Planar maximal covering location problem under block norm distance measure

2 authors, including:

Hassan Younies

28 PUBLICATIONS   234 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

none, moving out of country and looking for job. any job you have? View project

Planar maximal covering location problem under block norm distance measure

H Younies[1] and GO Wesolowsky[2],*

[1]*United Arab Emirate University, Al-Ain, UAE; and* [2]*McMaster University, Hamilton, Ontario, Canada*

This paper introduces a new model for the planar maximal covering location problem (PMCLP) under different block norms. The problem involves locating g facilities anywhere on the plane in order to cover the maximum number of n given demand points. The generalization, in this paper, is that the distance measures assigned to facilities are block norms of different types and different proximity measures. First, the PMCLP under different block norms is modelled as a maximum clique partition problem on an equivalent multi-interval graph. Then, the equivalent graph problem is modelled as an unconstrained binary quadratic problem (UQP). Both the maximum clique partition problem and the UQP are NP-hard problems; therefore, we solve the UQP format through a genetic algorithm heuristic. Computational examples are given.

1. Introduction

Consider n demand points, with equal weights, on a plane. We study the problem of citing g facilities anywhere on the plane such that the demand covered is maximized. This problem is the well-known maximal covering location problem (MCLP) introduced by Church and ReVelle (1974). When facilities can be located anywhere on the plane the problem has been called the planar maximal covering location problem (PMCLP) (Church, 1984). Church (1984) provided a solution technique for the PMCLP under the rectilinear and Euclidean distances (RPMCLP and EPMCLP, respectively). In the review of covering problems by Schilling *et al* (1993), the MCLP has been studied only under rectilinear, Euclidean, and spherical distance measures. No attempts were made to apply other distance measures in the PMCLP.

Distance functions can be classified according to their unit ball contours into block norms and round norms. The family of distance functions with contours made up from flat segments forming polytopes in N-dimensional (a polyhedral in R^2) are called block (or polyhedral) norms. The round norms are those norms with no flat spot on their unit ball contours (this includes Euclidean distance and the family of l_p norms ($1 < p < \infty$). Ward and Wendell (1980) introduced a block norm distance measure, called the one-infinity norm, as a linear combination of rectilinear and Tchebycheff norms. Let λ'_1 and λ'_2 be positive numbers, not both zero; then, the

one-infinity norm is defined as $l_{(\lambda'_1, \lambda'_2)} = \lambda'_1 l_1 + \sqrt{2}\lambda'_2 l_\infty$. Figure 1, shows the unit ball of the one-infinity norm defined by Ward and Wendell. The one-infinity norm can be interpreted as the weighted sum of the shortest rectilinear and diagonal distances—that is, the travelling distance either along or parallel to the one-infinity unit ball contour diagonals. The ratio λ'_1 / λ'_2 represents the proportion of the trip travelled along the rectilinear distance to the proportion travelled along the diagonal roads (the block norms diagonals). For (λ'_1, λ'_2) values of $(1, 0)$ and $(0, 1/\sqrt{2})$, the one-infinity norm changes to the rectilinear norm and Tchebycheff norm, respectively.

Ward and Wendell (1985), extended this idea to the family of block norms, which are polygonal in shape, and called them additive or mixed norms. As in the one-infinity norm, the directions of travel in an additive norm are those directions that are parallel to its unit ball contours diagonals.

In this paper, we refer to block norms with r diagonals and r allowed directions of travel as block norms of r degree. Therefore, we refer to the rectilinear and Tchebycheff norms as block norms of second degree, and the one-infinity norm as a fourth degree block norm. Hence, the unit ball of a block norm with $r = 3$ is hexagonal in shape.

Ward and Wendell (1980, 1985) show that the one-infinity norm is comparable with the weighted l_p norm as a distance predictor to geographical data. In addition to that, the one-infinity norm allows for a linear formulation of the Fermat–Weber problem and for the Rawls problem. One of the main results of Thisse *et al* (1984) is that block norms can represent a round norm as accurately as desired by a higher degree block norm, that is as $r \to \infty$ the block norm becomes

Correspondence: GO Wesolowsky, Faculty of Business, McMaster University, Hamilton, Ontario, Canada, L8S 4M4.
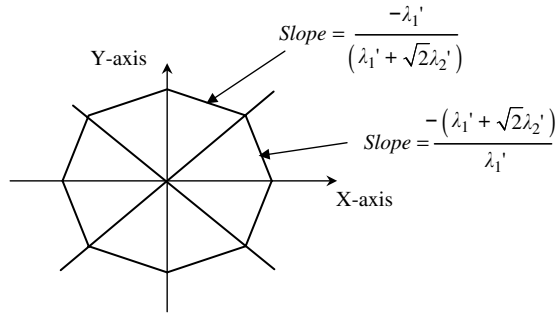E-mail: wesolows@mcmail.cis.mcmaster.ca

Figure 1 Unit ball (contour) of a one-infinity norm in R^2.

a round norm. Other block norm properties have been reported by Thisse *et al* (1984).

The mixed norms location problem concerns the location problem where facilities may have different coverage measure under different norms. Durier and Michelot (1985) investigated the Fermat–Weber problem under mixed norms; Plastria (1992) investigated the Fermat–Weber problem with mixed skewed norms; Nickel (1998) studied the centre problem under different polyhedral gauges (a gauge is a convex function that allows for asymmetric distance) where the set of demand points is allowed to have its own polyhedral norm. This work deals with the PMCLP, where facilities can have different covering measures under different block norms.

Since the block norms' unit ball shapes are polygons with opposite sides parallel, we will tackle the problem of the PMCLP under block norms as covering by parallel-sided polygonal shapes (PSP). Henceforth, we will use the term covering by block norms or covering by parallel-sided polygonal shapes interchangeably. In addition, we will use the term maximum covering by parallel-sided polygonal shapes (MCPSP) for maximum covering under different block norms. Thus the problem can be stated as the follows:

Problem statement Given n demand points in R^2, $a_i = (a_{ix}, a_{iy})$, $i = 1, ..., n$, with equal weights, find the locations of g parallel-sided polygons that will cover the maximum number of points. Any point covered is to be assigned to one polygon only.

Younies and Wesolowsky (2004) proposed a mixed integer program formulation (MIP) for MCLP using second-degree block norms (squares, rectangular shapes and parallelogram shapes). An extension of their MIP formulation to block norms of higher degrees is shown in the Appendix.

Unfortunately, while Younies and Wesolowsky (YW) MIP formulation is versatile and can be adapted to many cases, it cannot handle, efficiently and simultaneously, block norms of different degrees as well as large-size problems. Therefore, we propose a different approach to handle PMCLP under different block norms.

Our approach will be as follows: we will show in Section 2, a mathematical model for PMCLP, based on the idea of

interval graphs and multi-interval graphs, under different block norms as a maximum clique partition problem in an equivalent interval graph. The idea of using interval graph and maximum cliques in location problems has been used by Aneja *et al* (1988) for the *m*-center problem under the rectilinear distances measure. In Section 3, we convert the maximum clique partition problem into an unconstrained binary quadratic problem (UQP). This new problem representation, as a UQP, is more suitable for the heuristic approach implemented in this work, namely the genetic algorithms (GA) heuristic technique. Section 4 shows our approach to a GA implementation of the UQP of Section 3.

Finally, Section 5 reports on our computational experience, and we conclude this paper in Section 6.

2. Transforming maximum covering by PSP's to a clique problem on a graph

Let the undirected, unweighted arbitrary graph G with a set of vertices V and an edge set E be denoted as $G = (V, E)$. A subgraph $G' = (V', E')$ of a graph $G = (V, E)$ is a graph having all of its vertices and edges in G, that is, $V' \subseteq V$ and $E' \subseteq E$. A complete subgraph, also known as a clique, is a graph that has every pair of its vertices adjacent to each other, that is, $\forall (u, v) \in V'(u, v) \in E'$. The maximum clique ($MC$) is a complete subgraph $G' = (V', E')$, $V' \subseteq V$, $E' \subseteq E$, and $|V'|$ is at least as large as the vertex set of any other complete subgraph in G. The complementary graph $\overline{G} = (V, \overline{E})$ of a graph $G = (V, E)$ is a graph with the same set of vertices, but two vertices are adjacent in $\overline{G}$ if and only if they are not adjacent in G. A subset of vertices $V'' \subseteq V$ is said to be an independent set if no two vertices in the subset are adjacent, that is, $\forall (u, v) \in V''$, $(u, v) \notin E$. An independent set is maximal if any vertex not in the set is adjacent to at least one vertex in the independent set. The maximum independent set on a graph $\overline{G} = (V, \overline{E})$ is equal to the maximum clique on a graph $G = (V, E)$.

The transformation of maximum covering by block norms problems or, alternatively, the maximum covering by parallel-sided polygons (MCPSP) in a graph problem is based on the idea of an interval graph. There are many different ways to define an interval graph. The definition in Diestel (1997) is the best one to illustrate our purpose.

Definition 2.1. (Diestel (1997), p. 120) A graph $G = (V, E)$ is called an interval graph if there exists a set $\{t(v)|v \in V\}$ of real intervals such that for any two vertices, $(u, v) \in V$, $t(u) \cap t(v) \neq \varnothing$ if and only if $(u, v) \notin E''$.

For an extensive discussion of interval graphs, the reader may also refer to Fishburn (1985).

Figure 2 shows an example of an interval graph. Figure 2a shows a graph where vertices are connected if assigned
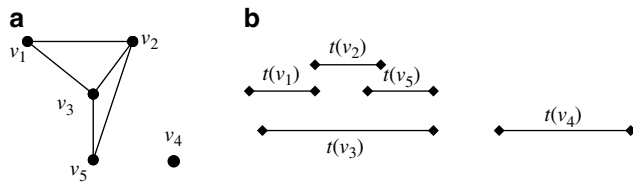
Figure 2   Intervals and the corresponding interval graph.

labeled intervals, shown in Figure 2b, overlap. Note that the size of intervals is not necessarily unique in Figure 2b and is given for illustration purposes only.

Consider n points in R^2; $a_i = (a_{ix}, a_{iy})$, $i = 1, \ldots, n$ and straight lines with inclination angle θ from the X-axis, passing through each point. Let the sorted Y-intercepts of the lines be $b_i, i = 1, \ldots, n$.

Construct a set of real intervals $t(a_i) = [b_i, b_i + \delta]$, $i = 1, \ldots, n$; where $\delta > 0$. Define an interval graph $G = (V, E)$, where $|V| = n$, from the intervals $t(a_i)$, $i = 1, \ldots, n$. The following proposition is self-evident.

Proposition 2.1. *If every demand point $a_i = (a_{ix}, a_{iy})$, $i = 1, \ldots, n$, in R^2 is represented by a vertex $v_i \in V$, then for any two demand points a_i, a_j there are equivalent vertices v_i, v_j such that $(v_i, v_j) \in E$ if and only if $t(v_i) \cap t(v_j) \neq \varnothing$.*

The proposition is illustrated in Figure 3a and b. Figure 3a shows an example of five points in space, their Y-intercepts, and the intervals assigned for each point. As evident from the figure, some points may have the same Y-intercept values, and the same real interval assigned to them. Figure 3b represents the equivalent interval graph. From proposition 2.1, assigning an interval δ to the Y-intercepts of each point would generate the interval graph $G = (V, E)$, where $t(v_i) \cap t(v_j) \neq \varnothing$ if the distance between the points' Y-intercepts is less than, or equal to, δ. If the distance between the points' Y-intercepts is greater than δ, then they will not be connected in G.

In definition 2.1 we obtained the interval graph by assigning a real interval, on the real line, for each vertex of the interval graph. Trotter and Harary (1979) extended definition 2.1 to double- and multiple-interval graphs. The multi-interval graph is that graph which represents the case where more than one set of real intervals, on more than one real line, are assigned for each vertex.

For the case when one has p intervals assigned to every vertex, on p-real lines, the graph is called an p-interval graph. In the next definition we give a formal definition for the multi-interval graph.

Definition 2.2.   Multi-interval graph. A graph $G = (V, E)$ is called a p-interval graph if one can assign to each $v \in V$ a set of real intervals $t_j(v), j = 1, \ldots, p$; $p \geqslant 2$ such that $\forall j \ t_j(v_i) \cap t_j(v_k) \neq \varnothing$; $i, k = 1, \ldots, n$ if and only if $(v_i, v_k) \in E$.
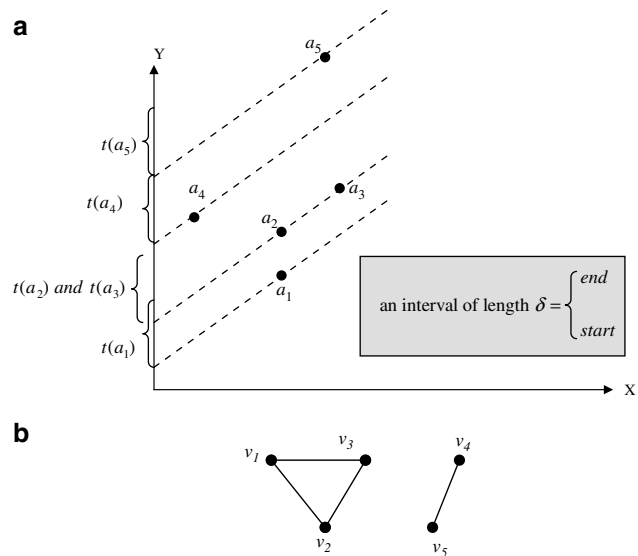

Figure 3   (a) Points in R^2 and the corresponding intervals assigned to their Y-intercepts. (b) The equivalent interval graph of (a).

In other words, given p-interval graphs on the same vertex set and possibly different edge sets $G_j = (V, E_j)$, $j = 1, \ldots, p$; the p-interval graph $G = (V, E)$ is a graph on the same vertex set V with an edge set defined as $E = \bigcap_{j=1}^p E_j \neq \varnothing$.

When $|t_j(v_i)| = |t_j(v_k)|$ for $j = 1, \ldots, p$; $i, k = 1, \ldots, n$; the graph is known as a unit- p-interval graph. Without loss of generality, we will call this graph a simple p-interval graph.

In this work we need to deal only with simple multi-interval graphs; that is, points will have equal length intervals assigned to them on the same real line.

Since every vertex represents one point in R^2, a maximum clique on the graph will represent the maximum number of points in R^2 with Y-intercepts within a distance δ from each other. Consider n points in R^2, a pair of parallel straight lines L and L' with inclination angle θ, and a perpendicular distance between the two lines d. Consider also the translation of the two lines in R^2 to enclose the maximum number of points. Let $\delta = I$ in proposition 2.1; then, for a pair of parallel straight lines we get the following corollary:

Corollary 2.1.   *For n points in R^2, finding the maximum number of points enclosed by a pair of parallel straight lines, L and L', with distance I between the Y-intercepts of the two lines, is equivalent to finding the maximum clique on an equivalent interval graph.*

Proof   Recall that every vertex in the equivalent interval graph G represents one point in R^2; that is, $|V| = n$. Suppose that the maximum number of points enclosed by the parallel straight lines is Ω. Then by proposition 2.1, assigning an interval $\delta = I$ to the Y-intercepts of lines passing through

every point in R^2 would generate an equivalent interval graph. This implies that there is a set of points, S, where $|S| = \Omega$, and $\max_{i \in S}(b_i) - \min_{i \in S}(b_i) \leqslant I$. Since all points in S are within a distance of I from each other, this graph will be a complete graph of size Ω; that is, it forms a clique of size Ω. Let $G = (V, E)$ represent the interval graph for the n points in R^2. Suppose that the maximum clique size is not equal to Ω; however, this would assume that there is another set of points S' of larger cardinality lying within the straight lines—which is a contradiction. Thus, the set of points, S, forms a maximum clique of size Ω. $\quad \square$

Consider a 2χ parallel-sided polygon, $\chi \geqslant 2$, with given side lengths and inclination angles from the X-axis. Using definition 2.2, one can transform the covering by polygon to a maximum clique problem in a simple p-interval graph. Note that for a diamond shape χ equals two; for a hexagon, χ equals three, and for one-infinity norm χ equals four.

Lemma 2.1. *Consider a polygonal shape of K parallel sides, where K is an even number. Let the graphs $G_j = (V, E_j)$, $j = 1, \ldots, K/2$ be the equivalent interval graph for every two polygon parallel sides. Construct a graph $G = (V, E)$ such that $E = \bigcap_{j=1}^{K/2} E_j$. The maximum covering by a parallel-sided polygonal shape is equivalent to a maximum clique problem on the equivalent multi-interval graph, $G = (V, E)$.*

Proof For every two parallel sides, construct an equivalent interval graph. One will get $K/2$ graphs with the same vertex set and different edge sets $G_j = (V, E_j)$, $j = 1, \ldots, K/2$. The vertices in G that are adjacent to each other are those with an edge on all $K/2$ interval graphs. This implies that these points are within all polygon sides. Therefore, the maximum number of points covered by the parallel-sides polygon are the points that are pairwise adjacent in a graph G; that is, they are the points corresponding to the maximum clique (MC) on graph $G = (V, E)$. $\quad \square$

The adjacency matrix $A = [\alpha_{ij}]$ is a $|V| \times |V|$ matrix in which $\alpha_{ij} = 1$ if $(v_i, v_j) \in E$ and 0 otherwise. For the purpose of our problem, the diagonal elements of the adjacency matrix will be ones; that is, $\alpha_{ii} = 1$. We denote the complementary adjacency matrix by $\overline{A} = [\overline{\alpha_{ij}}]$, in which $\overline{\alpha_{ij}} = 0$ iff $\alpha_{ij} = 1$ in A. A simple algorithm can generate the adjacency matrix for the equivalent p-interval graph.

In the case where the given g polygonal shapes are identical, each shape will generate a graph such that $E_k = E_j \ \forall k, \ j = 1, \ldots, g$, and the problem of maximal covering by polygonal shapes will be equivalent to partitioning the vertex set V in graph $G = (V, E)$ into nonempty disjoint complete subgraphs with vertex sets $V_1, V_2, \ldots, V_g$; such that $\sum_{k=1}^g |V_k| \leqslant |V|$ is maximized.

However, in the case of non-identical g polygonal shapes, each shape will generate a graph on the same vertex set and different edge sets $G_j(V, E_j)$, $j = 1, \ldots, g$. The problem of maximal covering by polygonal shapes will be equivalent to partitioning the vertex set V into nonempty disjoint complete subgraphs with vertex sets $V_1, V_2, \ldots, V_g$, such that $\sum_{k=1}^g |V_k| \leqslant |V|$ is maximized, and for $v_i, v_j \in V_k (v_i, v_j) \in E_k; \ i, j = 1, \ldots, n, \ k = 1, \ldots, g$.

Observation 1 Upper limit on the maximum number of points covered by g polygonal shapes.

Consider g different polygonal shapes and their corresponding g graphs $G_j(V, E_j), j = 1, \ldots, g$. Let MC_k^* be the maximum clique on graph k, and $|MC_k^*|$ be the size of such a clique. Denote the maximum number of points covered by the g parallel-sided polygonal (PSP's) shapes as maximum covering by parallel-sided polygons (MCPSP). Since each shape can cover at most MC_k^* points, it is evident that in the optimal solution of MCPSP (OPT_MCPSP) it is true that $OPT\_MCPSP \leqslant \sum_{k=1}^g |MC_k^*|$.

3. Formulating the maximum clique as an unconstrained quadratic binary problem (UQP)

In this section, we first show a zero–one formulation for finding the maximum covering by g PSP (MCPSP). We start by showing the zero–one formulation for the maximum clique problem (MCP) on a graph, then we extend the formulation to the MCPSP problem.

The following notation is used in the zero–one formulation for the MCP:

Inputs

$G = (V, E)$ is a graph with vertex set V and edge set E. $\overline{G} = (V, \overline{E})$ is the complement of graph G.

Decision variables

$$x_i = \begin{cases} 1, & \text{if vertex } i \text{ is chosen}; \quad i = 1, \ldots, n; \\ 0, & \text{otherwise} \end{cases}$$

$$\text{(P1)} \quad \text{Maximize} \quad \sum_{i=1}^n x_i \quad (1)$$

Subject to

$$x_i + x_j \leqslant 1, \quad \forall (i, j) \in \overline{E} \quad (2)$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, n \quad (3)$$

In the above zero–one formulation, the objective function (1) is to maximize the number of vertices chosen. Constraints (2) ensure that only adjacent points are chosen. Constraint (3) is the usual zero–one condition.

The following notation will be used for the zero–one formulation for partitioning vertex set V into nonempty

disjoint complete subgraphs with vertex sets $V_1, V_2, \ldots, V_g$, such that $\sum_{k=1}^{g} |V_k| \leqslant |V|$ is maximized, and for $v_i, v_j \in V_k (v_i, v_j) \in E_k; \; i, j = 1, \ldots, n, \; k = 1, \ldots, g.$

$G_k = (V, E_k)$ is multi-interval graph, $k = 1, \ldots, g$

$\overline{G_k} = (V, \overline{E_k})$ is the complement graph $G_k, \; k = 1, \ldots, g$

$$x_{ik} = \begin{cases} 1, & \text{if point (vertex) } i \text{ is assigned to graph } k; \\ & i = 1, \ldots, n; \; k = 1, \ldots, g \\ 0, & \text{otherwise} \end{cases}$$

A_k = adjacency matrix for graph k, $k = 1, \ldots, g$

α_{ijk} = row i and column j element in the adjacency matrix of graph k, $i = 1, \ldots, n; \; j = 1, \ldots, n;$ $k = 1, \ldots, g.$

The zero–one formulation will be

$$(\text{P2}) \quad \text{Maximize} \sum_{k=1}^{g} \sum_{i=1}^{n} x_{ik} \quad (4)$$

Subject to

$$x_{ik} + x_{jk} \leqslant 1, \quad \forall (i,j) \in \overline{E_k}, \; k = 1, \ldots, g \quad (5)$$

$$\sum_{k=1}^{g} x_{ik} \leqslant 1, \quad i = 1, \ldots, n \quad (6)$$

$$x_{ik} \in \{0, 1\}, \quad i = 1, \ldots, n, \; k = 1, \ldots, g \quad (7)$$

Constraints (6) ensure that a point is assigned to one shape only. Constraints (5) ensure only adjacent points can be chosen for any shape. This means that $x_{ik} + x_{jk} \leqslant 1$ if $\alpha_{ijk} = 0, \; i = 1, \ldots, n; \; j = 1, \ldots, n; \; k = 1, \ldots, g.$ The objective function (4) maximizes the sum of points assigned to different graphs. Constraint (7) is the usual zero–one assignment. Constraints (5) and (6) can be transformed to a penalty function in the unconstrained UQP.

The general UQP can be written as $f(x) = \mathbf{x}^{\mathrm{T}} Q \mathbf{x}$, where $\mathbf{x}$ is a binary vector and Q is an $n \times n$ matrix. Since Q is an identity matrix and all x's are binary, the equivalent UQP problem can be written as follows:

$$\text{UQP Maximize} \quad \sum_{k=1}^{g} \sum_{i=1}^{n} x_{ik} x_{ik} - P' - P'' \quad (8)$$

where $P' = \sum_{k=1}^{g} \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_{ik} x_{jk} \; \forall (i,j) \in \overline{E_k}$ is a penalty function representing Constraints (5), and $P'' = \sum_{k=1}^{g-1} \sum_{j=k+1}^{g} \sum_{i=1}^{n} x_{ik} x_{ij}$ is a penalty function representing Constraints (6).

In the UQP formulation above, the objective function is to maximize the total number of points assigned to shapes. However, the penalty function P' will increase if two non-adjacent points are chosen, forcing only adjacent points to be set to 1. Similarly, the penalty function P'' will increase if a point is assigned to more than one graph. Note that the j

index starts from $k + 1$ in the penalty function P''. This will avoid penalizing a point in the same graph, while penalizing the point if it is assigned to more than one graph.

In Section 4 we propose a GA to solve the UQP formulation—a GA that would maintain P'' as zero. This would maintain feasibility with respect to P'', easing the solution process. In order to improve the GA results, a local search on the final GA result is performed. We also discuss again the penalty function issue.

4. Genetic algorithm metaheuristic for the maximal covering by polygonal shapes

4.1. Introduction

In this section we will introduce a genetic algorithm developed specifically to solve the UQP formulation for maximal disjoint cliques on graphs having the same vertex set and different edge sets. The GA approach we discuss makes use of the idea of transforming covering by parallel-side polygons problem into maximum clique problems, introduced in Section 2. This transformation is necessary in order to allow handling of the many facilities with different block norms.

The genetic algorithm (GA) approach, introduced by Holland in 1975, mimics the natural evolution of species. Since then, it has been applied in many areas such as location problems by Jaramillo et al (2002), for capacitated location allocation models, as well as for uncapacitated models (Salhi and Gamal, 2003) and covering models (Aytug and Saydam, 2002). In addition, GA has been applied in set covering by Solar et al (2002), Aickelin (2002), set partitioning (Chu and Beasley, 1998), node partitioning problems (Chandrasekharam et al, 1993), UQPs (Katayama et al, 2000), and maximum clique problem (Balas and Niehaus, 1998).

A GA is based on the behaviour of genes in the adaptive process of nature and the mechanics of natural selection in nature. A GA starts with an initial population of strings, also known as individuals or chromosomes, which is considered a potential solution. The strings can be represented as a set of binary numbers or as strings with real valued numbers. A string's individual fitness is measured as either the individual objective value or its violation of constraints—which can also be represented as a penalty function value—or both. The fittest strings have a better chance of being chosen as parents for the production of offspring; they have a greater chance to survive and, consequently, transfer their characteristics to their offspring. Offspring can be mutated with a certain probability, and offspring fitness is evaluated. The offspring are then inserted in the population, replacing parents or individuals with lower fitness, and a new generation is produced. This process continues for a certain number of cycles, or until convergence is reached.

In general, a GA would have the ability to perform the following steps:

Step 1: Generate initial population (as strings).
Step 2: Evaluate individual string fitness.
Step 3: Establish a basis for deciding which parents to mate and, thus, generate offspring.
Step 4: Establish a framework for deciding which offspring and parents should constitute the next generation.
Step 5: Develop a method to imitate natural evolution.

4.2. The genetic algorithm for maximum covering by polygonal shapes

This section illustrates the adaptation of GA to solve the UQP formulation for the MCPSP on graphs having the same vertex set and different edge sets. Genetic operators for this problem can be summarized as follows:

(1) Create an initial population as a zero matrix of size gn, assign each row of the matrix to one shape.
(2) Randomly assign demand points to shapes in the initial population, such that if a point is assigned it will be assigned to one shape only.
(3) Use UQP formula, Equation (8) to evaluate strings fitness.
(4) Select parents from the strings population for mating.
(5) Perform a local search to improve the string fitness value.

The details for these genetic operators, initial population, fitness function, selection, crossover, and mutation are explained as follows:

(1) *Initial population*
This can be divided into two parts, population representation and population assignment.
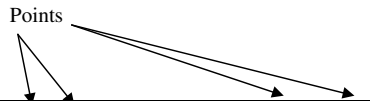
(1a) *Population representation*: In a UQP, the choice of strings in a binary format would be a natural choice. Each string takes n bits, each bit representing one point. The bit is assigned a value of 1 or zero according to whether it is assigned to a shape or not. Therefore, a matrix of size gn represents g shapes with strings of size n for each shape. This matrix of size gn represents one parent in the initial total population. An example of one parent in the initial population, as a matrix of size gn, is shown in Figure 4.

(1b) *Population assignment*: For every point there will be two choices: either to assign the point to one shape or to keep it unassigned. If the point is assigned to a shape, then the remaining bits in the column representing that point are set to zero. This will ensure that for any point assigned it will be assigned only for one shape, and will avoid the calculation of the penalty function P''. This population assignment will also ensure P'' will always be zero after crossover operations.

Figure 5 shows the initial population generation procedure. A brief explanation follows.

In Step 1 of the procedure a population of parents (matrices of size gn) is generated. Each parent represents g graphs, where each row in the parent matrix represents one graph (equivalent to one parallel-sided polygon (PSP)), while each column in the parent (matrix) represents one vertex (point). Therefore, given n points and g shapes, we say that a point, i, is assigned to shape k in parent j if $STR_j(k, i) = 1$. Selections of graph and points status, assigned or unassigned, are done in Steps 4 and 5. In Step 4, a list of g-indexed uniform random numbers, [0, 1], is generated.

In Step 5, if the value of the largest random number in ra_k is greater than a certain value, which we called 'cut value', then the point is assigned to that shape, otherwise it will remain unassigned in that matrix. Since it is expected that, in general, a larger number of shapes will cover a larger area and more points, then the cut value should be inversely proportional to the number of shapes. Therefore, we set the cut value as $1/cg$, where $c \geqslant 1$ is a constant, and g is the number of shapes. For a small number of shapes, we choose a large cut point, that is, $c \approx 1$; this will avoid our choosing too many points for a small number of shapes. It will also reduce the correction procedure performed for strings, as explained later. Experimentation with the cut point may be necessary.

	1	2		n-1	n
Shape 1	1	0		0	0
Shape 2	0	0		0	1
Shape g	0	1		0	0

Points

Figure 4 Initial population representation.

Input:
- A population *pop* of zero matrices of size gn, STR_{pop}, $|pop|$ = size of population
- g random numbers ($ra_k, k = 1, ..., g$)
- cut value
- $j = 0$
while not end of population
$j = j + 1$
For $i = 1 : n$
If max (ra_k) ⩾ cut value *then*
Set $STR_j(index\_of\_\max(ra_k), i) = 1$
End

Figure 5 Procedure initial population.

The procedure continues to the next point until all points are accounted for. Finally, if a shape has no points assigned to it—that is, if all the elements in the row are equal to zeros—then we assign to it a point from any remaining unassigned points.

(2) *Fitness function*

Each shape will have a number of points assigned to it as 1's in the matrix row. The shape fitness is the maximum number of points assigned that forms a complete graph. For any shape, which is represented by one row in the parent matrix, the shape fitness can be found as the difference between the total number of points assigned to that shape and the penalty function P'. If the penalty value is zero, then the shape fitness is the total number of 1's in the shape string (row). In this work, a correction procedure on each string was performed, setting points that lower the fitness value and increase penalty P' to zero. After this corrective procedure, the penalty P' becomes equal to zero and the fitness function for any string is the sum of the remaining 1's in that string. As an example, consider the graph shown in Figure 6 and its adjacency matrix A_1.

Let the string for polygon *poly*1 be as follows:

1	1	1	0	1

String for polygon *poly*1 before correction

The correction procedure will check the string bits that maximize the value of P'. From Figure 6, one can see that vertices v_1, v_2 are not adjacent to each other, that is, $(v_1, v_2) \in \overline{E_1}$; also $(v_1, v_3) \in \overline{E_1}$, $(v_2, v_5) \in \overline{E_1}$. Therefore, the first and second bits will contribute a penalty value of 2, while the fifth bit will have a penalty value of 1. Consequently, one of the bits with a higher penalty is changed from 1 to 0 (flipped), say the second bit, keeping bits 1, 3 and 5 as 1's. This procedure is repeated with a penalty of 1 for the first and third bits and a penalty of 0 for the fifth bit. Hence, the first bit is flipped from 1 to 0, leaving the third and fifth bits with penalty P' as zero.

0	0	1	0	1

String for polygon *poly*1 after correction

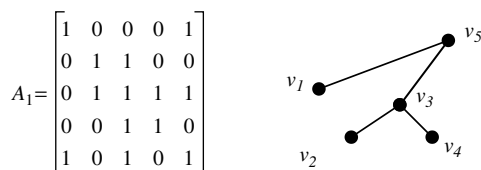The string's fitness value after this correction procedure is the sum of 1's in that string, which equals 2 in the above example. Since the penalty function P'' is set to zero then the parent fitness value will be the sum of shapes' fitness values, that is the sum of 1's in the matrix.

(3) S*election of parents for next offspring*

A truncation selection was used in this stage, where a certain percentage of the population is taken for the crossover stage. Parents are chosen from the fittest 50% of the population. Roulette-wheel selection was used for mating parents. The parents and the new offspring replaced the strings of lowest fitness.

(4) *Crossover*

In the crossover stage, genes from different parents are exchanged to create offspring with characteristics from both parents. From the different crossover schemes available in the literature, the simple one-point crossover was used. In a simple crossover, a random number is generated for each mating parent to find the crossover point. Our choice of initial population would maintain the feasibility with respect to the penalty function P''. A feasibility correction procedure was again performed for the offspring to maintain the penalty function P' as zero. Figure 7 shows how P'' would always be maintained as zero.

(5) *Mutation*

After each generation crossover we choose, randomly, a number of parents and a number of strings to be mutated. If a string is a candidate for mutation, one or more points in that string are randomly chosen. If the chosen point is 1, then it is flipped to 0. However, in order to maintain P'' as zero, if the point value is zero then it is flipped to 1 and all other bits in the matrix column are set to zero.
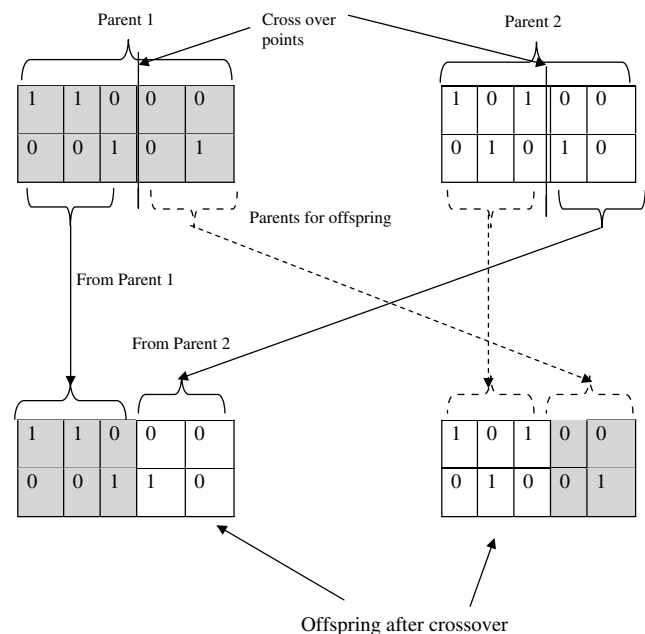
$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Figure 6 Adjacency matrices and equivalent graphs to illustrate the penalty function idea.

Figure 7 Crossover operation maintains P'' as zero.

(6) Local search

At the end of a certain number of cycles, the GA result for all shapes in the fittest parent is maintained in a tabu list. A search is then performed in the neighbourhood of the points assigned to each shape. If the neighbourhood of the shape string is not empty, and is not in the tabu list, a local search based on tabu search is performed and the points in the neighbourhood adjacent to all points in the string are added.

5. Computational example

In this section we present the results of experiments used to examine the GA algorithm. Computational tests for the GA and a one-shape exact algorithms developed by the authors were performed using Matlab and run on an IBM PC with an Intel Pentium III 600 MHz processor and 64 MB of memory. Six different shapes were taken, as presented in Figure 8. The shapes are as follows:

(1) A parallelogram as shown in Figure 8a.
(2) A diamond shape as shown in Figure 8c.
(3) A rectangle of sides length 3 and 2.5 as shown in Figure 8b.
(4) A hexagonal shape (a block norm of third degree) with a 2.5 units distance between its centre and corners as shown in Figure 8d.
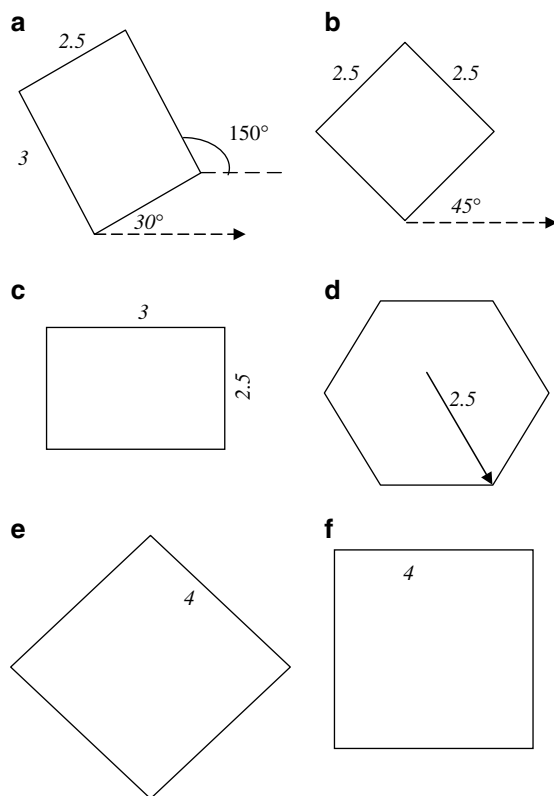
(5) A diamond shape of side length 4 as shown in Figure 8e.
(6) A square of side length 4 as shown in Figure 8f.

Computational experiments were performed for different sets of points in different areas. Table 1 shows the number of points and the area size.

Consider covering by g shapes, one shape at a time, where the demand points covered by the first shape are not considered for covering by the next one. A globally optimal single-shape algorithm such as the one in Younies (2004) could be used in the procedure. One may be tempted to try all g! possible orderings. Unfortunately, it can be shown that this approach does not guarantee an optimal solution. It can, however, provide a lower bound on the optimal value of the objective function. We call this the LB solution. Note that any ordering will by itself provide a lower bound. In our examples, we selected the best solution of all possible ordering as a lower bound for up to four shapes. To save computational time, we only used one permutation, {1, 2, 3, 4, 5, 6}, for the case of six shapes.

An estimated upper bound was found by performing exact search independently for each shape (where the demand points covered by the first shape are considered for covering by the next one), as discussed in Section 2, Observation 1. It must be noted, however, that this upper bound works better for the case of shapes of different size and inclination. This is due to the fact that similar shapes will tend to cover same demand points.

Although this upper bound may not be tight, since shapes may cover the same points, it is taken as guidance for the GA results. Moreover, if the shapes are identical, then we

Figure 8 The different shapes under study.

Table 1 Number of points and area

Shape number	Number of points	Area (x, y)
1	50	(20, 20)
2	50	(20, 20)
3	50	(20, 20)
4	50	(20, 20)
5	50	(20, 20)
6	50	(20, 20)
1	100	(30, 30)
2	100	(30, 30)
3	100	(30, 30)
4	100	(30, 30)
5	100	(30, 30)
6	100	(30, 30)
1	150	(40, 40)
2	150	(40, 40)
3	150	(40, 40)
4	150	(40, 40)
5	150	(40, 40)
6	150	(40, 40)
1	200	(40, 40)
2	200	(40, 40)
3	200	(40, 40)
4	200	(40, 40)
5	200	(40, 40)
6	200	(40, 40)

Table 2   Computational results for up to six shapes

1	2	3	4	5	6	No. of points	Upper bound	Best	Avg.	Time*	Value	Time†
						Shapes taken			*GA result*			*LB solution*
◆	◆					50	9	9	8	2	9	1
◆	◆	◆	◆			50	21	21	19	11.5	21	52
◆	◆	◆	◆			100	25	20	19	67.73	21	165.71
◆	◆	◆	◆			150	19	17	15	286.5	19	393.7
◆	◆	◆	◆			200	23	18	17	751.5	21	843.54
◆	◆	◆	◆	◆	◆	50	36	28	26	18.72	30	2.53†
◆	◆	◆	◆	◆	◆	100	40	33	28	128.4	31	10.98†
◆	◆	◆	◆	◆	◆	150	29	27	26	499	28	26.69†
◆	◆	◆	◆	◆	◆	200	36	33	30	1492	32	59.7†

*Time in seconds.
†Time for one ordering {1, 2, 3, 4, 5, 6}.

will have only one graph $G = (V, E)$ for all shapes, and the problem will be solvable as a maximum clique partition problem on the same graph.

In order to evaluate our GA results, we first compared the case of two shapes and 50 points with the MIP result from Younies and Wesolowsky (2004). Both the GA method and a two-shapes ordering based on a one-shape exhaustive search algorithm gave results similar to that of MIP for 50 points and two shapes. Results for four shapes and six shapes are reported in Table 2.

In Table 2 we show the computational results for both the GA and the LB algorithms. Where, shapes chosen from Figure 8 are indicated by ◆ under 'shapes taken' title.

We notice in Table 2 that the upper bound (solving each shape separately) is greater than the best GA solution for four shapes and 100 and 200 points. For four shapes, the GA results were lower than lower bound in three cases. The lower bound was similar to the upper bound for the case of 150 points, indicating that shapes did cover different points in the ordering and that this is an optimal answer as well. GA results were better than the results in the LB column in two cases, for six shapes and 100 and 200 points.

Run time for all possible orderings for six shapes can be estimated from the run time per one ordering. Our experiments with both the mutation rate and the cut value discussed in the previous section did not show a significant improvement. Therefore, the GA parameters were fixed in all computations to 40 generations, a uniform crossover, and a mutation rate of 0.1.

Finally, while the one-shape exact algorithm (taking one or all possible permutations) gave good results in Table 2, the exact algorithm results may be affected by many factors such as (1) the shape's size and inclinations, (2) the ordering of the solution and (3) by the spatial distribution and number of demand points taken. On the other hand, the GA approach will take all shapes, simultaneously, into account and will give a good solution regardless of the demand points' number, distribution or the shapes parameters (size, inclination).

6. Conclusions

The block norms as a distance measure have been applied for several problems in location theory, for example, the minimax and the Fermat–Weber problems. The work described in this paper is, to our knowledge, the first attempt to formulate the MCLP with block norms. In this paper we studied the PMCLP under different block norms. We showed that the multifacility PMCLP under different block norms, of any degree, can be solved as a maximum clique partition problem on different graphs. The transformation procedure to a graph theory problem introduces a new perspective to this set of problems. We formulated the problem as an unconstrained binary quadratic problem (UQP). The UQP formulation allowed us to make use of genetic algorithms as our workhorse solution technique. A performances comparison of the GA algorithm and a one shape complete enumeration algorithm are reported.

References

Aickelin U (2002). An indirect genetic algorithm for set covering problems. *J Opl Res Soc* **53**: 1118–1126.

Aneja YP, Chandrasekaran R and Nair KPK (1988). A note on the m-center problem with rectilinear distances. *Eur J Opl Res* **35**: 118–123.

Aytug H and Saydam C (2002). Solving large-scale maximum expected covering location problems by genetic algorithms: a comparative study. *Eur J Opl Res* **141**: 480–494.

Balas E and Niehaus W (1998). Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems. *J Heuristics* **4**: 107–122.

Chandrasekharam R, Subhramanian S and Chaudhury S (1993). Genetic algorithm for node partitioning problem and applications in VLSI design. *IEE* **140**: 255–260.

Chu PC and Beasley JE (1998). Constraint handling in genetic algorithms: the set partitioning problem. *J Heuristics* **11**: 323–357.

Church RL (1984). The planar maximal covering location problem. *Journal of Regional Science* **24**: 185–201.

Church RL and ReVelle CS (1974). The maximal covering location problem. *Papers Regional Sci* **32**: 101–118.

Diestel R (1997). *Graph Theory*. Springer-Verlag: New York.

Durier R and Michelot C (1985). Geometrical properties of the Fermat-Weber problem. *Eur J Opl Res* **20**: 332–343.

Fishburn PC (1985). *Interval Orders And Interval Graphs*. Wiley-Interscience Series in Discrete Mathematics: New York.

Holland J (1975). *Adaptions in Natural and Artificial Systems*. University of Michigan Press: Ann Arbor, MI.

Jaramillo JH, Bhadury J and Batta R (2002). On the use of genetic algorithms to solve location problems. *Comput Opns Res* **29**: 761–779.

Katayama K, Tani M and Narihisa H (2000). Solving large binary quadratic programming problems by effective genetic local search algorithm. In: Whitley LD, Goldberg DE, Cantú-Paz E, Spector L, Parmee I and Beyer H-G (eds). Proceeding of the Genetic and Evolutionary Computation Conference (Gecco 2000), Morgan Kaufman 2000, San Francisco, pp 643–650.

Nickel S (1998). Restricted center problems under polyhedral gauges. *Eur J Opl Res* **104**: 343–357.

Plastria F (1992). On destination optimality in asymmetric distance Fermat–Weber problems. *Ann Opl Res* **40**: 355–369.

Salhi S and Gamal MH (2003). A genetic algorithm based approach for the uncapacitated continuous location-allocation problem. *Ann Opns Res* **123**: 203–222.

Schilling D, Vaidynathan J and Barkhi R (1993). A review of covering problems in facility Location. *Location Science* **1**: 25–55.

Solar M, Parada V and Urrutia R (2002). A parallel genetic algorithm to solve the set covering problem. *Comput Opns Res* **29**: 1221–1235.

Thisse J-F, Ward JE and Wendell R (1984). Some properties of location problems with block and round norms. *Opns Res* **32**: 1309–1327.

Ward J and Wendell R (1980). A new norm for measuring distance which yields linear location Problems. *Opns Res* **28**: 837–844.

Ward J and Wendell R (1985). Using block norms for location modeling. *Opns Res* **33**: 1075–1090.

Trotter Jr WT and Harary F (1979). On double and multiple interval graphs. *J Graph Theory* **3**: 205–211.

Younies H (2004). *Planar maximal covering location problem under block norms*. PhD Thesis, McMaster University, Hamilton, Ontario, Canada.

Younies H and Wesolowsky G (2004). A mixed integer formulation for maximal covering by inclined parallelograms. *Eur J Opl Res* **159**: 83–94.

Appendix: A mixed integer formulation for covering under the one-infinity norm

Younies and Wesolowsky (2004) introduced a formulation that can handle the PMCLP for facilities with the rectilinear, parallelogram, or Tchebycheff coverage measure (second-degree block norms). In this appendix, we extend the analysis to the location of facilities with coverage defined by the one-infinity norm (a block norm of fourth degree). The formulation to be shown is limited to distances under the one-infinity norm (a norm with four directions of travel; $r = 4$). However, the formulation can be easily changed to handle coverage under block norms with other degrees.

As explained previously in Section 1, the one-infinity block norm is a linear combination of rectilinear and

Tchebycheff norms. Let λ_1' and λ_2' be non-negative numbers not both of which are zero. The one-infinity norm is defined as $l_{(\lambda_1', \lambda_2')} = \lambda_1' l_1 + \sqrt{2} \lambda_2' l_\infty$, where Let λ_1' represents the proportion of a trip made via rectilinear distance and Let λ_2' represent the proportion of trip made via a diagonal road as illustrated in Figure A1. The formulation is based on the difference between the Y-intercepts of the parallel sides of the one-infinity norm. The one-infinity norm sides' inclinations can be determined by Let λ_1' and Let λ_2', the magnitude of travel along the rectilinear and diagonal roads, respectively. We formulate the problem of planar maximal covering using one-infinity block norm, and follow this with a discussion.

Problem statement   Given n points in R^2, $a_i = (a_{ix}, a_{iy})$ with weights $w_i, i = 1, \ldots, n$, and g facilities with a coverage measure under the one-infinity block norm, the facility setup costs are $C_k, k = 1, \ldots, g$. We must find the locations of p facilities that will cover the maximum weight of points. Any point covered is to be assigned to one facility only.

To facilitate the presentation of our formulation, our discussion will focus on Figure A2 instead of Figure A1. Figure A2 shows a one-infinity unit ball, with an octagonal shape. The octagonal shape's upper sides are indexed counterclockwise as τ_1, τ_2, τ_3 and τ_4, and vertices denoted as $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ and Γ_5. The block norm unit ball contour
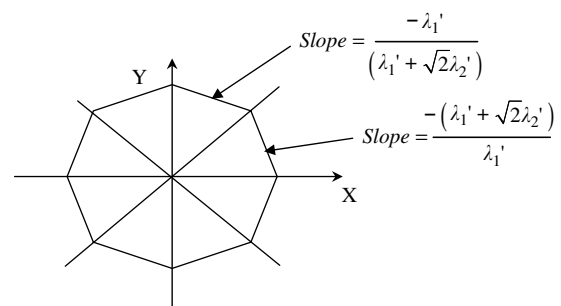

Figure A1   Contour of one-infinity block norm.
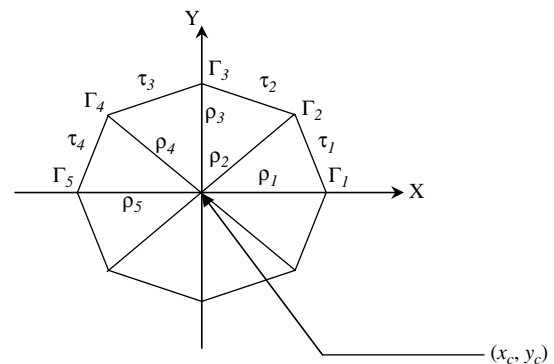

Figure A2   The one-infinity block norm unit ball.

center is denoted as (x_c, y_c), and the unit contour radius length as $\rho_{ik}, i = 1, \ldots, 5, \ k = 1, \ldots, g$.

For the PMCLP under the one-infinity norm (octagonal shapes), the following notation will be used:

Inputs

$m_{jk} =$ slope of side τ_j in facility k one-infinity unit norm contour, $k = 1, \ldots, g; j = 1, \ldots, 4$;

$I_{jk} =$ the distance between the Y-intercepts of side τ_j and the side parallel to it in facility k one-infinity unit norm contour, $k = 1, \ldots, g; j = 1, \ldots, 4$;

$\rho_{ik} =$ radius length in the one-infinity block norm of facility k, $i = 1, \ldots, 4; k = 1, \ldots, g$;

$\theta'_{1k} =$ angle inscribed between radius ρ_{1k} and radius ρ_{2k} in facility k one-infinity unit norm contour, $k = 1, \ldots, g$;

$\theta'_{2k} =$ angle inscribed between radius ρ_{4k} and radius ρ_{5k} in facility k one-infinity unit norm contour, $k = 1, \ldots, g$;

$w_i =$ weight associated with point i, $i = 1, \ldots, n$

Decision variables

$b_{jk} =$ the Y-intercept of side τ_j in the one-infinity block norm of facility k, $k = 1, \ldots, g; j = 1, \ldots, 4$;

$$u_{ik} = \begin{cases} 1, & \text{if point } i \text{ is assigned for facility } k, \\ & \quad k = 1, \ldots, g; \ i = 1, \ldots, n \\ 0, & \text{Otherwise}; \end{cases}$$

$$h_k = \begin{cases} 1, & \text{if facility } k \text{ is chosen}, \\ 0, & \text{Otherwise}; \end{cases}$$

$x_{ck} =$ the X-coordinate of facility k, $k = 1, \ldots, g$;
$y_{ck} =$ the Y-coordinate of facility k, $k = 1, \ldots, g$.

The I_{jk} values for each one-infinity block norm can be determined from the orthogonal distance between parallel sides. In the following formulation, a large number multiplier, M, is used in a manner similar to that in the inclined parallelograms formulation of Section 2.3. Points Γ_2 and Γ_4 in Figure A1 are used in the formulation to determine the facility location (the center of octagonal shape). The detailed MIP formulation for the PMCLP under the one-infinity norm is as follows:

$$\text{(P6)} \quad \text{Maximize} \sum_{k=1}^{g} \sum_{i=1}^{n} w_i u_{ik} - \sum_{k=1}^{g} C_k h_k \tag{A1}$$

Subject to

$$a_{iy} - m_{jk} a_{ix} - b_{jk} \leqslant M(1 - u_{ik}), \quad i = 1, \ldots, n; \\ j = 1, \ldots, 4; \quad k = 1, \ldots, g \tag{A2}$$

$$a_{iy} - m_{jk} a_{ix} - b_{jk} + I_{jk} \geqslant M(u_{ik} - 1), \\ i = 1, \ldots, n; \ j = 1, \ldots, 4; \ k = 1, \ldots, g \tag{A3}$$
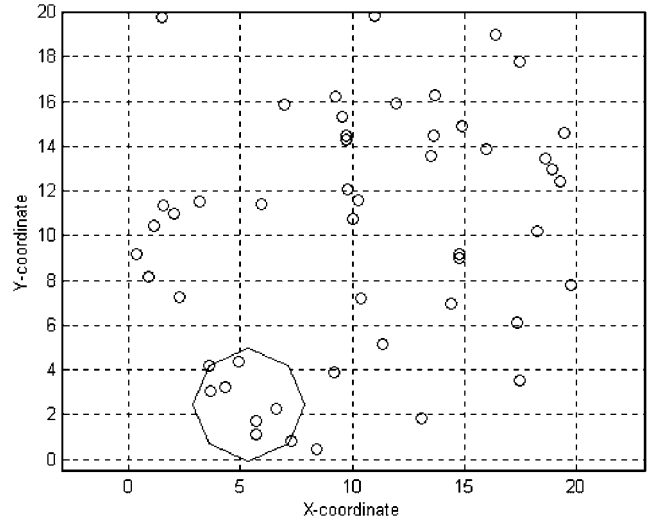

Figure A3 Output of covering by one-infinity block norm (octagonal shape).

$$y_{ck} + \rho_{2k} \sin \theta'_{1k} = m_{jk}(x_{ck} + \rho_{2k} \cos \theta'_{1k}) + b_{jk}, \\ k = 1, \ldots, g; \ j = 1, 2 \tag{A4}$$

$$y_{ck} + \rho_{4k} \sin \theta'_{2k} = m_{jk}(x_{ck} - \rho_{4k} \cos \theta'_{2k}) + b_{jk}, \\ k = 1, \ldots, g; \ j = 3, 4 \tag{A5}$$

$$\sum_{k=1}^{g} u_{ik} \leqslant 1, \quad i = 1, \ldots, n \tag{A6}$$

$$h_k - u_{ik} \geqslant 0, \quad i = 1, \ldots, n; \ k = 1, \ldots, g \tag{A7}$$

$$\sum_{k=1}^{g} h_k = p \tag{A8}$$

$$u_{ik}, h_k \in \{0, 1\} \quad \text{for} \quad i = 1, \ldots, n; k = 1, \ldots, g \tag{A9}$$

$$b_{jk}, x_{ck}, y_{ck} \geqslant 0 \tag{A10}$$

In the above formulation, constraints (A2), (A3), and (A6)–(A10) are similar to those in Section 2.3. The facilities locations are determined by constraints (A4) and (A5). In constraints (A4) the facilities location is determined from XY-coordinates of point Γ_2, while constraints (A5) determines the facilities location from the XY-coordinates of point Γ_4. There are $g(n + 1)$ binary variables for g facilities, and $6g$ continuous variables. The formulation illustrated for the one infinity block norm can be adapted to other block norms of degree greater than 2 or to other polygonal shapes with sides parallel to each other. Figure A3 shows the case for maximal covering by a single octagonal shape for 50 points of equal weights. The run time was 9 s on a Ultra-4 SPARC Sun station.