# New Variations of the Reverse Facility Location Problem

Bhaswar B. Bhattacharya* Subhas C. Nandy*

**Abstract**

In this paper we consider a natural extension of the so-called reverse facility location problem which was introduced by Cabello et al. [3]. Given a set of $n$ users and a set of $m$ facilities, where a user takes service from its nearest facility, the objective is to place two new facilities such that the total number of users served by these two new facilities is maximized. We refer to this problem as the 2-MaxCov problem. In the $L_1$ and $L_\infty$ metrics, the worst case time and space complexities of our proposed algorithm for solving this problem are both $O(n^2 \log n)$. In the $L_2$ metric, we propose an algorithm for solving the problem in $O(n \log n)$ time, if $m = 1$. We have also considered the obnoxious version of this problem, referred to as the 2-Farthest-MaxCov problem, where a user is served by its farthest facility. Our proposed algorithm for this problem runs in $O(n \log n)$ time for all the considered distance measures.

## 1 Introduction

The main objective in any facility location problem is to judiciously place a set of facilities, serving a set of users, such that certain optimality criterion is satisfied. Facilities and users are generally modeled as points in the plane. A facility can be *attractive*, like hospitals, schools, and supermarkets; or *obnoxious*, like garbage dumps and chemical plants. On the other hand, the set of users is either *discrete*, consisting of finitely many points, or *continuous*, that is, a region where every point is considered to be a user. Given that the facilities are equally equipped in all respects, a user always avails the service from its nearest facility. Consequently, each facility has its *service zone*, consisting of the set of users that are served by it. For a set $\mathcal{U}$ of users, finite or infinite, and a set $\mathcal{F}$ of facilities, define for every $f \in \mathcal{F}$, $\mathcal{U}(f, \mathcal{F})$ as the set of users in $\mathcal{U}$ that are served by the facility $f$ among the facilities in $\mathcal{F}$. Many variations of facility location problem in both the discrete and continuous user category, under several optimality criteria, have been studied [6]. Maximizing the cardinality or area of the service zone is one such criteria.

For continuous demand region, Dehne et al. [5] addressed the problem of locating a new facility $q$ amidst a set $\mathcal{F}$ of $n$ existing facilities, such that the area of the region served by $q$ is maximized. The problem reduces to placing a new point $q$ amidst a set of $n$ existing points $\mathcal{F}$ such that the Voronoi region of $q$ is maximized. Dehne et al. [5] showed that, when the given points are in convex position, the area function has only a single local maximum inside the region where the set of Voronoi neighbors do not change. For the same problem, Cheong et al. [4] gave a near-linear time algorithm that locates the new optimal point approximately, when the points in $\mathcal{F}$ are in general position. Variations of this problem, involving maximization of the area of Voronoi regions of a set of points placed inside a circle, have been recently considered by Bhattacharya [2].

The analogous version of this problem in the discrete user case is the problem of placing a new facility amidst a set of existing ones such that the number of users served by the new facility is maximized. This problem has been recently addressed by Cabello et al. [3]. They refer to this as the MaxCov problem. They showed that in the $L_1$ and $L_\infty$ metrics, the problem can be solved in $O(n \log n)$ time. In the $L_2$ metric, they proved that if the number of existing facilities $m \geq 2$, the MaxCov problem is 3SUM hard, and gave an algorithm for finding the set of all possible optimal placements of the new facility in $O(n^2)$ time. They also showed that for $m = 1$ the MaxCov problem in $L_2$ metric can be solved in $O(n \log n)$ time, and this is asymptotically optimal under the algebraic decision tree model.

Now, instead of placing one new facility, one may wish to place multiple facilities simultaneously such that they together serve the maximum number of users. This leads to the following generalization of the MaxCov problem.

$k$-MaxCov Problem: Given a set $\mathcal{U}$ of $n$ users, and a set $\mathcal{F}$ of $m$ existing facilities with $m < n$, find the placement of a set $F^*$ of $k$ ($\geq 1$) new facilities such that the total number of users in $\mathcal{U}$ served by the facilities in $F^*$ is maximized. In other words, we have to find the placement of a set $F^*$ of $k(\geq 1)$ new facilities such that $|\bigcup_{f \in F^*} \mathcal{U}(f, \mathcal{F} \cup F^*)|$ is maximized, for $F^* \subset \mathbb{R}^2 \backslash \mathcal{F}$.

Clearly, the 1-MaxCov problem is nothing but the MaxCov problem as discussed by Cabello et al. [3]. In this paper, we study the 2-MaxCov problem. Our objective is to place two new facilities $f$ and $f'$ such that the total number of users in $\mathcal{U}$ served by $f$ and $f'$ ($|\mathcal{U}(f, \mathcal{F} \cup \{f, f'\}) \cup \mathcal{U}(f', \mathcal{F} \cup \{f, f'\})|$) is maximized, for $f, f' \in \mathbb{R}^2 \backslash \mathcal{F}$. We begin by showing that if all the users

*Indian Statistical Institute, Kolkata, West Bengal, India, bhaswar.bhattacharya@gmail.com, nandysc@isical.ac.in.

and facilities are restricted to lie on a single straight line, then the 2-MaxCov problem can be solved in $O(n \log n)$ time. Using this idea, we give an algorithm for solving the general 2-MaxCov problem, in the $L_1$ and $L_\infty$ metrics, with worst case running time and space $O(n^2 \log n)$. Finally, we give an $O(n \log n)$ time algorithm for the 2-MaxCov problem in the $L_2$ metric for $m = 1$ (i.e., with only one existing facility). We also consider the obnoxious version of this problem and obtain an $O(n \log n)$ time algorithm for solving it in different metrics.

These problems can be interpreted as the simultaneous location of two new facilities in a competitive environment [7, 9]. Imagine that a set of already existing facilities are serving the users of a town. A new company, with the aim to compete with the existing facilities, now wishes to establish two outlets in the town simultaneously. The problem of maximizing the profit of the company, in the sense that it serves the maximum possible number of users, reduces to the 2-MaxCov problem.

## 2 The 2-MaxCov Problem

In this section we present our results for solving the 2-MaxCov problem in the $L_1$, $L_2$, and $L_\infty$ metrics. For a pair of points $p$ and $q$ in the plane the distances in $L_1$, $L_2$, and $L_\infty$ metric are denoted by $d_1(p, q)$, $d_2(p, q)$, and $d_\infty(p, q)$, respectively. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be the set of users and $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$ be the set of existing facilities. For every user $u_i \in \mathcal{U}$, we denote by $\phi(u_i)$ the nearest facility of $u_i$ in $\mathcal{F}$. The *nearest facility disk* $R_i$ of a user $u_i$ is the region such that if another facility $f$ is placed in that region, $\phi(u_i)$ will no longer remain the nearest facility for $u_i$, and $f$ becomes the nearest facility of $u_i$. Clearly, the interior of the nearest facility disk for each user in any metric does not contain any facility point.

Let $\mathcal{A}$ be the arrangement of the set of $n$ nearest facility disks $\{R_1, R_2, \ldots, R_n\}$, where $R_i$ corresponds to the user $u_i$. The 1-MaxCov problem can be solved by finding the cell $c_{\max}$ of maximum depth in the arrangement $\mathcal{A}$, where the depth of a cell is the maximum number of nearest facility disks that overlap on that cell.

In the 2-MaxCov Problem, we have to place two new facilities $f$ and $f'$ such that $|\mathcal{U}(f, \mathcal{F} \cup \{f, f'\}) \cup \mathcal{U}(f', \mathcal{F} \cup \{f, f'\})|$ is maximized, for $f, f' \in \mathbb{R}^2 \setminus \mathcal{F}$. Suppose one of the new facilities, say $f$, is placed at some cell $c$ of $\mathcal{A}$ where the disks $\{R_{i_1}, R_{i_2}, \ldots, R_{i_k}\}$ intersect, for some $\{i_1, i_2, \ldots, i_k\} \subset \{1, 2, \ldots, n\}$. Then the best possible position of another facility $f'$, given the placement of the facility $f$, such that $|\mathcal{U}(f', \mathcal{F} \cup \{f, f'\})|$ is maximized is the region of maximum depth in the arrangement of $\{R_1, R_2, \ldots, R_n\} \setminus \{R_{i_1}, R_{i_2}, \ldots, R_{i_k}\}$. Therefore, the optimum placement of the two facilities $f$ and $f'$ in the 2-MaxCov problem can be obtained by checking each cell $c \in \mathcal{A}$ as the position of $f$, and then compute the best position of $f'$ as mentioned above. In Figure 1, the optimal positions of $f$ and $f'$ are also shown using boxes ($\square$). Note that neither $f$ nor $f'$ is in the cell $c_{\max}$.

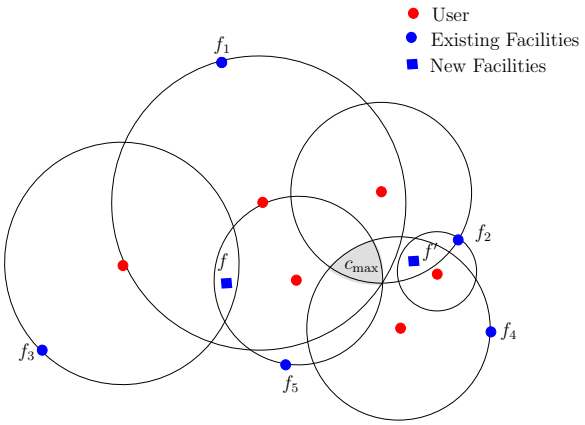


Figure 1: Demonstration of 2-MaxCov problem in $L_2$

### 2.1 The 2-MaxCov Problem on a Line

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be the set of users and $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$ be the set of existing facilities all lying on a straight line $L$. For every user $u_i \in \mathcal{U}$ denote by $I_i$ the interval on $L$ with center at the point $u_i$ and length $2|d(u_i, \phi(u_i))|$, where $d(a, b)$ is the distance between a pair of points $a, b$ on $L$. As mentioned earlier, the interior of this interval does not contain any other facility. The end-points of the intervals $\mathcal{I} = \{I_i | u_i \in \mathcal{U}\}$ split the line $L$ into $k$ cells, namely $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$, where $k \leq 2n - 1$. We consider all these regions for the placement of $f$. When $f$ is placed at a point $\rho$ in a cell, the subset of intervals $I_\rho \subseteq \mathcal{I}$ that overlap on the point $\rho$, are removed. Next, a point $\rho'$ on the line $L$ is identified for the placement of $f'$ where maximum number of intervals in $\mathcal{I} \setminus \mathcal{I}_\rho$ overlap. We now give a formal description of the algorithm.

Put a point $\pi_i$ in the proper interior of each cell $A_i$ of the arrangement $\mathcal{A}$. These points are referred to as *sites*. Each point in $\Pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$ is attached with a count $\chi_i$ indicating the number of intervals passing through the site $\pi_i$. If a facility is positioned at $\pi_i$, the nearest facility of $\chi_i$ users will be changed to $\pi_i$. So, we need to search for another facility location $\pi \in \Pi$ so that it can serve maximum number of users apart from these $\chi_i$ users. The $\chi_i$ values are calculated by considering the end-points of the intervals of $\mathcal{I}$ in order. We construct a leaf-search balanced binary tree $\mathcal{T}$ with the sites in $\Pi$ at its leaves in order. The internal nodes of $\mathcal{T}$ contain the discriminant values as is done in the interval tree [1]. Two integer fields, namely $max\_\chi$ and $\eta$ are attached with each non-leaf node of $\mathcal{T}$. The $max\_\chi$ field indicates the maximum $\chi$ value among the leaves in the subtree rooted at that node and $\eta$ indicates the excess count as in [8]. The sites in $\Pi$ are processed in left-to-right order. When a site $\pi \in \Pi$ is processed, let $I$ denote the set of interval containing $\pi$. For each interval $\mu \in I$, we identify the sites in $\Pi$ that are contained

in $\mu$, and reduce their count. During the processing, when one moves from one site $\pi$ to its next site, either a new interval $\mu$ begins or an existing interval $\nu$ ends. In the former case, the $\chi$ field of all the sites in the new interval $\mu$ is reduced, and in the latter case, the $\chi$ field of all the sites in that old interval are increased. Both these tasks can be done in $O(\log n)$ time by traversing the two paths corresponding to the two end-points of the concerned intervals ($\mu$ or $\nu$) from the root in $\mathcal{T}$, and using the excess fields $\eta$ attached to each node on these two paths. After the updates of $\chi$ values, we compute $max\_\chi$ and the corresponding site $\pi'$ by a bottom-up traversal along those two paths. If $\pi$ is considered for the placement of $f$, then $f'$ will be placed at $\pi'$, and both of them covers $\chi^*(\pi) = \chi(\pi) + \chi(\pi')$ users. Finally, we choose that site for which $\chi^*$ is maximum.

**Theorem 1** *The* 2-MaxCov *problem on a line can be solved in $O(n \log n)$ time.* □

### 2.2 The 2-MaxCov Problem in $L_1$ and $L_\infty$ Metrics

With an appropriate rotation of the coordinate axes by an angle $45^0$ in the $L_1$ metric, we may consider each $R_i \in \mathcal{R}$ as an axis-parallel square with center (intersection of two diagonals) at $u_i$, in both the $L_1$ and $L_\infty$ metrics,.

Let us consider the arrangement $\mathcal{A}$ of the members in $\mathcal{R}$. It may have $O(n^2)$ number of cells. Each cell in this arrangement is an axis-parallel rectangle. If the end-points of the lower boundary of a cell $C$ are $(x', y)$ and $(x'', y)$, then we choose a point $(\frac{x'+x''}{2}, y+\epsilon)$ inside that cell for the placement of a new facility, where $\epsilon$ is a very small positive constant. These points will be referred to as the *sites*. Note that, the sites are arranged in at most $n$ horizontal lines corresponding to the lower boundaries of $n$ squares. We store the sites in a range tree $\mathcal{T}$ [1], where each leaf is attached with the corresponding $\chi$ value (the number of squares containing that site), and each internal node is attached with two integer fields $max\_\chi$ and $\eta$. This needs $O(n^2 \log n)$ space. Note that while deleting/inserting a square the cumulative increment/decrement of $\chi/max\_\chi$ values of nodes of $\mathcal{T}$ is possible as we have done in Subsection 2.1. Thus, the time required for removing or adding a square, the corresponding updating of $max\_\chi$ values, and finding the site having maximum $max\_\chi$ value are all $O(\log^2 n)$ in the worst case. However, using fractional cascading [1] this can be reduced to $O(\log n)$.

We process the event-points on each horizontal line separately. While processing the sites on a horizontal line as the possible placements of $f$, $n$ squares may be deleted and inserted in the data structure for computing the best possible position for the corresponding placement of $f'$. This needs $O(n \log n)$ time. Thus, we have the following theorem.

**Theorem 2** *In the $L_1$ and $L_\infty$ metrics, the* 2-MaxCov *problem can be solved in $O(n^2 \log n)$ time and space.* □

If instead of range tree, 2D-tree is used to store the sites, the space complexity reduces to $O(n^2)$, but the time complexity increases to $O(n^{2.5} \log n)$.

### 2.3 2-MaxCov problem in $L_2$ metric

Here, the nearest facility disks are circles of different radii. The number of cells in an arrangement of $n$ circles can be $O(n^2)$, and can be computed in $O(n^2)$ time using $O(n^2)$ space; the region where the maximum number of circles overlap can also be computed in $O(n^2)$ time [3]. Thus, the naive approach for solving the 2-MaxCov problem in $L_2$ metric needs $O(n^4)$ time and $O(n^2)$ space. We present an efficient algorithm for the case $m = 1$.

#### 2.3.1 2-MaxCov problem with one existing facility

If there is only one existing facility, say $f_1$, all the circles $\{R_1, R_2, \ldots, R_n\}$ will pass through the point $f_1$.

**Observation 1** *Each of the new facilities $f$ and $f'$ will lie in a cell of $\mathcal{A}$ having $f_1$ as a vertex.* □

Consider a circle $\rho$ centered at $f_1$ and not containing any other vertex of the arrangement $\mathcal{A}$. Each $R_i$ creates an arc on $\rho$. Thus, we have an arrangement $\mathcal{A}^*$ of these arcs on the periphery of the circle $\rho$. For each user there is a point on the periphery of the circle $\rho$ which is closer to it than the existing facility. So, we need to consider each cell of $\mathcal{A}^*$ for the placement of $f$. The number of cells in $\mathcal{A}^*$ is $O(n)$. After placing $f$, the circles, that are covered by $f$, are removed from $\mathcal{A}^*$. The placement of $f'$ should be the cell in the arrangement of the remaining arcs having maximum degree. The naive algorithm for this problem needs $O(n^2 \log n)$ time, where that the arrangement of the remaining arcs is computed afresh in each time. This can be avoided by following the same technique as in Section 2.1 for processing the arcs $I_i \in \mathcal{A}^*$ to identify a pair of sites for the placement of $f$ and $f'$. Thus, we have the following theorem:

**Theorem 3** *The* 2-MaxCov *problem with one existing facility can be solved in $O(n \log n)$ time.* □

## 3 The Obnoxious Version

In the preceding sections, we assumed that every user avails the service from its nearest facility. However, in obnoxious facility location problems, the customer no longer finds a facility desirable and wants to stay as far way from it as possible. Given a set $\mathcal{U}$ of $n$ users and a set $\mathcal{F}_o$ of $m$ obnoxious facilities, for every facility $f \in \mathcal{F}_o$ let $\mathcal{U}(f, \mathcal{F}_o) = \{u \in \mathcal{U} | d(u, f) \geq d(u, f'), \forall\ f' \in \mathcal{F}_o \backslash \{f\}\}$. We now introduce the obnoxious version of the $k$-MaxCov problem as follows:

$k$-Farthest-MaxCov problem: Given a set $\mathcal{U}$ of $n$ users, a set $\mathcal{F}_o$ of $m$ existing obnoxious facilities with $m < n$, and a bounded region $C \subset \mathbb{R}^2$, find the placement of a set $F^*$ of $k$ ($\geq 1$) new obnoxious facilities such that $|\bigcup_{f \in F^*} \mathcal{U}(f, \mathcal{F}_o \cup F^*)|$ is maximized, for $F^* \subset C \backslash \mathcal{F}_o$.

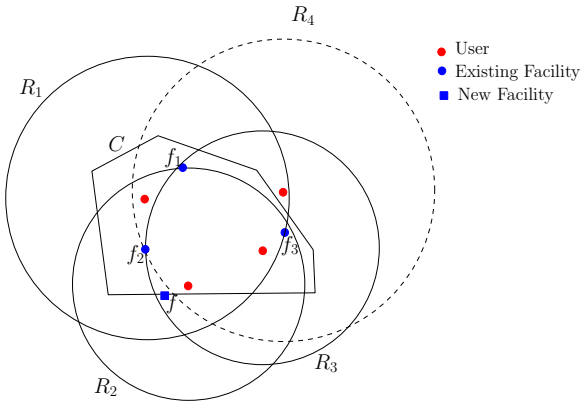


Figure 2: 2-Farthest MaxCov problem: the facility $f$ serves the user $u_4$ and $C \subset \bigcup_{i=1}^{3} R_i$.

The case $k = 1$ has been studied by Cabello et al. [3]. They referred it the Farthest-MaxCov problem, and gave an $O(n \log n)$ time algorithm for solving the problem in all the $L_1$, $L_2$ and $L_\infty$ metrices.

We study the 2-Farthest MaxCov problem in all the $L_1$, $L_2$, and $L_\infty$ metrics. For every user $u_i \in \mathcal{U}$ we denote by $\phi_o(u_i)$ its farthest facility. Let $R_i$ be the farthest facility disk with center at the point $u_i$ and radius $d(u_i, \phi_o(u_i))$. Let $\mathcal{A}_o$ denote the arrangement produced by the set of the $n$ such disks $\{R_1, R_2, \ldots, R_n\}$.

**Observation 2** *If a new facility $f^* \notin \mathcal{F}_o$ is placed in some cell $A_i \in \mathcal{A}_o$, then the number of users that are served by $f^*$ is the number of disks that do not contain the cell $A_i$.* □

**Lemma 4** *If the desired region $C$ is bounded, both the new facilities $f$ and $f'$ will lie on the boundary of $C$.*

**Proof.** Observe that all the facilities in $\mathcal{F}_o$ are contained in every circle $R_i$; this implies that $\bigcap_{i=1}^{n} R_i$ contains all the facilities $\mathcal{F}_o$. Let $f$ and $f'$ be any two points in $\mathcal{A}_o \cap C$ and $\alpha$ be a point in $\bigcap_{i=1}^{n} R_i$. Now, the rays $\overrightarrow{\alpha f}$ and $\overrightarrow{\alpha f'}$ can only leave disks one after one. Therefore, if both the directed lines $\overrightarrow{\alpha f}$ and $\overrightarrow{\alpha f'}$ intersect the boundary of $C$ at the points $a$ and $b$ respectively, then we have $|\mathcal{U}(a, \mathcal{F}_o \cup \{a, b\}) \cup \mathcal{U}(b, \mathcal{F}_o \cup \{a, b\})| \geq |\mathcal{U}(f, \mathcal{F}_o \cup \{f, f'\}) \cup \mathcal{U}(f', \mathcal{F}_o \cup \{f, f'\})|$. □

Now, suppose an obnoxious facility $f$ is already placed somewhere on $\mathcal{A}_o \cap \delta C$, where $\delta C$ is the boundary of the region $C$. The circles $R_i$ which do not contain the facility $f$ will correspond to the users which will be served by $f$. Suppose the point $f$ lies inside the circles $R_{i_1}, R_{i_2}, \ldots, R_{i_p}$, for some $\{i_1, i_2, \ldots, i_p\} \subset \{1, 2, \ldots, n\}$. Here the following two cases can arise:

- $\delta C \setminus \bigcup_{j=1}^{p} R_{i_j} \neq \emptyset$ : Here a region on $\delta C$ exists where no circle of $\{R_{i_1}, R_{i_2}, \ldots, R_{i_p}\}$ overlap. In this case, the best possible location of the next facility $f'$, given the placement of $f$, is any point on $\delta C \setminus \bigcup_{j=1}^{p} R_{i_j}$. In this case, the new facilities $f$ and $f'$ serve all the users in $\mathcal{U}$.
- $\delta C \setminus \bigcup_{j=1}^{p} R_{i_j} = \emptyset$: Here $f \in \bigcap_{j=1}^{p} R_{i_j}$ and $C \subset \bigcup_{j=1}^{p} R_{i_j}$ (see Figure 2). In other words, every point on $\delta C$ is covered by at least one of the circles in $\{R_{i_1}, R_{i_2}, \ldots, R_{i_p}\}$. Here, we need to place $f'$ on $\delta C$ where minimum number of members in $\bigcup_{j=1}^{p} R_{i_j}$ overlap.

**Theorem 5** *The* 2-Farthest MaxCov *problem in any one of $L_1$, $L_2$ and $L_\infty$ metrics can be solved in $O(n \log n)$ time.*

**Proof.** Note that $\delta C$ is a closed curve of constant complexity, say $k$. We compute the segments of $\delta C$ intersected by $R_{i_j}$ for $j = 1, 2, \ldots, p$. Since each segment of $\delta C$ can intersect $R_{i_j}$ in at most one segment in each of the aforesaid three metrics, the number of arc-segments of $R_{i_j}$ on $\delta C$ is at most $k$. Thus, the total number of arc-segments created on $\delta C$ by the circles $R_{i_j}$ for $j = 1, 2, \ldots, p$ is at most $kn$, since $p \leq n$.

Similar to the 2-MaxCov problem with one existing facility, we have an arrangement of at most $kn$ arcs on $\delta C$, which creates at most $2kn$ cells. Each cell is covered by at least one arc. We need to find two cells such that the total number of distinct arcs covering these two cells is minimum. Using the same data structure as in the Subsection 2.1, this problem can be solved in $O(n \log n)$ time. □

## References


[1] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwartzkopf, Computational Geometry - Algorithms and Applications, Springer, 1997.

[2] B. B. Bhattacharya, *Maximizing Voronoi regions of a set of points enclosed in a circle with applications to facility location*, J. Math. Model. Algor., to appear, http://dx.doi.org/10.1007/s10852-010-9142-0.

[3] S. Cabello, J. M. Diáz-Báñez, S. Langerman, C. Seara, and I. Ventura, *Facility location problems in the plane based on reverse nearest neighbor queries*, Eur. J. Operations Research, Vol. 202(1), 99–106, 2010.

[4] O. Cheong, A. Efrat, and S. Har-Peled, *On finding a guard that sees most and a shop that sells most*, Discrete Computational Geometry, vol. 37, pp. 545-563, 2007.

[5] F. Dehne, R. Klein, and R. Seidel, *Maximizing a Voronoi region: the convex case*, Int. J. Computational Geometry and Applications, vol. 15, pp. 463-475, 2005.

[6] Z. Drezner and H. W. Hamacher (Eds.), Facility Location: Applications and Theory, Springer, 2002.

[7] H. A. Eiselt, G. Laporte, and J. F. Thisse, *Competitive location models: A framework and bibliography*, Transportation Science vol. 27, pp. 44-54, 1993.

[8] S. C. Nandy and B. B. Bhattacharya, *A unified algorithm for finding maximum and minimum object enclosing rectangles and cuboids*, Computers and Mathematics with Applications, vol. 29, pp. 45-61, 1995.

[9] F. Plastria, *Static competitive location: An overview of optimisation approaches*, Eur. J. Operations Research, vol. 129, pp. 461-470, 2001.