in the plane. It is possible to preprocess D in time $O(n^{1+\varepsilon})$, into a linear-size data structure, such that determining whether a constant-size query polygon Q is fully covered by the disks in D can be done in time $O(n^{\frac{1}{2}+\varepsilon})$.*

Remark 1: As known, the n^{ε} factors in the theorem above can be replaced by slightly smaller factors. Also the standard storage/query tradeoff can be applied to construct a data structure of size $n \leq m \leq n^2$ with query time $O(n^{1+\varepsilon}/m^{1/2})$. In particular, if the query polygons are of linear size, then we can construct a data structure of size and query cost roughly $O(n^{4/3})$.

2.2 Rectangular region queries

In this subsection we consider the special case where the query regions are axis-parallel rectangles. This is the case to which we refer in Section 3. For this case, we can obtain better bounds for the preprocessing time and query cost, by replacing the general range searching and ray shooting data structures with standard specialized data structures. More precisely, we use a data structure for orthogonal range searching over the set V' [6], and a data structure for horizontal/vertical ray shooting over the set E' [2]. We thus obtain:

Theorem 4 *Let D be a set of n unit disks in the plane. It is possible to preprocess D in time $O(n \log n)$, into a linear-size data structure, such that determining whether a query rectangle Q is fully covered by the disks in D can be done in time $O(\log n)$.*

Remark 2: The bounds for the somewhat simpler problem, where D is a set of unit axis-parallel squares instead of unit disks remain the same.

3 Obnoxious Facilities

In this section we solve several instances of the *Placing Obnoxious Facilities* problem stated in the Introduction. In all the problem instances that we consider, the set of regions $\mathcal{R}$ consists of unit axis-parallel squares. We consider the two problems in which the number of facilities k is one or two, respectively, under the L_∞ metric as well as under the Euclidean metric. For $k \geq 3$ we show two examples which in some sense imply that there is not much hope for a subquadratic solution for $k = 3$ or for any other value of k greater than 3. Obviously, if the set $\mathcal{R}$ is not k-pierceable, then there is no solution. Therefore, we assume that $\mathcal{R}$ is k-pierceable. We can check whether $\mathcal{R}$ is k-pierceable, $1 \leq k \leq 2$, in $O(m)$ time [19].

When solving a problem, we first present a solution to the corresponding decision problem, and then apply the sorted matrices technique of Frederickson and Johnson [10] or the parametric searching technique of Megiddo [14] to obtain a solution to the original problem. That is, we first solve a problem of the form: Determine whether there exist k locations, such that, for each of the m unit squares $r \in \mathcal{R}$, at least one of these locations is in r, and the minimal distance between the n demand points and these locations is at least d, where d is a parameter of the problem. We then apply one of the above techniques to obtain the maximal value d^* for which the decision problem returns a positive answer.

3.1 The L_∞ case

3.1.1 $k = 1$

In this problem we wish to place only one facility. This problem is relatively easy, and we present a solution to the weighted version of the problem as well.

In the non-weighted version of the problem, we simply compute the L_∞ Voronoi diagram of P restricted to the non-empty rectangle $R = \cap\mathcal{R}$. It is easy to see that the desired location is a vertex of the restricted diagram, so, for each of these vertices, we compute its corresponding distance, and select the vertex with largest distance. Clearly, all this can be done in $O(n \log n)$ time.

In the weighted version of the problem, each point $p_i \in P$ has two weights associated with it: $w_1(p_i)$ and $w_2(p_i)$. Solving the decision problem for d, each point $p_i \in P$ defines a *forbidden* region F_i, where the facility may not reside. F_i is the rectangle

$$\{q \in \mathbb{R}^2 \mid d_x(q, p_i) < d \cdot w_1(p_i) \ , \ d_y(q, p_i) < d \cdot w_2(p_i)\} \ .$$

Let U denote the union of the forbidden regions $F_1, \ldots, F_n$. Let $R = \cap\mathcal{R}$. R is a non-empty rectangle (since, by assumption, $\mathcal{R}$ is 1-pierceable). An allowed location for the facility exists if and only if U does not completely cover R. Since it is possible to determine whether a set of n rectangles covers another rectangle R, using a segment tree, in $O(n \log n)$ time, we obtain an $O(n \log n)$ solution to the decision problem.

Bespamyatnikh et al. [3] apply a result of Megiddo and Tamir [15] to obtain in $O(n \log^2 n)$ time the maximal value d^* for which the corresponding decision problem returns "YES". (A parallel algorithm for the decision problem is presented in [3]; it employs $O(n \log n)$ processors and computes the answer in $O(\log n)$ time.) We thus obtain:

Theorem 5 *Under the L_∞ metric and for $k = 1$, the problem can be solved in $O(n \log n)$ time, and the weighted version of the problem can be solved in $O(n \log^2 n)$ time.*

A lower bound. We obtain a lower bound of $\Omega(n \log n)$ for the decision problem (of both the weighted and non-weighted versions), by showing that even the one-dimensional version of the problem "determine whether a set of n unit squares covers another square" has a lower bound of $\Omega(n \log n)$. Consider the GAP-EXISTENCE problem: Given a set A of n real numbers $A = \{a_1, \ldots, a_n\}$, determine whether there exist two consecutive numbers in the sorted sequence obtained from A, such that the difference between them is greater than 1. Sharir and Welzl [19] observed that this problem has a lower bound of $\Omega(n \log n)$. We transform a_i, $i = 1, \ldots, n$, to the one-dimensional rectangle $[a_i, a_i + 1]$, thus obtaining a set

$\mathcal{R}$ of n rectangles. We define $R = [\min_{a_i \in A} a_i \ , \ \max_{a_i \in A} a_i]$. It is clear that R is not covered by the rectangles in $\mathcal{R}$ if and only if there exist two consecutive numbers as above.

3.1.2 $k = 2$

Assuming $\mathcal{R}$ is 2-pierceable (but not 1-pierceable), we need to determine whether there exist two points p_1, p_2, such that $\{p_1, p_2\}$ is a piercing pair for $\mathcal{R}$, and neither p_1 nor p_2 lie in the interior of U, where U is the union of the squares of edge length $2d$ centered at the points of P. Assume $\{p_1, p_2\}$ is a piercing pair for $\mathcal{R}$, then we can divide the squares in $\mathcal{R}$ into two disjoint subsets $\mathcal{R}_{p_1}$ and $\mathcal{R}_{p_2}$, such that $p_1 \in R_1 = \cap\mathcal{R}_{p_1}$ and $p_2 \in R_2 = \mathcal{R}_{p_2}$. (If some of the squares in $\mathcal{R}$ are pierced by both p_1 and p_2, then there are many ways to do this.) Therefore, we could search for a "good" piercing pair $\{p_1, p_2\}$ (i.e., a piercing pair such that both points are not in the interior of U) by considering all possible partitions of $\mathcal{R}$ into two subsets $\mathcal{R}_1, \mathcal{R}_2$ such that the rectangles $R_1 = \cap\mathcal{R}_1$ and $R_2 = \cap\mathcal{R}_2$ are non-empty. For each such partition, we would have to check for each of the corresponding two rectangles whether it contains a point that is not covered by U. However, this method is very inefficient. Fortunately, the following lemma that is taken from [11] allows us to restrict our search to a quadratic number of partitions. Denote by $X_{\mathcal{R}}$ the centers of the squares in $\mathcal{R}$, sorted by their x-coordinate (left to right), and by $Y_{\mathcal{R}}$ the centers of the squares in $\mathcal{R}$, sorted by their y-coordinate (low to high).

Lemma 6 ([11]) *If p_1 and p_2 are a piercing pair for $\mathcal{R}$, then $\mathcal{R}$ can be divided into two subsets $\mathcal{R}_1$ and $\mathcal{R}_2$, $p_1 \in \cap\mathcal{R}_1$, $p_2 \in \cap\mathcal{R}_2$, such that $\mathcal{R}_1$ can be represented as the union of two subsets $\mathcal{R}_1^x$ and $\mathcal{R}_1^y$ (not necessarily disjoint, and one of them might be empty), where the centers of squares of $\mathcal{R}_1^x$ form a consecutive subsequence of the list $X_{\mathcal{R}}$, starting from its beginning, and the centers of squares of $\mathcal{R}_1^y$ form a consecutive subsequence of $Y_{\mathcal{R}}$, starting from the list's beginning.*

According to the lemma it is enough to consider partitions in which one of the subsets is obtained by taking the i_x leftmost squares in $\mathcal{R}$ together with the i_y bottommost squares in $\mathcal{R}$, $0 \leq i_x, i_y \leq n$. (In [11] the lemma is proven under the assumption that the piercing

points p_1 and p_2 are centers of squares in $\mathcal{R}$. However, it is easy to see that the lemma is also true without this assumption.)

We further restrict our search by employing a technique, due to Sharir [18], that resembles searching in monotone matrices; for a recent refinement of this technique and applications see [7, 11]. Let M be an $(m+1) \times (m+1)$ matrix, whose rows (skipping row 0) correspond to $X_{\mathcal{R}}$ and whose columns (skipping column 0) correspond to $Y_{\mathcal{R}}$. An entry M_{xy} in the matrix is defined as follows. Let D_x be the set of squares in $\mathcal{R}$ such that the x-coordinate of their centers is smaller or equal to x, and let D_y be the set of squares in $\mathcal{R}$ such that the y-coordinate of their centers is smaller or equal to y. Let $D^l_{xy} = D_x \cup D_y$ and $D^r_{xy} = (\mathcal{R} - D^l_{xy})$, and let $R^l_{xy} = \cap D^l_{xy}$ and $R^r_{xy} = \cap D^r_{xy}$.

$$M_{xy} = \begin{cases} \text{`}YY\text{'} & \text{if } R^r_{xy} \not\subseteq U \text{ and } R^l_{xy} \not\subseteq U \\ \text{`}YN\text{'} & \text{if } R^r_{xy} \not\subseteq U \text{ but } R^l_{xy} \subseteq U \\ \text{`}NY\text{'} & \text{if } R^r_{xy} \subseteq U \text{ but } R^l_{xy} \not\subseteq U \\ \text{`}NN\text{'} & \text{if } R^r_{xy} \subseteq U \text{ and } R^l_{xy} \subseteq U \end{cases}$$

where we assume of course that the empty set is contained in U. It follows that the answer to our decision problem is "YES" if and only if M contains a 'YY' entry.

In order to apply Sharir's technique the lines and columns of M^1 must be non-decreasing (assuming 'Y' > 'N'), and the lines and columns of M^2 must be non-increasing, where M^i is the matrix obtained from M by picking from each entry only the i'th letter, $i = 1, 2$. In our case this property clearly holds, since, for example, if for some x_0 and y_0, $M^1_{x_0,y_0}$ ='Y', then for any $x' \geq x_0$ and $y' \geq y_0$, $M^1_{x',y'}$='Y'. Thus we can determine whether the *implicit* matrix M contains an entry 'YY' by inspecting only $O(m)$ entries in M, advancing along a *connected* path within M [7]. For each entry along this path, we need to determine whether R^z_{xy} is fully covered by U, $z \in \{l, r\}$. This can be done in $O(\log n)$ time by dynamically maintaining the intersection $\cap D^z_{xy}$, and by utilizing the data structure of Section 2.2 (see Remark 2 there). Thus in $O(n \log n)$ time we can determine whether M contains a 'YY' entry.

Optimization. We show how to find the smallest value d^* for which the matrix M above contains a 'YY' entry. It is easy to verify that d^* is either (i) half the difference between the x-coordinates (alternatively, y-coordinates) of a pair of points in P, or (ii) the horizontal (respectively, vertical) distance between a vertical (respectively, horizontal) edge of a square

in $\mathcal{R}$ and a point in P. All these potential values can be represented by four (implicit) sorted matrices; two matrices for each axis.

We define the two sorted matrices corresponding to the x-axis. Let L_x be the sorted list consisting of the x-coordinates of the points in P and the x-coordinates of the vertical edges of the squares in $\mathcal{R}$. Entry (i, j) in matrix M_1 stores the value $(x_j - x_i)/2$, where x_i, x_j are the i-th and j-th elements in L_x, and entry (i, j) in the matrix M_2 stores the value $(x_j - x_i)$. Clearly, these matrices contain several values that do not belong to the set of potential solutions, but this does not affect the running time. We define the two sorted matrices M_3 and M_4 corresponding to the y-axis analogously.

We now apply the Frederickson and Johnson technique [10] to each of the four matrices in order to find the smallest value in these matrices for which the decision algorithm returns "Yes". We thus obtain:

Theorem 7 *Under the L_∞ metric and for $k = 2$, the problem can be solved in $O(n \log^2 n)$ time. The corresponding decision problem can be solved in $O(n \log n)$ time.*

3.1.3 $k \geq 3$

The largest integer l for which there exists a linear-time algorithm that determines whether $\mathcal{R}$ is l-pierceable (and, if yes, computes an l-piercing set) is 3 [16, 17, 19]. This fact caused us to believe that Lemma 6 is also true for a piercing triplet. That is, if p_1, p_2, p_3 is a piercing triplet for $\mathcal{R}$, then $\mathcal{R}$ can be divided into three subsets, such that one of them can be represented as the union of two subsets as in Lemma 6. Unfortunately, we came up with a counterexample that is depicted in Figure 1(a). All piercing triplets for the 8 squares of Figure 1(a) must consist of a point from each of the three black rectangles, and it is easy to verify that we cannot divide the set of squares as required.

Figure 1(a) is not, however, a counterexample for $k \geq 4$. Since by adding squares and increasing the number of piercing points, the desired property might reappear. For completeness, we also provide a counterexample for $k \geq 4$ depicted in Figure 1(b). Assume that each of the four pairs in the figure lies near the corresponding corner of some huge square region s. Then we can add any number of squares around the middle of s and increase the number of piercing points accordingly, without ruining the counterexample.

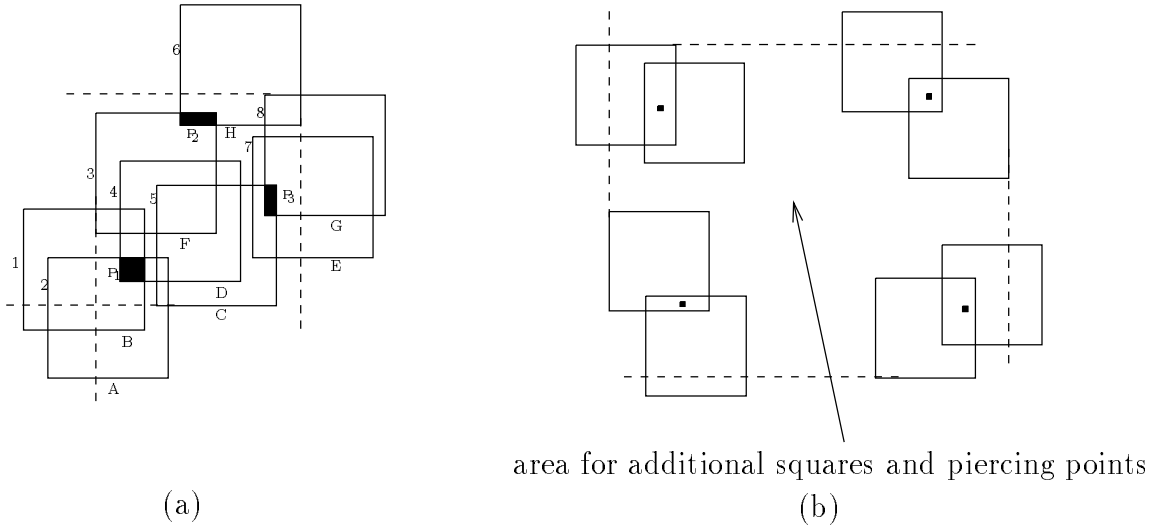

Figure 1: Lemma 6 is false for $k \geq 3$

The above counterexamples provide, in some sense, evidence that it is apparently impossible to obtain subquadratic solutions for $k \geq 3$.

3.2 The L_2 case

3.2.1 k=1

As in the L_∞ case, we can solve this problem in $O(n \log n)$ time by computing the Voronoi diagram of P restricted to the non-empty rectangle $R = \cap\mathcal{R}$. We thus obtain:

Theorem 8 *Under the L_2 metric and for $k = 1$, the problem can be solved in $O(n \log n)$ time.*

3.2.2 k=2

The decision algorithm is identical to the decision algorithm in the L_∞ case (Section 3.1.2), except for the component that deals with queries of the form: determine whether a query rectangle is fully contained in U. Now U is the union of n disks, each of radius d, so we use our solution to the Reception Problem with axis-parallel rectangular queries (Section 2.2). We obtain an $O(n \log n)$-time decision algorithm.

We apply the parametric searching technique to obtain in $O(n \operatorname{polylog} n)$ time the maximal value d^* for which the decision problem returns "YES"; we omit the details from this version. (The problematic set of potential values is the one consisting of the radii of all circles that pass through three points in P. This set is also one of the sets of potential values in the planar 2-center problem, solved by Sharir [18] with parametric searching in $O(n \operatorname{polylog} n)$ time. Sharir's solution has been improved by Eppstein to randomized expected time $O(n \log^2 n)$ [8].) We thus obtain:

Theorem 9 *Under the L_2 metric and for $k = 2$, the problem can be solved in $O(n \operatorname{polylog} n)$ time. The corresponding decision problems can be solved in $O(n \log n)$ time.*

Conclusion

We have considered several instances of the *Placing Obnoxious Facilities* problem, to which we have presented efficient solutions. A natural question that arises is is it possible to devise subquadratic solutions for instances with values of k greater than 2.

The dual problem, where the facilities are "friendly" or desirable, is also interesting. For $k = 2$ and under the L_∞ (resp. L_2) metric, this problem (actually, its corresponding decision problem) becomes: Find a pair of points which serves as a piercing pair for both the set $\mathcal{R}$ of unit squares and the set of squares (resp. disks) of radius d centered at the demand points. In the L_∞ case, this can be done by simply finding in $O(n)$ time a piercing pair for the union of the two sets of squares. In the Euclidean case, we would like to employ Lemma 6 in a sophisticated way, as we did in Section 3.1.2. Here, we need to determine, for each pair of rectangles that is generated, whether the set of disks can be pierced by choosing a point in each of the two rectangles. This can be done apparently by adopting Sharir's solution to the (decision problem of the) planar 2-center problem [18] (see also [8]). We are still working on the details.

References

[1] P.K. Agarwal and J. Matoušek "Ray shooting and parametric search" *SIAM J. Computing* 22 (1993), 794–806.

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

[3] S. Bespamyatnikh, K. Kedem and M. Segal "Optimal facility location under various distance functions" Tech. Report 98-07, Dept. of Math and Comp. Science, Ben-Gurion University.

[4] B. Bhattacharya and H. Elgindy "An efficient algorithm for an intersection problem and an application" Tech. Report 86-25, Dept. of Comp. and Inform. Sci., University of Pennsylvania, 1986.

[5] J. Brimberg and A. Mehrez "Multi-facility location using a maximin criterion and rectangular distances" *Location Science* 2 (1994), 11–19.

[6] B. Chazelle "Filtering search: a new approach to query-answering" *SIAM J. Computing* 15 (1986), 703–724.

[7] O. Devillers and M. Katz "Optimal line bipartitions of point sets" *Int. J. Comput. Geom. and Appls*, to appear.

[8] D. Eppstein "Faster construction of planar two-centers" in *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pp. 131–138, 1997.

[9] F. Follert, E. Schömer and J. Sellen "Subquadratic algorithms for the weighted maximin facility location problem" in *Proc. 7th Canad. Conf. Comput. Geom.*, pp. 1–6, 1995.

[10] G.N. Frederickson and D.B. Johnson "Generalized selection and ranking: sorted matrices" *SIAM J. Computing* 13 (1984), 14–30.

[11] M.J. Katz, K. Kedem and M. Segal "Constrained Square-Center Problems" in *6th Scandinavian Workshop on Algorithm Theory*, Lecture Notes in Computer Science 1432, pp. 95–106, 1998.

[12] M.J. Katz, K. Kedem and M. Segal "Improved algorithms for placing undesirable facilities" manuscript.

[13] J. Matoušek "Efficient partition trees" *Discrete Comput. Geom.* 8 (1992), 315–334.

[14] N. Megiddo "Applying parallel computation algorithms in the design of serial algorithms" *Journal of ACM* 30 (1983), 852–865.

[15] N. Megiddo and A. Tamir "New results on the complexity of p-center problems" *SIAM J. Comput.* 12(4) (1983), 751–758.

[16] D. Nussbaum "Rectilinear p-Piercing Problems" in *Proc. 1997 International Symposium on Symbolic and Algebraic Computation*, pp. 316–323, 1997.

[17] M. Segal "On the piercing of axis-parallel rectangles and rings" in *Proc. 5th European Symp. on Algorithms*, Lecture Notes in Computer Science (1284), pp. 430–442, 1997.

[18] M. Sharir "A near-linear algorithm for the Planar 2-center problem" *Discrete Comput. Geom.* 18 (1997), 125–134.

[19] M. Sharir and E. Welzl "Rectilinear and polygonal p-piercing and p-center problems" In *Proc. 12th ACM Symp. on Computational Geometry*, pp. 122–132, 1996.