PLANAR MAXIMAL COVERING LOCATION PROBLEM UNDER DIFFERENT BLOCK NORMS

By

HASSAN ZEIDAN YOUNIES

A Thesis
Submitted to the School of Graduate Studies in
Partial Fulfillment of the Requirements
for the Degree
Doctor of Philosophy

McMaster University

PLANAR MAXIMAL COVERING LOCATION PROBLEM UNDER DIFFERENT BLOCK NORMS

DOCTOR OF PHILOSOPHY (2004) McMaster University

Management Science/Systems Hamilton, Ontario

TITLE: Planar Maximal Covering Location Problem Under Different Block Norms

AUTHOR: Hassan Zeidan Younies

B.Eng. (Aden University)

M.Eng. (University of New Brunswick)

SUPERVISORY COMMITTEE: Professor George O. Wesolowsky (Chair)

Professor George Steiner

Professor Prakash Abad

NUMBER OF PAGES: xii, 125

Dedicated to the memory of my mother

ACKNOWLEDGEMENTS

I received help from many individuals in the course of performing the research described in this thesis. First and foremost, I sincerely thank my mentor and supervisor, Dr. George Wesolowsky, for his unfailing help, guidance, patience and other assistance. I am grateful for having had the opportunity of being his student, and for the extensive knowledge I have acquired from him.

I would like also to extend heartfelt thanks to my thesis committee, Dr. George Steiner and Dr. Prakash Abad, whose discussions and comments helped me profoundly in developing this thesis and steering it to its final format.

And last but not least, I would like to thank my family, especially my wife, for their support and understanding during the many stressful hours I spent bringing this thesis to a successful conclusion.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iv

LIST OF TABLES viii

LIST OF FIGURES ix

ABSTRACT xi

CHAPTERS

1 INTRODUCTION 1

1.1 Background 1

1.2 Distance Measures in Location Theory 5

1.3 Block Norms 8

1.4 Contiuous Covering Location Problems 11

1.4.1 Rectilinear Distance Minimax Problem 13

1.4.2 Rectilinear Planar Maximal Covering Problem (RMCLP) 15

1.5 Computational Aspects in Covering problems 19

1.6 Scope and Outline of the Thesis 20

2 A MIXED INTEGER PROGRAM FORMULATION FOR PLANAR MAXIMAL COVERING LOCATION PROBLEM UNDER BLOCK NORMS 25

2.1 Introduction 25

2.2 From Minimax to Maximal Covering 27

2.3 The Mixed Integer formulation for the PMCLP with Second Degree Block Norms 29

2.3.1 Parallelogram Norms 29

2.3.2 A Mixed Integer Program Formulation 32

2.3.3 Computational Example 40

2.4 Covering by Rectangles with Sides Parallel to the Axes 41

2.4.1 Computational Example 44
2.5 Covering under the One-Infinity Norm 45

3 EXACT ALGORITHMS FOR PLANAR MAXIMAL COVERING LOCATION PROBLEM UNDER DIFFERENT BLOCK NORMS 52
3.1 Introduction 52
3.2 Alternative Algorithm for Maximal Covering by a Parallelogram 53
3.2.1 Computational Results 58
3.3 Transforming Maximum Covering by PSP's to a MaximumClique Problem on a Graph 61
3.4 A Guided Search Algorithm for the Maximum Clique Problem 72
3.4.1 Introduction 72
3.4.2 Guided Search Algorithm for the Maximum Clique Problem: 74
3.4.3 Guided search algorithm for the maximum clique problem "GSAMCP" . 76
3.4.4 Example and Computational Results 81

4 GENETIC ALGORITHM SOLUTION APPROACH FOR PLANAR MAXIMAL COVERING LOCATION PROBLEM UNDER DIFFERENT BLOCK NORMS 88
4.1 Introduction 88
4.2 Genetic Algorithms: Background and Principles 90
4.2.1 Population Generation 91
4.2.1.1 Representation 91
4.2.2 Selection 91
4.2.2.1 Fitness Evaluation 92
4.2.2.2 Parents Selection 93
4.2.3 Crossover 94
4.2.4 Mutation 96
4.3 Maximal Covering by Many Shapes as a Graph Formulation 98

4.3.1 Formulating the Maximum Clique as an Unconstrained Quadratic Binary Problem (UQP) 99
4.3.1.1 A Small Example 102
4.4 Genetic Algorithm for Maximum Covering by Polygonal Shapes 104
4.4.1 Computational Experiment 111
5 CONCLUSIONS AND FURTHER RESEARCH 116
5.1 Conclusions 116
5.2 Suggested Future Research 118
BIBLIOGRAPHY 120

LIST OF TABLES

Table 3.1: Computational comparisons for Maximal covering by a diamond shape. 60

Table 3.2: Vertices indices and degrees for the *GSAMCP* example..... 81

Table 3.3: Computational results of comparison between *GSAMCP* and *GS*. 87

Table 4.1. Number of points and area under study. 113

Table 4.2. Computational results for up to six shapes. 114

LIST OF FIGURES

Figure 1.1: Euclidean distance contour, radius = 1 9
Figure 1.2: Unit balls for rectilinear and Tchebycheff norms in the plane. 9
Figure 1.3: Unit ball (contour) of one-infinity norm in R^2 11
Figure 1.4: Full covering by diamond shape. 15
Figure 1.5.a: An example of diamond intersection points. 18
Figure 1.5.b: An example of DIPS points when two diamond shapes intersection is a line segment. 19
Figure 1.5: The possible intercept result of two diamond shapes. 18-19
Figure 2.1: Equi-distance contour for rectilinear norm and traveling directions. 31
Figure 2.2: Equi-distance contour of parallelogram norm and traveling direction. 31
Figure 2.3 : Illustration of Property 2.1 for $0 \le \theta \le \frac{\pi}{2}$ 33
Figure 2.4 : Illustration of Property 2.1 for $\frac{\pi}{2} \le \theta \le \pi$ 34
Figure 2.5: An inclined parallelogram with the Y-intercepts of its sides. 35
Figure 2.6: Solving for inclined parallelogram corner (x_{ck}, y_{ck}) 39
Figure 2.7: Maximum covering location for one inclined parallelogram and one inclined square. 41
Figure 2.8: Result for choosing the minimal cost maximal cover. 45
Figure 2.9: Contour of one infinity block norm. 47
Figure 2.10: The one-infinity block norm unit ball. 47
Figure 2.11: Output of covering by one-infinity block norm (octagonal shape). 51

Figure 3.1: Computation of the solution list SL_1 55
Figure 3.2: Result from MIP and algorithm..... 59
Figure 3.3: Intervals and the corresponding interval Graph. 62
Figure 3.4.a: Points in R^2 and the corresponding intervals assigned to their Y-intercepts..... 64
Figure 3.4.b: The equivalent interval graph of Figure 3.4.a. 64
Figure 3.5.a: Two sets of real intervals of different length and their equivalent 2-interval graph..... 66
Figure 3.5.b: Two sets of real intervals with the same interval length on each real line and the equivalent simple 2-interval graph..... 66
Figure 3.6.a: A rectangle with the axis-intercepts of its sides and points intervals..... 70
Figure 3.6.b: The equivalent graph for intervals on Y-axis..... 70
Figure 3.6.c: The equivalent graph for intervals on X-axis. 71
Figure3.6.d: The equivalent two-interval graph 71
Figure 3.7: A Matlab program to find adjacency matrix for a rectangular shape..... 72
Figure 4.1: Single point Crossover. 95
Figure 4.2: Multi-point crossover. 95
Figure 4.3: Mutation for binary string. 96
Figure 4.4: Flow diagram of GA..... 97
Figure 4.5: Adjacency matrices and equivalent graphs to illustrate the penalty function idea..... 103
Figure 4.6: Initial population representation..... 105
Figure 4.7: Procedure initial population. 106
Figure 4.8: Crossover operation maintains p^* as zero..... 110
Figure 4.9: The different shapes under study 112

ABSTRACT

This study introduces a new model for the planar maximal covering location problem (PMCLP) under different block norms. The problem involves locating p facilities anywhere on the plane in order to cover the maximum number of n given demand points. The generalization we introduce is that distance measures assigned to facilities are block norms of different types and different proximity measures.

This problem is handled in three phases. First, a simple model based on the geometrical properties of the block norms' unit ball contours is formulated as a mixed integer program (MIP). The MIP formulation is more general than previous PMCLP's and can handle facilities with different coverage measures under block norm distance and different setup cost, and capacity.

Second, an exact solution approach is presented based on:

1- An exact algorithm that is capable of handling a single facility efficiently.

2- An algorithm for an equivalent graph problem – the maximum clique problem (MCP).

Finally, the PMCLP under different block norms is formulated as an equivalent graph problem. This graph problem is then modeled as an unconstrained binary quadratic problem (UQP) and solved by a genetic algorithm.

Computational examples are provided for the MIP, the exact algorithm, and the genetic algorithm approaches.

CHAPTER ONE

INTRODUCTION

1.1 Background

Decisions made on the location of facilities play an important role in the economical and social development of society. Political, economical and social factors often influence location decisions. These influencing factors should be abstracted into mathematical models by location theory practitioners. Multiobjective mathematical models that integrate these factors along with other aspects of the firm activities (such as transportation, production, finance etc.) are frequently used. Location theory practitioners should be provided with good mathematical formulations which accurately represent real-life situations and all decision-influencing factors, as well as appropriate solution techniques for mathematically formulated problems (Owen and Daskin (1998)). These mathematical models with the help of computer visual systems play an integral part of decision support systems (DSS) for decision makers (i.e. politician, public servant).

Some location models are general in that they provide a basic conceptual structure one can build on. These framework models, such as the *p*-center, the *p*-median, and the uncapacitated facility location problem, have been extensively studied, and the theoretical and computational aspects are well known to experts in the field. Other location models are tailored to specific applications, or to a specific setting (for example, a bank account

problem, Cornuejols et al. 1977), and they are usually a variation of the framework models.

Location models can be classified, according to their spatial setting, into discrete models, network models, and continuous location models. In discrete location models, the locations of new facilities are restricted to a set of points or to the nodes of a given network structure. Hence, in discrete location models, optimal sites for new facilities are limited to a set of given predetermined candidate sites. Discrete location models require information of the actual traveling distances between points and, thus, need large data storage. The availability of transportation network data and the advent of modern computers make discrete location models attainable for many real-life problems. Some of the most common models are the *p*-median (Lorena and Senne 2003), *p*-center (Hochbaum and Pathria 1997), quadratic assignment (Burkard 1984), set covering (Aickelin 2002), and hub-location models (Campbell 1994).

Location models that can site facilities anywhere on the plane, or on the network, are known as continuous location models.

Other classifications of location models can be categorized under these two spatial settings, the discrete and the continuous location models.

Some examples of these classifications are:

1) Location models under uncertainty (such as stochastic location models, Louveaus and Peters 1992).

2) Evolving demand over time (such as multi-period location, Wesolowsky and Truscott 1975).

3) Classification based on objective function such as (i) minimize total traveling cost (minisum, Chiu 1987), (ii) minimize maximum traveling distance (minimax, Michilot and Plastria 2002).

4) Classification based on facility type (uncapacitated facility, Gao et al. 1994 versus capacitated facility location, Lorena and Senne 2003).

In all location models, researchers pursue common basic objectives. One location problem objective could be to locate the factory in a manner that minimizes transportation costs to demand points and satisfies their demand (minisum). Alternatively, the objective might be to find the location of an emergency facility that minimizes the maximum traveling time to a set of customers (minimax), or to site a radio transmitter that covers the maximum number of users (maximum coverage).

Whether the model objective is to increase financial profit (such as the case in private sector), or increase some measure of satisfaction – such as service level, or shorter service time (such as pizza delivery or emergency medical service) – some performance measure based on distance is used. In fact, the concept of distance is the major commonality in most location models, and extensive studies have been done on that topic. Section 1.2 discusses different distance measures, paying special attention to distance measures from the family of block norms in Section 1.3.

One type of location model which has many applications in public and private sectors is the covering location model. Covering problems can be divided, according to the proportion of population covered, into full covering and maximal covering location problems (MCLP). Both problems – the full covering and the maximal covering – have

been modeled as discrete and continuous location problems; these have been studied under the rectilinear and Euclidean distance measures.

In full covering models, the model objective is to locate a single facility or multiple facilities that can cover all demand points while, in maximal covering models, facilities are located with the objective to maximize the total demand covered within a specified service standard (time or distance). In both models, the measure of covering (satisfaction) is based on a proximity measure that varies from one covering model to another. This covering measure is usually taken as distance traveled, or time to travel, from demand points to the facility. Therefore, in the full covering and the maximal covering models, a demand point is said to be covered (satisfied) if it is within a certain distance, or traveling time, from the facility. In both models – the full covering and the MCLP – demand could be stochastic or deterministic, and could be represented either as points in space or on a network (discrete), or as an area demand (continuous). In addition, varying positive weights can be associated with demand points. In the case of demand points of equal weights the MCLP is also known as the partial covering location problem.

For a review of some location problems, one can refer to papers by Brandeau and Chiu (1989) and Beaumont (1981). More detailed information on the subject is found in books by Drezner and Hamacher (2002); Love, Morris and Wesolowsky (1988); and Francis, McGinnis and White (1992), as well as in the book on discrete location models edited by Mirchandani and Francis (1990).

This work deals with deterministic covering problems where demand points are represented as points in space and facilities can be sited anywhere on the plane. This

covering model, known as planar maximal covering location problem (PMCLP), can be classified under continuous location models. This thesis deals with the PMCLP, where facilities can have different covering measures under different block norms.

Section 1.4 deals with two location problems related to the work in this thesis: the rectilinear minimax location problem and the rectilinear planar maximal covering problem. Section 1.5 studies computational issues in covering models. Finally, Section 1.6 describes the related literature and delineates the scope and outline of this thesis.

1.2 Distance Measures in Location Theory

Location problems deal with distance between two points in space in two different forms. In the first form, the actual traveling distance is given as input data, while in the second, the distance is calculated from coordinates of the two points. The two different distance forms are used in the discrete location models and in the continuous location models, respectively.

This thesis is dedicated to continuous location models; henceforth, our discussion will be limited to location models where a facility can be located anywhere on the plane, R^2.

Continuous location models are known as site generation models because they allow facility siting anywhere on the plane. Some of the most common continuous location models are the well-known Fermat-Weber problem, uncapacitated facility location, and warehouse location allocation models. In the Fermat-Weber problem

model, for example, an optimal site for the new facility is found by minimizing the sum of distance functions between the given data (demand) points and the new facility site.

Euclidean distance and rectilinear distance – also known as rectangular or Manhattan distance – are the most common distance measures used in continuous location models. The Euclidean distance between two points is the length of the Euclidean path – that is, the straight line – joining the two points. Thus, the Euclidean distance is the shortest distance between any two points. Examples of location models where Euclidean distance is an accurate representation of distance are some cases of radio transmitter coverage, emergency helicopter systems, and the cases where the traveling path between facilities is an actual Euclidean path.

On the other hand, the rectilinear distance between two points is the length of a traveling path parallel to either the X or Y axes. Therefore, the traveling directions are perpendicular to each other, and, hence, we have the name city grid, or Manhattan distance. The rectilinear distance measure is often an accurate representation of the traveling distance along city-street grids. Exceptions include cases of one-way-street grids or natural barriers to travel. Rectilinear distances can also be applied in location models that deal with factories and warehouses where travel is along aisles.

In reality, the traveling distance between two facilities rarely follows the Euclidean or rectilinear distance measure exactly; instead, in most cases, Euclidean and rectilinear distance measures approximate actual traveled distance. This has led to other distance measures, mainly the family of l_p norms. The l_p norm is a more general distance

measure, covering the Euclidean (l_2) and the rectilinear (l_1) norms for p values of two and one, respectively.

For any three points p', q', and h' in R^2, we define a distance (metric) as a function that satisfies the following basic properties (Ward and Wendell (1985) among others):

$d'(p',q') \geq 0$ (Nonnegativity)

$d'(p',q') = 0 \Leftrightarrow p' = q'$ (Identity)

$d'(p',h') \leq d'(p',q') + d'(q',h')$ (Triangle inequality)

$d'(p',q') = d'(q',p')$ (Symmetry)

For any two points a_1 and a_2, with coordinates (a_{1x}, a_{1y}) and (a_{2x}, a_{2y}), the l_p norm is given as

$$l_p(a_1,a_2) = \left[|a_{1x} - a_{2x}|^p + |a_{1y} - a_{2y}|^p \right]^{1/p}, \quad 1 \leq p \leq \infty .$$

Thus, the Euclidean and the rectilinear distances can be given as

$l_2(a_1,a_2) = \sqrt{(a_{1x} - a_{2x})^2 + (a_{1y} - a_{2y})^2}$ and $l_1(a_1,a_2) = \left[|a_{1x} - a_{2x}| + |a_{1y} - a_{2y}| \right]$, respectively.

When $p \to +\infty$, we get the Tchebycheff norm as $l_\infty(a_1,a_2) = \max\left\{ |a_{1x} - a_{2x}|, |a_{1y} - a_{2y}| \right\}$.

For better distance estimation, the value of p in the l_p norm can be tailored to a specific region by empirical studies, Brimberg et al. (1996). Unfortunately, the gained accuracy from empirical p value, in the l_p norm, is local to the region under study, motivating research into various approximations for distance measures. One area of

distance approximations that has been considered is the weighted distance measure, such as the weighted l_p norm and the weighted mixed norm. Extensive references for weighted distance measures can be found in Brimberg (1989) and Üster (1999).

Some other distance measures have been proposed, such as the block norm distance measure (Ward and Wendell (1980), Hamacher and Klamroth (2000)), and spherical distance for aircraft refueling (Yamani 1990). The distance measures of relevance to our research are from the family of block norms. These distance measures are described in the next section.

1.3 Block Norms

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) is called block (or polyhedral) norms. The round norms are those norms with no flat spot on their unit ball contours. The l_p norm is a round norm for $1 < p < \infty$. The Euclidean distance unit ball in R^2 is a circle of radius 1, as shown in Figure 1.1.

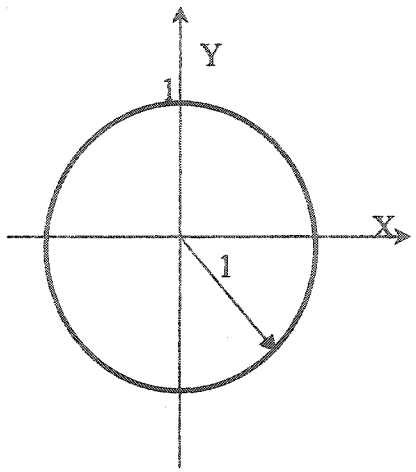


Figure 1.1: Euclidean distance contour, radius = 1.

The unit ball contours of the rectilinear (l_1) distance is a diamond shape, a square tilted by 45 degrees, with corner points at (1, 0), (-1, 0), (0, 1), and (0, -1). The Tchebycheff (l_∞) unit ball contours is a square with corners at (1, 1), (-1, 1), (1, -1), and (-1, -1), as shown in Figure 1.2.

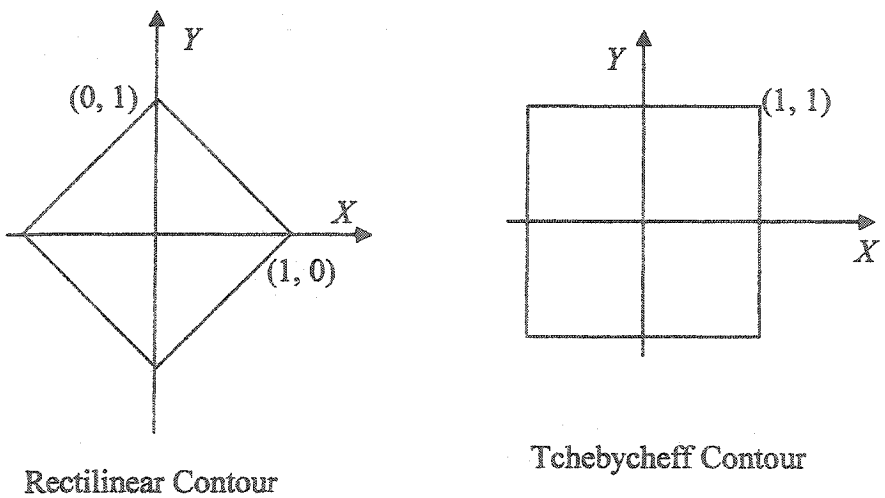


Figure 1.2: Unit balls for rectilinear and Tchebycheff norms in the plane.

Ward and Wendell (1980) introduced a block norm, 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 1.3 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 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 traveled along the rectilinear distance to the proportion traveled along the diagonal roads (the block norms diagonals). For (λ_1', λ_2') values of $(1,0)$ and $(0, \frac{1}{\sqrt{2}})$, 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 thesis, we refer to block norms with r diagonals and r allowed directions of travel as block norms of degree r (r degree). Therefore, we refer to the rectilinear and Tchebycheff norms as block norms of degree two (or a block norm of second degree), and the one-infinity norm as a block norm of degree four (or a block norm of fourth degree). Hence, the unit ball of a block norm with $r = 3$ is hexagonal in shape.

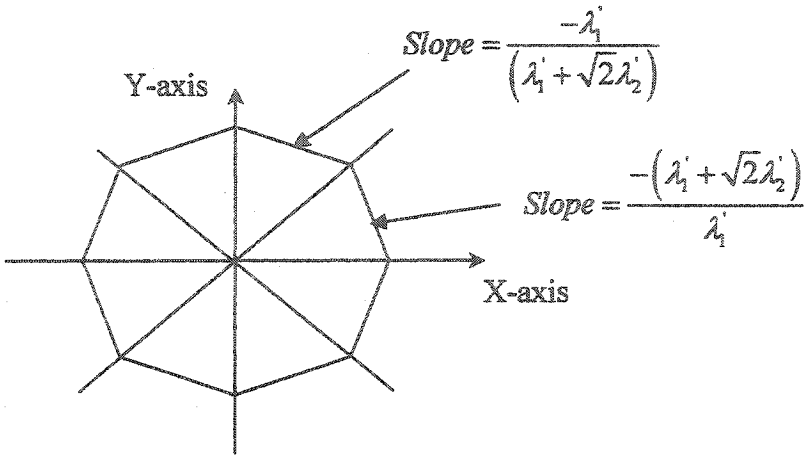


Figure 1.3: Unit ball (contour) of a one-infinity norm in R^2.

Ward and Wendell (1980, 1985) shows that the one-infinity norm is comparable with the weighted l_p norm as a distance predictor to geographical data. In addition to that one-infinity norm allows for linear formulation of the Fermat-Weber problem and to Rawls problem. Other block norm properties have been reported by Thisse, Ward and Wendell (1984). One of the main results of Thisse, Ward and Wendell (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 a round norm.

1.4 Continuous Covering Location Problems

From a location theory perspective, a demand point is said to be covered (serviced or satisfied) if it lies within a certain distance or travel time from the covering, or serving, facility. There are two types of continuous covering location problems: the full covering problem and the partial-covering problem. In full covering problems, all demand points

must be covered by a single facility or multiple facilities. An example of full covering is the diamond shape covering problem, which is equivalent to the unweighted rectilinear distance minimax location problem.

In minimax problems, facility location is optimized to minimize the maximum distance from the facility to demand points, a problem well studied under the Euclidean and rectilinear distance measures. The single facility minimax problem under the Euclidean distance measure is equivalent to the so-called circle covering problem. The location of the smallest diamond shape covering all demand points becomes the solution to the single facility minimax problem under the rectilinear distance measure. The minimax problem under the rectilinear distance measure will be discussed briefly in Section 1.4.1.

In practice, however, it is either impossible or uneconomical to cover, or serve, all demand points by a fixed number of facilities. The maximal covering location problem (MCLP) is the problem of locating a single facility or multiple facilities that cover the maximum weight of demand points. When the weights associated with demand points are equal, one has the simple partial covering problem.

As in other location problems, the maximal covering location problem can be formulated as a discrete or a continuous location problem. Most of the work done on MCLP can be categorized into discrete and network location models. Covering on networks can also be divided into continuous coverage – where the potential facility can be located on nodes or arcs – and discrete coverage, where the locations of the potential

facilities are limited to network nodes. Our discussion will be limited to the work done on planar maximal covering location problems (PMCLP).

PMCLP problems have been categorized, according to the distance measure used, into the Euclidean planar maximal covering problem (EPMCLP) and the rectilinear planar maximal covering problem (RPMCLP). This thesis is concerned with planar maximal covering problems with facilities under different block norms. In Section 1.4.2, we will discuss the PMCLP, focusing mainly on the RPMCLP.

1.4.1 Rectilinear Distance Minimax Problem

In this section, we consider the single facility rectilinear distance minimax problem. This problem is equivalent to a full covering problem where all demand points must be covered. The single facility rectilinear minimax problem is therefore equivalent to covering all demand points by the smallest single diamond shape – the unit ball of the rectilinear distance function. The mathematical formulation for the minimax problem and the geometrical approach to solve the minimax problem will be shown in this section. The minimax problem can be stated as follows:

Problem statement: Consider n demand points $a_i = (a_{ix}, a_{iy})$ in R^2, $i = 1,...,n$, we must find a new facility location (x, y), that will minimize the maximum traveling distance (time) from the new facility to any demand point.

This problem has applications in both the private sector and the public sector. In the public sector, the new facility could be an ambulance, a fire station or a public school; where the objective is to locate a facility within an acceptable traveling time from demand

points. In the private sector, the location model could be to, for example, locate a convenience store, or a restaurant, in such a way that the total population living within a certain distance (or traveling time) is maximized.

The unweighted single facility minimax location problem for the rectilinear distance measure can be formulated as follows:

$$(P1) \qquad \text{Minimize } z \tag{1.1}$$

Subject to

$$[|x-a_{ix}|+|y-a_{iy}|]\le z \qquad i=1,\ldots,n \tag{1.2}$$

Problem *P1* can be solved by expanding inequality (1.2) into the following equivalent inequalities.

$$\begin{aligned} z &\ge [(x-a_{ix})+(y-a_{iy})] \\ z &\ge [(x-a_{ix})-(y-a_{iy})] \\ z &\ge [-(x-a_{ix})+(y-a_{iy})] \\ z &\ge [-(x-a_{ix})-(y-a_{iy})] \qquad i=1,\ldots,n \end{aligned} \tag{1.3}$$

By rearranging inequalities (1.3) as described in Love et al., (1988, Chapter 6), a closed form solution can be found. The geometrical solution approach for diamond shape covering starts with constructing the smallest rectangle that covers all points, with rectangle sides making an angle of ±45 degrees with an axis. The rectangle sides that are farthest apart are then extended in length to obtain a square. The geometrical approach for the diamond-covering problem is shown in Figure 1.4.

1.4.2 Rectilinear Planar Maximal Covering Problem (RPMCLP)

In this section, we discuss the situation when new facilities are unable to cover all demand points. This situation occurs when there are time and distance constraints on the coverage measure and a budget constraint on the number of the new facilities to be located. It can also occur when there are capacity constraints on the demand that can be served by each facility. A demand point is said to be covered if it can be served within a given distance or time limit. In the private sector, a facility with a coverage constraint could be a restaurant with a time limit on delivery. In the public sector, locating an emergency medical service station (EMS) or a fire station that covers most of the population is an example of an MCLP.

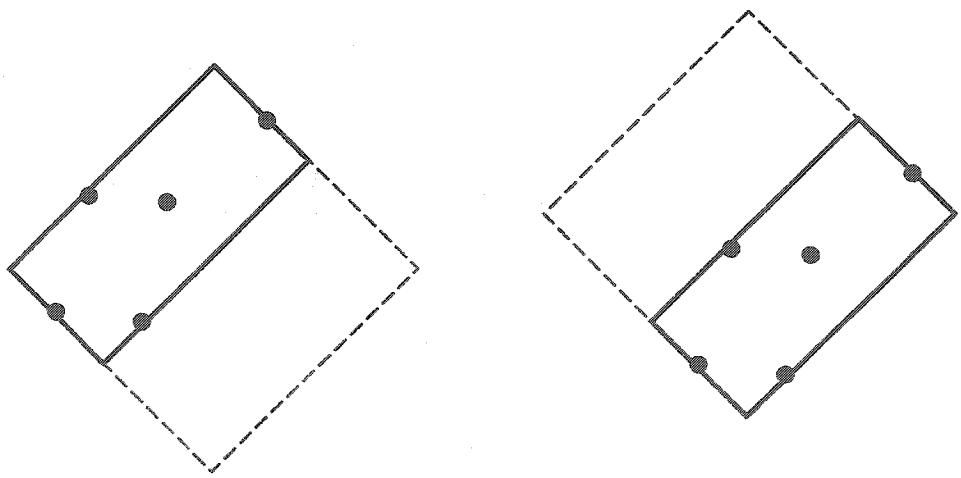


Figure 1.4: Full covering by diamond shape.

The MCLP can be stated as follows:

Problem statement: Consider n demand points $a_i = (a_{ix}, a_{iy})$ in R^2 with associated weights $w_i, i = 1,...,n$; find the location of p facilities that will cover the maximum demand weight.

Church (1984) introduced the planar maximal covering model (PMCLP). His method is mainly an extension of the original work on the MCLP by Church and ReVelle (1974). The MCLP, defined by Church and ReVelle, is structured to minimize the number of demand points left uncovered. The mathematical notation and formulation of MCLP are as follows:

Inputs:

DP = the set of demand points,

PS = the set of potential facility sites,

CM_i = coverage measure (distance or time) beyond which a demand point i is considered uncovered,

sd_{ij} = shortest distance from point i to site j, $i = 1,...,n; j \in PS$

w_i = weight associated with demand point $i, i = 1,...,n;$

p = the number of facilities to be located

$FS_i = \{j \in PS \mid sd_{ij} \leq CM_i\}$, the set of facility sites eligible to provide coverage to point i,

Decision variables:

$$h_j = \begin{cases} 1 & \text{if a facility is allocated to site } j, \\ 0 & \text{otherwise} \end{cases}$$

$$u_i = \begin{cases} 1 \text{ if demand point } i \text{ is not covered,} \\ 0 \text{ if demand point } i \text{ is covered} \end{cases}$$

$$(P2) \quad \text{Minimize} \sum_{i \in DP} w_i u_i \tag{1.4}$$

Subject to

$$\sum_{j \in FS_i} h_j + u_i \geq 1 \qquad \forall i \in DP \tag{1.5}$$

$$\sum_{j \in PS} h_j = p \tag{1.6}$$

$$u_i, h_j = (0,1) \quad \forall i \in DP, \forall j \in PS \tag{1.7}$$

The objective function of problem *P2* is to minimize the uncovered weight. If a point is covered, then the binary variable u_i is assigned a value of zero. Constraint (1.5) ensures that at least one facility is located in the set of facility sites eligible to provide coverage for a demand point *i*, FS_i. The number of facilities to be sited is limited to *p* by constraint (1.6).

Given *n* demand points, let CM_i be a rectilinear covering measure for point *i* where *i* is one of *n* demand points. Draw a diamond shape centered at point *i* with a rectilinear distance CM_i to its boundaries. The diamond shapes intersect set (DIPS) is defined as the set consisting of all demand points in addition to all points where some pair of diamond shapes intersects. The Church (1984) solution approach for the rectilinear maximal covering location problem (RPMCLP) is based on the fact that the optimal solution for the RPMCLP always belongs to set of intersection points of diamond shapes around demand points (the DIPS).

Consider any pair of intersecting diamond shapes; they either intersect at two points or share a line segment on one of their boundaries, Figure 1.5. In the case of two diamond shapes sharing a line segment, while any point on the line segment could be used, the two endpoints of the line segment are taken in the DIPS, Figure 1.5.b. Thus for n demand points, the DIPS will be finite set with at most n^2 members (demand points plus diamond shapes intercepts). The size of the DIPS can be further reduced by using the dominance rule where a facility site FS_i is said to dominate facility site FS_j if it covers the same demand as FS_j plus possibly more. Formulation *P2* and linear programming is then applied to find the optimal facilities location.

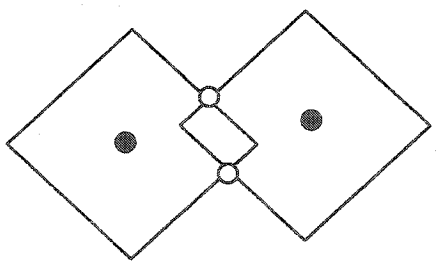


Figure 1.5.a: An example of diamond intersection points.

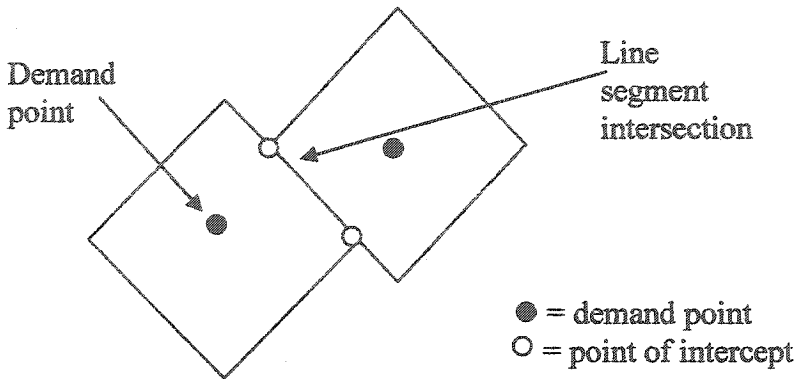


Figure 1.5.b: An example of DIPS points when two diamond shapes intersection is a line segment.

Figure 1.5: A possible intercept result of two diamond shapes.

1.5 Computational Aspects in Covering Problems

Location practitioners should provide, if possible, an optimal solution to location problems within a reasonable computational effort. In certain settings, some location models tend to be easy to solve; for example, covering location models on tree networks have polynomial time solutions (Karive and Hakimi (1979)). In other settings, however, location models can not be optimized in a reasonable amount of time.

The rectilinear maximal covering location problem (RPMCLP) is closely related to the *k*-center problem, an NP-Hard problem (Minieka 1970). On another setting, Megiddo and Supowit (1984) found that the geometrical covering by squares and rectangles is NP-Hard. This conclusion was derived from the 3-SAT problem (Gary and Johnson 1979). A similar approach was also used by Fowler et al. (1981).

On the other hand, in the Church covering model, the set of intersection points of diamond shapes was used as the set of possible optimal locations. This fact changed the search from locating the facility anywhere on the plane to a finite (and large) number of possible locations. The maximum size of DIPS in the Church method for RPMCLP is n^2. Church (1984) reported that a linear relaxation for formulation *P2* provides an optimal solution most of the time. However, in few cases, a branch and bound algorithm was needed. Thisse et al. (1984) provided the number of such block norms unit ball intersect set as $(n+n(n-1)(r(r-1)/2))$, where n is the number of demand points and r is the number of allowed directions of travels (that is, the number of the block norm diagonals).

1.6 Scope and Outline of the Thesis

Covering problems have been studied from different perspectives for different applications in different areas of science. For example, Barequet et al. (1995) studied convex polygon containment. The objective was to contain the maximum number of points, with possible applications in material cutting and line detection. Later, this problem was extended by Dickerson and Scharstein (1998) to the optimal placement of a *single c*onvex polygon (allowing both rotation and translation) to cover the maximum number of points. The algorithm complexity of Dickerson and Scharstein (1998) was a function of the number of points contained, the number of points on the plane, the polygon width and length, and the number of vertices of the polygon. Ventura and Dung (1993) studied parts inspection with rectangular and square shapes, using a Euclidean

least squares method to determine the optimal parameters of the sides of squares and rectangles.

Other applications that borrow from covering models have been reported in literature such as the apparel trim placement problem. Grinde and Daniels (1999) applied covering models from location theory to the apparel trim placement problem. The goal of this problem is to place the most trim pieces in the available container space. Grinde and Daniels used a Lagrangian heuristic to solve the MIP formulation based on the maximum covering location problem. Further, Brotcorne et al. (2002) proposed a heuristic for large-scale covering-location problems for cytological screening tests.

On the other hand, other researchers have studied the minimal covering problem for material salvage applications. For example, Drezner and Wesolowsky (1994) introduced mixed integer formulation to find the location of the minimum weight containment rectangle with sides parallel to the axes.

In addition to Church's (1984) PMCLP formulation, Mehrez et al. (1983-1985) studied PMCLP, developing in Mehrez et al (1985) a procedure for locating a facility that is "somewhat desirable". This problem known as "maximin-minimax" facility location is applicable in public service centers where the facility services are desirable and should be placed close to the residential area, but not too close in order to prevent noise and other disturbances (some examples are shopping malls and schools). The procedure of Mehrez et al (1985) finds the set of intersection points of any two lines forming the equi-rectilinear distances from the demand points. Other researchers used heuristic techniques and MCLP formulation for locating ambulances (Diaz and Rodriguez 1997).

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 use other distance measures in the PMCLP. Block norms as a distance measure have been applied in Fermat-Weber and Minimax location problems (Ward J. and Wendell R. (1980, 1985)). On the other hand, the mixed norms location problem concerns the location problem where facilities may have different coverage measure under different norms. Durier R., 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 center 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.

According to Avella et al. (1998), the location problem where demand points have different distance functions has not been studied thoroughly so far, so this issue is considered one of the niches of future research. The author's review of the literature indicates that the multiple facility planar maximal covering problems under different block norms has not been studied in the literature. Introducing block norms to the PMCLP would fill a gap in our knowledge of location theory.

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 polygonal shapes. Chapter Two starts by introducing a mixed integer program formulation (MIP) for block norms with two allowed directions of travel; these norms have been called block norms with $r = 2$. Without loss of generality, in this thesis we refer to second degree

block norms as block norms of degree two, or as parallelogram norms. The MIP of Chapter Two is more general than the Church MCLP formulation in that it allows facilities with different norms and coverage measure. The findings of Sections 2.2-2.4 and Section 3.2 have been accepted for publication in the *European Journal of Operational Research* (Younies and Wesolowsky 2004.a).

In Section 2.5, we extend the MIP formulation from the PMCLP for parallelogram shapes (block norms of degree two) to block norms with four allowed directions of travel (block norms of degree four). The block norm with four allowed directions of travel is the one-infinity norm which is octagonal in shape.

Unfortunately, while the MIP formulation is versatile and can be adapted to many cases, it cannot handle large-size problems efficiently. Therefore, one should seek an alternative solution method, such as problem specific algorithms or heuristic techniques.

Chapter Three introduces two exact solution methods that can handle one shape or a few shapes efficiently. The first algorithm, from Younies and Wesolowsky (2004.a), can handle only block norms with two directions of travel. The second algorithm, based on graph theory, can handle higher degree block norms. The algorithm of Section 3.4 is not limited to covering problems; this has been submitted for publication by Younies (2004).

When the number of demand points and the number of facilities considered increases, the MIP approach and the exact algorithm approach will not be sufficient. In Chapter Four, we use a heuristic approach based on genetic algorithms (GA) and the ideas set forth in Section 3.3. First, we present in Section 4.2 a brief review of the basic

elements of GA, and then, in Section 4.3, we convert PMCLP under different block norms into an unconstrained binary quadratic problem (UQP). This new representation as a UQP is a more appropriate format for the GA algorithm. Section 4.4 explains the different issues related to applying GA to solve the PMCLP under different block norms and its UQP equivalent. The material of Section 3.3 and Chapter Four will be submitted for publication by Younies and Wesolowsky (2004.b).

Finally, the work performed in this thesis is aimed at introducing block norms to PMCLP, introducing an alternative perspective, methodology, and solution technique for the PMCLP. In Chapter Five, we provide a summary of this work and its contributions and some recommendations for further future research.

CHAPTER TWO

A MIXED INTEGER PROGRAM FORMULATION FOR PLANAR MAXIMAL COVERING LOCATION PROBLEM UNDER BLOCK NORMS

2.1 Introduction

In this chapter, we introduce a zero-one mixed integer formulation (MIP) for a maximal covering problem, where demand points are discrete and a facility can be sited anywhere on the plane. The MIP formulation utilizes some of the geometrical properties of block norms contours, mainly the fact that the block norms considered are those with a polygonal like contour shape, where opposite sides are parallel. The formulation strategy is based on the difference between Y-intercepts of parallel straight lines. Therefore, to simplify references, we will refer to such even-sided polygons that have opposite sides parallel, as parallel-sided polygons (PSPs). In addition, we will use the terms "shape", "block norm contour" and "parallel-sided polygon" (or simply "polygon") interchangeably.

In Section 1.4.1, we will discuss the rectilinear single facility minimax problem and its relationship with the diamond shape full covering problem. To emphasize this relationship between the unweighted rectilinear distance minimax problem and the covering problem, we will show in Section 2.2 how some slight modifications will transform formulation *P1* of Section 1.4.1 into the single facility rectilinear maximal covering location problem (RPMCLP).

In rectilinear distance, the directions of travel are those parallel to the X- axis and the Y-axis. Nevertheless, the distance traveled parallel to the X-axis, or Y-axis, is Euclidean. This idea – the additive norm of Euclidean distance in two directions – can be generalized to block norms with unit balls in the shape of a parallelogram. The mixed integer program (MIP) formulation that handles both the rectilinear and the parallelogram covering problems is discussed in Section 2.3. The MIP formulation proposed in Section 2.3 can only be used with inclined parallelograms – although, of course, it can also be used for parallelograms with inclinations close to $\pi/2$. The special case of covering rectangles or squares with sides parallel to the axis is treated with a separate formulation in Section 2.4.

In Section 2.5, we extend the ideas of the MIP formulation of Section 2.3 to handle one-infinity norms. The formulation of Section 2.5 can handle facilities with block norms of the same degree and different coverage measures. Finally, the formulations proposed in this chapter can easily be adapted for other location and material cutting applications.

2.2 From Minimax to Maximal Covering

In Chapter One, the unweighted minimax problem under the rectilinear norm distance measure was shown to be equivalent to a full covering problem with diamond shapes. The diamond's size can be found with the help of the geometrical solution approach and the demand points' coordinates. Therefore, the maximum traveling distance in minimax problems is governed by the demand points' coordinates. This assumes that demand points are seeking the facility's service, regardless of the traveling distance. In the following, we show a variation of the diamond shape covering problem where

1) The facility cannot provide (quality) service beyond a certain distance (examples: emergency medical services (EMS) or pizza delivery)

2) The facility's service is not pursued by customers beyond that distance (example: a grocery store).

The change in coverage from covering all demand points in minimax problem into covering the maximum number of demand points within certain distance (in maximal covering), alters the problem from one of finding the parameters of the smallest diamond shape that covers all points to one of finding the location of a fixed size diamond shape that covers the maximum number of points. To illustrate, we will show how the minimax model described in Section 1.4.1 can be slightly modified to handle a single facility RPMCLP. In addition to notation used in Section 1.4.1 we will define the following notation:

Inputs

M = a large number

Z = Coverage distance measure beyond which a point is said to be uncovered.

Decision Variables

$$u_i = \begin{cases} 1 & \text{if demand point is covered} \\ 0 & \text{otherwise} \end{cases}$$

The single facility RPMCLP will be

$$(P3) \qquad \text{Maximize} \sum_{i=1}^{n} u_i \tag{2.1}$$

Subject to

$$\left[\left|x - u_i a_{ix}\right| + \left|y - u_i a_{iy}\right|\right] \leq Z + (1 - u_i)M \quad i = 1,\ldots,n \tag{2.2}$$

$$u_i \in \{0,1\} \tag{2.3}$$

$$x \geq 0, y \geq 0 \tag{2.4}$$

The objective in problem *P3* is to maximize the number of points enclosed by a diamond shape. Coverage of demand point i is achieved when $u_i = 1$ in (2.2). Therefore, the large number, *M*, will ensure that constraint (2.2) is redundant for the cases when a point is not covered. The weighted version of problem *P*3 can be easily achieved by introducing different weights for different points.

2.3 The Mixed Integer Formulation for the PMCLP with Second Degree Block Norms

In this section we start by introducing the idea of inclined parallelogram norms in Section 2.3.1. We introduce a mixed integer formulation (MIP) capable of choosing among many rectilinear and parallelogram shaped norms with different inclination, covering distance (size of the shape) and different placement costs in Section 2.3.2. Our MIP formulation for the PMCLP with inclined parallelograms makes use of a straight line equation and its Y-intercept. A detailed derivation of the approach and a zero-one mixed integer formulation will be shown in Section 2.3.2.

Finally, a computational example for covering 50 demand points in R^2 is presented in Section 2.3.3.

2.3.1 Parallelogram Norms

Consider any two points in R^2, a_1 and a_2 with coordinates (a_{1x}, a_{1y}) and (a_{2x}, a_{2y}). The rectilinear (l_1) norm can be written as $l_1(a_1, a_2) = \left[\left|a_{1x} - a_{2x}\right| + \left|a_{1y} - a_{2y}\right|\right]$. Note $\left|a_{1x} - a_{2x}\right| = \sqrt{(a_{1x} - a_{2x})^2}$ which is the Euclidean distance between the X-coordinates of points a_1 and a_2. That is

$l_2(a_{1x}, a_{2x}) = \left|a_{1x} - a_{2x}\right| = \sqrt{(a_{1x} - a_{2x})^2}$ and $l_2(a_{1y}, a_{2y}) = \left|a_{1y} - a_{2y}\right| = \sqrt{(a_{1y} - a_{2y})^2}$.

Hence, we can rewrite the rectilinear distance between a_1 and a_2 as $l_1(a_1, a_2) = l_2(a_{1x}, a_{2x}) + l_2(a_{1y}, a_{2y})$, where $l_2(a_{1x}, a_{2x})$ is the distance traveled parallel to the X-axis and $l_2(a_{1y} - a_{2y})$ is the distance traveled parallel to the Y-axis. Therefore, the

rectilinear distance norm can be considered as an additive norm of two axis-parallel Euclidean distances, Figure 2.1. Ward and Wendell (1985) realized that the rectilinear and Tchebycheff norms are block norms with $r = 2$.

Consider a road network that is dense in two non-orthogonal directions, and entailing unequal traveling speed in the two directions. Following the same analogy as in the one-infinity and rectilinear norms, the block norm shape will be a parallelogram with the parallelogram diagonals as the allowed traveling directions. This generalizes the idea of an additive norm of Euclidean distance in two directions into block norms with unit balls in the shape of a parallelogram. The rectilinear and Tchebycheff block norms can be considered as special cases of the parallelogram norm. Figure 2.2 shows a parallelogram norm and the possible directions of travel.

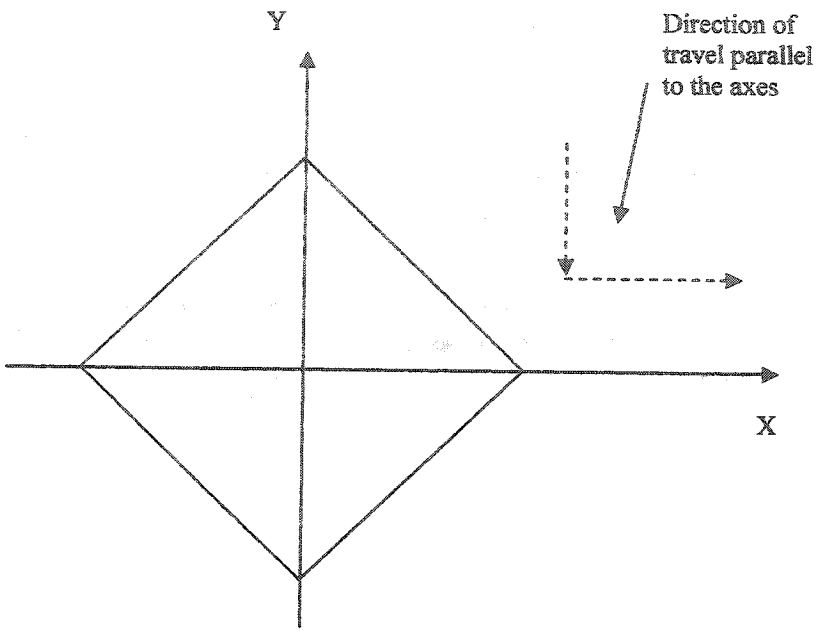


Figure 2.1: Equi-distance contour for rectilinear norm and traveling directions.

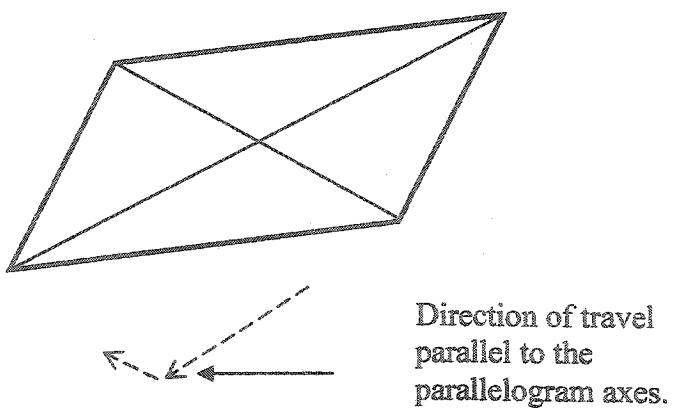


Figure 2.2: Equi-distance contour of parallelogram norm and traveling direction.

2.3.2 A Mixed Integer Program Formulation

This section considers maximal facility covering problems under second degree block norms. The common second degree block norms are the rectilinear and Tchebycheff norms, which have contours in a shape of parallelograms with 45 degrees inclination and sides parallel to the axis, respectively. Therefore, the general PMCLP with a second degree block norm is similar to covering by inclined parallelograms. We start this section by stating the problem; then we discuss the formulation. Computational results for this formulation are shown in Section 2.3.3.

Problem statement

Given n demand points (a_{ix}, a_{iy}) in R^2 with weights w_i, $i = 1,...,n$, and g facilities with coverage measure under block norms of second degree, and a setup cost for each facility C_k, $k = 1,...,g$, we must then find the locations of p facilities that will cover the maximum weight of points. Any given point is assigned to one facility only.

Analysis

Second degree block norm contours are parallelograms in shape, and, hence, the problem is similar to that of the covering problem by inclined parallelograms. Therefore, we would tackle the problem as if it were a maximal covering problem by parallelogram shapes. The inclined parallelogram maximal covering formulation assumes that none of the parallelogram sides are strictly parallel to the Y-axis. The case of square or rectangular shapes with sides parallel to the axes is discussed in Section 2.4.

Two straight lines parallel to each other can be defined with two straight-line equations that differ only in their Y-intercepts. In a parallelogram, two non-parallel sides

with larger Y-intercepts than the sides parallel to them can be selected. Therefore, a parallelogram can be defined by straight-line equations of these intersecting sides and the differences in Y-intercepts for parallel sides. The following property will relate the difference in Y-intercepts of these parallel lines, the inclination angle of the lines, and the perpendicular distance between them.

Property 2.1: *Consider a pair of parallel straight-lines L and L' with Y-intercepts b and b', respectively. The distance between the respective Y-intercepts b and b' can be given by* $I = |b - b'| = \left|\frac{d}{\cos\theta}\right|$ *for* $0 \le \theta \le \pi, \theta \ne \frac{\pi}{2}$, *where d is the perpendicular distance between the two lines and* θ *is the counterclockwise inclination angle of the lines from the X-axis.*

Proof: Easily derived from Figure 2.3 and Figure 2.4.

The case of $\pi/2 < \theta \le \pi$ is similar and is shown in Figure 2.4.

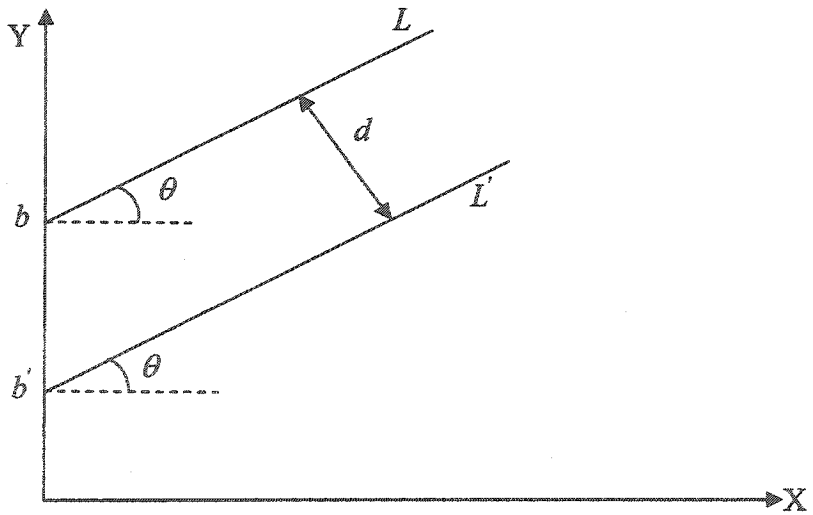


Figure 2.3: Illustration of Property 2.1 for $0 \le \theta < \frac{\pi}{2}$

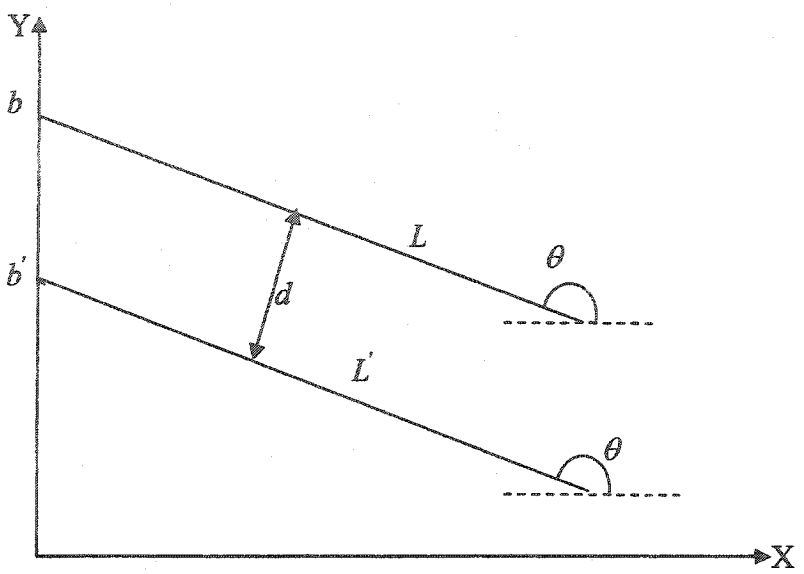


Figure 2.4: Illustration of Property 2.1 for $\frac{\pi}{2} < \theta \leq \pi$

Figure 2.5 illustrates a parallelogram with sides L_1 and L_2, with Y-intercepts b_1 and b_2; and the lines parallel to them L'_1 and L'_2, with Y-intercepts b'_1 and b'_2, respectively. In addition, the distances between the Y-intercepts of the parallel sides are denoted as I_1 and I_2.

Let L_1 and L_2 be two intersecting sides of a parallelogram such that the Y-intercepts of each are larger than that of its corresponding parallel side. The formulation to be shown will find the Y-intercepts of intersecting parallelogram sides with larger intercepts and, by property 2.1, the Y-intercepts for the other parallel sides can be found

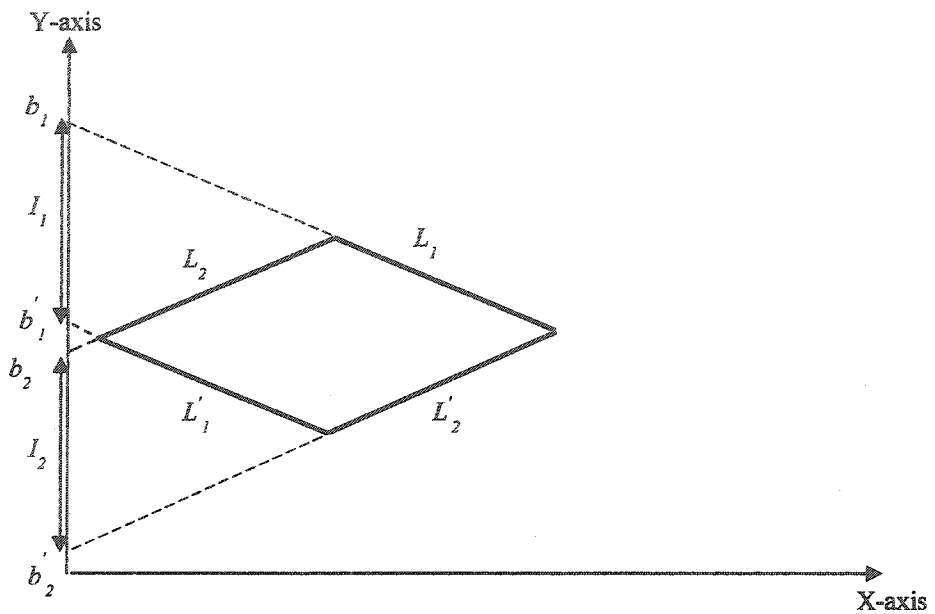


Figure 2.5: An inclined parallelogram with the Y-intercepts of its sides.

It can be shown from the straight line equation that the bounds on the maximum and minimum possible values of the Y-intercepts of lines through the points (a_{ix}, a_{iy}), $i = 1,...,n$, can be given by

$$\min_{i=1,...,n}\{a_{iy} - ma_{ix}\} \le b \le \max_{i=1,...,n}\{a_{iy} - ma_{ix}\}, \text{ where } m = \tan\theta \text{ is the slope of the line.}$$

This relation will be useful in the following formulation.

Using property 2.1 and the bounds on the Y-intercept, we develop an MIP formulation for locating p parallelograms selected from g parallelograms – each with known inclination angle, side lengths, and the angle between those sides. If the intercepts of the sides are given, the parallelogram's four corners can be found as the point of intersection of the intersecting sides and, hence, the parallelogram location can be found. The following notation will be used:

Inputs

L_{jk} parallelogram intersecting sides with larger Y-intercepts, $j = 1, 2; k = 1,\ldots, g$;

L'_{jk} parallelogram intersecting sides with smaller Y-intercepts, $j = 1, 2; k = 1,\ldots, g$;

θ_{jk} counterclockwise inclination angle of line L_{jk} in parallelogram k from the X-axis, $j = 1, 2; k = 1,\ldots, g$;

$m_{jk} = \tan\theta_{jk}$, slope of line L_{jk} in parallelogram k, $j = 1, 2; k = 1,\ldots, g$;

d_{jk} = the perpendicular distance between the parallel sides L_{jk} and L'_{jk} in parallelogram k, $j = 1, 2; k = 1,\ldots, g$;

I_{jk} = distance separating the Y-intercepts of sides L_{jk} and L'_{jk} in parallelogram k,

$$I_{jk} = \left|\frac{d_{jk}}{\cos\theta_{jk}}\right|,\ j = 1, 2; k = 1,\ldots, g;$$

C_k = cost of choosing parallelogram k, $k = 1,\ldots, g$;

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

Decision variables

$$u_{ik} = \begin{cases} 1 & \text{if demand point } i \text{ is covered by parallelogram } k, \\ 0 & \text{otherwise} \end{cases}$$

$$h_k = \begin{cases} 1 \text{ if parallelogram } k \text{ is chosen} \\ 0 \text{ otherwise} \end{cases}$$

b_{jk} = the Y-intercept of line L_{jk} in parallelogram k with the Y-axis, $j = 1, 2; k = 1,\ldots, g$.

Following the above notation, we will denote the sides of parallelogram k with slope m_{1k} as L_{1k}, L'_{1k}, and the intersecting sides with slope m_{2k} as L_{2k} and L'_{2k}.

In the following formulation, a large number M is used as a multiplier. The minimum possible value of the multiplier can be determined from the bounds on the Y-intercepts and the distance separating the Y-intercepts of the given parallelograms. As explained below, it can be shown that $M \geq 2 \max_{k=1,\ldots,g} \left\{ \left| \min_{i=1,\ldots,n} (a_{iy} - m_k a_{ix}) \right|, \left| \max_{i=1,\ldots,n} (a_{iy} - m_k a_{ix}) \right| \right\}$ is adequately large.

The inclined parallelogram maximal covering formulation is as follows:

$$(P4) \text{ Maximize } \sum_{k=1}^{g} \sum_{i=1}^{n} w_i u_{ik} - \sum_{k=1}^{g} C_k h_k \tag{2.5}$$

Subject to

$$a_{iy} - m_{jk} a_{ix} - b_{jk} + I_{jk} \geq M(u_{ik} - 1) \qquad j = 1,2; i = 1,\ldots,n; k = 1,\ldots,g; \tag{2.6}$$

$$a_{iy} - m_{jk} a_{ix} - b_{jk} \leq M(1 - u_{ik}) \qquad j = 1,2; i = 1,\ldots,n; k = 1,\ldots,g; \tag{2.7}$$

$$\sum_{k=1}^{g} u_{ik} \leq 1 \qquad i = 1,\ldots,n; \tag{2.8}$$

$$h_k - u_{ik} \geq 0 \qquad i = 1,\ldots,n; k = 1,\ldots,g; \tag{2.9}$$

$$\sum_{k=1}^{g} h_k = p \tag{2.10}$$

$$u_{ik}, h_k \in \{0,1\} \text{ for } i = 1,\ldots,n;\ k = 1,\ldots,g. \tag{2.11}$$

$$b_{jk} \geq 0, j = 1,2, k = 1,\ldots,g. \tag{2.12}$$

In the above formulation, constraints (2.6) and (2.7) refer to each of the parallelogram's parallel sides. Constraint (2.6) represents the side with lower Y-intercept,

and constraint (2.7) represents the side with larger Y-intercept. If a parallelogram is chosen and a point is assigned to a parallelogram, the right hand side (RHS) in constraints (2.6) and (2.7) will equal zero, indicating that the point is on or between the parallelogram parallel sides. If a parallelogram is not chosen or a point is not assigned to it, then constraints (2.6) and (2.7) will be satisfied because of the multiplier *M*. Constraints (2.8) ensure that if a point is assigned, it will be assigned to one parallelogram only, while constraint (2.9) ensures that no point is assigned if a parallelogram is not chosen. Finally, the number of parallelograms chosen is governed by constraint (2.10).

Instead of using an arbitrarily large number, it can be easily shown that *M* in (2.6) can be replaced by $2\max_{k=1,\ldots,g}\left\{\min_{i=1,\ldots,n}(a_{iy}-m_{jk}a_{ix})\right\}$ and that *M* in (2.7) can be replaced by $2\max_{k=1,\ldots,g}\left\{\left|\max_{i=1,\ldots,n}(a_{iy}-m_{k}a_{ix})\right|\right\}$.

Once the intercepts of a parallelogram are found by the program, the parallelogram location can be found. To find the corners of the parallelogram, one can solve for the coordinates from the straight-line equations of two intersecting sides. In the following, we will illustrate how to find the corner location for an inclined parallelogram from the Y-intercepts.

Figure 2.6 shows an inclined parallelogram, *k*, and the notation for the sides. The corner location (x_{ck}, y_{ck}) can be found from the intersection of straight line equations of lines L'_{1k} and L'_{2k} in Figure 2.6:

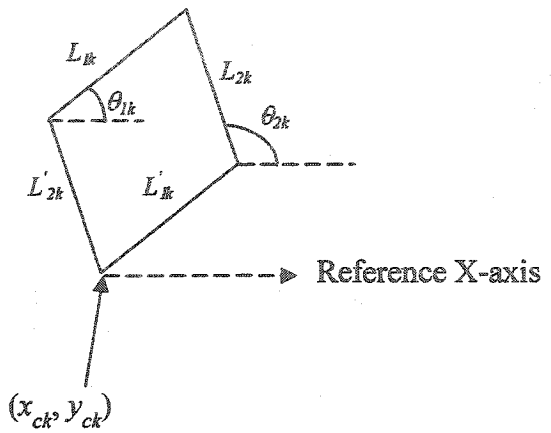


Figure 2.6: Solving for inclined parallelogram corner (x_{ck}, y_{ck}).

$$y_{ck} = m_{1k}x_{ck} + b_{1k} - I_{1k}\ , \qquad (2.13)$$

$$y_{ck} = m_{2k}x_{ck} + b_{2k} - I_{2k}\ . \qquad (2.14)$$

By solving the above two equations, the corner location is

$$x_{ck} = \frac{b_{1k} - b_{2k} - I_{1k} + I_{2k}}{m_{1k} - m_{2k}} \qquad and \quad y_{ck} = m_{1k}\left(\frac{b_{1k} - b_{2k} - I_{1k} + I_{2k}}{m_{1k} - m_{2k}}\right) + b_{1k} - I_{1k}\ .$$

Note that one can modify the previous formulation to solve directly for one of the parallelogram corners. In a modified formulation that finds the corner between the parallelogram sides with lower Y-intercepts, constraints (2.6) and (2.7) in (*P4*) will be changed as follows:

$$a_{iy} - m_{jk}\left(a_{ix} - x_{ck}\right) - y_{ck} \geq M(u_{ik} - 1) \qquad j = 1,2; i = 1,\ldots,n; k = 1,\ldots,g \quad (2.15)$$

$$a_{iy} - m_{jk}(a_{ix} - x_{ck}) - y_{ck} - I_{jk} \le M(1-u_{ik}) \qquad j=1,2; i=1,...,n; k=1,...,g \quad (2.16)$$

For problems with g parallelograms and n demand points, the formulation takes $g(n+1)$ binary decision variables and $2g$ continuous variables for the Y-intercepts.

2.3.3 Computational Example

While the proposed formulations can handle weighted points, points with equal weights are easier to present in a graph. Figure 2.7 shows the case of finding the location of two shapes. The first shape is an inclined parallelogram with side lengths of 2.5 and 3, and counterclockwise inclination angles of 30 and 150 degrees, respectively. The second shape is an inclined square with 2.5 side length and a 45 degrees inclination angle. The MIP solver used was XPRESS-MP software on an Ultra-4 SPARC Sun station. The time for the best solution was 59 seconds, and the total time for the program to complete its search was 333 seconds.

The inclined parallelogram formulation was tested for inclination angles close to $\pi/2$. We tested for inclination angles of 90.01 degrees and 89.99 degrees – with satisfactory results. It should be mentioned that, to avoid numerical errors, the multiplier value should not be taken arbitrarily large.

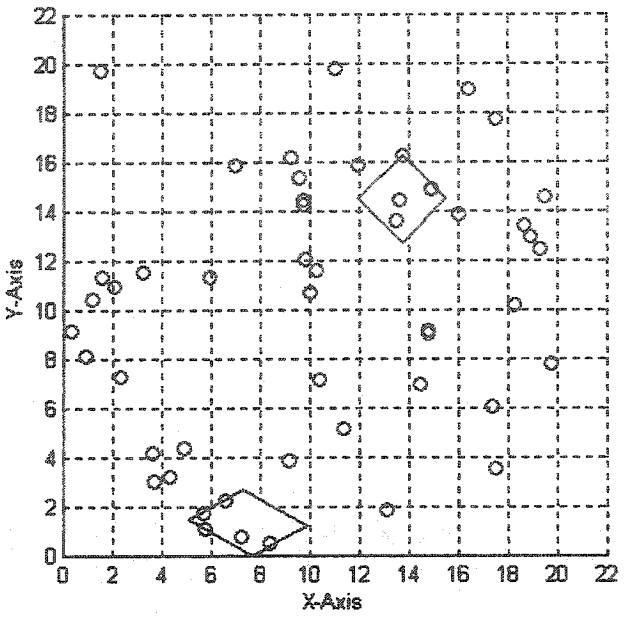


Figure 2.7: Maximum covering location for one inclined parallelogram and one inclined square.

2.4 Covering by Rectangles with Sides Parallel to the Axes

In Section 2.3 we introduced an MIP formulation for inclined parallelograms, based on the slope of parallelogram sides. If the shape were rectangular with sides parallel to the axes, the previous formulation would fail for a rectangle with a side exactly parallel to the Y-axis. In this section, an MIP formulation for this special case is introduced. The formulation for rectangles with sides parallel to the axes will be based on finding the lower left corner coordinates of the rectangles that cover the maximum weight. The Drezner and Wesolowsky formulation (1994) solves for the same rectangle

corner point. Their formulation requires $5n$ binary variables and two continuous variables to find the minimum number of points covered by one rectangular shape.

The formulation introduced in this section requires only $(n+1)$ binary variables and two continuous variables for each rectangular shape. The following notation will be used for the case of rectangles with sides parallel to the axes.

Inputs

l_{xk} = side length parallel to the X-axis for rectangle k, $k = 1,\ldots, g$;

l_{yk} = side length parallel to the Y-axis for rectangle k, $k = 1,\ldots, g$;

C_k = cost of choosing rectangle k, $k = 1,\ldots, g$;

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

Decision variables

$$u_{ik} = \begin{cases} 1 & \text{if demand point } i \text{ is covered by rectangle } k, \\ 0 & \text{otherwise} \end{cases}$$

$$h_k = \begin{cases} 1 \text{ if rectangle } k \text{ is chosen,} \\ 0 \text{ otherwise.} \end{cases}$$

x_{ck} = X-coordinate of lower left corner of rectangle k, $k = 1,\ldots, g$;

y_{ck} = Y-coordinate of lower left corner of rectangle k, $k = 1,\ldots, g$.

As in the formulation for inclined parallelograms in Section 2.3, a large number multiplier, M, is used. It can be shown that M to be sufficient large if $M \geq \max_{i=1,\ldots,n} \{a_{ix}, a_{iy}\}$.

The formulation for maximum covering by rectangles with sides parallel to the axes is

$$(P5)\quad \text{Maximize} \sum_{k=1}^{g}\sum_{i=1}^{n} w_i u_{ik} - \sum_{k=1}^{g} C_k h_k \tag{2.17}$$

Subject to

$$l_{xk} - a_{ix} + x_{ck} \geq M(u_{ik} - 1) \qquad i = 1,...,n; k = 1,...,g; \tag{2.18}$$

$$l_{yk} - a_{iy} + y_{ck} \geq M(u_{ik} - 1) \qquad i = 1,...,n; k = 1,...,g; \tag{2.19}$$

$$a_{ix} - x_{ck} \geq M(u_{ik} - 1) \qquad i = 1,...,n; k = 1,...,g; \tag{2.20}$$

$$a_{iy} - y_{ck} \geq M(u_{ik} - 1) \qquad i = 1,...,n; k = 1,...,g; \tag{2.21}$$

$$h_k - u_{ik} \geq 0 \qquad i = 1,...,n; k = 1,...,g; \tag{2.22}$$

$$\sum_{k=1}^{g} h_k = p \tag{2.23}$$

$$\sum_{k=1}^{g} u_{ik} \leq 1 \qquad i = 1,...,n; \tag{2.24}$$

$$u_{ik}, h_k \in \{0,1\} \text{ for } i = 1,...,n;\ k = 1,...,g; \tag{2.25}$$

$$x_{ck}, y_{ck} \geq 0 \qquad k = 1,...,g. \tag{2.26}$$

A brief explanation of the above formulation follows. If a rectangle is chosen and a point is assigned to it, then constraints (2.20) and (2.21) will ensure that the point lies to the right and above the lower left corner. Similarly, constraints (2.18) and (2.19) will ensure that the point lies to the left and below the top right corner. While constraint (2.23) sets the number of rectangles used, constraints (2.22) ensure that no points are assigned if a rectangle is not chosen. If a rectangle is not chosen, or if a point is not inside the rectangle, then the RHS of constraints (2.18) to constraints (2.21) will be negative, while

the minimum value for the LHS of the constraints will not exceed *M*. Finally, constraint (2.24) will ensure any point assigned is assigned to one rectangle only.

Since the objective in Sections 2.3 and 2.4 formulations is to find the maximum weight enclosed by one or more parallelograms or rectangles, the formulation will allow the shapes to overlap. However, any point will be assigned once only.

2.4.1 Computational Example

In this section, we illustrate the problem of choosing *p* shapes out of a given *g* shapes. A special example – with 20 equally weighted demand points – is shown in Figure 2.8. Three rectangular shapes, with sides parallel to the axis, were considered: rectangle **A** with side lengths of 2 and 1; square **B** with side length of 2; and rectangle **C** with side lengths of 4 and 2 – where all sides are parallel to the X and Y-axis, respectively. To exemplify the cost of choosing any of these shapes, consider the case where one wants to cut the maximum weight set of points from a given piece of material. If the time taken to cut a rectangular piece is in direct proportion to the total distance taken to cut the rectangular shape, then it will be reasonable to take the cost of choosing a rectangle as proportional to its perimeter. The optimal solution shown in Figure 2.8 shows that rectangles **A** and **B** have been chosen. The figure shows what might be the extra points if rectangle **C** were chosen. For the case of Figure 2.8, with 63 binary variables and 6 continuous variables, the time taken was 13 seconds to find the optimal solution, and 26 seconds for the solver to complete the search process.

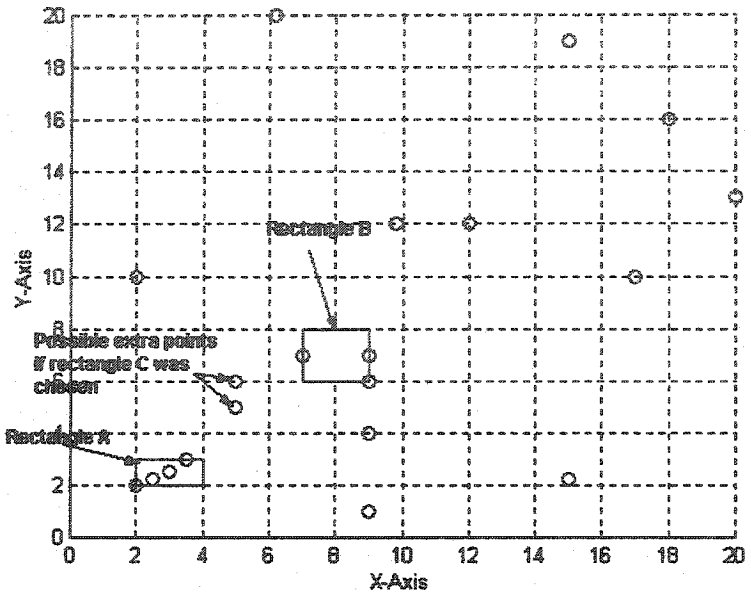


Figure 2.8: Result for choosing the minimal cost maximal cover.

2.5 Covering under the One-Infinity Norm

In Sections 2.3 and 2.4 we introduced a formulation that can handle the PMCLP for facilities with the rectilinear, parallelogram, or Tchebycheff coverage measure (second degree block norms). In this section, we extend the analysis to location of facilities with coverage measure under the one-infinity norm (a block norm of degree four). 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 another degree. In Figure 2.9, we

review the one-infinity block norm contours diagram previously given in Section 1.3. The one-infinity block norm is a linear combination of rectilinear and Tchebycheff norms. Let λ_1' and λ_2' be nonnegative 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 λ_1' represent the proportion of trip made via rectilinear distance and λ_2' represent the proportion of trip made via diagonal road. 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 λ_1' and λ_2', the degree of travel along the rectilinear and diagonal roads, respectively. First, we state the problem of planar maximal covering under one infinity block norm, and follow this with a discussion.

Problem statement:

Given n points (a_{ix}, a_{iy}) in R^2 and weights $w_i, i = 1,...,n$, g facilities with a coverage measure under the one-infinity block norm, and a facility setup cost $C_k, k = 1,...,g$, 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 2.10 instead of Figure 2.9. Figure 2.10 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 center is denoted as (x_c, y_c), and the unit contour radius length as $\rho_{ik}, i = 1,...,5, k = 1,...,g$.

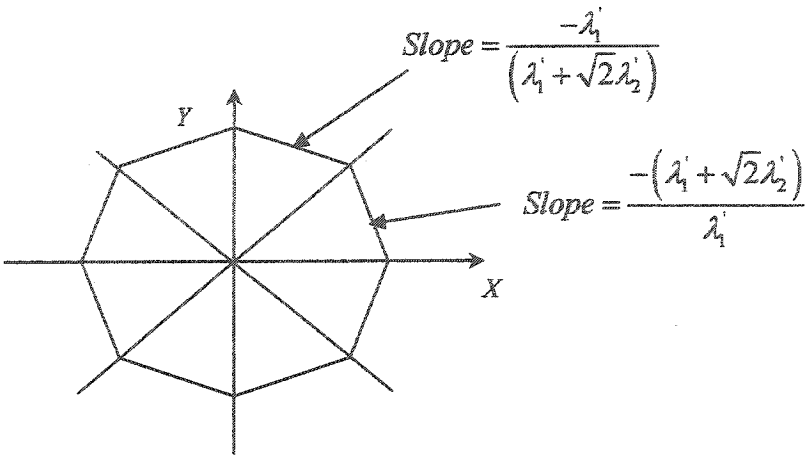


Figure 2.9: Contour of one infinity block norm

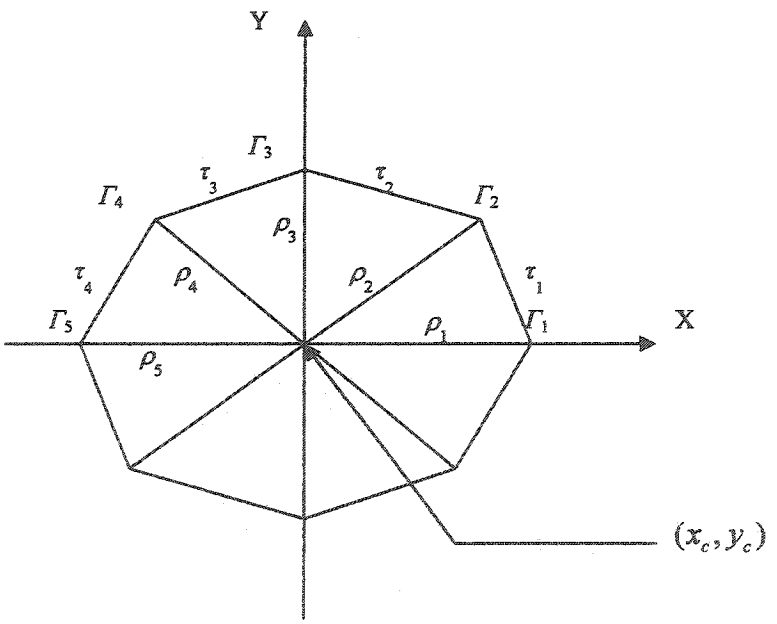


Figure 2.10: The one-infinity block norm unit ball.

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, $k = 1,...,g; j = 1,...,4;$

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, $k = 1,...,g; j = 1,...,4;$

ρ_{ik} = radius length in the one infinity block norm of facility k, $i = 1,...,4$, $k = 1,...,g$.

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

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

w_i = weight associated with point i, $i = 1,...,n$.

Decision variables

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

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

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

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

y_{ck} = the Y -coordinate of facility k, $k = 1,...,g$.

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 2.10 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:

$$(P6)\quad \text{Maximize} \sum_{k=1}^{g}\sum_{i=1}^{n} w_i u_{ik} - \sum_{k=1}^{g} C_k h_k \tag{2.27}$$

Subject to

$$a_{iy} - m_{jk}a_{ix} - b_{jk} \le M(1-u_{ik}) \qquad i=1,...,n;\ j=1,...,4;\ k=1,...,g; \tag{2.28}$$

$$a_{iy} - m_{jk}a_{ix} - b_{jk} + I_{jk} \ge M(u_{ik}-1) \qquad i=1,...,n;\ j=1,...,4;\ k=1,...,g; \tag{2.29}$$

$$y_{ck} + \rho_{2k}\sin\theta'_{1k} = m_{jk}(x_{ck} + \rho_{2k}\cos\theta'_{1k}) + b_{jk} \qquad k=1,...,g;\ j=1,2; \tag{2.30}$$

$$y_{ck} + \rho_{4k}\sin\theta'_{2k} = m_{jk}(x_{ck} - \rho_{4k}\cos\theta'_{2k}) + b_{jk} \qquad k=1,...,g;\ j=3,4; \tag{2.31}$$

$$\sum_{k=1}^{g} u_{ik} \le 1 \qquad i=1,...,n; \tag{2.32}$$

$$h_k - u_{ik} \ge 0 \qquad i=1,...,n;\ k=1,...,g; \tag{2.33}$$

$$\sum_{k=1}^{g} h_k = p \tag{2.34}$$

$$u_{ik}, h_k \in \{0,1\} \text{ for } i=1,...,n;\ k=1,...,g. \tag{2.35}$$

$$b_{jk}, x_{ck}, y_{ck} \ge 0\,. \tag{2.36}$$

In the above formulation, constraints (2.28), (2.29), and (2.32)-(2.36) are similar to those in Section 2.3. The facilities locations are determined by constraints (2.30) and (2.31). In constraints (2.30) the facilities location is determined from XY-coordinates of point Γ_2, while constraints (2.31) 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 2.11 shows the case for maximal covering by a single octagonal shape for 50 points of equal weights. The run time was 9 seconds on Ultra-4 SPARC Sun station.

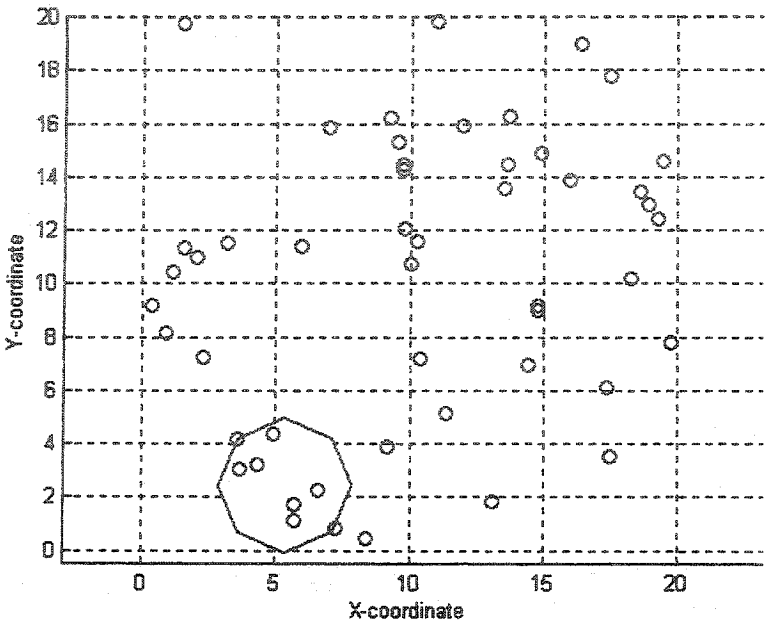


Figure 2.11: Output of covering by one-infinity block norm (octagonal shape).

CHAPTER THREE

EXACT ALGORITHMS FOR PLANAR MAXIMAL COVERING LOCATION PROBLEM UNDER DIFFERENT BLOCK NORMS

3.1 Introduction

In Chapter Two we introduced a MIP formulation for the rectilinear planar maximum covering problem (RPMCLP), parallelogram PMCLP, and a one-infinity norm PMCLP. In this chapter we introduce alternative exact algorithms for a single facility planar maximum covering problem (PMCLP) under different block norms. First, in Section 3.2, we introduce an exact algorithm that can handle covering by a single inclined parallelogram or a rectangular shape with sides parallel to the axis. For other block norms, such as the one-infinity norm, an alternative approach based on the maximum clique problem (MCP) is also shown. In Section 3.3 we discuss the equivalence of maximum covering by a block norm and the maximum clique problem. In Section 3.4, we illustrate an implicit enumeration algorithm for the maximum clique problem. The material in this chapter, especially that in Section 3.3, acts as an introduction to Chapter Four.

3.2 Alternative Algorithm for Maximal Covering by a Parallelogram

In Chapter Two, it has been shown that the PMCLP for facilities under second degree block norms is a special case of covering by parallelogram shapes. The following algorithm solves for the optimal location of a single parallelogram of known side lengths, inclination angle from the X-axis, and inscribed angles between parallelogram sides. The algorithm uses the idea that the equations of the parallelogram's parallel sides differ only in their Y-intercepts. In a given parallelogram, there are two differences between Y-intercepts – one for each pair of parallel sides. The difference between the Y-intercepts of the parallelogram parallel sides can be found as shown in Property 2.1. Thus, the Y-intercepts of lines parallel to the parallelogram sides and passing through points inside the parallelogram, would be bounded by the Y-intercepts of those respective parallelogram parallel sides. Therefore, the maximum weight set of points that can be covered by the parallelogram will be the point set, with Y-intercepts of lines through them and parallel to the parallelogram sides – within the parallelogram Y-intercepts.

The algorithm consists of two parts. The first part creates two sorted Y-intercept lists for lines parallel to the parallelogram's parallel sides through points $a_i = (a_{ix}, a_{iy}), i = 1,...,n.$ Then, sorted solution sets from each sorted Y-intercepts list are found. The second part of the algorithm finds, from the intersection of the solution sets, the point set of maximum total weight. The algorithm can be used for parallelograms of any inclination and for points with any weight. The algorithm can also be used for rectangles with sides parallel to the axes by replacing the two sorted Y-intercepts list of

the lines passing through the demand points with the sorted XY- coordinates of the demand points.

Part A: Create the lists

1. Using Property 2.1, find the Y-intercept differences between the parallelogram's parallel sides, that is I_1 and I_2.
2. Find the Y-intercepts of lines parallel to the parallelogram's sides, with side slopes m_1 and m_2 through the points $a_i, i = 1,...,n$. Save the points' index, its corresponding Y-intercept, and points' weights, in two three-column lists, B_1 and B_2. That is, $B_j(i,1) = i,\ B_j(i,2) = a_{iy} - m_j a_{ix}; B_j(i,3) = w_i, i = 1,...,n; j = 1,2$
3. Sort the B_1 and B_2 lists in descending order according to the Y- intercepts values, and name these new lists Bs_1 and Bs_2, respectively.
4. Find the points that are I_1 or less from each other in the Y-intercepts column in Bs_1, compute their total weight, and save an indexed solution in a solution list, SL_1. Figure 3.1 sketches, in MATLAB, the procedure to find this solution list.
5. Repeat step 4 for the Bs_2 list and intercept separation I_2, and save the indexed solution in a second solution list, SL_2.
6. Sort the solution lists, SL_1 and SL_2, in descending order according to the total weight column. Save the sorted lists in $SL_1^{'}$ and $SL_2^{'}$, respectively.

Sorting in steps 3 and 6 is performed four times with complexity of $O(n \log n)$. Step 2 is done twice for each parallel-sides slope, with n computations each. The number of

computations of steps 4 and 5 depends on the points distributions, and on the value of the Y-intercepts difference. It will take $\left(\frac{n}{2}(n+1)\right)$ computations in the worst case for steps 4 and 5. This worst case occurs when all the points lie within the parallelogram's parallel sides. However, it must be noted that, for such a case, the search process in the second part of the algorithm would be trivial.

points = 0
$i = 1$
$j = i$
weights = 0
While $i \le n$
While $abs\left(Bs_1(i,2) - Bs_1(j,2)\right) \le I_1$
points = points + 1
$weights = weights + Bs_1(j,3)$
$j = j + 1$
if $j > n$
break
else
end
end
$SL_1(i,1) = i$
$SL_1(i,2) = \text{points}$
$SL_1(i,3) = weights$
$i = i + 1$
$j = i$
weights = 0
points = 0
end

Figure 3.1: Computation of the solution list SL_1.

Part B: Finding the Location of the Maximum Covering Parallelogram

1. Set the final solution to an initial value, and denote this value as FV. Also, set a counter to zero, and indexes Λ_1 and Λ_2 to 1, and create empty final-answer lists B_{1final} and B_{2final}.
2. Prune any elements in $SL_1^{'}$ and $SL_2^{'}$ that have a total weight less than FV.
3. Set $j = SL_1^{'}(\Lambda_1, 1)$ and $j_{final} = j + SL_1^{'}(\Lambda_1, 2)$. Similarly, set $k = SL_2^{'}(\Lambda_2, 1)$ and $k_{final} = k + SL_2^{'}(\Lambda_2, 2)$. (Recall from Part A, that $SL_1^{'}$ and $SL_2^{'}$ are three-column lists)
4. Compare $Bs_1(j,1)$ with $Bs_2(k,1)$
   a) If $Bs_1(j,1) = Bs_2(k,1)$, that is, if the points indices are equal, add point weight to counter; save the respective Y-intercepts values, that is, $Bs_1(j,2)$ and $Bs_2(k,2)$ in temporary-answer lists, and go to (b). Otherwise, go to (c).
   b) Set $j = j+1$; if $j < j_{final}$, set $k = SL_2^{'}(\Lambda_2, 1)$ and $k_{final} = k + SL_2^{'}(\Lambda_2, 2)$, and go to (a). Otherwise, go to step 5.
   c) Set $k = k+1$; if $k < k_{final}$, go to (a). Otherwise, go to (d).
   d) Set $j = j+1$; if $j < j_{final}$, set $k = SL_2^{'}(\Lambda_2, 1)$ and $k_{final} = k + SL_2^{'}(\Lambda_2, 2)$, and go to (a). Otherwise, go to step 5.
5. If the counter is greater than FV, save counter in FV. Place the Y-intercepts values from the temporary answer lists in B_{1final} and B_{2final}, and re-initialize counter and temporary answer lists. Otherwise, if FV is greater than counter, then set counter to zero and re-initialize the temporary answer-list elements.

6. Set $\Lambda_1 = \Lambda_1 + 1$; if $SL_1'(\Lambda_1, 3) < FV$ and $\Lambda_1 < |SL_1'|$, go to step 6. Otherwise, go to step 7.
7. Set $\Lambda_2 = \Lambda_2 + 1$; if $SL_2'(\Lambda_2, 3) < FV$ and $\Lambda_2 < |SL_2'|$, go to step 7. Otherwise, go to step 8.
8. Repeat steps 3 to 7 until $\Lambda_1 = |SL_1'|$ and $\Lambda_2 = |SL_2'|$.

The results in B_{1final} and B_{2final} are the Y-intercepts of lines passing through maximum weight points in a parallelogram. Without loss of generality, since the points with extreme Y-intercepts (the covered points with the smallest or largest Y-intercepts) can be located on the parallelogram sides, the parallelogram can be translated in X and Y directions to have at least one point on each side of two intersecting sides. To find the location of the parallelogram, one can use the parallelogram's given information and the location of one of its corners. As an example of finding the corner between the parallelogram's sides with larger Y-intercepts, take the maximum of B_{1final} and B_{2final} and, using the straight line equations as described in Section 2.3.2, solve for the corner coordinates.

The algorithm starts with the element of maximum weight in the solution lists, and, to avoid total enumeration, FV is used as a lower bound and is updated in the search procedure. The computational complexity of part B of the algorithm will depend on the solution lists found in steps 4 and 5 in part A of the algorithm. The worst case occurs when the optimal solution is at the end of the solution lists SL_1' and SL_2'.

For the case of rectangular shapes with sides parallel to the axes B_1 and B_2, lists will be replaced by the points' Y-coordinates for B_1 and X-coordinates for B_2.

3.2.1 Computational Results

Figure 3.2 shows the results, for one parallelogram shape, of both an MIP formulation from Chapter Two and the algorithm from Section 3.2. In both cases, the same points have been covered by the parallelogram. Without loss of generality, the parallelograms in Figure 3.2 can be translated in both X-axis and Y-axis directions to have an exact overlap. The running time for the algorithm was 2 seconds and the running time for the MIP was 5 seconds.

Table 3.1 shows the running time for diamond shapes of different sizes for both the MIP and the algorithm of Section 3.2. Note that the MIP software running time did not exceed the 5 seconds running time for one shape. In fact, our experiments show that the running time for the MIP decreases as the size of the diamond shape increases. For the algorithm, the running time will initially increase with the shape size; however, it will eventually start to decrease when the shape size increases enough to cover more than half of the points. Table 3.1 also shows that computational time was much less than one second in the last three cases. Finally, while the running time for the algorithm is somewhat greater than that for the MIP for diamond shape cases, it was less for parallelogram shape. In fact, for shapes of different sizes and equal inclination angles (as

in Table 3.1), the Y-intercept lists need to be created only once. In addition the algorithm can handle points of negative weight which was not possible in the case of the MIP.

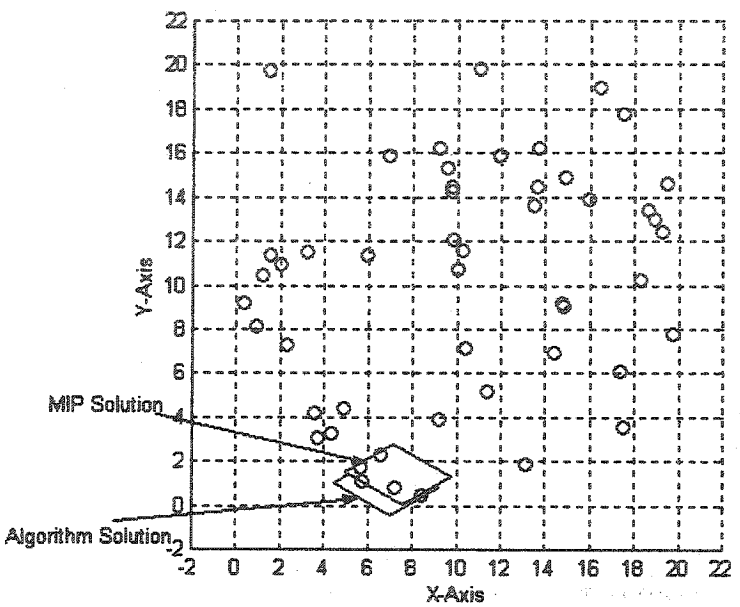


Figure 3.2: Result from MIP and algorithm.

Table 3.1: Computational comparisons for maximal covering by a diamond shape.

Side Length	Running Time (sec)		Objective Value	
	MIP	Algorithm	MIP	Algorithm
3.5	5	3	6	6
4.5	5	4	8	8
5.5	5	5	10	10
6.5	4	5	13	13
7.5	5	6	14	14
8.5	3	6	16	16
9.5	4	6	19	19
10.5	1	6	22	22
12.5	1	4	28	28
14.5	0.6	3	32	32
16.5	0.4	0.5	41	41
18.5	0.4	0.28	46	46

3.3 Transforming Maximum Covering by PSP's to a Maximum 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 the maximum covering by block norms problems or, alternatively, the maximum covering by parallel-sided polygon (MCPSP) to 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 3.1 (Diestel page 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 $(u,v)\in E$".

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

Figure 3.3 shows an example of an interval graph. Figure 3.3.a shows a graph where vertices are connected if assigned labeled intervals, shown in Figure 3.3.b, overlap. Note that the size of intervals is not necessarily unique in Figure 3.3.b and is given for illustration purposes only.

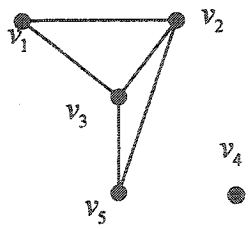


Figure 3.3.a

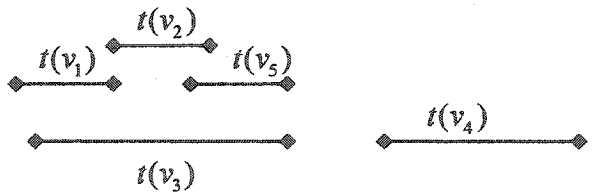


Figure 3.3.b

Figure 3.3: Intervals and the corresponding interval Graph.

Consider n points in R^2; $a_i=(a_{ix},a_{iy})$ $i=1,...,n$ 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, $i=1,...,n$.

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

Proposition 3.1: If every demand point $a_i=(a_{ix}, a_{iy})$, $i=1,...,n$, 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 Figures 3.4.a and 3.4.b. Figure 3.4.a 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 3.4.b represents the equivalent interval graph. From proposition 3.1, 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 δ.

If the distance between the points' Y-intercepts is greater than δ, then they will not be connected in *G*.

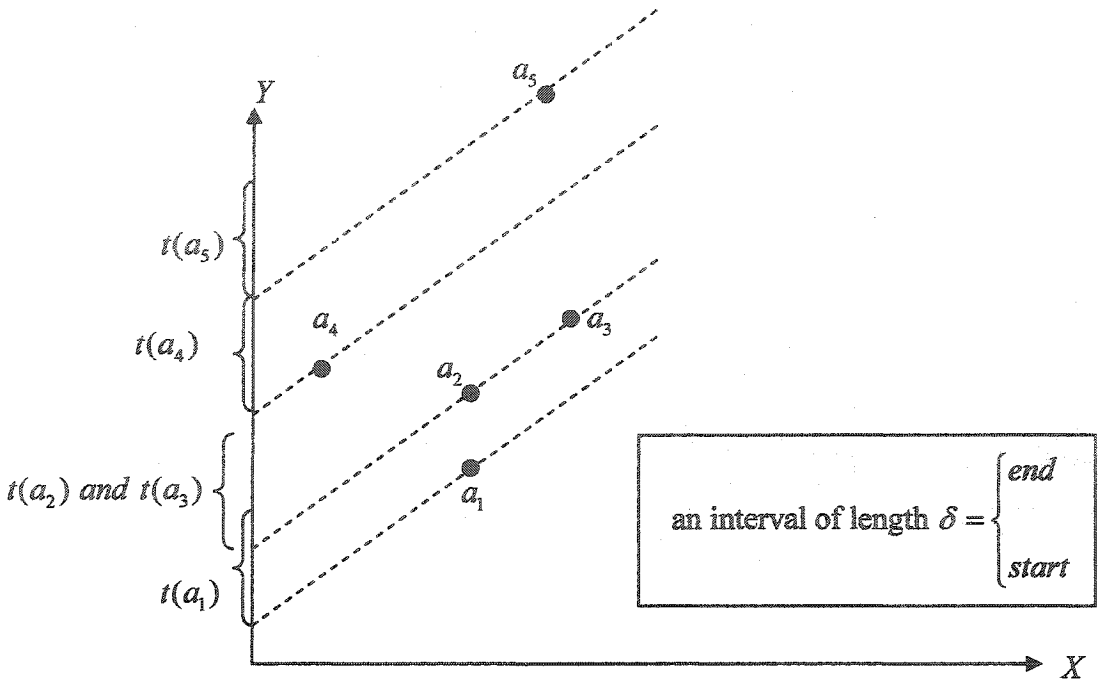


Figure 3.4.a: Points in R^2 and the corresponding intervals assigned to their Y-intercepts.

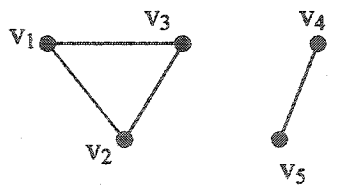


Figure 3.4.b: The equivalent interval graph of Figure 3.4.a.

In definition 3.1 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 3.1 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 3.2: *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 $t_j(v), j=1,...,p; p \geq 2$ such that $\forall j \ t_j(v_i) \cap t_j(v_k) \neq \varnothing; i,k=1,...,n;$ if and only if $(v_i, v_k) \in E$.

In other words, given p interval graphs on the same vertex set and possibly different edge sets $G_j=(V,E_j), j=1,...,p$; 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 $j=1,...,p;\ i,k=1,...,n;$ 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 thesis we deal only with simple multi-interval graphs; that is, points will have equal length intervals assigned to them on the same real line. Figure 3.5 shows two 2-interval graphs. In Figure 3.5.a, the intervals' sizes on the real lines (1) are not equal, nor are the intervals' sizes on the real line (2) equal. Figure 3.5.b shows a simple 2-interval graph where the intervals assigned for vertices on real line (1) are equal, and where the intervals assigned for vertices on real line (2) are equal.

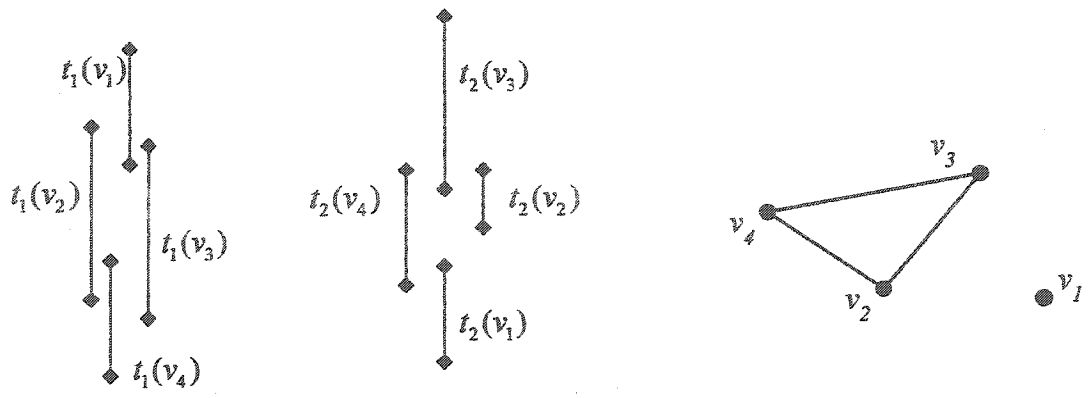


Figure 3.5.a: Two sets of real intervals of different length and their equivalent 2-interval graph.

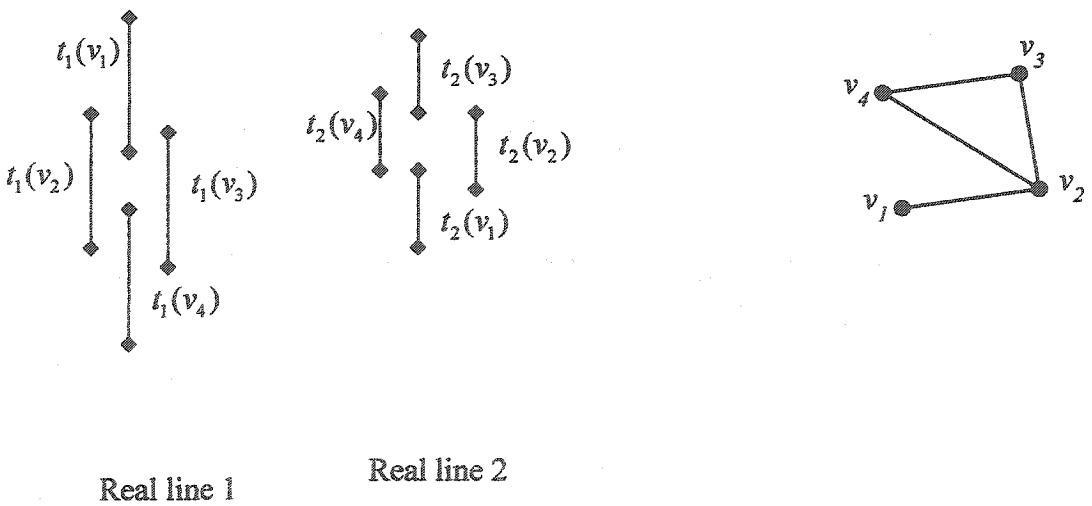


Figure 3.5.b: Two sets of real intervals with the same interval length on each real line and the equivalent simple 2-interval graph.

As shown in Figures 3.5.a and 3.5.b, the intervals on both real lines must overlap in order for an edge on the graph to exist.

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 3.1; then, for a pair of parallel straight lines we get the following corollary:

Corollary 3.1: 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 3.1, 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 $\max_{i \in S}(b_i) - \min_{i \in S}(b_i) \leq I$. 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. ■

Consider a 2χ parallel-sided polygon, $\chi \geq 2$, with given side lengths and inclination angles from the X-axis. Using definition 3.2, one can transform the covering by polygon to a maximum clique problem in a simple *p*-interval graph. Note that for a parallelogram, shape χ equals two; for a hexagon, χ equals three.

Lemma 3.1*:* Consider a polygonal shape of K parallel sides, where K is an even number. Let the graphs $G_j = (V, E_j),\ j = 1,...,\frac{K}{2}$ 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 $\frac{K}{2}$ graphs with the same vertex set and different edge sets $G_j = (V, E_j),\ j = 1,...,\frac{K}{2}$. The vertices in G that are adjacent to each other are those with an edge on all $\frac{K}{2}$ 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 $G = (V, E)$. ■

The adjacency matrix $A=[\alpha_{ij}]$ is a $|V|\text{x}|V|$ 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, $\alpha_{ii}=1$. The complementary graph $\overline{G}=(V,\overline{E})$ of 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. We denote the complementary adjacency matrix by $\overline{A}=\left[\overline{\alpha_{ij}}\right]$, in which $\overline{\alpha_{ij}}=0$ iff $\alpha_{ij}=1$ in A.

Figure 3.6.a shows a rectangular shape and two interval sets on the Y- and X-axes. The difference between the intercepts of the rectangle's parallel sides on the Y-axis and X-axis are given as I_1 and I_2, respectively. Therefore, the size of intervals assigned for each point on the Y-axis equals I_1, while the size of those assigned for each point on the X-axis equals I_2. Each point is assigned two intervals, one for the X-axis and another for the Y-axis. Let b_{ix} be the X-intercept of the line parallel to the Y-axis, passing through point i, and let b_{iy} be the Y-intercept of the line parallel to the X-axis, passing through point i. The intervals' start and end points on the X-axis and Y-axis can be given as $t(a_{ix})=[b_{ix},b_{ix}+I_2]$ and $t(a_{iy})=[b_{iy},b_{iy}+I_1]$ for $i=1,...,n$, respectively.

The points are shown as dots, while their intervals are shown with diamond-shaped endpoints. The interval start is the points projection on the X and Y axis, while the endpoints of the intervals are shown as $t(a_{ix})_e$ and $t(a_{iy})_e$ $i=1,...,4$, for the X and Y axis, respectively. Figure 3.6.b shows the equivalent graph from the intervals on Y-axis, while Figure 3.6.c shows the equivalent graph from the intervals on X-axis. The two-

interval graph and the adjacency matrix, A_{rec}, for the rectangular shapes are shown in Figure 3.6.d.

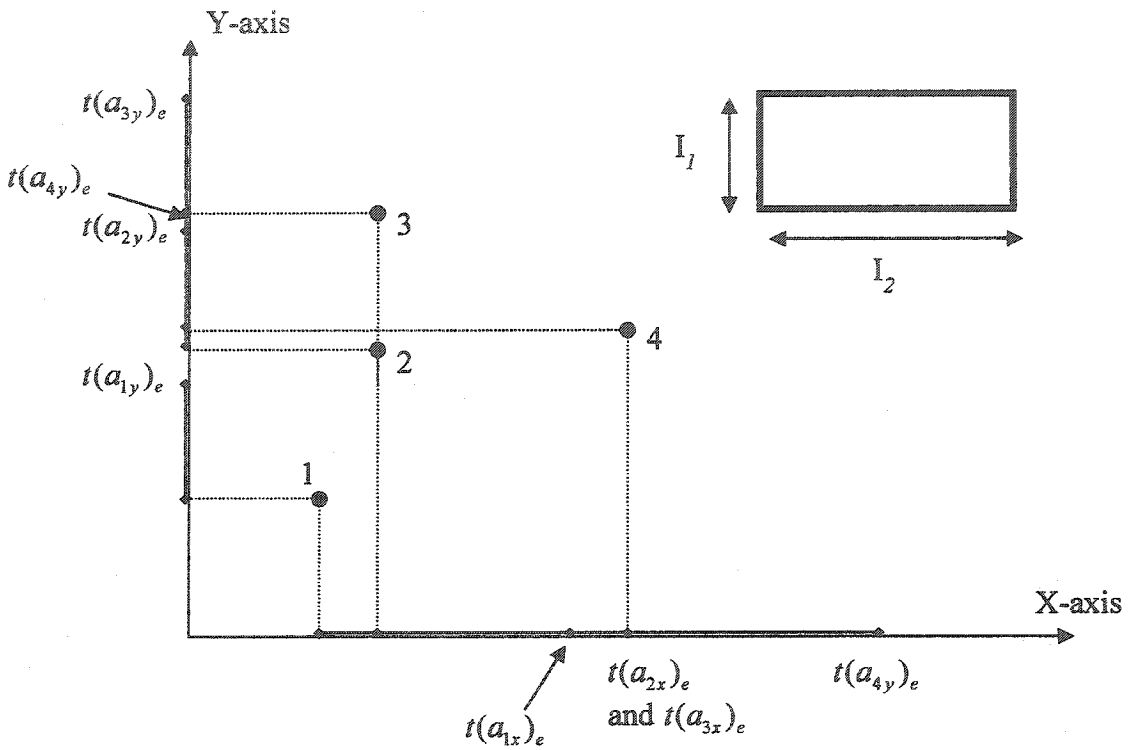


Figure 3.6.a: A rectangle with the axis-intercepts of its sides and points intervals.

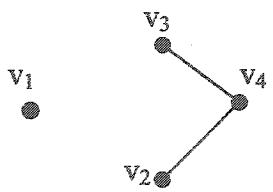


Figure3.6.b: The equivalent graph for intervals on Y-axis.

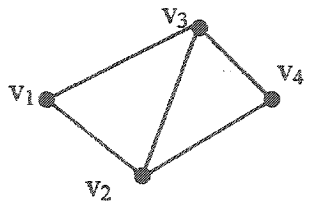


Figure3.6.c: The equivalent graph for intervals on X-axis.

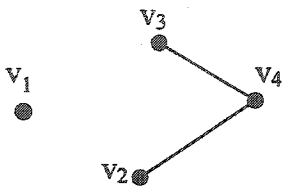


Figure3.6.d: The equivalent two-interval graph.

The adjacency matrix associated with Figure 3.6.d is $A_{rec} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$.

A simple algorithm can generate the adjacency matrix for the equivalent 2-interval graph. A MATLAB program that finds the adjacency matrix from the XY-intercepts, b_x and b_y, respectively is shown in Figure 3.7.

In the following procedure, the number of computations to find the adjacency matrix is n^2. The program in Figure 3.7 can be easily modified to handle parallel-sided polygonal shapes with any number of sides.

```
i = 1;
while i ≤ n
  for j = i : n;
  if abs(b_iy − b_jy) ≤ I_1
  if abs(b_ix − b_jx) ≤ I_2
     α_ij = 1;
     α_ji = 1;
  else
     α_ij = 0;
     α_ji = 0;
  end
end
end
i = i + 1;
end
```

Figure 3.7: A MATLAB program to find the adjacency matrix for a rectangular shape.

3.4 A Guided Search Algorithm for the Maximum Clique Problem

3.4.1 Introduction

In previous section, we concluded that finding the maximum covering by PSP is equivalent to solving the maximum clique problem on its equivalent graph. The maximum clique problem (MCP) is a known NP-hard problem, with a wide range of applications in a variety of fields (Pardalos and Xue (1994)). It has been also used in facility location by Aneja et al. (1988) to solve the *m*-center problem as covering nodes by the minimum number of cliques. However, the work of Aneja et al. was limited to

rectilinear distance measurement, and for facilities with the same covering measure - diamond shapes of equal size.

Many researchers introduced efficient exact algorithms with a modest time requirement for the MCP (Carraghan and Pardalos (1990), Pardalos and Rodgers (1992), Babel (1994)). Others used heuristic techniques to find a good solution (close to optimal) for the MCP (Soriano and Gendreau (1996), Battiti and Protasi (2001), Friden et al. (1990), Jagota (1995), Jagota and Sanchis (2001) and Balas and Niehaus (1998)). Soriano and Gendreau (1996) used a Tabu search method (TS) for MCP. They showed that TS for the MCP could compete with other solution techniques. Friden et al. (1990) proposed an exhaustive search algorithm, using some tabu search protocols, for the maximum independent set problem. Since the maximum independent set on a graph $\overline{G}$ is equal to the maximum clique on graph G, then the Friden et al. (1990) method can be used for the MCP as well. A complete survey of the maximum clique problem can be found in Pardalos and Xue (1994).

The natural question that arises in any "TS based heuristic" or exact algorithm for the MCP concerns the choice of the initial solution set, or the choice of the node (vertex) at which any enumerative method should start. To answer this question, we propose, in this section, a guided search algorithm for the maximum clique problem – or, briefly, the GSAMCP. We propose selection criteria by which a vertex $v \in V$ is chosen as a candidate for any search technique. Therefore, the GSAMCP can be used in conjunction with other heuristic search techniques like TS, or implicit enumeration algorithms like branch and bound. In addition, it guides the search by an effective bounding technique, accelerating

the overall search process. Tabu search heuristic and branch and bound algorithms are described in previously mentioned references. We explain the basic features of the GSAMCP in Section 3.4.2; then, in Section 3.4.3, we introduce the GSAMCP algorithm and discuss the main issues in the GSAMCP. A small example and computational experience with randomly generated graphs will be shown in Section 3.4.4.

3.4.2- Guided Search Algorithm for the Maximum Clique Problem:

The Basics

Given an arbitrary graph $G=(V,E)$ with a set of vertices V, $|V|=n$. A vertex v_j is said to be in the neighborhood of vertex v_i if the vertex v_j is adjacent to it; that is, v_j belongs to the neighborhood of v_i if and only if $(v_i,v_j)\in E$. Let $N(v_i)\subseteq V$ be the set of neighboring vertices to vertex $v_i\in V$, and let the degree of a vertex v_i, $\Delta(v_i)$, be the number of vertices in the neighborhood of vertex v_i; then clearly, $\Delta(v_i)=|N(v_i)|$. Further, let the vertex index be assigned according to the vertex degree, in ascending order; that is, $\Delta(v_i)\geq\Delta(v_j)$ for $i>j, i=2,...,n, j=1,...,n-1$.

In addition, let $\omega(k)$ be the number of vertices of degree k, and $\omega(\geq k)$ be the number of vertices of degree greater than, or equal to, k. Denote the size of the maximal clique that includes vertex v_i as $|MC(v_i)|$, and the size of the maximum clique of the graph G as $|MC^*|$.

To explain the GSAMCP, first we state the following self-evident facts, without a proof:

Fact 1: In any graph $G=(V,E)$, $|MC^*| \le \omega(\ge |MC^*|)$.

Fact 2: Given an undirected, un-weighted arbitrary graph $G=(V,E)$, a non-zero positive integer ζ, and a set of vertices $V'' = \left\{ V'' \subset V \mid \underset{v_i \in V''}{\Delta}(v_i) \le \zeta \right\}$, then no $v_i \in V''$ belongs to any clique of size greater than ζ.

Fact 2 is similar to rules 4 and 5 in Pardalos and Rodgers (1992). Fact 1 provides an easy way to find, and update, an upper bound on the graph's maximum clique. This is stated in the next lemma.

Lemma 1: *An upper bound on the size of the graph's maximum clique,* $|MC^*|$, *can be found according to the following simple rule:* $UB = \max\{i \mid \omega(\ge i) \ge i\}$ *where i is a non-negative integer greater than zero.*

Proof: Evident from Facts 1-2.

The GSAMCP algorithm initially chooses a vertex v_i where $\Delta(v_i) = UB$. An exhaustive search, using any well known method, is then performed in $N(v_i)$, returning $MC(v_i)$. If $|MC(v_i)|=UB$, then an optimal solution is found, and the search process stops. However, when the maximal clique found is smaller than *UB*, the GSAMCP algorithm updates *UB* and selects another vertex with degree as large as *UB* according to a given set of rules.

In general, all previous TS algorithms for the MCP deal mainly with the neighborhood search of a selected vertex. Vertices were selected according to their degree – a well known greedy rule. The motivation behind the greedy selection rule is the preassumed

increased probability of having the graph maximum clique in the neighborhood of the largest degree vertices. On the other hand, Pardalos' and Rodgers' (1992) branching rule ("Rule 6") was to select the vertex of smallest degree – which activates certain bounding rules and yields an overall tree reduction. Both arguments (greedy selection versus the Pardalos and Rodgers selection rule) have some merit to them. We realize, however, that in applying an exhaustive local search – or any other enumerative algorithm – it is faster to find $MC(v_i)$ in a smaller neighborhood than in a larger one. It is also more probable, however, that the graph maximum clique, MC^*, will lie in the neighborhood of larger degree vertices. Therefore, we devise the GSAMCP as a compromise between these two selection criteria (greedy selection and Pardalos and Rodgers rules), guiding the search to where it is more probable to have MC^*, while keeping the exhaustive search burden reasonable.

3.4.3 Guided search algorithm for the maximum clique problem "GSAMCP"

The major features of the GSAMCP can be summarized as follows:

1- The GSAMCP algorithm starts by finding an upper bound for MC^*, which is updated during the search process.

2- The GSAMCP algorithm guides the search to vertices with degree equal to, or slightly larger than, the maximum clique upper bound.

First, we state the GSAMCP algorithm and follow with an explanation. We finish this section with a small example that illustrates and clarifies the main steps in the GSAMCP algorithm. For the GSAMCP algorithm the following indicators will be used

Ng = Number of vertices with degree greater than upper bound

Ne = Number of vertices with degree equal to the upper bound

$NUB = Ne + Ng$

$\Delta^{'}(v_i) =$ an indicator of vertices degree $i = 1,...,n.$

Algorithm GSAMCP

1- *Initialize Lower bound (LB) and Upper Bound (UB),*

$LB \leftarrow 0, UB \leftarrow 2; \Delta^{'}(v_i) \leftarrow \Delta(v_i)\ i = 1,...,n; sol = \varnothing$

2- *While stopping criteria not met.*

3- *If vertices are not sorted, then sort vertices according to their degree (in ascending order). Let* $v_i, i = 1,...,n$ *be the sorted vertices such that* $\Delta(v_{i+1}) \geq \Delta(v_i)\ \ i = 1,...,n-1.$

4- *Using* **Lemma 1,** *find the upper bound, UB, and the number of vertices with degree greater than UB (Ng) and the number of vertices with degree equal to UB (Ne).*

a. $UB = \max\{i \mid \omega(\geq i) \geq i\}$

b. $Ng = \omega(> UB), Ne = \omega(= UB), NUB = Ng + Ne.$

5- *If* $Ne \neq 0$, $k \leftarrow \max\{j \mid \Delta^{'}(v_j) = UB)\}$, *otherwise* $k \leftarrow \min\{j \mid \Delta^{'}(v_j) > UB)\}$

6- *Apply a neighborhood search (procedure NS), or any local search or enumerative algorithm, in the neighborhood of* v_k, *return* $|MC(v_k)|$, *the maximum clique in the neighbor of* v_k.

7- *If* $|MC(v_k)| = UB$, $sol = MC(v_k) = MC^{*}$ *Stop*

else

a. *if* $|MC(v_k)| \geq LB$ *then* $LB \leftarrow |MC(v_k)|, \Delta(v_k) = LB$, $sol = MC_k$

else

$\Delta^{'}(v_k) = LB$

8- Use Fact 2, $\forall v_j$, if $\Delta^{'}(v_j) \leq LB, j = 1,...,k$ *then*

(i) $\Delta^{'}(v_j) = LB$

(ii) $\alpha_{ij} = 0 \;\; i = 1,...,n$

9- choose next vertex to search and update upper bound

a. $NUB = NUB - 1$

b. If $NUB > UB, k = k - 1$, *go to 6 ;else*

c. If $NUB = UB$, *then* $k = \min\left\{ j \mid \Delta^{'}(v_j) \geq UB) \right\}$, *go to 6; else*

d. If $NUB < UB$, then $UB = UB - 1$; if $UB \leq LB$, stop, else

Sort, then go to step 4

Fact 3*: In step 4,* $Ng \leq UB$.

Stopping criteria

1- $LB \geq UB$

2- Maximum clique found (step 6)

Procedure *NS*, mentioned in step 6, can be summarized as follows:

Procedure NS (Neighborhood Search)

Input: *LB, k*

If $\left|N(v_k)\right| > LB$

Perform exact local search in $N(v_k)$

Return: $MC(v_k), |MC(v_k)|$

Otherwise, go to step 8 in GSAMCP

Steps 1, 3 and 6 in the GSAMCP algorithm are straightforward. In step 4.a, Lemma1 is used to find an upper bound (*UB*) on the maximum clique size. The indicators Ng, Ne, of vertices with degree greater than upper bound and of vertices with degree equal to upper bound, respectively, are used to find the number of vertices greater and equal to the upper bound (*NUB*). These three indicators divide the vertices into three sets: the set of vertices with degree greater than upper bound, the set of vertices with degree less than upper bound, and the set of vertices with degree equal to upper bound. The search will always occur in the set of vertices of degree equal to *UB*, or degree just larger than *UB*.

In step 5, the vertex at which one performs a neighborhood search is chosen according to the following rules:

1- If there are no vertices of degree equal to *UB*, the search starts at the vertex of degree just greater than *UB*.

2- If there is one or more vertices of degree equal to *UB*, the search starts at the vertex of the largest index in the sorted list.

If an optimal solution is not found in step 7, the lower bound (*LB*) is updated and the degree of the visited vertex is forced to change according to step 7.a. Changing the vertex degree, along with the other vertex degree changes at step 8, has the following benefits:

1- It will guarantee that a visited vertex will not be searched again.

2- It will reduce the computational time for reordering vertices, (step 9.d).

Using Fact 2, the visited vertex, as well as all vertices of degree lower than the lower bound, are pruned from search space in step 8.

Step 9.b indicates that there are vertices of degree equal to *UB*, hence the index k is reduced by just one, and the search starts at the next vertex of degree equal to the upper bound. However, if $NUB = UB$ as in step 9.c, the index k is updated according to a different rule. The search index k will go to the vertex with degree equal to *UB*, if any exists, or to vertex of degree just greater than upper bound.

Finally, when $NUB < UB$, this situation indicates that the current upper bound is not tight enough, and it is updated in step 9.d. The algorithm stops if stopping criteria are met, (step 9.d); otherwise, a new upper bound indicator is found. Since the vertices degrees have been updated in steps 7.a and 8, the new upper bound will guide the search to new vertices. Care must be taken to track the changes in the original vertex indexing. The computational cost of reordering vertices according to their new degree can be justified by the following arguments:

1- Updating vertex degrees, after pruning vertices with degree equal to *LB*, would guide the GSAMCP into a new search space.

2- An updated, most probably smaller, *UB* may be found, which, when compared with *LB*, may stop further search.

3- It will reduce the adjacency matrix size, thus reducing the memory requirements.

In fact, we can perform *UB* updates more often than proposed in the GSAMCP algorithm. To avoid adding a lot of computational cost by performing *UB* updates, we may limit *UB* updates to

1- Every $\lfloor |V|/\Psi \rfloor$ or $\lfloor Ne/\Psi \rfloor$ steps, where $\lfloor |V|/\Psi \rfloor$ or $\lfloor Ne/\Psi \rfloor$ is the largest integer less than or equal to $|V|/\Psi$ or Ne/Ψ, respectively, and Ψ is a positive integer number that can be determined experimentally.

2- Every ϑ upper bound changes, where ϑ is a positive integer number.

3.4.4 Example and Computational Results

The next two examples show the mechanism of choosing the candidate vertex for the *NS* procedure. For simplicity, we will not perform a sorting in the first example. In addition, to perform enough steps, we will assume that stopping criteria are not met in the first example. In the second example the adjacency matrix will be given.

Example 1:

Consider a graph with the vertex degrees and indices as shown in Table 3.2. In step 1, we show all the GSAMCP parameters. Only those parameters of interest will be shown in further steps.

Table 3.2: Vertices indices and degrees for the *GSAMCP* example.

Vertex Index	Vertex Degree
1	1
2	4
3	4
4	4
5	4
6	5
7	5
8	6
9	6
10	8

1- Find the parameters ($Ns = \omega(< UB)$), or simply $Ns = n - NUB$

UB	*Ng*	*Ne*	*Ns*	*NUB*	*Index*
5	3	2	5	5	7

Perform a full search in $N(v_7)$, and return $|MC(v_7)|$.

2- Assume optimal solution not found – and no vertex degree update – except for visited vertex. We get $NUB < UB$; hence, step 9.d is activated.

UB	*Ng*	*Ne*	*Ns*	*NUB*	*Index*
4	4	4	1	8	5

3- Since $NUB > UB$, we reduce *Index* by one (*Index* = 4). New *NUB* = 7.

4- Same as (4) above, new *NUB* = 6, *Index* = 3.

5- *NUB* = 5 , *Index* = 2 .

6- *NUB* = 4, *Index* = 6 (step 9.c)

7- *NUB* = 3, *UB* updated, cause of step 9.d.

Example 2:

Given the adjacency matrix for a graph $G_{eg} = (V, E)$ as A_{eg}, the graph vertices degree is also given in as shown below:

$$A_{eg} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$,vertex indices and degrees $$\begin{bmatrix} v_1 & 3 \\ v_2 & 3 \\ v_3 & 4 \\ v_4 & 4 \\ v_5 & 5 \\ v_6 & 5 \\ v_7 & 5 \\ v_8 & 6 \\ v_9 & 6 \\ v_{10} & 6 \\ v_{11} & 6 \\ v_{12} & 7 \end{bmatrix}$$

1- Find the parameters ($Ns = \omega(< UB)$), or simply $Ns = n$ - NUB

UB	*Ng*	*Ne*	*Ns*	*NUB*	*Index*
5	5	3	4	8	7

Perform a full search in $N(v_7)$, and return $|MC(v_7)|$.

We can easily find that $MC(v_7) = \{v_6, v_7, v_{10}\}$, and $|MC(v_7)| = 3$.

Update LB, and set $\alpha_{ij} = 0, i = 1,...,12; j = 1,2,7;$ and set $\Delta'(v_k) = LB, k = 1,2,7.$

2- $NUB = NUB - 1$, $NUB > UB$; hence, step 9.b is activated and $k = k$-1

$(NUB = 7, k = 6)$

Perform a full search in $N(v_6)$, and return $|MC(v_6)|$.

We can easily find that $|MC(v_6)| = 2 < LB.$

Set $\alpha_{ij} = 0, i = 1,...,12; j = 6,$ and set $\Delta'(v_6) = LB.$

2- $NUB = NUB - 1$, $NUB > UB$; hence, step 9.b is activated and $k = k$-1

($NUB = 6$, $k = 5$).

Perform a full search in $N(v_5)$, and return $|MC(v_5)|$.

We can easily find that $MC(v_5) = \{v_5, v_8, v_9\}$, and $|MC(v_5)| = 3.$

Set $\alpha_{ij} = 0, i = 1,...,12; j = 5,$ and set $\Delta'(v_5) = LB.$

3- $NUB = NUB - 1, NUB = UB$; hence step 9.c is activated, $k = 8$

Perform a full search in $N(v_8)$, and return $|MC(v_8)|$.

We can easily find that $MC(v_8) = \{v_8, v_9, v_{12}\}$, and $|MC(v_8)| = 3.$

Set $\alpha_{ij} = 0, i = 1,...,12; j = 8,$ and set $\Delta'(v_8) = LB.$

4- $NUB = NUB - 1, NUB < UB$, then $UB = UB - 1 = 4 > LB$, then update

$\Delta'(v_i)$ $i = 1,...,12.$

The new updated degrees can be given as follows

$$\begin{bmatrix} v_1 & 3 \\ v_2 & 3 \\ v_3 & 4 \\ v_4 & 4 \\ v_5 & 3 \\ v_6 & 3 \\ v_7 & 3 \\ v_8 & 3 \\ v_9 & 4 \\ v_{10} & 3 \\ v_{11} & 2 \\ v_{12} & 3 \end{bmatrix} \text{ and after sorting } \begin{bmatrix} v_{11} & 2 \\ v_1 & 3 \\ v_2 & 3 \\ v_5 & 3 \\ v_6 & 3 \\ v_7 & 3 \\ v_8 & 3 \\ v_8 & 3 \\ v_{10} & 3 \\ v_{12} & 3 \\ v_3 & 4 \\ v_4 & 4 \end{bmatrix}$$

4- Find a new UB (step 4.a)
$UB = 3 = LB$, Stop search optimal solution found and $MC(v_7) \equiv MC^*$

Computational Experience

A comparison between GSAMCP algorithm and the enumerative search based on the greedy selection rule (GS) is shown in this example. We considered randomly generated graphs (from uniform distribution) with different sizes and densities. All programming and computations were done using MATLAB, on the same computer platform and under the same conditions. The neighborhood search procedure (*NS*) used follows the Friden et al. (1990) and Gendreau et al. (1993) approaches (utilizing Tabu Search). In GSAMCP, we forced UB to be updated every $\lfloor |V|/20 \rfloor$ steps – on top of the normal GSAMCP algorithm update. Searched vertex pruning was done in GS as well.

The computational results are shown in Table 3.3. The first two columns show the number of vertices and edges in the graph. Column 3 indicates the number of times the

UB has been forced to be updated. This will also indicate, indirectly, the number of vertices visited before optimal solution, MC^*, is found. The initial values of *Ng*, *Ne*, and *UB* values are in the fourth, fifth, and sixth columns respectively.

Table 3.3 shows that in all cases the search performed following the GSAMCP selection rules is computationally more efficient than the search done by following the GS rules. Except in a few cases, the *UB*-forced update was not activated more than twice, reflecting the efficiency of GSAMCP in restricting the search to where it does matter. Furthermore, we noticed that the running time was influenced by both the size of the graph maximum clique and the size of the graph.

We used the same *NS* procedure to perform the neighborhood search in both GSAMCP and in the GS selection rules, and the only difference was where to perform the search (vertex selection). Therefore, using another neighborhood search procedure or improving the current on (*NS* procedure) would be reflected in the running time of GSAMCP algorithm and GS selection rule.

Table 3.3: Computational results of comparison between the *GSAMCP* algorithm and *GS*.

$\|V\|$	$\|E\|$	# Ψ activated	*Ng*	*Ne*	*UB*	*MC**	*CPU time* (sec)	
							GS	*GSAMCP*
50	162	2	6	8	6	5	0.16	0.05
50	550	2	12	1	13	9	2.69	0.17
50	964	2	17	5	22	15	16.58	0.44
50	1598	3	28	4	31	24	102	2.58
50	1890	4	31	4	35	27	207	5.11
100	446	2	6	9	7	6	0.44	0.16
100	1380	2	11	7	18	11	7.8	0.44
100	3328	6	36	4	36	19	217	8.45
100	5898	3	53	4	56	35	1065.3	55.24
150	6796	5	50	2	51	24	887.7	35.21
150	10566	5	70	6	72	39	3623	83.43
200	10682	2	60	5	60	31	2272	43.17

CHAPTER FOUR

GENETIC ALGORITHM SOLUTION APPROACH FOR PLANAR MAXIMAL COVERING LOCATION PROBLEM UNDER DIFFERENT BLOCK NORMS

4.1 Introduction

In Chapter Two we introduced a new MIP formulation for the planar maximum covering location problem (PMCLP) under parallelogram norms and the one-infinity norm. The MIP formulation, however, is limited to facilities of the same degree block norm – that is, block norms of $r = 2$ in the case of rectilinear, Tchebycheff, or parallelogram norms, and block norm of $r = 4$ in the case of the one-infinity norm. Handling facilities of different distance type coverage by MIP is not possible without an added computational cost. In addition, our experimentations shows that MIP is efficient only for small problems (fifty demand points and two shape), and an alternative method is needed to handle larger problems. The exact algorithms of Chapter Three can handle one shape at a time and, for locating g facilities, the algorithm may be used with other heuristic techniques such as TABU search method.

In this chapter, we attempt to overcome the formulation and exact solution restrictions delineated in Chapters Two and Three. We propose to accomplish that using a meta-heuristic technique, mainly the genetic algorithm approach (GA), a meta-heuristic technique widely used to overcome computationally difficult problems.

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), unconstrained binary quadratic problems (Katayama et al., 2000), and maximum clique problem (Balas and Niehaus, 1998).

We start this chapter by reviewing in Section 4.2 the basic elements and techniques of GA. The GA approach we discuss makes use of the idea of transforming covering by parallel-side polygons problem into maximum clique problems, introduced in Chapter Three. This transformation is necessary in order to allow handling of the many facilities with different block norms. Section 4.3 shows the extension of the ideas presented in Section 3.3, formulating our problem in a format suitable for GA. In Section 4.4, we introduce our approach in applying GA to PMCLP under different block norms. We provide computational examples also in Section 4.4.1.

4.2 Genetic Algorithms: Background and Principles

A GA is a random search heuristic that mimics the natural evolution of species. It emanated from the behavior 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, that 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 time or cycles, or until convergence is reached.

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

1- Generate initial population (as strings).

2- Evaluate individual string fitness.

3- Establish a basis for deciding which parents to mate and, thus, generate offspring.

4- Establish a framework for deciding which offspring and parents should constitute the next generation.

5- Develop a method to imitate natural evolution.

We will now discuss the main steps in a GA. The steps are named population generation and representation, selection, crossover, and mutation.

4.2.1 Population Generation

This step is usually known as initialization or seeding. In initialization, we want to determine the initial population (Generation [0]) before starting the GA. A population generated from a uniform distribution is frequently used in the literature, and is equivalent to sampling randomly from the space of possible solutions.

Another alternative is to start with a possibly good solution obtained from a problem characteristic or from another heuristic technique. However, this approach may bias the random nature of GA and cause a premature convergence.

4.2.1.1 Representation

Representation refers to the form used to represent individuals and possible solutions in the GA. Different representations would affect the population encoding. The most popular way to encode a population is binary representation, in which each individual in the population is represented as a string of zeros and ones. The choice of different representations affects the performance and the techniques used in other GA operations. In general, we should use a representation that is easy to handle in subsequent GA operations. In this thesis we use binary strings in our GA implementation and, hence, we will limit our discussion to binary strings.

4.2.2 Selection

Once an initial generation has been initiated, the question is which part of this population has the characteristics deemed good enough to transfer to future generations.

Apparently good characteristics are favored and bad characteristics are penalized. This step should answer the following questions:

1- How can we judge which characteristics are favored?

2- Which parent string, as well as how many parents, is selected from the given population?

3- What factors determine which strings mate with each other?

This process of selecting parents to generate offspring has been done in many different ways. We will briefly discuss a few of them.

4.2.2.1 Fitness Evaluation

Also known as the evaluation function, this provides a way to judge the quality of individuals in the given population. While individual fitness is important, it should be viewed as a part of total population fitness, allowing the random selection process as discussed in Section 4.2.2.2.

String evaluation can be performed from the optimization problem objective function, while maintaining problem constraints satisfied. In cases where constraints are violated, different measures can be taken:

1- Rejecting infeasible strings.

2- Repairing infeasible strings.

3- Penalizing infeasible strings.

In a GA, evaluation of strings is considered the greatest computational burden; string representation usually plays a role on the data structure used and the speed of evaluation.

4.2.2.2 Parents Selection

The selection of strings as parents for offspring is a fitness-based operation. Different selection strategies have been proposed by different researchers; some of the most common selection strategies are listed below:

1- Roulette selection

This technique takes its name from the analogy of the roulette gambling game. The individuals in the population are mapped to segments of a line, where each individual segment is equal in size to its fitness. A random number, from a uniform distribution, is generated. The individual whose segment spans this random number is then selected as a parent for mating.

2- Rank-based selection

In this method, the population is sorted according to their objective values. The fitness assigned to each individual depends only on its position in the overall rank. Position fitness can be found according to a linear or a non-linear function – linear or non-linear ranking, respectively – of position and number of individuals in the population. Individuals are then selected randomly according to their normalized fitness (individual fitness normalized by the total fitness of the population).

3- Truncation selection

In this method, a threshold is set, and individuals with fitness value below this threshold are ignored. Individuals to be parents of offspring are chosen at random from individuals with fitness value greater than this threshold.

In addition to selection, a strategy should be developed upon which offspring will replace some or all of the old population (reinsertion). Some of these strategies are

1- Offspring replace parents (offspring produced are as many as parents).

2- Offspring randomly replace some parents (offspring produced are fewer than parents).

3- Offspring replace the worst parents (offspring produced are fewer than parents).

4- Reinsert only the best offspring (offspring produced are more than needed-i.e original population).

4.2.3 Crossover

For binary valued strings, the recombination step, where reproduction takes place, is known as crossover. Parents' genes are randomly exchanged to create offspring. This exchange can happen in different ways; one of the oldest and most commonly used is the one point crossover, also called as a simple crossover. Let the string length be $strl$, and select an integer k such that $1 \le k \le strl - 1$. Offspring are then generated by swapping all genes between positions $k+1$ and $strl$, Figure 4.1. Note that k can be selected randomly or can be set for a certain value during all the generations.

Other types of crossover can be used, such as the multi-point crossover and shuffle crossover. In multi-point crossover a number of crossover positions are generated randomly between 1 and the $strl$-1. The offspring are generated by exchanging genes between the first crossover position and the second one, then between the third and fourth and so on. The 3-point crossover operation is shown in Figure 4.2.

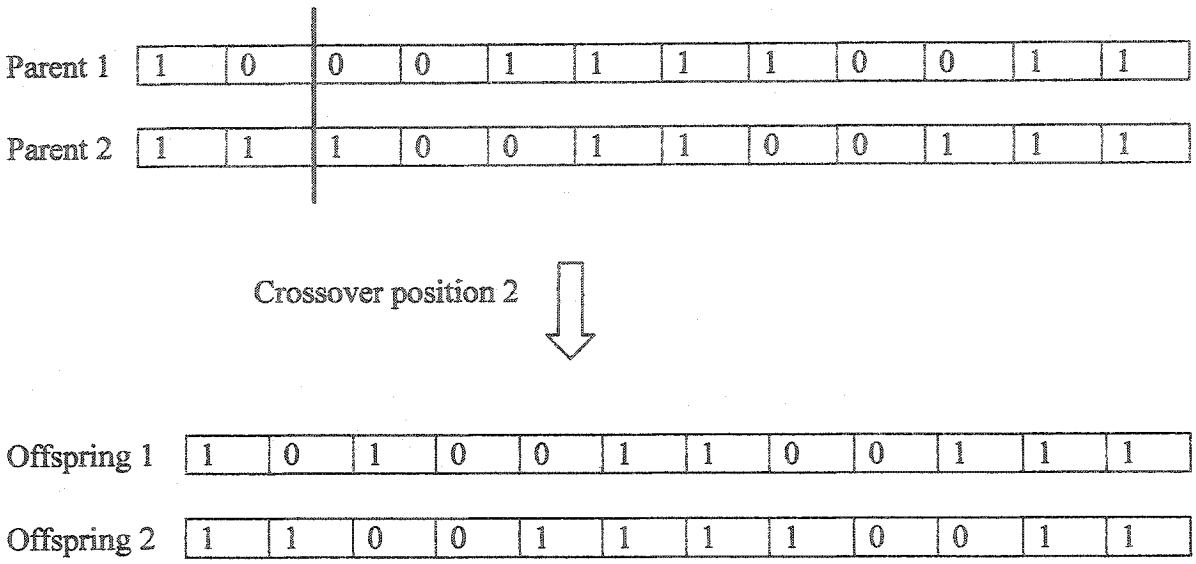


Figure 4.1: Single point crossover.

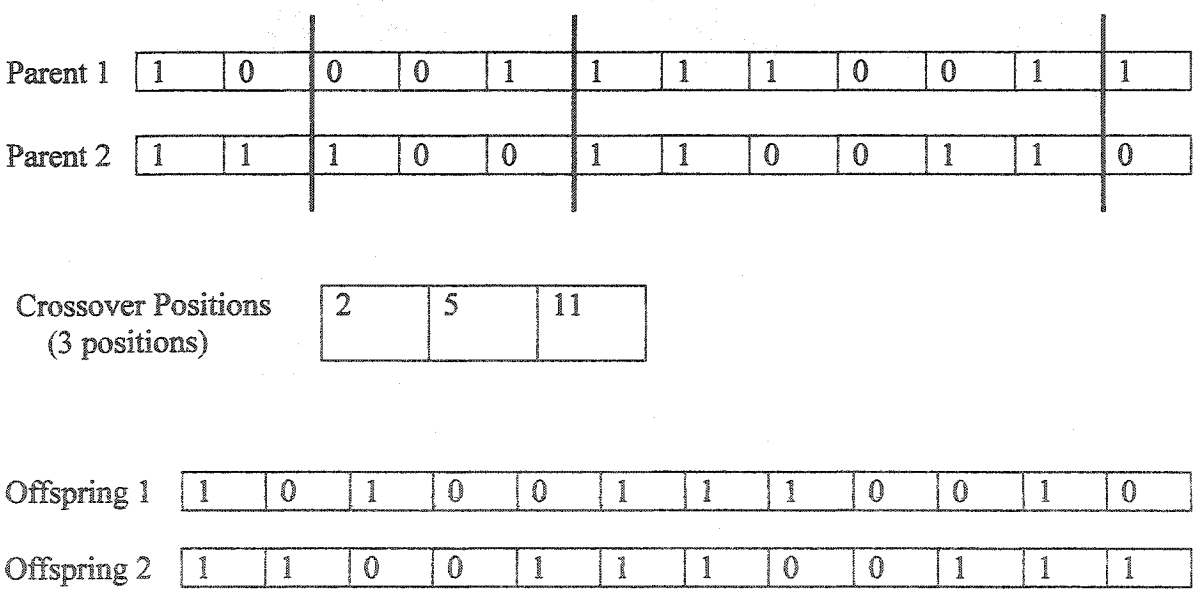


Figure 4.2: Multi-point crossover.

4.2.4 Mutation

Mutation is the changing of the value or position of one or more numbers in the string. For a binary format, string mutation can be as simple as flipping the gene from 0 to 1, Figure 4.3. Mutation plays a role in insuring that the GA does not converge prematurely to a local optimum; it usually occurs randomly, with a small probability.

Figure 4.3: Mutation for binary string.

Figure 4.4 shows the flow chart for a typical GA. For a detailed discussion on GA and its main components, refer to the classical book by Goldberg (1989) and the book edited by Reeves (1995), among many others.

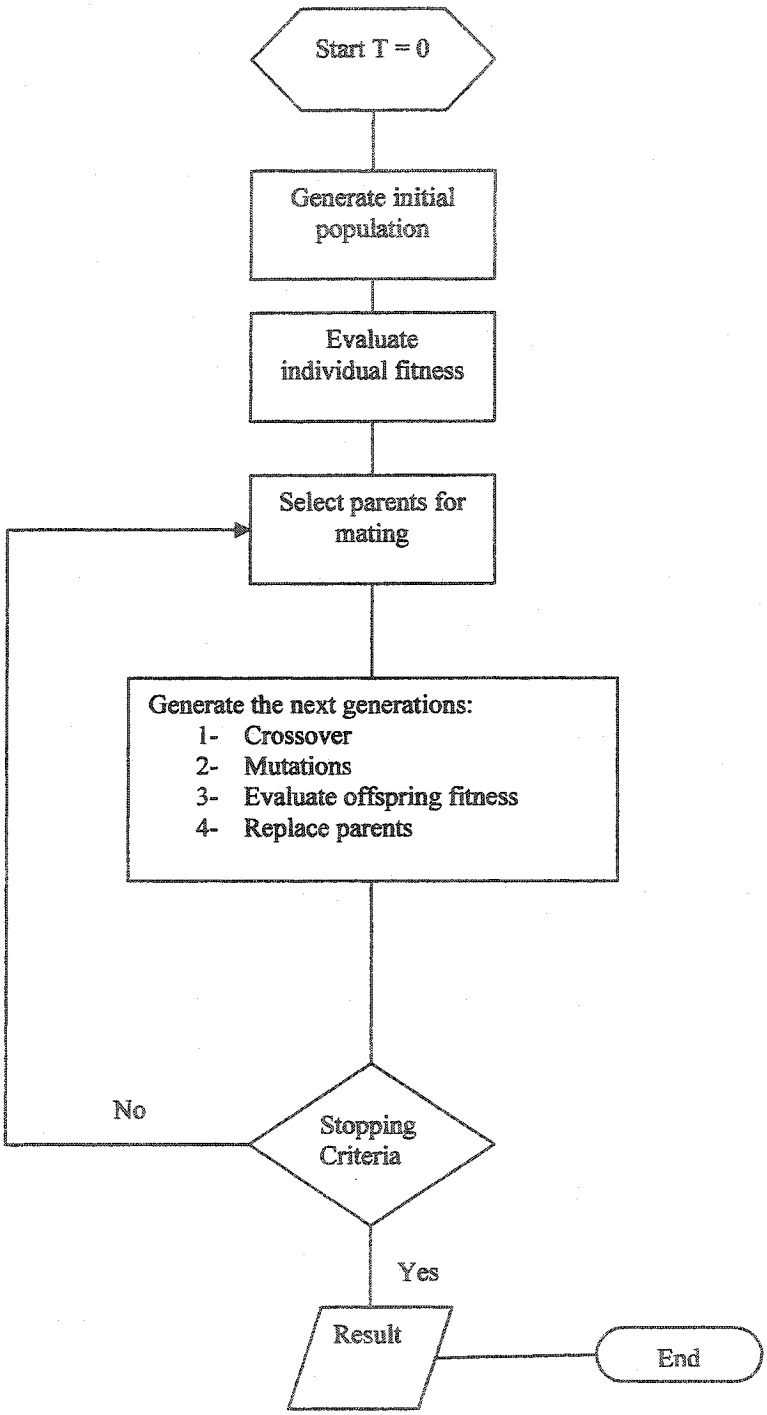


Figure 4.4: Flow diagram of GA.

4.3 Maximal Covering by Many Shapes as a Graph Formulation

In Chapter Three, Section 3.3, we developed the equivalence between the maximum covering by a single shape, the facility, and the maximum clique problem. In this section, we develop covering by different shapes as a graph equivalent problem. We extend the classical zero-one maximum clique formulation to handle this case, then we transform it to an equivalent unconstrained quadratic binary problem (UQP). This transformation plays an important role in the GA implementation.

In the case where the given g polygonal shapes are identical, each shape will generate a graph such that $E_k = E_j \ \forall k, j = 1,...,g$, 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 $V_1, V_2, ..., V_g$; such that $\sum_{k=1}^{g} |V_k| \leq |V|$ 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 $G_j(V, E_j),\ j = 1,...,g$. 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 $V_1, V_2, ..., V_g$, such that $\sum_{k=1}^{g} |V_k| \leq |V|$ is maximized, and for $v_i, v_j \in V_k \ (v_i, v_j) \in E_k$; $i, j = 1,...,n, k = 1,...,g$.

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 $G_j(V, E_j)$, $j = 1,...,g$. 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^* point, it is evident that the optimal solution of *MCPSP* (*OPT_MCPSP*) follows that $OPT\_MCPSP \le \sum_{k=1}^{g} |MC_k^*|$.

4.3.1 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 for 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

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

$$(P7) \quad \text{Maximize} \sum_{i=1}^{n} x_i \tag{4.1}$$

Subject to:

$$x_i + x_j \leq 1 \quad \forall (i,j) \in \overline{E}, \tag{4.2}$$

$$x_i \in \{0,1\} \; i = 1,...,n. \tag{4.3}$$

In the above zero-one formulation, the objective function (4.1) is to maximize the number of vertices chosen. Constraints (4.2) ensure that only adjacent points are chosen. Constraint (4.3) 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 $V_1, V_2, ..., V_g$, such that $\sum_{k=1}^{g} |V_k| \leq |V|$ is maximized, and for $v_i, v_j \in V_k \; (v_i, v_j) \in E_k \; ; i, j = 1,...,n, k = 1,...,g.$

For the zero-one formulation of the *MCPSP* the following notation will be used.

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

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

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

A_k = adjacency matrix for graph k, $k = 1,...,g$

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

The zero-one formulation will be:

$$(P8) \quad \text{Maximize} \sum_{k=1}^{g}\sum_{i=1}^{n} x_{ik} \tag{4.4}$$

Subject to:

$$x_{ik} + x_{jk} \le 1 \quad \forall (i,j) \in \overline{E_k}; k = 1,...,g, \tag{4.5}$$

$$\sum_{k=1}^{g} x_{ik} \le 1 \quad i = 1,...,n, \tag{4.6}$$

$$x_{ik} \in \{0,1\} \; i = 1,...,n, k = 1,...,g. \tag{4.7}$$

Constraints (4.6) ensure that a point is assigned to one shape only. Constraints (4.5) ensure only adjacent points can be chosen for any shape. This means that $x_{ik} + x_{jk} \le 1$ if $\alpha_{ijk} = 0, i = 1,...,n; \; j = 1,...,n; \; k = 1,...,g$. The objective function (4.4) maximizes the sum of points assigned to different graphs. Constraint (4.7) is the usual zero-one assignment. Constraints (4.5) and (4.6) can be transformed to a penalty function in the unconstrained UQP.

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

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

Where

$\mathrm{P}' = \sum_{k=1}^{g}\sum_{i=1}^{n}\sum_{j=i+1}^{n} x_{ik}x_{jk} \quad \forall (i,j) \in \overline{E_k}$ is a penalty function representing constraints (4.5).

$\mathrm{P}'' = \sum_{k=1}^{g-1}\sum_{j=k+1}^{g}\sum_{i=1}^{n} x_{ik}x_{ij}$ is a penalty function representing constraints (4.6).

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.

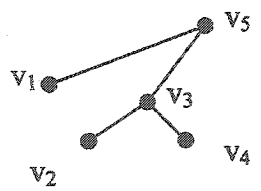
In Section 4.4 we propose a GA to solve this problem – a GA that would maintain $P^{''}$ as zero. This would maintain feasibility with respect to $P^{''}$, easing the solution process. In order to improve the GA results, a local search based on a Tabu search heuristic is performed. We also discuss again the penalty function issue.

4.3.1.1 A Small Example

The following illustrates, by means of a small hypothetical example, the penalty function idea discussed so far. The same example will also be used to illustrate the genetic algorithms of the next section.

Consider five points in R^2 and two parallel-sided polygonal shapes (PSP's), *poly1* and *poly2*. Assume that the adjacency matrices for the graphs generated from polygons *poly1* and *poly2* be A_1 and A_2, respectively. The adjacency matrices and the equivalent graphs are shown in Figure 4.5.

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$



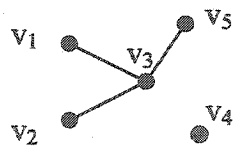


Figure 4.5: Adjacency matrices and equivalent graphs to illustrate the penalty function idea.

As can be seen from the graphs in Figure 4.5, the maximum clique in each graph is 2 and in the best solution, each shape will cover a maximum of two points ((v_1, v_5) for *poly*1 and (v_3, v_2) for *poly*2). For illustrative purposes, suppose that we initially choose vertices {2,3,4,5} for shape *poly1* and vertices {1,2,3} for shape *poly2*. Since $(v_2, v_4) \in \overline{E_1}$ *and* $(v_2, v_5) \in \overline{E_1}$ in *poly1*, this will add a penalty, $P^{'}$ of 2; also, $(v_1, v_2) \in \overline{E_2}$ in *poly2*, so an additional penalty of 1 will be added to $P^{'}$. Similarly, one can find $P^{''}$ to be equal to 2 – simply because v_2, v_3 are chosen in both graphs. Finally, note that in the optimal solution, both penalty functions $P^{'}$ and $P^{''}$ should be equal to zero.

4.4 Genetic Algorithm for Maximum Covering by Polygonal Shapes

In Section 4.2 we introduced the basics of GA. 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.

The genetic operations, initial population, selection, crossover, and mutation are as follows.

1- Initial population: This can be divided into two parts, population presentation and population assignment.

1.a 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 4.6.

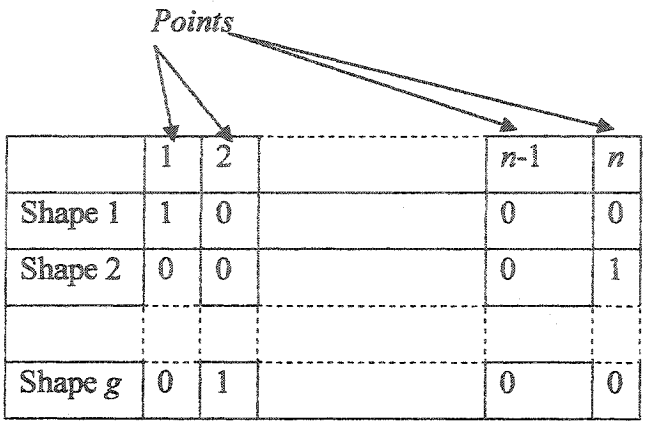


Figure 4.6: Initial population representation.

1.b: 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 4.7 shows the initial population generation procedure. A brief explanation follows.

1- Generate a population of zero matrices of size gn,
and call this STR_j, $j = 1,...,end\_of\_population$.

2- For $j = 1,...,end\_of\_population$

3- For $i = 1:n$

4- Generate g random numbers $(ra_k, k = 1,...,g)$

5- If max $(ra_k) \geq$ cut value

Set $STR_j(index\_of\_\max(ra_k), i) = 1$

end

end

end

Figure 4.7: Procedure initial population.

In step 1 of the procedure a population of parents (matrices of size gn) are 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], are 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 $c \geq 1$ 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.

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.

1- 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 adjacency matrix A_1 from Section 4.2, and 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 4.5, 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.

2- Selection 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.

6- 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 penalty function $P^{'}$ as zero. Figure 4.8 shows how $P^{"}$ would always be maintained as zero.

7- Mutation:

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.

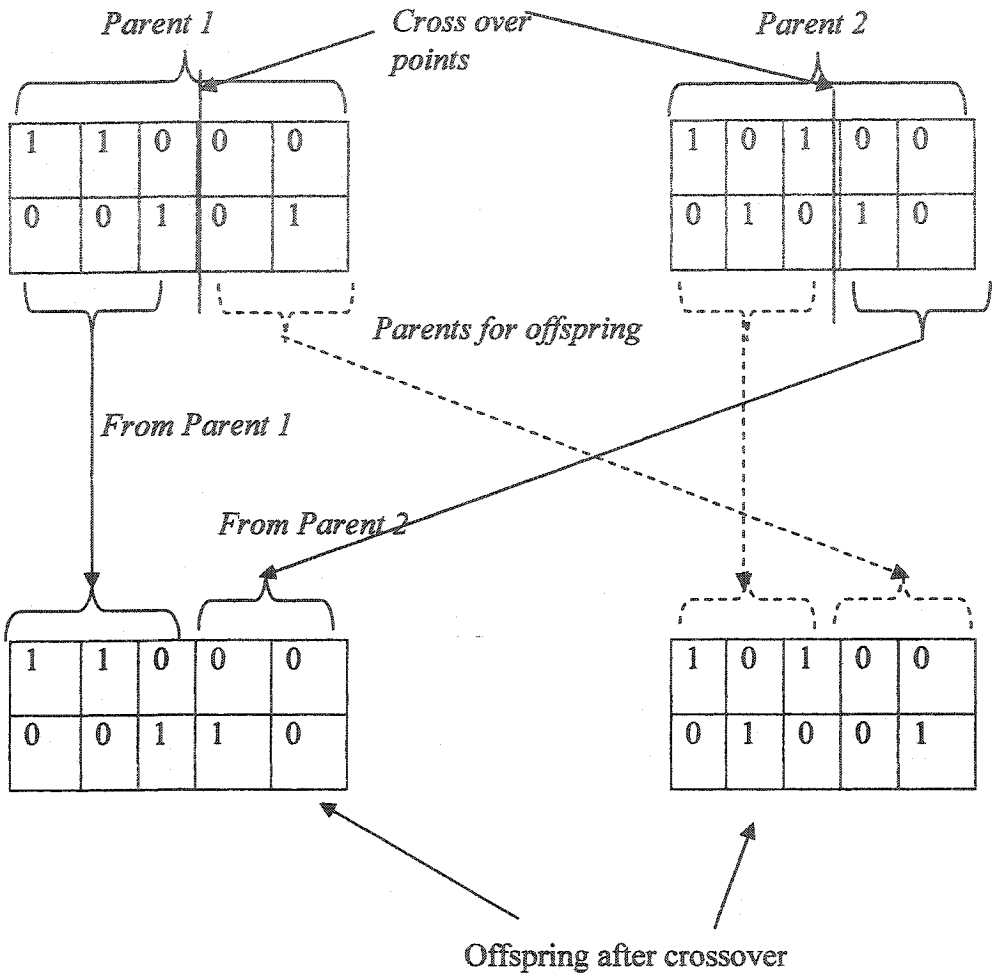


Figure 4.8: Crossover operation maintains P^{*} as zero.

8- 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 neighborhood of the points assigned to each shape. If the neighborhood of the shape string is not empty, and is not in the tabu list, a tabu search is performed and the points in the neighborhood adjacent to all points in the string are added.

4.4.1 Computational Experiment

In this section we present the results of experiments used to examine the GA algorithm. Computational tests for the GA and TS algorithms developed by the author were performed using Matlab and run on an IBM PC with an Intel Pentium III 600 MHz processor and 64MB of memory. Six different shapes were taken, as presented in Figure 4.9. The shapes are as follows:

1- A parallelogram as shown in Figure 4.9.1.

2- A diamond shape as shown in Figure 4.9.2.

3- A rectangle of sides length 3 and 2.5 as shown in Figure 4.9.3.

4- A hexagonal shape (a block norm of third degree) with a 2.5 units distance between its center and corners as shown in Figure 4.9.4.

5- A diamond shape of side length 4 as shown in Figure 4.9.5.

6- A square of side length 4 as shown in Figure 4.9.6.

Computational experiments were performed for different sets of points in different areas. Table 4.1 shows the number of points and the area size.

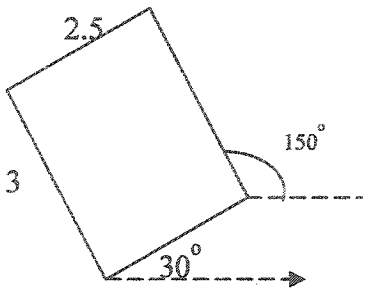


Figure 4.9.1

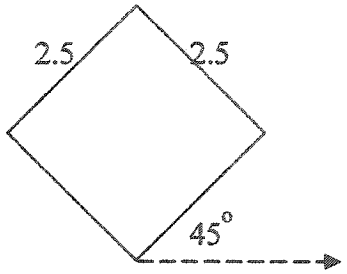


Figure 4.9.2

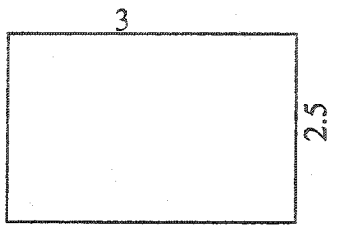


Figure 4.9.3

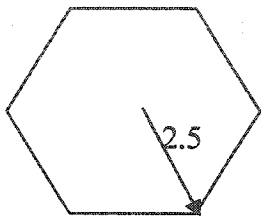


Figure 4.9.4

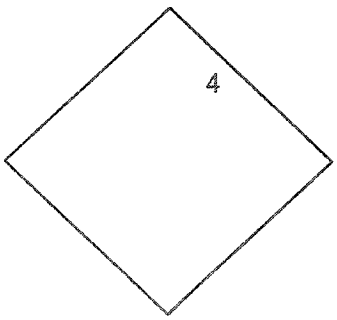


Figure 4.9.5

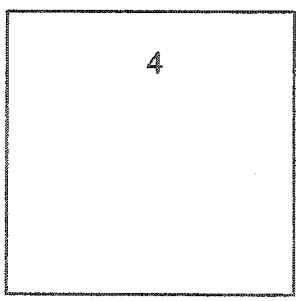


Figure 4.9.6

Figure 4.9: The different shapes under study.

Table 4.1. 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)

In order to evaluate our GA and TS results, we first compared case of two shapes and 50 points with the MIP result from Younies and Wesolowsky (2004.a). Both GA and TS gave results the same as that of MIP, for 50 points and two shapes.

The exact algorithm, based on the ideas of Friden et al. (1990) and Gendreau et al. (1993), was used thereafter for comparisons. For four graphs, we examined by solving a twenty-four different permutation ordering by the exact search algorithm. However, an

exact search was also performed independently for each shape, and an upper bound for each shape was found.

In Table 4.2 we show the computational results for both the GA and the exact search algorithms. For six shapes, we performed exact search for an arbitrary one possible ordering for comparison reasons only. The ordering {1, 2, 3, 4, 5, 6} was performed by exact search.

Table 4.2. Computational results for up to six shapes.

Shapes taken						# of	Upper	GA Result			Exact Algorithm	
1	2	3	4	5	6	Points	Bound	Best	Avg.	Time	Value	Time
.	.					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/run
.	.	.	.	.	.	100	40	33	28	128.4	31	10.98/run
.	.	.	.	.	.	150	29	27	26	499	28	26.69/run
.	.	.	.	.	.	200	36	33	30	1492	32	59.7/run

From Table 4.2 we notice the upper bound (solving each shape separately) is greater than the best solution for four shapes and 100 and 200 points. For four shapes, the GA results were lower than best result by the exact algorithm in three cases. This permutation ordering results were better than the GA result in two cases, and close to upper bound for the case of 150 points. The exact algorithm run time for six shapes is the

run time per one permutation ordering. Our experiments with both the mutation rate and the cut value discussed in previous section did not show a significant improvement. Therefore, the GA parameters were fixed in all computations to forty generations, a uniform crossover, and a mutation rate of 0.1.

Finally, while the exact algorithm (taking one or all possible permutations) gave good results in Table 4.2, the exact algorithm results may be affected by many factors such as: 1) the shapes 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 simultaneously all shapes into account and will give a good solution regardless of the demand points number, distribution or the shapes parameters (size, inclination).

CHAPTER FIVE

CONCLUSIONS AND FURTHER RESEARCH

In this chapter we summarize the work described in this thesis and propose avenues for future related research.

5.1 Conclusions

The research described in this thesis is aimed at introducing block norms to the planar maximal covering location problem (PMCLP). It offers an alternative perspective and methodology, as well as new solution techniques.

First, we introduced an alternative mixed integer formulation (MIP) based on the block norms geometrical properties presented in Chapter Two. Our MIP formulation is more general than any previously published formulations for the rectilinear planar maximal covering location problem (RPMCLP) in that it can handle facilities with block norms of different size (coverage measure), setup cost, and capacity. The formulation can be used for cases of siting facilities with coverage measure under block norms of the same degree. The main advantage of our MIP formulation is its adaptability to many problem variations. From the authors' experience, the MIP formulation would solve for

one shape and fifty demand points in 4-5 seconds on an Ultra-4 SPARC Sun station. The MIP formulation can be easily implemented in many standard commercial packages.

In Section 3.2 we provided an exact solution approach for sitting a single facility with coverage measure under a block norm of order 2. The algorithm presented is also capable of handling points with negative weights, which was not possible in the MIP formulation. For facilities with coverage under block norms of higher degree, the maximum clique approach will be more appropriate than the exact algorithm of Section 3.2. The computational cost of solving for the maximum clique depends on the number of demand points, number of vertices in the graph, and the graph density (number of edges). However, the maximum clique is an extensively studied problem: programs and algorithms and different solution techniques have been developed, and published in the literature. In Chapter Three, Section 3.4, we contributed to this vast body of literature by introducing the guided search algorithm for maximum clique problems (GSAMCP). It is more expensive to perform an enumerative search in a large neighborhood than in a small one; however, we are more likely to find larger cliques in the neighborhood of larger degree vertices. In GSAMCP, we addressed this fact and the problem of selecting a starting vertex where an enumerative local search can be employed. GSAMCP is a promising idea, offering plenty of opportunities for future improvements.

In Chapter Three we showed that a single facility PMCLP under different block norms can be solved as a maximum clique problem. Chapter Four extended this idea for the multifacility PMCLP under different block norms of any degree. In order to achieve that goal, we first 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. The transformation procedure to a graph theory problem introduces a new perspective to this set of problems. Performances of the GA algorithm and complete enumeration are reported.

5.2 Suggested Future Research

The block norms as a distance measure have been applied for a few problems in location theory, namely, the minimax and the Fermat-Weber problem. The work described in this thesis is the first attempt to formulate the MCLP with block norms. The thesis introduces a new model, a new guided search algorithm for the maximum clique problem, and applies genetic algorithms to facility location problems in a new way. Many avenues for future research are possible.

In Chapter Two, it was shown that the PMCLP with block norms of the same order can be formulated as an MIP using the geometrical properties of these block norms contours. In addition to this application, one can investigate the problem of optimal placement of parallel-sided polygons, allowing both rotation and translation of these polygons.

Also, the idea of the difference in Y-intercepts can be used for other polygonal shapes. In fact, in the course of this research, the author formulated maximum covering by a single non-convex shape using the geometrical properties of that shape. Additional research could be conducted on the MIP formulation and different solution methodologies.

The main area for further research arising from Chapter Three is the multilevel guided search algorithm for maximum clique problem (GSAMCP), where search in the neighborhood of selected vertex can also be guided.

Chapter Four dealt with the GA method, which, in turn, presents many steps that can be further investigated, such as selection of the initial population according to the facility coverage measure (reflected in the unit contour size). The PMCLP under different block norms has been modeled as an unconstrained quadratic problem (UQP). Therefore, by developing an efficient general solution technique for the UQP, one can apply it to the PMCLP under different block norms. Several alternative heuristic techniques for UQP have been described in the literature, such as the one-pass heuristic technique and the TABU search technique. These techniques, applied to UQP, can also be utilized for the PMCLP under different block norms.

BIBLIOGRAPHY

Aickelin, U. (2002). An indirect genetic algorithm for set covering problems. *Journal of the Operational Research Society* 53, 1118–1126.

Aneja, Y.P., R. Chandrasekaran, and K.P.K. Nair (1988). A note on the *m*-center problem with rectilinear distances. *European Journal of Operational Research* 35, 118-123.

Avella, P. et al. (1998). Some personal views on the current state and the future of locational analysis. *European Journal of Operational Research* 104, 269-287.

Aytug, H. and Cem Saydam (2002). Solving large-scale maximum expected covering location problems by genetic algorithms: A comparative study. *European Journal of Operational Research* 141, 480-494.

Babel, L. (1994). A fast algorithm for the maximum weight clique problem. *Computing* 52, 31-38.

Balas, E. and W. Niehaus (1998). Optimized Crossover-Based Genetic Algorithms for the Maximum Cardinality and Maximum Weight Clique Problems. *Journal of Heuristics* 4, 107-122.

Barequet, G., M. Dickerson, and P. Pau (1995). Translating a convex polygon to contain a maximum number of points. *Proceedings 7th Canadian Conference on Computational Geometry*, 61-66.

Battiti, R. and M. Protasi (2001). Reactive local search for the maximum clique problem. *Algorithmica* 29, 610-637.

Beaumont, John R. (1981). Location-allocation problems in a plane a review of some models. *Socioeconomic Planning Sciences* 15, 217-229.

Brandeau, M. L. and S. S. Chiu (1989). An overview of representative problems in location research. *Management Science* 35, 645-674.

Brimberg, J. (1989). *Properties of distance functions and minisum location models*. Ph.D. Thesis, McMaster University, Hamilton, Ontario.

Brimberg, J., P. Dowling, and Love, R. F. (1996). Estimating the parameters of the weighted l_p norm by linear regression. *IIE Transactions* 28, 363-367.

Brotcorne, L., G. Laporte, and F. Semet (2002). Fast heuristic for large scale covering-location problems. *Computers and Operations Research* 29, 651-665.

Burkard, R, E. (1984). Quadratic assignment problems. *European Journal of Operational Research* 15, 283-289.

Campbell, J. F. (1994). Integer programming formulations of discrete hub location problems. *European Journal of Operational Research* 72, 387-405.

Carraghan, R. and P. M. Pardalos (1990). An exact algorithm for the maximum clique problem. *Operations Research Letters* 9, 375-382.

Chandrasekharam, R., S. Subhramanian, and S. Chaudhury (1993). Genetic algorithm for node partitioning problem and applications in VLSI design. *Computers and Digital Techniques IEE Proceedings E* 140, 255-260.

Chiu, S. (1987). The minisum location problem on an undirected network with continuous link demand. *Computers and Operations Research* 14, 369-383.

Chu, P. C. and J. E. Beasley (1998). Constraint handling in genetic algorithms: The set partitioning problem. *Journal of Heuristics* 11, 323-357.

Church, R. L. (1984). The planar maximal covering location problem. *Journal of Regional Science* 24, 185-201.

Church, R. L. and C. S. ReVelle (1974). The maximal covering location problem. *Journal Regional Science Association* 32, 101-118.

Cornuejols, G. M., M. Fisher and G. Nemhauser (1977). Location of bank account to optimize float- An analytic study of exact and approximate algorithms. *Management Science* 23, 789-810.

Diaz, B and F. Rodriguez (1997). A simple search heuristic for the MCLP: Application to the location of ambulance bases in a rural region. *Omega* 25, 181-187.

Dickerson, M. and D. Scharstein (1998). Optimal placement of convex polygons to maximize point containment. *Computational Geometry* 11, 1-16.

Diestel, R. (1997). *Graph Theory*. Springer-Verlag.

Drezner, Z. and H. Hamacher (Eds.) (2002). *Facility Location, Application and Theory*. Springer, New York.

Drezner, Z. and G. O. Wesolowsky (1994). Finding the circle or rectangle containing the minimum weight of points. *Location Science* 2, 83-90.

Durier, R. and C. Michelot. (1985). Geometrical properties of the Fermat-Weber *problem. European Journal of Operational Research* 20, 332-343.

Fishburn, P. C. (1985). *Interval Orders and Interval Graphs*. Wiley Interscience Series in Discrete Mathematics.

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

Francis, R.L., F. Leon, J. McGinnis and J. A. White (1992). *Facility Layout and Location: An Analytical Approach, 2nd ed.* Prentice-Hall, New York.

Friden, C., A. Hertz and D. DE Werra (1990). Tabaris: An exact algorithm based on Tabu search for finding a maximum independent set in a graph. *Computers and Operations Research* 17, 437-445.

Gao, L., Powell R. Jr.(1994). Uncapacitated facility location: General solution procedure and computational experience. *European Journal of Operational Research* 76, 410-427.

Gary, M. R. and D. S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: Freeman.

Gendreau, M., P. Soriano and L. Salvail (1993). Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research* 41, 385-403

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Massachusetts, Addison-Wesley.

Grinde, R. B. and K. Daniels (1999). Solving an apparel trim placement using a maximum cover problem approach. *IIE Transactions* 31, 763-769.

Hamacher, H. and K. Klamroth (2000). Planar Weber Location Problems with Barriers and Block Norms. *Annals of Operations Research* 96, 191-208.

Hochbaum, D. S. and A. Pathria (1997) "Generalized *p*-center problems: Complexity results and approximation algorithms. *European Journal of Operational Research* 100, 594-607.

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

Jagota, A. (1995). Approximating maximum clique with a Hopfield network. *IEEE Transactions on Neural Networks* 6, 724-735.

Jagota, A. and L. A. Sanchis (2001). Adaptive, restart, randomized greedy heuristics for maximum clique. *Journal of Heuristics* 7, 565-585.

Jaramillo, J. H., J. Bhadury, and R. Batta (2002). On the use of genetic algorithms to solve location problems. *Computers and Operations Research* 29, 761- 779.

Kariv, O. and L. Hakimi (1979). Algoithmic approach to network location problems. I: The *p*-centers. *Siam Journal of Applied Mathematics* 37, 513-538.

Katayama, K., M. Tani, and H. Narihisa (2000). Solving large binary quadratic programming problems by effective genetic local search algorithm. *In Proceeding of the Genetic and Evolutionary Computation Conference (GECCO 00*, 643-650.

Lorena, L. A. N. and E. L. F. Senne (2003). Local search heuristics for capacitated *p*-median problems. *Networks and Spatial Economics* 3, 407-419.

Louveaus, F. V. and D. Peeters (1992). A dual-based procedure for stochastic facility location. *Operations Research* 40, 564-573.

Love, R. F., J. G. Morris, and G. O. Wesolowsky (1988). *Facilities Location: Models & Methods*. North-Holland, New York.

Megiddo, N. and K. J. Supowit (1984). On the complexity of some common geometric location problems. *Siam Journal Computing* 13, 183-196.

Mehrez, A. (1983). A note on the linear integer formulation of the maximal covering location problem with facility placement on the entire plane. *Journal of Regional Science* 23, 553-555.

Mehrez, A. and A. Stulman (1984). An extended continuous maximal covering location problem with facility placement. *Computers and Operations Research* 11, 19-23.

Mehrez, A., Z. Stern, and A. Stulman (1985). A single facility location problem with a weighted maximin-minimax rectilinear distance. *Computers and Operations Research* 12, 51-60.

Michilot, C. and F. Plastria (2002). An extended multifacility minimax location problem revisted. *Annals of Operations Research* 111, 167-179.

Minieka, E. (1970). The *m*-center problem. *Siam Review* 12, 138-139.

Mirchandani, P. B. and R. L. Francis (Eds.) (1990). *Discrete Location Theory*, Wiley Interscience, New York.

Nickel, S. (1998). Restricted center problems under polyhedral gauges. *European Journal of Operational Research* 104, 343-357.

Owen, S., and M. S. Daskin (1998). Strategic facility location: A review. *European Journal of Operational Research* 111, 423-447.

Pardalos, P. M. and R. G. Rodgers (1992). A branch and bound algorithm for the maximum clique problem. *Computers and Operations Research* 19, 363-375.

Pardalos, P. M. and J. Xue (1994). The maximum clique problem. *Journal of Global Optimization* 4, 301-328.

Plastria, F. (1992). On destination optimality in asymmetric distance Fermat-Weber problems. *Annals of Operational Research* 40, 355-369.

Reeves, R. C. (Ed.) (1995). *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill,UK.

Salhi, S. and M. H. Gamal (2003). A genetic algorithm based approach for the uncapacitated continuous location-allocation problem. *Annals of Operations Research* 123, 203-222.

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

Solar, M., V. Parada, and R. Urrutia (2002). A parallel genetic algorithm to solve the set covering problem. *Computers and Operations Research* 29, 1221-1235.

Soriano, P. and M. Gendreau (1996). Tabu search algorithms for the maximum clique problem. *Dimacs Series in Discrete Mathematics* 26, 221-237.

Thisse, J. F., J. E. Ward, and R. E. Wendell (1984). Some properties of location problems with block and round norms. *Operations Research* 32, 1309-1327.

Trotter, W. T. and F. Harary (1979). On double and multiple interval graphs. *Journal of Graph Theory* 3, 205-211.

Üster, H. (1999). *Weighted Sum of Order p and Minisum Location Models*. Ph.D.Thesis, McMaster University, Hamilton, Ontario, Canada.

Ventura, J. A. and Z. Dung (1993). Algorithms for computerized inspection of rectangular and square shapes. *European Journal of Operational Research* 68, 265-277.

Ward, J. E. and R. E. Wendell (1980). A new norm for measuring distance which yields linear location problems. *Operations Research* 28, 837-844.

Ward, J. E. and R. E. Wendell (1985). Using block norms for location modeling. *Operations Research* 33, 1075-1090.

Wesolowsky, G. O. and W. G. Truscott. (1975). The multiperiod location allocation problem with relocation of facilities. *Management Science* 22, 57-65.

Yamani, A., T. J. Hudgson, and L. A. Vega (1990). Single aircraft mid-air refueling using spherical distances. *Operations Research* 38, 792-780.

Younies, H. (2004). Guided search algorithm for maximum clique problem. Submitted.

Younies, H. and G. O. Wesolowsky (2004.a). Maximal covering by parallelogram shapes. *European Journal of Operational Research* 159, 1, 83-94.

Younies, H. and G. O. Wesolowsky (2004.b). Genetic algorithm for Maximal covering under different block norms. To be Submitted.