Maximizing the number of obnoxious facilities to locate within a bounded region

Shimon Abravaya[*] Michael Segal[†]

Abstract. This paper deals with the problem of locating a maximal cardinality set of obnoxious facilities within a bounded rectangle in the plane such that their pairwise L_∞-distance as well as the L_∞-distance to a set of already placed demand sites is above a given threshold. We employ techniques and methods from computational geometry to design an optimization algorithm and an efficient 1/2-approximation algorithm for the problem, and employ the optimization algorithm to design a PTAS based on the shifting strategy [7]. As a byproduct we improve the algorithm for placing obnoxious facilities given by Katz et al. [9].

1 Introduction

An *obnoxious* facility incorporates both attractive and noxious location aspects. On one hand, it is necessary to the development of its customers due to the essential service it supplies, and hence should be located somewhat close to its customers. On the other hand, however, it poses some by-product negative external effects to the area nearby and hence better not reside within a predefined distance from each customer. General examples of obnoxious facilities are nuclear power plants, garbage dump sites, polluting industrial factories, and chemical plants.

The problem of locating one or several obnoxious facilities inside a bounded region in the plane amidst existing demand points has been extensively investigated in the literature under various constraints, distance functions, and objective criteria (for surveys see [5, 12]). Note that while many single facility versions of this problem have been strongly investigated, multifacility instances have hardly been considered. Furthermore, in most of the problems dealing with a multifacility location, the number of obnoxious facilities to be located is known in advance and the goal is to maximize the distance between them. Ben-Moshe et al. [3] deal with the problem of locating a set of obnoxious facilities with respect to a given set of demand sites and a given set of regions, where the goal is to maximize the minimal distance between a demand site and facility under the constraint that each of the given regions must contain at least one facility. Qin et al. [14] solve the problem of locating a set of obnoxious facilities amidst a set of demand sites placed within a convex polygon, so as to maximize the minimum distance between the demand sites and the unified set of demand sites and obnoxious facilities. By

[*]Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel.
[†]Department of Communication Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel. Partially supported by INTEL and REMON consortium.

employing Voronoi diagrams, they design a general method that enables solving approximately the problem when the demand sites are either points or weighted convex polygons. Tamir [15] presents subquadratic algorithms for the problem of locating two obnoxious facilities within a compact polygonal set in the plane, so as to maximize the minimum of all the weighted distances between the two new facilities and a given set of demand points placed in the compact polygonal set, and the distance between the pair of new facilities. Welch et al. [16] introduce two branch-and-bound algorithms for solving the problem of locating a set of obnoxious facilities in the plane under the maximin criterion.

In this paper we employ techniques and methods from computational geometry to solve obnoxious multifacility problems with a number of facilities not known in advance. This, in turn, emphasizes the essentiality and contribution of our paper. For instance, the problem of locating trash sites within an urban area where the goal is to locate as many as possible sites such that each site is far enough from any population concentration, motivates our research. A more specified motivation for the problem is given in Section 1.3.

1.1 Model and results

Let $\{p_1, \ldots, p_n\}$ be a set of demand points enclosed in a bounded rectangle $R \subset \mathbf{R}^2$. We define the *obnoxious multifacility location problem* as follows. Locate as many as possible obnoxious facilities within R under the constraints that the smallest L_∞-distance between each demand point and the facilities is at least a given r, and the L_∞-distance between any two facilities is at least a given d. We supply an optimization algorithm, a 1/2-approximation algorithm, and a PTAS for the problem, and a solution for its corresponding decision version where the problem is to determine whether it is possible to locate a given number of obnoxious facilities within R under the problem distance constraints. We denote by k the number of facilities in a maximal location when dealing with the optimization problem, and the given number of facilities to be located when dealing with the decision problem. Clearly, in the optimization problem, k is not known in advance, whereas in its corresponding decision problem, k is given in advance as input. Each of our solutions repeatedly locates a facility under the defined constraints and computes the resulting sub-region of R, which is valid for the next facility to locate. We denote by $N = O(n+k)$ the number of vertices on the boundary of each such sub-region of R computed throughout our algorithms. The following theorems summarize our results:

Theorem 1.1 *There is an $O\left((N/4)^k \log N \log\log N\right)$ time algorithm for the obnoxious multifacility location problem.*

Theorem 1.2 *The decision version of the obnoxious multifacility location problem, where k is known in advance and the problem is to determine whether it is possible to locate k facilities within R under the problem distance constraints, can be solved in $O\left((N/4)^{k-2} \log N\right)$ time.*

Theorem 1.3 *A 1/2-approximation for the obnoxious multifacility location problem can be computed in $O\left(\max(n \log n, k \log N \log\log N)\right)$ time.*

Theorem 1.4 *There is a Polynomial Time Approximation Scheme for the obnoxious multifacility location problem, such that the approximation algorithm with parameter l has an error ratio of at least $(1-1/l)^2$ and runs in $O\left(l^2(N/4)^{l^2}\log N\log\log N\right)$ time.*

Remark 1 To improve clearness and readability, we regard throughout this paper the special case of common thresholds r and d. However, each of our solutions solves, without further extensions, the general case where the distance between each demand point $p_i, 1 \leq i \leq n$, and the new facilities must be above a given corresponding threshold r_i.

1.2 Previous related work

In the decision version of the obnoxious multifacility location problem, the number of new facilities to be located k is known in advance. The problem is to determine whether it is possible to locate k obnoxious facilities inside R under the same constraints. Brimberg et al. [4] solve this problem under the L_∞ norm in $O(N^{2k})$ time. Katz et al. [9] deal with the same decision problem and supply $O(n\log n)$-time algorithms in cases $k=2$ and $k=3$, and an $O(N^{k-2}\log N)$ time algorithm in case $k\geq 4$. The algorithm for the case $k\geq 4$ employs a recursive pattern of locating a facility within the valid region and then computing the resulting sub-region that is valid for the next facility location. This recursive algorithm runtime results from the formula

$$T(k) = N\cdot\left(O(N)+T(k-1)\right), \tag{1}$$

where $T(3)=O(N\log N)$. One may be tempted to employ the binary search technique that repeatedly executes the decision algorithm of Katz et al. [9] in order to solve the optimization problem itself as follows. First, execute the decision algorithm with the increasing values of $k=2^i, i=0,1,2,\dots$ as long as the algorithm returns *YES*. Let j be the first value of i such that the execution of the decision algorithm with $k=2^j$ returns *NO*. Clearly, 2^{j-1} and 2^j are the lower bound and the upper bound, respectively, of the maximal number of obnoxious facilities that may reside within R under the problem constraints. Hence, continue with a binary search within $[2^{j-1},2^j]$ to find the required value. Apparently, this approach requires $O(N^{k-2}\log N\log k)$ time due to the $O(\log k)$ executions of the decision algorithm of Katz et al. [9]. However, in practice, this may not be the case. For instance, denote by $k_{\max}$ the maximal number of obnoxious facilities that may be located within R, and consider the case of $k_{\max}=2^{j-1}+1$ for some natural $j>0$. In this case, the execution of the decision algorithm outputs *YES* with $k=k_{\max}-1=2^{j-1}$, and *NO* with $k=2(k_{\max}-1)=2^j$. Note that the last execution alone requires $O(N^{k-2}\log N)$ time, which, in turn, is equal to $O(N^{2k_{\max}-4}\log N)$ time. This case specifically describes the inefficiency incurred by using this approach to solve the optimization problem, and emphasizes the need for an efficient solution.

Our paper forms a natural continuation of the work done in [9]. A closer examination of the decision algorithm for more than three facilities of Katz et al. [9] shows that it can be easily extended to solve the optimization problem itself in $O(N^{k+1})$ time, where k is the number of facilities in a maximal solution, by testing the $O(N^2)$-time recursion stop criterion defined by equation (7). The runtime of this approach is represented by the formula in (1) where $T(1) = O(N^2)$ is the runtime complexity of the stop criterion. This criterion is explicitly specified in the description of our optimization algorithm and extensively used by each of our algorithms. Based on this solution, we design an optimization algorithm and a 1/2-approximation algorithm for the problem as stated in Theorem 1.1 and Theorem 1.3, respectively. Moreover, the runtime of our optimization algorithm is represented by the formula

$$T(k) = N/4 \cdot (O(\log N \log\log N) + T(k-1))\,, \tag{2}$$

where $T(1) = O((N/4)\log N \log\log N)$. Clearly, equation (2) provides a faster solution than equation (1), which emphasizes the practical efficiency of our solution compared to the solution given in [9] with the above required extension. Note that our solution for the optimization problem consists of two major improvements to the decision algorithm locating more than three facilities by Katz et al. [9], and these improvements can be easily implemented in the decision algorithm itself. This yields an $O((N/4)^{k-2}\log N)$ time solution for the decision problem, such that its runtime is represented by the formula

$$T(k) = N/4 \cdot (O(\log N \log\log N) + T(k-1))\,, \tag{3}$$

where $T(3) = O(N \log N)$. As with the solution for the optimization problem, equation (3) provides a faster solution than equation (1), which emphasizes the practical efficiency of our solution for the decision problem compared to the solution presented in [9].

The *dispersion* and *packing* problems deal with locating a set of objects in a region containing obstacles such that no two objects intersect. While the dispersion problem demands that the center of each object will reside within the given region, the packing problem demands that each object will be fully contained in this region. These problems are some of the classic problems in mathematics and computer science and have been studied extensively under the L_∞ and the L_2 norms. Baur and Fekete [1] present an $O(n^{38})$ time 2/3-approximation solution to the problem of packing n given identical squares in a rectangular region containing obstacles such that their edge length is as big as possible. Benkert et al. [2] supply a 2/3-approximation algorithm to the problem of packing a maximal cardinality set of disks within a rectangular region containing possibly intersecting unit disks as obstacles.

We define a t-square to be an axis-aligned square of size $t \times t$. The obnoxious multifacility location problem constraints imply that each demand point defines a forbidden $2r$-square centered at the point's location. Namely, for each such $2r$-square, the new facilities must not reside within this square. Let $P_i, 1 \le i \le n$, be these $2r$-squares, then the restriction of $\overline{\cup_{i=1}^{n} P_i}$ to R forms the region valid for the future facility placements. Here and throughout the paper we denote by V the rectilinear polygon resulted by $R - \cup_{i=1}^{n} P_i$ as shown in Figure 1. We define the *max-k-dispersion* problem and the *max-k-packing* problem to be the problems of dispersing and packing, respectively, a maximal number of d-squares in V. Clearly, the former problem is equivalent to our obnoxious multifacility location problem.

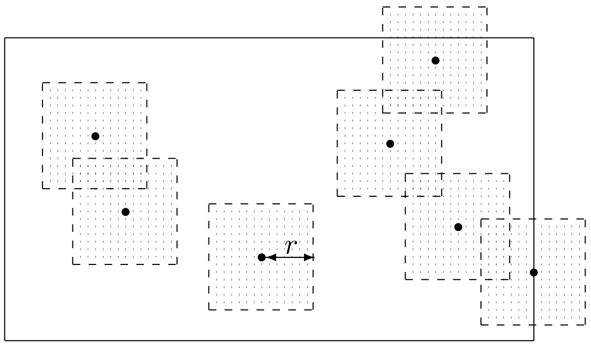

Figure 1: The rectangle R; the given demand points; around each point there is a forbidden square of length size $2r$; the white region is V.

Fowler et al. [6] show that the decision version of max-k-packing is NP-complete, which immediately yields the NP-completeness of max-k-packing and max-k-dispersion. Hochbaum et al. [7] supply a PTAS, based on the *shifting strategy*, for the following discrete version of the max-k-packing problem. Given a bounded region with a rectilinear grid of unit size placed on it and a natural number t, locate a maximal set of t-squares within the region so that their sides coincide with the lines of the overlaying grid. The resulting approximation of their solution has an error ratio of at least $(1-1/l)^2$ and runs in $O(d^2 l^2 n^{l^2})$ time, where l is the shifting parameter. The natures of max-k-dispersion and max-k-packing imply that designing a PTAS based on the shifting strategy for the max-k-dispersion problem (equivalently, for the obnoxious multifacility location problem) is a more complex task. This, in turn, emphasizes the essentiality of the PTAS we supply, as states Theorem 1.4.

Remark 2 Consider the general case where the distance between each $p_i, 1 \leq i \leq n$, and the new facilities must be above the given corresponding threshold r_i. Each of our solutions forms a similar runtime solution to this general problem with the following slight adjustment. Each forbidden square $P_i, 1 \leq i \leq n$, is now a $2r_i$-square centered at the location of p_i. No further adjustments are required.

1.3 Motivation

Our interest in the problem arises from the physical design process of electronic circuits [10], where an electronic board with several already placed components is given and some new components are to be located (this problem is also known as finding the optimal layout of chips in VLSI). The performance of the circuit is measured by two criteria. The major criterion is the total throughput of components that immediately emphasizes the importance of locating as many new components as possible. The minor criterion is the system delay incurred due to the components functioning and the rectilinear connections between them. This criterion implies that the new components should be located somewhat close to each other and to the already existing components. However, the new components produce heat that spreads along their connections and may damage other components. Hence, their mutual distance as well as the distance to the set of already placed components should be above a given threshold. For better understanding of these criteria, regard the case where each new component to be located is a processor in a multi-processor parallel system.

Consider a rectangular board R with components $p_1, \dots, p_n$ already placed. For each $1 \leq i \leq n$, let β_i be the maximal permitted heat at the location of p_i. For ease of presentation we regard the special case where each of these components can tolerate the same amount of heat β. Namely, for each $1 \leq i \leq n$, we have $\beta_i = \beta$. Each new component f_l produces δ amount of heat and can tolerate additional heat of amount γ. We denote by w the linear rate of the heat decrease along the circuit rectilinear connections starting from the heat production point. Let $d(p_i, f_l)$ be the rectilinear distance between p_i and f_l, then a valid location for f_l achieves

$$\delta - wd(p_i, f_l) \leq \beta, \tag{4}$$

which, in turn, yields

$$d(p_i, f_l) \geq \frac{\delta - \beta}{w}. \tag{5}$$

Similarly, a valid location for any two new components f_l and f_t achieves

$$d(f_l, f_t) \geq \frac{\delta - \gamma}{w}, \tag{6}$$

where $d(f_l, f_t)$ is the rectilinear distance between f_l and f_t. Transforming the system into L_∞ metric distances we get a special case of our problem such that r is the resulting transformation of the right handside of (5), and d is the resulting transformation of the right handside of (6).

2 Obnoxious multifacility

Our problem is to locate as many obnoxious facilities as possible within V such that the L_∞-distance between any two facilities is greater than d. Our solutions employ a repeating pattern of locating a facility within the region valid for the current location, and then computing the resulting sub-region valid for the next facility location. This creates the necessity of a dynamic algorithm for updating V upon placement of a new facility. In Section 2.1 we supply an $O(n \log n)$ time algorithm for computing the initial V. Moreover, this algorithm supports dynamic updates upon location of a new facility in $O(\log N)$ time. Section 2.2 begins with an introduction of an $O(N^k \log N)$ time optimization algorithm for the problem and continues with a presentation of an improvement to the same algorithm that requires only $O\left((N/4)^k \log N \log\log N\right)$ time. Section 2.2 ends with the presentation of the $O((N/4)^{k-2} \log N)$-time algorithm for the corresponding decision version of the problem. In Section 2.3 we give a 1/2-approximation algorithm for the problem that runs in $O\left(\max(n \log n, k \log N \log\log N)\right)$ time. Finally, we present a PTAS in Section 2.4.

2.1 Computing the region valid for future facility locations

Recall that each demand point defines the forbidden $2r$-square where the facilities must not reside. Similarly, each facility defines a forbidden $2d$-square centered at its location where the other facilities must not reside. Denote by F_i the forbidden $2d$-square defined by f_i. For any given t, let $f_1, \ldots, f_t$ be a set of already located facilities within V, and f_{t+1} be the next facility to be located. The definitions of V and $F_1, \ldots, F_t$ immediately show that the restriction of $\overline{\cup_{i=1}^{t} F_i}$ to V forms the region valid for the location of f_{t+1}. This, in turn, yields that once f_{t+1} is located within $V - \cup_{i=1}^{t} F_i$, the restriction of $\overline{\cup_{i=1}^{t+1} F_i}$ to V forms the region valid for the location of f_{t+2}, and so on. Based on this observation, we supply below an $O(n \log n)$ time algorithm for the initial computation of V that supports the dynamic update of the region valid for future facility locations in $O(\log N)$ time.

Huang et al. [8, Section 5] introduce a graph representation of the regions resulting from the complement of a given squares union. The vertices and edges of the graph G in the representation correspond to the vertices and edges comprising the boundaries of the resulting regions as follows. Each stand alone square in the union is represented by eight vertices and eight edges as shown in Figure 2 ([8]). For every corner x of a square, two vertices x and x' are considered. Thus, G consists of $O(m)$ vertices and edges, where m always denotes the current number of squares in the union. The extra vertices help to efficiently maintain the graph when squares start to move and intersect on the plane. When two squares meet, at most two pairs of line segments of their boundaries intersect. Without loss of generality, suppose a vertical edge $b'c$ intersects with a horizontal edge $e'f$ at a point z, and the new boundary comprises edges $b'z$ and zf. Then we simply reallocate vertices c and e' to z, insert an edge ce' and remove edges cc' and ee' from G. This operation is illustrated in Figure 2. The cases of a vertical edge intersecting with a vertical edge, and of a horizontal edge intersecting with a horizontal edge are analogous, and thus handled by at most two edge insertions and at most two edge deletions.

Huang et al. [8] proved that such graph supports, edge Insertions, and deletions that may occur upon insertion of a new square into the squares union, in $O(\log m)$ time, where m is the current number of squares in the union. We first compute the forbidden $2r$-squares corresponding to the n demand points, and then compute the complement of their union restricted to R. We construct the graph representation of the resulting regions as in [8]. This preprocessing phase requires $O(n \log n)$ time. Locating a new facility in one of the resulting regions yields the insertion of its corresponding forbidden $2d$-square into the squares union, which actually yields the update of the graph in the representation in $O(\log N)$ time. The resulting graph represents the region valid for the next facility to be located.

2.2 Optimization and decision algorithms

The following lemma is equivalent to [9, Lemma 2.5]. For the sake of completeness, we present below its proof, which is very similar (but yet simpler and more elegant) to the proof of [9, Lemma 2.5], since we use it later to design an improved optimization algorithm for the problem.

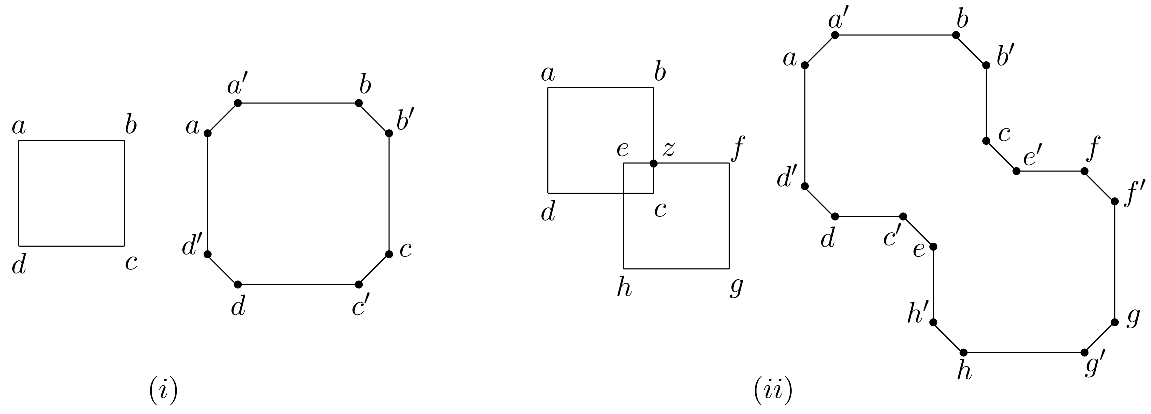

Figure 2: Graph G of the representation. (i) A square represented by 8 vertices. (ii) Dynamics of vertices and edges when two squares intersect. Vertices c, c', e, and e' are relocated. Edges cc' and ee' are removed, and edges ce' and ec' are inserted.

Lemma 2.1 *At least one of the facilities in any maximal location is located at a vertex on the boundary of* V*.*

Proof. By contrary, assume there exists a maximal location with facilities $f_1, \dots, f_k$ within V, such that none of the facilities is located at a vertex on the boundary of V. For each facility f_i, let c_i be a d-square centered at the location of f_i. Clearly, in the given initial positioning the c_i's do not intersect. Our approach is to move the c_i's such that at least one of the f_i's will eventually coincide with some vertex on the boundary of V. Moreover, at the end of the movement, the c_i's remain pairwise disjoint. We first push the c_i's as much to the left as possible, so that they still do not intersect and the center of each square resides within V throughout the motion. If at any point throughout this movement one of the f_i's coincides with some vertex on the boundary of V, then we are done. At the end of this stage, the leftmost facility f_i is lying on a vertical edge of the boundary of V (if there are several leftmost facilities, then f_i is the lowest among them). We continue pushing the c_i's down as much as possible so that they still do not intersect and the center of each square resides within V throughout the motion. As with the horizontal movement, we stop if at some point one of the f_i's coincides with some vertex on the boundary of V. At the end of this stage, the bottommost facility f_j is lying on a horizontal edge of the boundary of V (if there are several bottommost facilities, then f_j is the leftmost among them).

If $c_i = c_j$, then the facility f_i is located at a vertex v on the boundary of V, and we are done. Otherwise, c_i is blocked below by another square c_t, and c_j is blocked on its left by another square c_l. Note that f_t is not lying on any horizontal edge of the boundary of V, since then we can push c_t to the left until f_t coincides with a vertex of V. Thus, f_t is lying on a vertical edge of the boundary of V. Considering similar arguments, we conclude that f_l is lying on a horizontal edge of the boundary of V. Repeating this process, we obtain a sequence of blocking squares from below with their corresponding facilities lying on vertical edges of the boundary of V, and a sequence of blocking squares on the left with their corresponding facilities lying on horizontal edges of the boundary of V. Since the boundary is finite, at some point one of these facilities coincides with some vertex v on the boundary of V. $\square$

The following algorithm is based on the algorithm in [9] for the decision problem in the case of $k \geq 4$. We first compute the graph representation of V as shown in Section 2.1, and then consider positioning a facility at each vertex v on the boundary of V. For each such positioning, we first locate a forbidden $2d$-square Q_v at v and then compute $V - Q_v$, the region valid for the next facility to locate, using the graph representation. We continue recursively with the updated $V = V - Q_v$. The algorithm stops only when the computation of the valid region for the next facility to locate with each vertex on the boundary of the current region yields an empty set. Hence, the stop criterion is given by the formula

$$\forall v \in V: \quad V - Q_v = \emptyset, \tag{7}$$

where V is the current region valid for the facility location. Once the algorithm stops, it returns the maximal cardinality set of facility locations over all the recursively computed sets. The running time for a maximal solution with k facilities is given by $T(k) = N \cdot (O(\log N) + T(k-1))$. Observe that $T(1) = O(N \log N)$, which yields $T(k) = O(N^k \log N)$. Recall that the optimization problem can be solved in a similar manner by the algorithm for the decision version in [9] with the same worst-case time complexity, as suggested in Section 1.2. However, this worst case time complexity is given by $T(k) = N \cdot (O(N) + T(k-1))$, and hence our solution forms a more efficient solution.

A closer examination of the proof of Lemma 2.1 produces the following conclusion. Let v be the vertex on the boundary of V found at the end of the movement process. The vertex v forms the left endpoint of some horizontal edge bounding V on its bottom, and the bottom endpoint of some vertical edge bounding V on its left. We call such vertex a *left-bottom* vertex of V. The proof of the lemma yields that at least one of the facilities is located at a left-bottom vertex of V. Consider the analogous definitions of *right-bottom*, *left-top*, and *right-top* vertices of V. To clarify these vertex definitions, we specify in Figure 3 the type of each vertex on the boundary of the region valid for the facility locations from Figure 1. The proof of the lemma can be easily changed to prove that at least one of the facilities is located at a right-bottom, a left-top, or a right-top vertex of V, by changing the direction of the c_i's movement. Hence,

Lemma 2.2 *For each type of vertex, i.e., a left-bottom, a right-bottom, a left-top, or a right-top vertex of V, at least one of the facilities in a maximal location is located at a vertex on the boundary of V of this type.*

This lemma implies that it is sufficient to consider only vertices of the above defined four types instead of all the vertices on the boundary of V throughout the algorithm. Note that for at least one of these types, the number of vertices of this type on the boundary of V is at most

$\lfloor |V|/4 \rfloor$, which in turn is bounded by $N/4$. Thus, considering only vertices of this type significantly improves our algorithm. To support this improvement we maintain four orthogonal range trees [13]. An orthogonal range tree consists of a balanced binary tree according to the x-coordinate of the points. Each node w of this tree corresponds to the balanced binary tree (secondary tree) according to the y-coordinate of points whose x-coordinate belongs to the subtree rooted at w. In our solution each orthogonal range tree stores all the points corresponding to same type vertices on the boundary of the region that is valid for future facility locations.

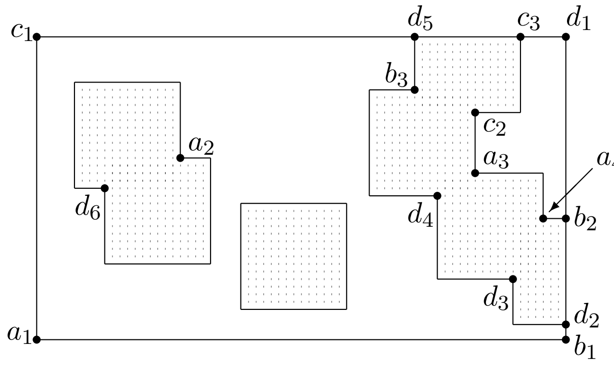

Figure 3: The region valid for future facility locations V is the resulting white region within R; each of the highlighted vertices on the boundary of V is a left-bottom, a right-bottom, a left-top, or a right-top vertex of V. The a_i's are left-bottom vertices, the b_i's are right-bottom vertices, the c_i's are left-top vertices, the d_i's are right-top vertices.

Once V is computed, we initialize the orthogonal range trees as follows. For each vertex v on the boundary of V, we store its corresponding point p_v in the appropriate tree according to its type. Clearly, at the end of this initialization phase, each of the orthogonal range trees contains at least one point. Let T be the tree containing the minimal number of points. For each point p_v stored in T, we consider positioning a facility at v. For each such positioning, we first compute the region $V - Q_v$ valid for the next facility location and then update each of the four orthogonal range trees accordingly. The update of each orthogonal range tree includes removing points related to vertices that have been erased from the boundary of V throughout the update $V = V - Q_v$, and inserting points related to new vertices that have been added to the boundary of V throughout the same update. Such an update occurring upon a facility location considering the example presented in Figure 1 and Figure 3 is illustrated in Figure 4. We continue recursively, as defined previously, with the slight variation that at any point of the algorithm we consider only vertices corresponding to points in the orthogonal range tree of minimal cardinality. Note that we can improve the performance of the orthogonal range trees by using the *Dynamic fractional cascading* technique [11]. This gives us a speedup of $O(\log N/\log\log N)$ in running time of queries and updates. Observe that the number of tree updates incurred due to the location of a new facility is constant, and the cost of each such update, assuming we use orthogonal range trees with the dynamic fractional cascading technique, is bounded by $O(\log N \log\log N)$ time. Thus, the running time of the algorithm, involving this improvement, is given by $T(k) = N/4 \cdot (O(\log N \log\log N) + T(k-1))$. Since $T(1) = O((N/4)\log N \log\log N)$ time, our improved algorithm requires $O((N/4)^k \log N \log\log N)$ time.

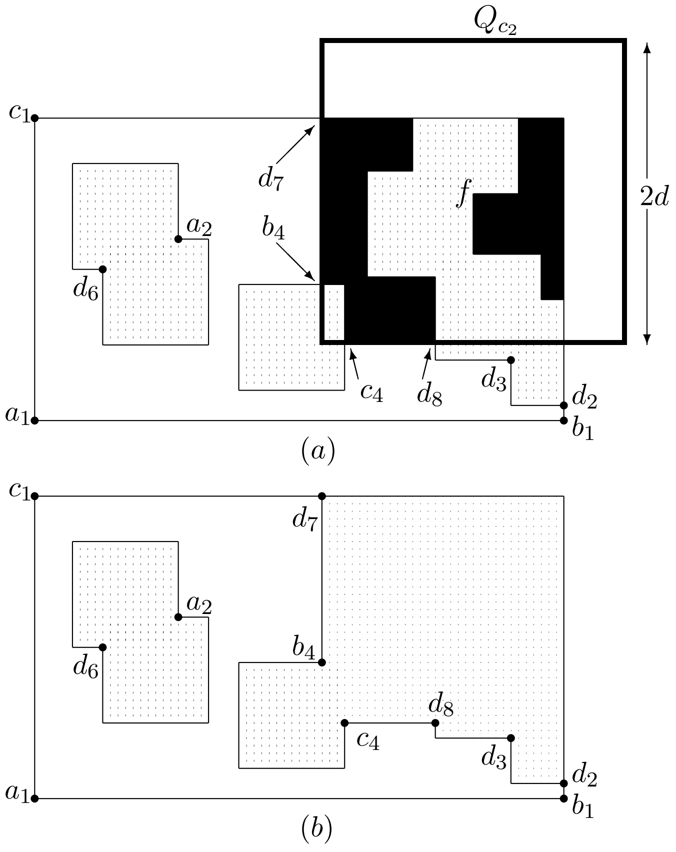

Figure 4: Locating a new facility within the region valid for the current facility location V, as shown in Figure 3; the orthogonal range tree containing the points related to the c_i's is of minimum cardinality. Part (a) illustrates the location of a new facility f at the chosen left-top vertex c_2 and the corresponding location of the $2d$-square Q_{c_2} at c_2; the black regions are the parts removed from V throughout the computation of $V - Q_{c_2}$. Part (b) illustrates the resulting region valid for the next facility location $V - Q_{c_2}$; each of the points related to the vertices $a_3, a_4, b_2, b_3, c_2, c_3, d_1, d_4$ and d_5 has been removed from its corresponding containing orthogonal range tree; the points related to the new right-bottom vertex b_4, the new left-top vertex c_4, and the new right-top vertices d_7 and d_8 are inserted into their corresponding orthogonal range trees.

Next, we show how to improve the running time of the corresponding decision algorithm given by Katz et al. [9]. The algorithm for the decision version is similar to the above algorithm for the optimization problem with the following slight variation. Our decision algorithm employs the $O(N \log N)$-time decision algorithm for three facilities of Katz et al. [9] as follows. For the first $k-3$ facilities, our decision algorithm is similar to the above optimization algorithm. If the algorithm fails in locating all these $k-3$ facilities, our decision algorithm stops and returns *NO*. Otherwise, these $k-3$ facilities have been successfully located, and our decision algorithm executes the decision algorithm for three facilities of Katz et al. [9] with the remaining three facilities. If this execution returns *NO*, our decision algorithm returns *NO* and exits. Otherwise, it returns *YES*. Clearly, this approach is equivalent to implementing the above two improvements (i.e., the improvement presented in Section 2.1, and the improvement presented in Lemma 2.2 and the consequent use of the orthogonal range trees) directly in the decision algorithm for more than three facilities of Katz et al. [9]. Applying similar considerations to those taken for the above optimization algorithm, we conclude that our decision algorithm requires $O((N/4)^{k-2}\log N)$ time, where this runtime results in the formula given in (3) with $T(3) = O(N \log N)$. This proves Theorem 1.2.

2.3 A 1/2-approximation solution

Let v be the leftmost vertex on the boundary of V. If there exist few leftmost vertices, then v is the lowest among them. We define v to be the *leftmost-bottommost* vertex on the boundary of V. The 1/2-approximation factor of our solution implicitly resulted from the following lemma together with the derived Observation 2.4.

Lemma 2.3 *Let k be the number of obnoxious facilities in a maximal location within V (namely, for any feasible location with t obnoxious facilities, it holds that $t \leq k$), then there either exists a maximal location within V with one of the facilities located at v, or there exists a feasible location with $k-1$ facilities within V with one of the facilities located at v.*

Proof. Let $f_1, \dots, f_k$ be the facilities participating in a maximal location within V, and $c_1, \dots, c_k$ be their corresponding d-squares as defined in the proof of Lemma 2.1. Clearly, in the given maximal location, the c_i's do not intersect. We show that if we center a d-square at v, then it may intersect at most 2 squares out of the c_i's. Namely, at most two facilities of $f_1, \dots, f_k$ are located at a distance that is smaller than d from v. This, in turn, yields that by locating a new obnoxious facility at v and removing all the facilities located at a distance that is smaller than d from v, we get a feasible location with k or $k-1$ facilities (clearly, if it is a feasible location with k facilities, then it also forms a maximal location). Let s_v be the d-square centered at v that represents the forbidden area related to the new facility that must not intersect with any of the c_i's. Since the location of $f_1, \dots, f_k$ is maximal and v is the leftmost-bottommost vertex in V, only one of the following cases is possible:

1. One of the facilities f_i is located at v. Thus, its corresponding square c_i coincides with s_v. Replacing this facility with the new one, we obtain the same maximal location.

2. None of the facilities is located at v and exactly one of them, denoted by f_i, is located at a point, such that its corresponding square c_i intersects s_v (see Figure 5-(a)). Replacing this facility with the new one, we obtain a maximal location with a facility located at v.

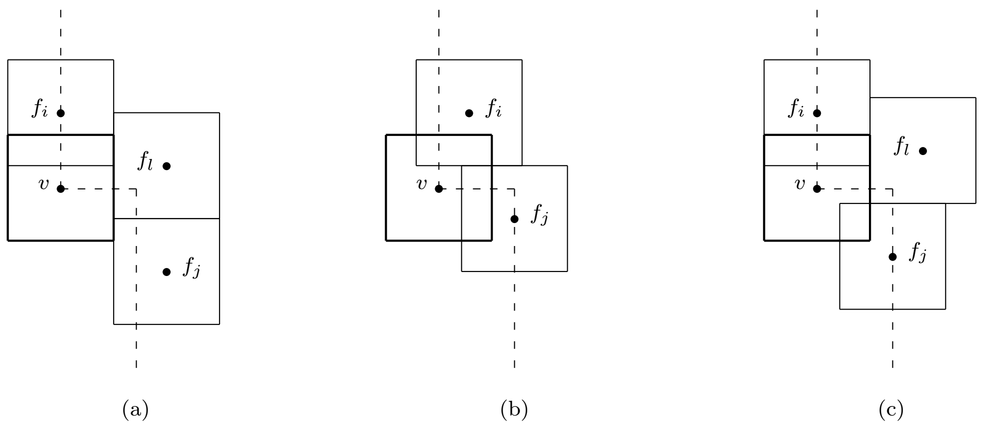

Figure 5: Replacing one or two squares of the c_i's with s_v.

3. None of the facilities is located at v and exactly two of them, denoted by f_i and f_j, are located at points such that their corresponding squares c_i and c_j, respectively, intersect s_v (see Figures 5-(b) and 5-(c)). Replacing these facilities with the new one, we obtain a feasible location with $k-1$ facilities such that one of them is located at v.

This completes the proof. $\square$

Consider a maximal location within V with facilities $f_1, \ldots, f_k$ and the corresponding feasible location within V from Lemma 2.3. Recall that each facility $f_i, 1 \leq i \leq k$, defines the forbidden $2d$-square F_i where the other facilities must not reside. Let F be the union of the F_i's of the at most two facilities that have been removed from the maximal location according to the proof of Lemma 2.3. Consider the location of these facilities only within V, such that each facility is located at its exact location from the maximal location. Clearly, $V - F$ is the region valid for the next facility location. Now consider location of a facility at v only. As stated previously, $V - Q_v$ is the region valid for the next facility location resulted by this location. Thus, we obtain

Observation 2.4 *The cardinality of a maximal location within $V - Q_v$ is no less than the cardinality of a maximal location within $V - F$.*

Since $(Q_v \cap V) \subseteq (F \cap V)$, we get $(V - F) \subseteq (V - Q_v)$. Thus, the sub-location resulting from the restriction of the maximal location to $V - F$, which in turn is a maximal location within $V - F$, forms also a feasible location within $V - Q_v$. Clearly, the cardinality of this feasible location within $V - Q_v$ is smaller than or equal to the cardinality of any maximal location within $V - Q_v$.

The outline of the algorithm is as follows. We first compute the graph representation of V, and insert the points corresponding to the left-bottom vertices of V into an orthogonal range tree. Then, we locate a facility at the leftmost-bottommost vertex of V. We position at v the

$2d$-square Q_v, and update V to be $V - Q_v$, which is the region valid for the location of the next facility. We continue recursively with the updated V, where the stop criterion is similar to the one used by the optimization algorithm in Section 2.2. Employing Lemma 2.3 and Observation 2.4 at each recursive step, we get the 1/2-approximation factor of our algorithm for the problem. Applying similar considerations to those taken for the time complexity analysis of the optimization algorithm, this approximation algorithm requires $O(\max(n \log n, k \log N \log \log N))$ time.

2.4 A PTAS

The shifting strategy of Hochbaum et al. [7] is a powerful approach for devising polynomial approximation schemes for many NP-complete problems. It allows us to bound the error of

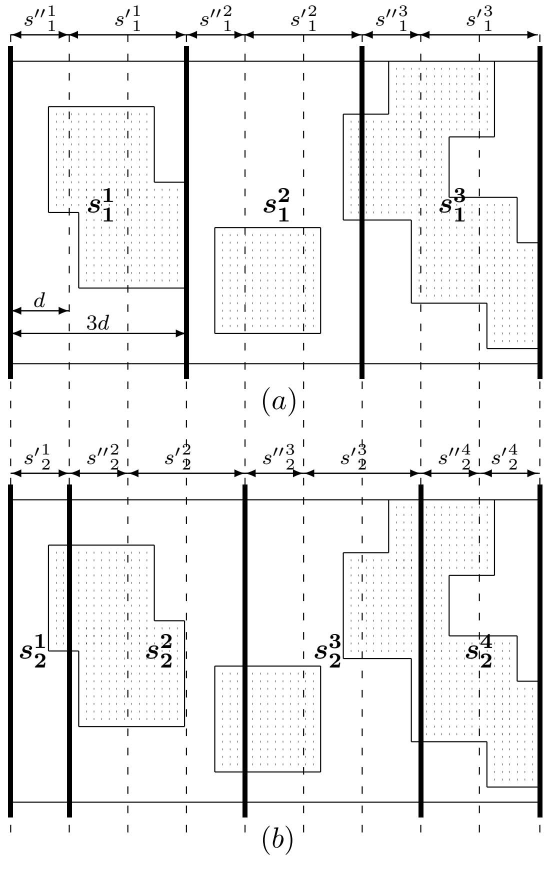

Figure 6: The region valid for future facility locations is V, as shown in Figure 3; the shifting parameter $l = 3$. The upper part (a) illustrates the partition s_1 and its comprising $3d$-strips s^1_1, s^2_1, and s^3_1; each $s^j_1, 1 \le j \le 3$, consists of the d-strip s''^j_1 and the $2d$-strip s'^j_1; The lower part (b) illustrates the partition s_2 resulting from shifting the partition s_1 to the right over one d-strip; s_2 is consist of the d-strip s^1_2, the $3d$-strips s^2_2 and s^3_2, and the $2d$-strip s^4_2; Recall that at most two ld-strips may contain less than l consecutive d-strips; indeed, the leftmost strip s^1_2 contains the single d-strip s'^1_2, and the rightmost strip s^4_2 contains the d-strips s''^4_2 and s'^4_2; each inner strip $s^j_2, 1 < j < 4$ is a $3d$-strip containing the d-strip s''^j_2 and a $2d$-strip s'^j_2

the simple divide-and-conquer approach by applying it repeatedly and selecting the single most favorable resulting solution. The framework of this technique consists of the following steps. First, a shifting parameter l that yields a corresponding $(1-1/l)$-approximation factor is chosen. Then, the problem is repeatedly partitioned into smaller sub-problems that can be solved approximately in polynomial time, and can be combined into one feasible solution to the problem. The maximal solution over all these repetitions forms the solution for the whole problem. Note that we re-employ this technique in each of the smaller sub-problems in order to supply a local approximation solution.

Consider a division of V into vertical, left-closed, and right-opened strips of width d. We regard each of these strips as a d-strip, and define a td-strip to be a vertical strip of width td resulting from the union of t consecutive d-strips. Let l be the shifting parameter, then for a fixed division of V into d-strips, there are l different ways to partition it into consecutive ld-strips such that each partition is derived from the previous one by shifting it to the right over one d-strip. Note that by repeating this right shift l times, we end up with the same partition we started from. For ease of presentation, we regard each of the strips comprising a partition as an ld-strip, even though a partition may consist of one or two td-strips with $t < l$. Denote these shift partitions by $s_1, \ldots, s_l$. For each partition $s_i, 1 \leq i \leq l$, we define $|s_i|$ to be the number of consecutive ld-strips comprising it, and $s_i^1, \ldots s_i^{|s_i|}$ to be these ld-strips. For each ld-strip $s_i^j, 1 \leq i \leq l, 1 \leq j \leq |s_i|$, we define s'^j_i to be the $(l-1)d$-strip resulting by the union of the $l-1$ rightmost d-strips of s_i^j, and s''^j_i to be the leftmost d-strip of s_i^j. In order to ensure a legal combination the local solutions at two consecutive ld-strips, the removal of each facility lying in the leftmost d-strip of the right ld-strip is required (alternatively, the removal of each facility lying in the rightmost d-strip of the left ld-strip is required). This implies that for any partition $s_i, 1 \leq i \leq l$, and each $2 \leq j \leq |s_i|$, the $(l-1)d$-strip s'^j_i is used to find a local solution within s_i^j, and the d-strip s''^j_i is used to enable the combination of this local solution with the one of s_i^{j-1}. The first partition s_1 of V, as shown in Figure 3, and the partition s_2 resulting from shifting this partition to the right over one d-strip is illustrated in Figure 6 above.

Let A be any algorithm that solves the obnoxious multifacility location problem in the restriction of V to any strip of width ld or less. For each partition $s_i, 1 \leq i \leq l$, let $A(s_i)$ be the algorithm that invokes $A(s_i^1)$ and $A(s'^j_i), 2 \leq j \leq |s_i|$, and outputs the set of all the computed locations. Clearly, this set of points computed by $A(s_i)$ forms a feasible solution for the obnoxious multifacility location problem defined on V. The shift algorithm S_A invokes $A(s_i)$ for each partition $s_i, 1 \leq i \leq l$, and outputs the set of points of maximal cardinality out of the l computed sets.

We denote by MAX a maximal solution and by $|MAX|$ its value (recall that $|MAX| = k$ throughout the paper). For an algorithm B, let Z^B be the value of the solution computed by it, and r_B be its performance ratio, i.e., the infimum of $Z^B/|MAX|$ over all the problem instances. The following Lemma forms the core of the PTAS we design.

Lemma 2.5 (The Shifting Lemma)

$$r_{S_A} \geq r_A \left(1 - \frac{1}{l}\right), \tag{8}$$

where A is a local algorithm and l is the shifting parameter.

Proof. We follow the notations from [7]. Consider a partition $s_i, 1 \leq i \leq l$. The value of the solution computed by $A(s_i)$ is given by

$$Z^{A(s_i)} = Z^{A(s_i^1)} + \sum_{j=2}^{|s_i|} Z^{A({s'}_i^j)}.$$

For each $j, 1 \leq j \leq |s_i|$, we denote by MAX_i^j and ${MAX'}_i^j$ the maximal solutions for the obnoxious multifacility location problem in the bounded polygons defined by the restrictions of V to s_i^j and ${s'}_i^j$, respectively. Clearly, for each $j, 1 \leq j \leq |s_i|$, it holds that $\left|{MAX'}_i^j\right| \leq \left|MAX_i^j\right|$. By this and the definition of the performance ratio r_A, we have

$$Z^{A(s_i)} \geq r_A \sum_{j=1}^{|s_i|} \left|{MAX'}_i^j\right|. \tag{9}$$

Observe that $\cup_{j=1}^{|s_i|} {MAX'}_i^j$ forms a feasible solution for the obnoxious multifacility location problem defined on V. Thus,

$$\sum_{j=1}^{|s_i|} \left|{MAX'}_i^j\right| \leq |MAX|. \tag{10}$$

Consider any $j, 1 \leq j \leq |s_i|$. We define $MAX|s_i^j, MAX|{s'}_i^j$ and $MAX|{s''}_i^j$ to be the restrictions of MAX to the strips $s_i^j, {s'}_i^j$, and ${s''}_i^j$, respectively. Namely, for a strip $s \in \{s_i^j, {s'}_i^j, {s''}_i^j\}$, the restriction $MAX|s$ contains all the location points from MAX that also reside inside s. Clearly, $\left|MAX|{s'}_i^j\right| = \left|MAX|s_i^j\right| - \left|MAX|{s''}_i^j\right|$. The maximality of ${MAX'}_i^j$ yields $\left|{MAX'}_i^j\right| \geq \left|MAX|{s'}_i^j\right|$ which, in turn, yields

$$\sum_{j=1}^{|s_i|} \left|{MAX'}_i^j\right| \geq \sum_{j=1}^{|s_i|} \left|MAX|s_i^j\right| - \sum_{j=1}^{|s_i|} \left|MAX|{s''}_i^j\right|. \tag{11}$$

For each partition $s_i, 1 \leq i \leq l$, put ${MAX''}^{(i)} = \cup_{j=1}^{|s_i|} MAX|{s''}_i^j$. Using these notations we have

$$\sum_{j=1}^{|s_i|} \left|{MAX'}_i^j\right| \geq |MAX| - \left|{MAX''}^{(i)}\right|. \tag{12}$$

We observe that the sets ${MAX''}^{(1)}, \ldots, {MAX''}^{(l)}$ are disjoint and hence can add up to MAX at most. Thus,

$$\sum_{i=1}^{l} \sum_{j=1}^{|s_i|} \left|{MAX'}_i^j\right| \geq \sum_{i=1}^{l} |MAX| - \sum_{i=1}^{l} \left|{MAX''}^{(i)}\right| \geq (l-1)\,|MAX|. \tag{13}$$

Expressions (9) and (13) imply that

$$\max_{1\le i\le l} Z^{A(s_i)} \ge \frac{1}{l}\sum_{i=1}^{l} Z^{A(s_i)} \ge \frac{r_A}{l}\sum_{i=1}^{l}\sum_{j=1}^{|s_i|}\left|MAX'^{j}_{i}\right| \ge r_A\left(1-\frac{1}{l}\right)|MAX|. \tag{14}$$

By the definition of Z^{S_A} and (14) we obtain

$$Z^{S_A} = \max_{1\le i\le l} Z^{A(s_i)} \ge r_A\left(1-\frac{1}{l}\right)|MAX|, \tag{15}$$

which establishes (8). $\square$

We use two nested applications of this shifting strategy with a shifting parameter l to design the required PTAS for the obnoxious multifacility location problem as follows. We first divide the plane into vertical ld-strips, and then apply the shifting strategy horizontally in order to solve the obnoxious multifacility location problem at each restriction of V to these ld-strips. Thus, we divide each of the vertical ld-strips into squares of side length ld. The set of horizontal partitions of the $3d$-strip s_1^1 from Figure 6 is illustrated in Figure 7. We solve the obnoxious multifacility location problem at each bounded polygon defined by the restriction of V to the corresponding ld-square using the optimization algorithm presented at Section 2.2. By the shifting lemma, this process yields a $(1-1/l)$-approximation solution for the obnoxious multifacility location problem defined at each restriction of V to a vertical ld-strip, and thus a $(1-1/l)^2$-approximation solution for the obnoxious multifacility location problem defined at V.

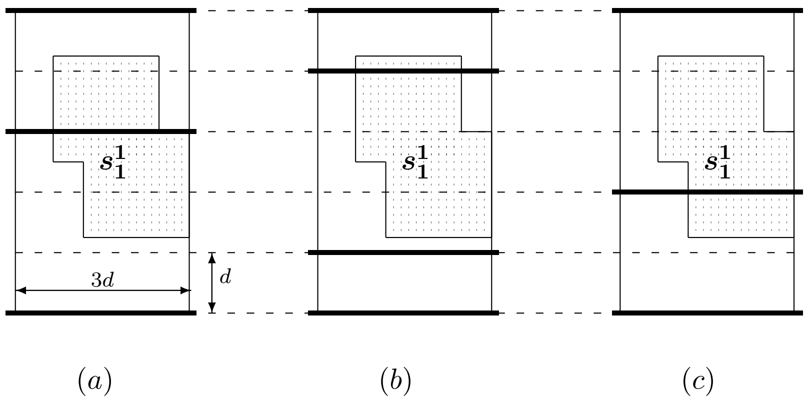

Figure 7: The set of all the horizontal partitions of the $3d$-strip s_1^1 from Figure 6. The leftmost part (a) illustrates the first horizontal partition of s_1^1. The middle part (b) illustrates the second horizontal partition of s_1^1 resulting from shifting the first horizontal partition to the top over one d-strip. The rightmost part (c) illustrates the third (and last) horizontal partition of s_1^1 resulting from shifting the second horizontal partition to the top over one d-strip.

Let $\widetilde{P}$ be a bounded polygon defined by the restriction of V to some ld-square, and $\widetilde{n}$ the number of vertices on its boundary. Clearly, the size of a maximal solution for the obnoxious multifacility location problem within $\widetilde{P}$ is at most l^2. Thus, our optimization algorithm from Section 2.2 solves this local problem in $O((\widetilde{N}/4)^{l^2} \log \widetilde{N} \log\log \widetilde{N})$ time, where $\widetilde{N} = O(\widetilde{n}+l^2)$ is the upper bound on the number of vertices comprising the boundary of the valid region for facility locations throughout the optimization algorithm execution. Recall that this optimization algorithm determines in $O(\widetilde{n} \log \widetilde{n} \log\log \widetilde{n})$ time if $\widetilde{P}$ is not valid for a facility location. Thus, we may conclude that the number of such invalid local polygons is negligible, and omit any related consideration from the complexity analysis. For any two horizontal partitions of some ld-strip, the number of the ld-squares participating in these partitions are different by at most 1. Similarly, for any two vertical partitions of the plane, the amounts of the ld-strips participating in these partitions are different by at most 1. We define t_1 to be the size of any such horizontal partition of a given ld-strip, and t_2 to be the size of any such vertical partition of the plane. This, in turn, yields that our algorithm executes the optimization algorithm $l^2 t_1 t_2$ times, such that each execution applies the optimization algorithm on a different local polygon $\widetilde{P}$. Note that the time required for $t_1 t_2$ local executions of the optimization algorithm, where each execution occurs within a different polygon $\widetilde{P}$, is bounded by $O\left(((N+t_1t_2)/4)^{l^2} \log(N+t_1t_2) \log\log(N+t_1t_2)\right)$ time. Let k be the number of obnoxious facilities in a maximal location, then the above considerations yield $t_1 t_2 \leq k$. Thus, the algorithm S_A requires $O\left(l^2\left((N+k)/4\right)^{l^2} \log(N+k) \log\log(N+k)\right)$ time, which yields $O\left(l^2 (N/4)^{l^2} \log N \log\log N\right)$ time.

3 Conclusions

In this paper we presented a number of efficient solutions for maximizing the number of obnoxious facilities inside a given region under the L_∞ metric. One possible future direction is to extend them for arbitrary metric. It is also challenging to make them work in dynamic settings, i.e., when the demand points are allowed to be inserted or deleted.

References

[1] C. Baur and S.P. Fekete, Approximation of geometric dispersion problems, *Algorithmica* 30 (2001), 451–470.

[2] M. Benkert, J. Gudmundsson, C. Knauer, E. Moet, R. van Oostrum and A. Wolff, A polynomial-time approximation algorithm for a geometric dispersion problem, *22nd European workshop on computational geometry* (2006), 141–144.

[3] B. Ben-Moshe, M. J. Katz and M. Segal, Obnoxious facility location: Complete service with minimal harm, *International Journal of Computational Geometry and Applications* 10 (2000), 581-592.

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

[5] P. Cappanera, A Survey on obnoxious facility location problems, *Technical Report* (1999).

[6] R.J. Fowler, M.S. Paterson and S.L. Tanimoto, Optimal packing and covering in the plane are NP-complete, *Information Processing Letters* 12 (1981), 133–137.

[7] D.S. Hochbaum and W. Maass, Approximation schemes for covering and packing problems in image processing and VLSI, *Journal of the ACM* 32 (1985), 130–136.

[8] H. Huang, A. W. Richa and M. Segal, Dynamic coverage in ad-hoc sensor networks, *Mobile Networks and Applications* 10 (2005), 9–17.

[9] M. Katz, K. Kedem and M. Segal, Improved algorithms for placing undesirable facilities, *Computers and Operations Research* 29 (2002), 1859–1872.

[10] M. Lorenzetti and B. Preas, *Physical Design Automation of VLSI Systems*, Benjamin/Cummings Publishing Company, California (1988).

[11] K. Mehlhorn and S. Näher, Dynamic fractional cascading, *Algorithmica*, Springer, New-York (2005), 215–241.

[12] F. Plastria, Continuous covering location problems, in H. Hamacher and Z. Drezner, *Location analysis: theory and applications*, Springer, New-York (2001), 39–83.

[13] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, NY, 1985.

[14] Z. Qin, Y. Xu and B. Zhu, On some optimization problems in obnoxious facility location, *Lecture Notes in Computer Science*, Springer, 2000.

[15] A. Tamir, Locating two obnoxious facilities using the weighted maximin criterion, *Operations Research Letters* 34 (2006), 97–105.

[16] S. B. Welch, S. Salhi and Z. Drezner, The multifacility maximin planar location problem with facility interaction, *IMA Journal of Management Mathematics* (2006), 1–16.