OPTIMAL LOCATION OF SINGLE AND MULTIPLE OBNOXIOUS FACILITIES: ALGORITHMS FOR THE MAXIMIN CRITERION UNDER DIFFERENT NORMS

IOANNIS GIANNIKOS

London School of Economics and Political Science

Thesis submitted to the University of London
for the degree of Doctor of Philosophy
in the Faculty of Economics

March 1993

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

This thesis investigates the computational problem of locating obnoxious (undesirable) facilities in a way that minimizes their effect on a given set of clients (e.g. population centres). Supposing that the undesirable effects of such a facility on a given client are a decreasing function of the distance between them the objective is to locate these facilities as far away as possible from the given set of clients, subject to constraints that prevent location at infinity. Emphasis is given to the MAXIMIN criterion which is to maximize the minimum client-to-facility distance. Distances are measured either in the Euclidean or the rectilinear metric.

The properties of the optimal solution to the single facility problem are viewed from different, seemingly unrelated, perspectives ranging from plane geometry to duality theory. In particular, duality results from a mixed integer programming model are used to derive new properties of the optimal solution to the rectilinear problem.

A new algorithm is developed for the rectilinear problem where the feasible region is a convex polygon. Unlike previous approaches, this method does not require linear programming at all. In addition to this, an interactive graphical approach is proposed as a site-generation tool used to identify potential locations in realistic problems. Its main advantages are that it requires minimal user intervention and makes no assumptions regarding the feasible region. It has been applied in large scale problems with up to 1000 clients, whereas the largest reported application so far involved 10 clients.

Alternative models are presented for the multi-facility problem as well. Each of them is based on different assumptions and is applicable to specific situations. Moreover, an algorithm is established for the two-facility problem based on the properties of the optimal solution. To the best of our knowledge this is the first attempt to address this problem in the plane.

Finally, a number of unresolved issues, especially in the multi-facility problem, are outlined and suggested as further research topics.

ACKNOWLEDGMENTS

I am grateful to Dr Gautam Appa, for his friendship and cooperation without which this thesis would never have been possible.

I must also thank the Alexander S. Onassis public benefit foundation for financing this course of study.

Thanks are also due to Eleni, Dimitris and Karen for their help and friendship and all the happy times we have had.

My special thanks to Panos for his invaluable support throughout this effort and his encouragement when things went wrong.

I am indebted to Eleni, my best friend who was always ready to listen and offer her advice and, more importantly, her moral support whenever I needed it.

I really cannot find words to describe my gratitude towards my parents. This thesis is dedicated to them because, simply, I owe them everything.

Στους γονείς μου

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

1.1 Problem Outline

Although location theory can be traced back as far as the 17th century, most of the literature refers to the location of desirable facilities such as hospitals or police stations. However, as a result of the extensive industrialization of the early 1960's, modern societies have become increasingly concerned about the problem of locating obnoxious (undesirable) facilities such as nuclear power plants, chemical factories or dump sites for waste disposal. The accidents at Bopal, Chernobyl and elsewhere shocked the whole world by revealing the disastrous effects of such facilities on nearby population centres. TIME magazine (1989) devoted a special issue to environmental problems and especially the growing problem of waste disposal. More recently, the Environment Secretary in Britain had to reject plans for a large scale toxic waste plant in South Yorkshire because of the threat to unpolluted water supplies (see THE GUARDIAN, 13 November 1991).

Despite the increasing number of undesirable facilities throughout the world, very little research has been done on the problems associated with them. A recent survey revealed that only 2% of the location literature deals with obnoxious facilities.

Obviously, the problem of locating obnoxious facilities in a way that minimises their undesirable effects on a given set of clients (e.g. population centres) is extremely complex since it involves environmental, economic and social issues. Most of the attempts to address the problem so far have assumed that the effect of an undesirable facility on a client is a decreasing function of the distance between them, thus reducing the problem to a distance maximisation one.

In addition to this, distance maximisation models can be used for the location of some desirable facilities which, for some reason, must be kept apart from each other. A typical application is the need to disperse business franchises to achieve maximum penetration in a market area.

The objective of this thesis is to investigate distance maximisation models in the two-dimensional plane and show how their characteristics can be viewed from different perspectives. In addition to this, our purpose is to propose new efficient algorithms for some of these models and exploit ideas suggested by other researchers to develop a graphical method, applicable to realistic large-scale problems. Moreover, we discuss the particular issues associated with multiple facility location and review different models applicable to different situations.

It should be kept in mind that this thesis does not intend to provide the ultimate answer to the problem of locating undesirable facilities which is very complicated anyway. It merely attempts to present theoretical and algorithmic tools which will help the decision maker understand the structure of the problem and, possibly, select several candidate solutions which he/she can then assess based on whatever criteria he/she considers important.

1.2 Structure of the Thesis

In chapter 2 we discuss the single facility problem in the two-dimensional plane under two alternative distance metrics (Euclidean and rectilinear) and two maximisation criteria (MAXISUM and MAXIMIN). We state the problem formally and analyze the properties of the optimal solution using different, seemingly unrelated techniques. We also present a mixed integer programming (MIP) formulation for the rectilinear version of the MAXIMIN problem and use duality results from the MIP to prove known and establish new properties of the optimum.

In chapter 3 we concentrate on the MAXIMIN problem and review previous attempts to address it. For each existing algorithm we discuss its complexity and its applicability to realistic problems.

Chapter 4 discusses linear programming (LP) based approaches to the single facility rectilinear MAXIMIN problem. After proving that one of the most efficient methods in the literature is not entirely correct, we introduce a new algorithm which solves the problem without using LP at all.

In chapter 5 we introduce an interactive graphical approach to the single facility MAXIMIN problem. This method is based on previous graphical techniques and uses two simple heuristics to minimise user intervention. We also demonstrate that the method is applicable to realistic problems and that it offers the possibility to experiment with various parameters of these problems and compare the results.

Chapter 6 considers the multifacility problem and discusses the issues raised by the introduction of more than one undesirable facility. We present several alternative models, each appropriate to particular applications and introduce a new algorithm which solves the two-facility rectilinear problem without using LP.

Finally, chapter 7 summarises our results and poses several future research questions which, in our opinion, are of great interest.

CHAPTER TWO

THE SINGLE FACILITY PROBLEM

2.1 Introduction

Like most good mathematical problems the single obnoxious facility problem is very easy to state: given a set of demand points (existing facilities) on the plane and a bounded permissible region S, locate a new obnoxious facility within S in a way that minimizes its undesirable effects on the demand points. The terms existing facility, client or customer will all be used to denote a demand point. Examples of undesirable facilities include industrial plants which emit pollutants, noise or radiation, pieces of hazardous equipment within a working environment, dump sites for waste disposal etc.

Assuming that the effect of the obnoxious facility on a given demand point is a decreasing function of the distance between them, the problem is to locate the new facility within S as far away as possible from the demand points. Clearly, in facility location problems distance from the customers is only one of the factors which should be taken into consideration. Travel costs to and from the facility as well as operating and maintenance costs may be equally important. As a result, the problem becomes extremely complex and analytical methods can only treat a small fraction of the relevant issues.

The models we present assume that the perceived cost of "living" near an undesirable facility outweighs all other costs. Consequently, distance is considered to be the most significant factor and the objective is to place the new facility as far away as possible from the demand points.

Section 2.2 presents several models for the single facility problem. Sections 2.3 and 2.4 discuss the properties of the optimal solution under alternative objectives

and distance metrics. Section 2.4, in particular, introduces the Euclidean and rectilinear bisectors i.e. the lines of equidistance from two demand points and explains how the properties of the optimal solution can be viewed from that perspective. Finally, section 2.5 shows how a mixed integer programming (MIP) formulation can be used to prove these properties for the rectilinear problem and reveal ways of making existing algorithms more efficient.

2.2 Description of the Problem

Let (x_i, y_i) for i=1,2,...,n be the coordinates of n demand points in the two-dimensional plane. The problem is to locate a single obnoxious facility at X(x, y) so as to maximize its distance from the n demand points. Mathematically the problem can be formulated as follows:

$$\max G(X) \qquad \text{(P1)}$$
$$\text{s.t.} \quad X \in S$$

where G(X) is a measure of the system's effectiveness and S is a bounded closed permissible region which prevents location at infinity.

Problem (P1) actually represents a whole class of problems depending on the maximization criterion and the distance metric. More specifically, G(X) could be the total weighted distance of the undesirable facility to all demand points, which is known as the MAXISUM criterion, i.e.

$$G(X) = \sum_{1}^{n} w_i * d(X, P_i)$$

where:

- w_i is a positive weighting factor expressing the importance of demand point P_i or, equivalently, the relative incompatibility between P_i and the undesirable facility to be located.

- $d(X, P_i)$ denotes the distance between location X and demand point P_i (in some distance metric).

The distance metric could be either Euclidean, i.e.

$$d_E(X, P_i) = ((x - x_i)^2 + (y - y_i)^2)^{1/2}$$

or rectilinear, i.e.

$$d_r(X, P_i) = |x - x_i| + |y - y_i|$$

The rectilinear metric, also known as the Manhattan metric, is adopted when travelling is possible along a grid of streets or corridors. Typical applications include locating a piece of hazardous equipment in an industrial plant or warehouse that is arranged into rectangular bays, or locating a chemical factory along a rectangular network of water canals.

Melachrinoudis and Cullinane (1986a) proved theorem 2.1 below stating that when the MAXISUM criterion is used with either distance metric the optimal solution to (P1) is one of the vertices of the convex hull H of S.

Theorem 2.1

Let H be the convex hull of S. If the MAXISUM criterion is used, then the set N of vertices of H contains the optimal solution to problem (P1).

Proof:

Since $S \subset H$

$$\max_{X \in H} G(X) \geq \max_{X \in S} G(X) \qquad (1)$$

Clearly, G(X) is convex in H and S; hence, the maxima on each side of (1) occur on extreme points of H and S i.e. on the vertices of H and S respectively. However, N is a subset of the set of vertices of S, therefore it contains the optimal solution to (P1).

Hence, Melachrinoudis and Cullinane (1986a) proved that in order to solve the MAXISUM problem it suffices to evaluate G(X) on all vertices of H and take the one which maximizes it. However, the MAXISUM criterion can be viewed as an aggregate measure of efficiency since it focuses on the "average" demand point. Since an obnoxious facility may even have lethal effects on its nearest demand point, it would seem more appropriate to adopt the MAXIMIN criterion, which is concerned with the distance to the nearest rather than the average customer.

Using the MAXIMIN criterion and assuming that the permissible region S is a two dimensional polygon, not necessarily convex, (P1) can be formulated as follows:

$$\max L \qquad \text{(P2)}$$

$$\text{s.t.} \quad L \leq w_i \, d(X, P_i) \quad i=1,\ldots,n \qquad (2)$$

$$X \in S \qquad (3)$$

The properties of the optimal solution to (P2) are discussed in the following section.

2.3 Properties of the Optimal Solution in the MAXIMIN Problem

2.3.1 The MAXIMIN criterion with Euclidean distances.

The MAXIMIN location problem using Euclidean distances is stated as follows:

$$\max L \qquad \text{(P3)}$$

$$\text{s.t.} \quad L \leq w_i \left((x - x_i)^2 + (y - y_i)^2 \right)^{1/2} \quad i=1,\ldots,n \qquad (4)$$

$$X \in S \qquad (5)$$

Although the objective function of (P3) is linear, the problem is nonlinear and nonconvex because of constraints (4) which define regions outside circular cones in the (x, y, L) space that have their vertices on the x-y plane and their axes of symmetry perpendicular to that plane. Hence, it is possible to have more than one local maximum for (P3).

Definition 2.1

A **local maximum** for (P3) is a location X(x, y) at distance L from its nearest points such that in the epsilon (ϵ) neighbourhood of X there is no better solution.

Melachrinoudis and Cullinane (1982) state and prove four properties of local solutions for the Euclidean MAXIMIN problem using simple geometric arguments. These properties are illustrated in figure 2.1 where it is assumed, without loss of generality, that $w_i = 1$ for all i.

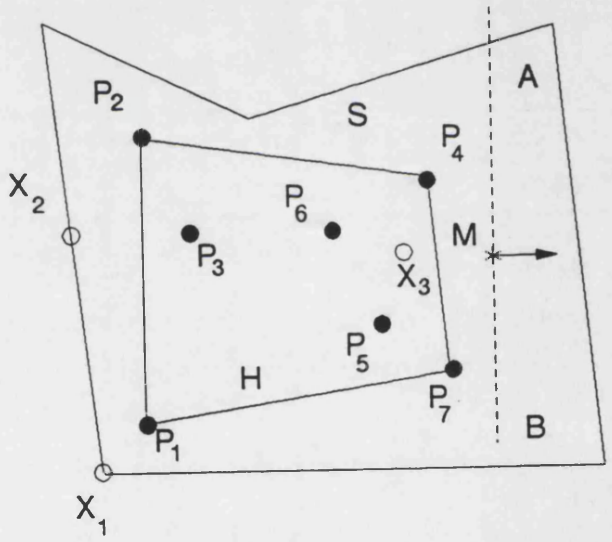

Figure 2.1: *Local optima in the Euclidean metric*

Property 0

A local optimum to (P3) will lie either on the boundary of the feasible region S or within the convex hull H defined by the n demand points.

Proof:

Point M which is outside the convex hull H in figure 2.1 cannot be a local solution. Since $M \notin H$ there exists a line AB separating M and H. Moving

infinitesimally perpendicular to this line away from H increases the distance to all clients, hence M cannot be a local optimum.

Property 1

If a local optimum X is at a vertex of S, at least one point is at distance L. See vertex X_1 in figure 2.1 whose nearest demand point is P_1. Note that although there is a nearest point to every vertex of S, not every vertex provides a local solution.

Property 2

A local solution X (x, y) on the boundary of S (but not at a vertex) is at distance L from at least two demand points. Point X_2 in the above figure is equidistant from two customers, namely P_2 and P_3.

Property 3

A local solution in the interior of S is equidistant from at least three demand points. Point X_3 in the interior of S is equidistant from P_4, P_5 and P_6.

2.3.2 The MAXIMIN criterion using the rectilinear metric.

When the rectilinear distance metric is adopted, the problem can be formulated as follows:

$$\max L \qquad \text{(P4)}$$

$$\text{s.t.} \quad L \le w_i (| x - x_i | + | y - y_i |) \quad i=1,\ldots,n \qquad (6)$$

$$X \in S \qquad (7)$$

Problem (P4) is also nonlinear and nonconvex because of constraints (6) that define regions outside pyramids in the (x, y, L) space which have their vertices on the x-y plane and their axes of symmetry perpendicular to the plane. Since (P4) is nonconvex, it is likely to have more than one local optimum. The form of these local solutions and their properties, as investigated by Melachrinoudis and Cullinane

(1986a), Melachrinoudis (1988) and Appa and Giannikos (1992), are illustrated in figure 2.2 where it is assumed for simplicity that $w_i = 1$ for all i.

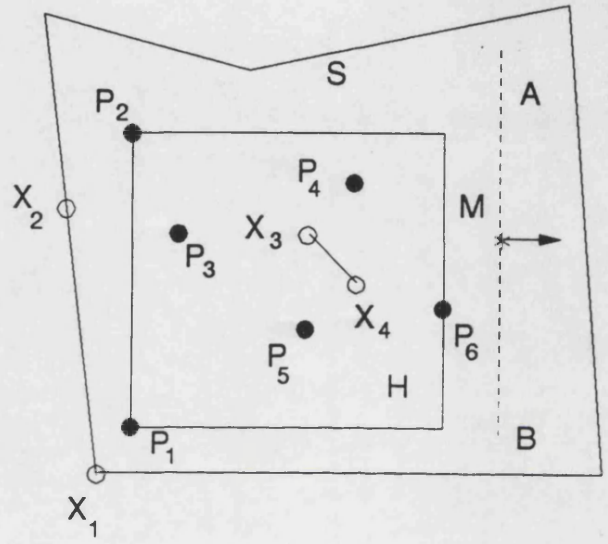

Figure 2.2: *Local optima in the rectilinear metric*

Property 0

A local solution to (P4) will lie either on the boundary of S or within S ∩ H, where H is the smallest rectangle encasing all demand points. Clearly, point M in figure 2.2 cannot be a local solution since by the same argument used in the Euclidean case, there exists a movement away from M towards the boundary of S that increases the value of the objective function.

Property 1

A local solution on a vertex of S is at distance L from at least one demand point (see X_1 in figure 2.2 with closest point P_1).

Property 2

A local solution on an edge of S (but not on a vertex) is equidistant from at least two demand points (see X_2 in the same figure with closest points P_2 and P_3).

Property 3

A local solution within S ∩ H occurs along a $\pm 45^o$ edge at which at least two constraints from set (6) of (P4) are binding (e.g. edge X_3X_4 equidistant from P_4 and P_5).

Proof:

Let X^* be a local optimum inside S ∩ H. Also, let L^* be the corresponding value of the objective function. If only one constraint from set (6) is binding at X^*, e.g. the i-th one, X^* cannot be a local maximum since moving away from X^* and P_i by a small distance δ either in the x or in the y direction would improve L^*. Suppose that two constraints from (6), say the i-th and the k-th are binding at X^*. It can be shown that the locus of points, whose weighted rectilinear distance from P_i is L^*, is a diamond with centre P_i and semi-diagonal distance equal to L^*/w_i, as illustrated in figure 2.3. Consequently, X^* which is at weighted distance L^* from both P_i and P_k, must lie on the 45^o segment AB, all points of which are at weighted distance L^* from both demand points.

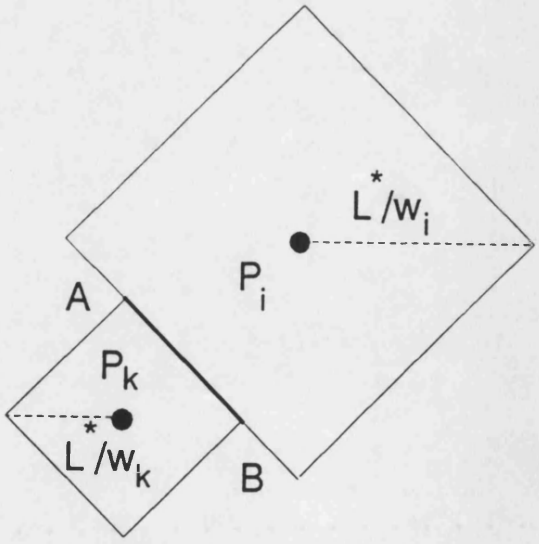

Figure 2.3: *Two intersecting diamonds*

These properties form the basis of several solution techniques either to the Euclidean or to the rectilinear problem some of which are discussed in the following chapter.

2.4 Bisectors: an Alternative Framework

2.4.1 Definition and Properties

In the previous section we showed that, apart from solutions on the vertices of S, a local solution to the single facility problem is equidistant from at least two existing facilities. Consequently, it would make sense to investigate the locus of points which are equidistant from two customers, known as the bisector. A formal definition of bisectors is given below.

Definition 2.2

Let P_i and P_j be two demand points on the plane and w_i and w_j be the corresponding weights. The **bisector** B_{i-j} of P_i and P_j is the locus of points equidistant from P_i and P_j, i.e.

$$B_{i-j} = \{ X \mid w_i d(X, P_i) = w_j d(X, P_j) \},$$

where d (P, Q) denotes the distance between P and Q in any distance metric.

2.4.1.1 The Unweighted Bisector.

Lee (1980) contains an excellent description of the bisectors in the unweighted case, i.e. when the weights corresponding to all demand points are equal. It can be seen that in the Euclidean case B_{i-j} is simply the perpendicular bisector between P_i and P_j (see figure 2.4).

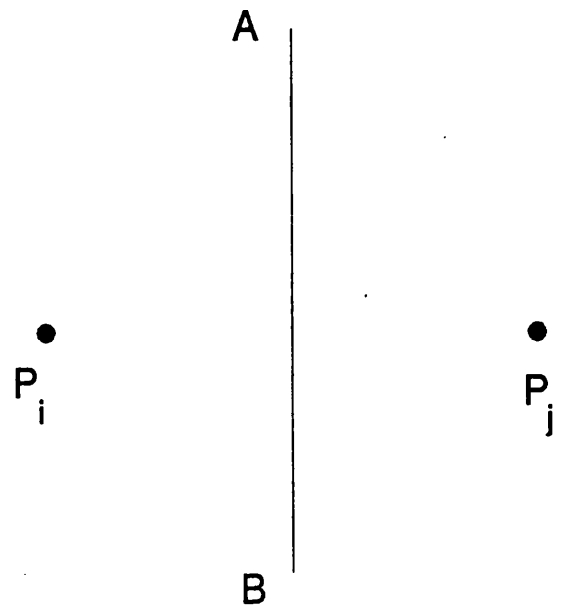

Figure 2.4: *Euclidean unweighted bisector*

In the rectilinear case the bisector depends on the relative position of the two demand points. Definition 2.3 distinguishes between two cases.

Definition 2.3

Two points $P_i(x_i, y_i)$ and $P_j(x_j, y_j)$ are said to form a **tall box** if $|x_i - x_j| < |y_i - y_j|$ and a **long box** if $|x_i - x_j| > |y_i - y_j|$.

As shown in figure 2.5, in the normal case where $|x_i - x_j| \neq |y_i - y_j| \neq 0$, the rectilinear unweighted bisector defined by P_i and P_j consists of one diagonal segment and two horizontal lines when P_i and P_j form a tall box, or two vertical ones when they form a long one. More specifically, returning to figure 2.5, there is a 45° line denoted as BC, along which the distance from P_i and P_j remains the same; along BA and CD the distance between the two is equal and increasing.

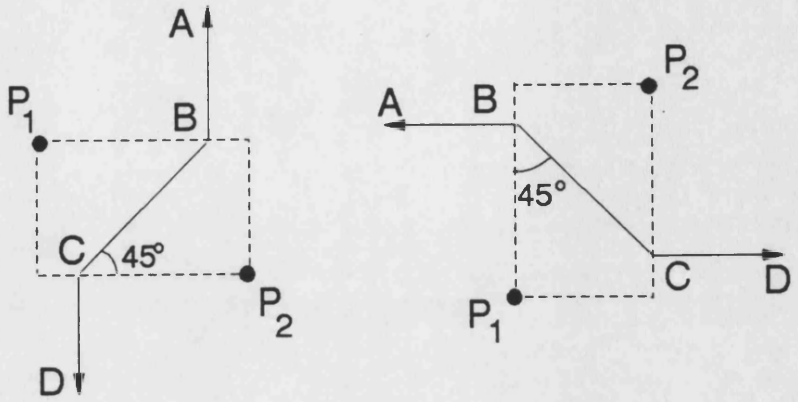

Figure 2.5: *Rectilinear unweighted bisector in the normal case*

When $| x_i - x_j | = 0$ or $| y_i - y_j | = 0$, the diagonal segment collapses into one point and the bisector is a vertical or a horizontal line respectively. If $| x_i - x_j | = | y_i - y_j |$, the bisector, as illustrated in figure 2.6, consists of the diagonal segment BC and areas A and D, in which the distance from both demand points is equal but increasing as we move away from B or C.

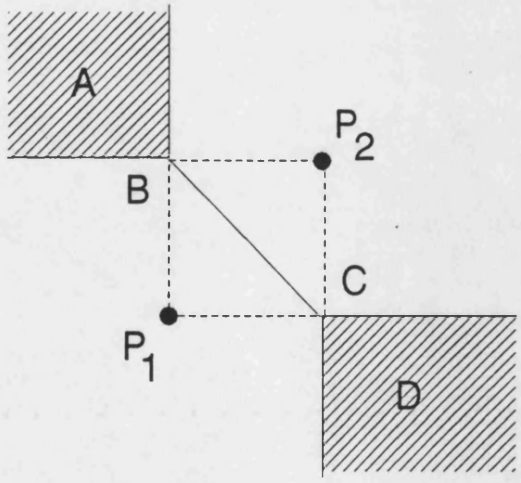

Figure 2.6: *Rectilinear bisector when P_1 and P_2 form a square*

It can be seen that in the normal case of figure 2.5 each bisector B_{i-j} divides the plane in two half planes. The half plane h (P_i , P_j) defined by B_{i-j} and containing P_i is the locus of points closer to P_i than to P_j , i.e.
h (P_i , P_j) = { X | d(X, P_i) ≤ d(X, P_j) }.

2.4.1.2 The Weighted Bisector.

Things are much more complicated in the weighted case, i.e. when each demand point has been assigned a different weight. Let us consider two existing facilities P_i and P_j with weights w_i and w_j respectively. Without loss of generality, let us assume that $w_j > w_i$.

It can be shown that in the Euclidean case the weighted bisector between P_i and P_j is the circle with centre $(w_j^2 P_j - w_i^2 P_i)/(w_j^2 - w_i^2)$ and radius $w_i w_j d_E (P_i , P_j)/(w_j^2 - w_i^2)$ where $d_E (P_i , P_j)$ denotes the Euclidean distance between P_i and P_j (see appendix A). Figure 2.7 shows the Euclidean bisector between P_i and P_j with $w_i = 1$ and $w_j = 2$.

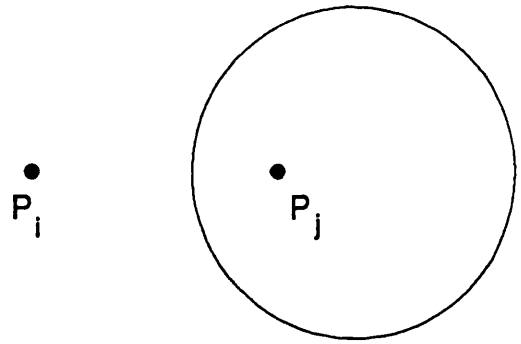

Figure 2.7: *Euclidean weighted bisector when w_i=1 and w_j=2*

In the rectilinear metric the bisector is a closed polygon containing the demand point with the largest weight and consisting of $\pm 45°$ segments and also segments with slope related to the respective weights of the two demand points, instead of the horizontal or vertical lines. See figure 2.8 where $w_i = 1$ and $w_j = 2$; segments BC, DE and AF are diagonal, whereas segments AB and CD have a slope of $\pm(w_i + w_j)/w_i = 3$ and segment EF a slope of $w_i/(w_i + w_j) = 1/3$.

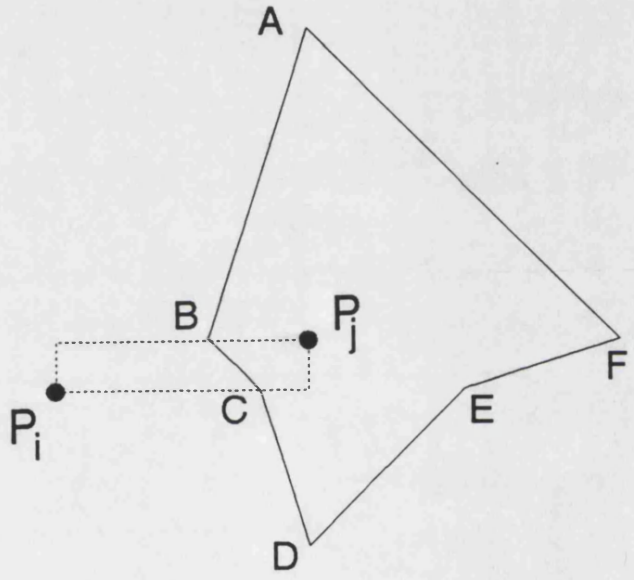

Figure 2.8: *Rectilinear weighted bisector when $w_i=1$ and $w_j=2$*

2.4.2 Interpretation of the Bisectors

The concept of the bisectors enables us to give a geometrical interpretation to properties 1 to 3 of section 2.3, both in the Euclidean and in the rectilinear case. More simply, supposing that a local solution X^* is at distance L^* from its nearest demand points, the properties presented in the previous section can be restated as follows:

2.4.2.1 The Euclidean case.

X^* will either be on a vertex of the feasible region S, or on the intersection of a bisector with an edge of the boundary of S, or on the intersection of three bisectors in the interior of S.

2.4.2.2 The rectilinear case

Similarly, a local solution X^* will either be on a vertex of S, or on the intersection of a bisector with a boundary edge of S, or in the interior of S where it must lie on the $\pm 45°$ section of a bisector $B_{i\text{-}j}$.

2.5 A Mathematical Programming Formulation for the Rectilinear Problem

2.5.1 Introducing the Model

When the feasible region S is a convex polygon defined by the intersection of m linear constraints, the properties of the Euclidean problem can be investigated using the Kuhn-Tucker theorem to characterize all possible local optima (see Melachrinoudis (1985)). More specifically, problem (P3) can be written as follows:

$$\max L \qquad \text{(P5)}$$

$$\text{s.t.} \quad L \le w_i \left((x - x_i)^2 + (y - y_i)^2 \right)^{1/2} \qquad i=1,\ldots,n \qquad (8)$$

$$a_j x + b_j y \le c_j \qquad j=1,\ldots,m \qquad (9)$$

Let u_i for $i=1,..,n$ and v_j for $j=1,..,m$ be the Lagrangean multipliers corresponding to constraints (8) and (9) respectively of problem (P5). Melachrinoudis (1985) observes that all local maxima can be generated by allowing three (or more) Lagrangean multipliers from (8) and (9) to be nonzero and solving the corresponding system of simultaneous equations. Three nonzero multipliers corresponding to

constraint set (8) define a local maximum within the convex hull of S, whereas one or more nonzero multipliers from set (9) define a local solution on the boundary of S.

A similar analysis for the rectilinear problem can be carried out using a mixed integer programming (MIP) formulation for problem (P4)[1] when S is a convex polygon. Let pdx_i represent the deviation of x from x_i when $x \geq x_i$, and ndx_i the absolute deviation when $x < x_i$. Define pdy_i and ndy_i similarly. Moreover, let zx_i and zy_i be zero-one variables which ensure that the positive and negative deviations are correctly represented by the previously defined deviation variables. More simply:

let $zx_i = 1$ imply $ndx_i = 0$ and $zx_i = 0$ imply $pdx_i = 0$, and

let $zy_i = 1$ imply $ndy_i = 0$ and $zy_i = 0$ imply $pdy_i = 0$.

If ux and uy are upper limits on x and y respectively, (P4) can be formulated as follows:

	max L		(P6)
s.t.	$x - x_i = pdx_i - ndx_i$	$i = 1,.., n$	(10)
	$y - y_i = pdy_i - ndy_i$	"	(11)
	$a_j x + b_j y \leq c_j$	$j = 1,.., m$	(12)
	$pdx_i \leq ux * zx_i$	$i = 1,.., n$	(13)
	$ndx_i \leq ux * (1-zx_i)$	"	(14)
	$pdy_i \leq uy * zy_i$	"	(15)
	$ndy_i \leq uy * (1-zy_i)$	"	(16)
	$pdx_i \geq 0, ndx_i \geq 0$	"	(17)
	$pdy_i \geq 0, ndy_i \geq 0$	"	
	$zx_i = 0$ or 1, $zy_i = 0$ or 1	"	(18)
	$L \leq w_i ((pdx_i + ndx_i) + (pdy_i + ndy_i))$	"	(19)

[1]This formulation and some of the results in 2.5.2 are borrowed from Dr. Appa's lecture notes for the Mathematical Programming II course at the LSE (1989-90). The formulation also appears in Appa and Giannikos (1992).

Constraints (13) to (18) allow either the positive or the negative deviation (but not both) to be non-zero while constraints (19) are the distance constraints which ensure that each demand point is at least distance L away from the obnoxious facility.

The model presented above is by no means the most efficient way to solve the problem since the number of zero-one variables, namely 2n, can get extremely large for realistic applications. However, at this stage we are interested in the MIP formulation merely to investigate the properties of the problem, as proposed by Appa and Giannikos (1992), rather than to solve it efficiently.

2.5.2 Using Extreme Point Properties of the MIP

Problem (P6) is solved by implicitly solving each LP derived by assigning value zero or one to each integer variable. Obviously, many of these LPs will be infeasible. Let P be a typical feasible LP. At least one solution to P must be an extreme point. Since each zx_i and zy_i is assigned a value zero or one before defining P, it has 4n+3 non-negative variables and an extreme point of P is a feasible intersection of 4n+3 independent constraints. Each $zx_i = 0$ or 1 implies that one of the corresponding pair of constraints (13) or (14) is satisfied as an equality, while the other is satisfied as a strict inequality. The same applies to each zy_i and the corresponding pair of constraints (15) and (16). Hence, in all we have 2n constraints satisfied as equalities for each problem P derived from (P6). These constraints, together with 2n equalities given by (10) and (11) give a total of 4n independent constraints for P which are satisfied as equalities. Consequently, at each extreme point of P, three independent constraints out of the remaining m+n constraints given by (12) and (19) must be satisfied as equalities. Moreover, at most two out of the m border constraints from set (12) can be independent. Hence, each extreme point is defined by k distance constraints from set (19) and 3-k border constraints, where k=1, 2 or 3, proving theorem 2.2. below.

Theorem 2.2

Let X(x, y) be a local solution to the rectilinear single obnoxious facility problem and let L be the corresponding value of the objective function. Then X is either:

(Case 1) at a vertex of S with at least k=1 point at distance L, or

(Case 2) on an edge of S with at least k=2 points at distance L, or

(Case 3) in the interior of S with at least k=3 demand points at distance L.

If the solution is degenerate there will be more than k points at distance L in all three cases.

Case 3 of theorem 2.2 seems to be in contrast with property 3 given in section 2.3.2. However, lemma 2.2 proven below based on the dual of problem P reveals that theorem 2.2 and the properties of section 2.3.2 are equivalent.

2.5.3 Using Duality Information from the MIP

Let us define the dual variables corresponding to the constraints of P as follows:

Table 2.1:Primal Constraints and Dual Variables for P

Constraints	(10)	(11)	(12)	(13)	(14)	(15)	(14)	(17)
Dual Vars.	r_i	s_i	p_j	pt_i	nt_i	pu_i	nu_i	q_i

Note that r_i and s_i for $i=1,\ldots,n$ are unrestricted variables, while the remaining ones are restricted to be non-negative. It turns out that the dual constraints corresponding to the primal variables of P (obtained after eliminating the zero-one variables from (P6)) are as follows:

Table 2.2: Primal Variables and Dual Constraints for P

Primal Vars.	Dual Constraints
L	$\sum_i q_i = 1$
x	$\sum_i r_i + \sum_j a_j * p_j = 0$
y	$\sum_i s_i + \sum_j b_j * p_j = 0$
pdx_i	$-r_i + pt_i - w_i * q_i \geq 0$
ndx_i	$r_i + nt_i - w_i * q_i \geq 0$
pdy_i	$-s_i + pu_i - w_i * q_i \geq 0$
ndy_i	$s_i + nu_i - w_i * q_i \geq 0$

Note that if in the primal optimal solution $zx_i = 1$ then $ndx_i = 0$ in the primal and $pt_i = 0$ in the dual. Also, pdx_i is basic and ndx_i non-basic so that by complementary slackness we have:

$-r_i - w_i q_i = 0$ or $r_i = -w_i q_i$.

On the other hand, if $zx_i = 0$ then $r_i = w_i q_i$.

Based on duality information for cases 1 to 3 of theorem 2.2 Appa and Giannikos (1992) established four lemmas which reveal several interesting properties of local optima.

The first lemma shows how duality information can be used to prove a result which is intuitively obvious.

Lemma 2.1

Let points P_1 , P_2 and P_3 define a local solution (x, y) in the interior of S and let $x_{min} = \min (x_1 , x_2 , x_3)$ and $x_{max} = \max (x_1 , x_2 , x_3)$. Define y_{min} and y_{max} similarly. Then $x_{min} \leq x \leq x_{max}$ and $y_{min} \leq y \leq y_{max}$.

Proof:

Suppose $x < x_{min}$. Then $zx_i = 0$ and $r_i = w_i q_i$ for i=1, 2, 3. Consequently, the first two dual constraints are:

$q_1 + q_2 + q_3 = 1$ and $w_1 q_1 + w_2 q_2 + w_3 q_3 = 0$,

which are inconsistent, since $q_i \geq 0$ (for dual feasibility) and $w_i > 0$. Similarly it can be shown that $x > x_{max}$ or $y < y_{min}$ or $y > y_{max}$ give inconsistent equations.

The second lemma also refers to local solutions in the interior of S and is based on the dual of case 3 of theorem 2.2.

Lemma 2.2

The basic solution corresponding to any local solution in the interior of S is multiply optimal.

Proof:

If points P_1 , P_2 and P_3 define a local solution (x, y) at distance L from all three of them, the following dual constraints must be satisfied:

$q_1 + q_2 + q_3 = 1$, $r_1 + r_2 + r_3 = 0$, $s_1 + s_2 + s_3 = 0$,

$q_i \geq 0$, $r_i \geq 0$ and $s_i \geq 0$ for i=1,2,3.

Looking at all possible locations for (x, y) within the bounds set by lemma 2.1, this is equivalent to the following system of equations:

$$q_1 + q_2 + q_3 = 1 \qquad (20)$$

$$\pm w_1 q_1 \pm w_2 q_2 \pm w_3 q_3 = 0 \qquad (21)$$

$$\pm w_1 q_1 \pm w_2 q_2 \pm w_3 q_3 = 0 \qquad (22),$$

with $q_i \geq 0$ required for dual feasibility. If all three terms in equation (21) or (22) have the same sign, the three equations are inconsistent. Hence, at least one term in

both (21) and (22) must have a positive sign and at least one must have a negative sign. It can be shown that in all such cases one of the q_i's has value zero.

For example, if $zx_1 = zx_2 = zy_1 = zy_3 = 0$ and $zx_3 = zy_2 = 1$, we have:

$$q_1 + q_2 + q_3 = 1$$

$$w_1 q_1 + w_2 q_2 - w_3 q_3 = 0$$

$$w_1 q_1 - w_2 q_2 + w_3 q_3 = 0$$

which is solved by $q_1 = 0$, $q_2 = w_3/(w_2 + w_3)$ and $q_3 = w_2/(w_2 + w_3)$. Since a basic variable in the dual is equal to zero, the dual is degenerate or, equivalently, the primal has multiple optimal solutions.

Geometrically, this implies that although local solutions along the diagonal segment of a bisector are equidistant from two customers, there is always one solution, corresponding to the basic solution of P, which is equidistant from three customers, as shown in figure 2.9.

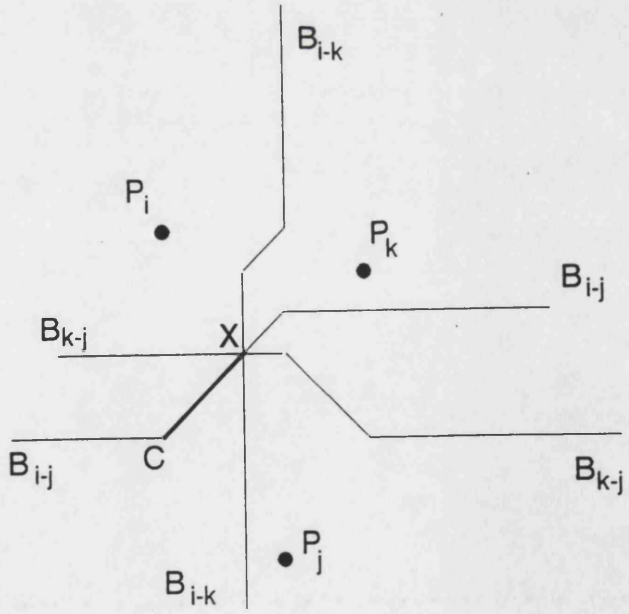

Figure 2.9: *Three intersecting bisectors*

This solution is defined by the intersection of three bisectors, as illustrated by point X in figure 2.9. Clearly, such a solution, at distance L from its nearest points, must be on the diagonal segment of at least one bisector, defined by say, P_i and P_j, since three distinct bisectors cannot intersect in any other way. All points of the segment which are further away from the third point i.e. segment XC from P_k in the figure, are also local solutions, since they are at least L away from all three points. However, note that the end of the segment, namely point C in the figure is <u>not</u> a local solution since a slight movement away from C along the horizontal part of B_{i-j} would increase the distance from all three demand points.

Clearly, as Melachrinoudis (1988) observes, if such a diagonal edge is globally optimal both of its endpoints must be local optima, as shown in figure 2.10.

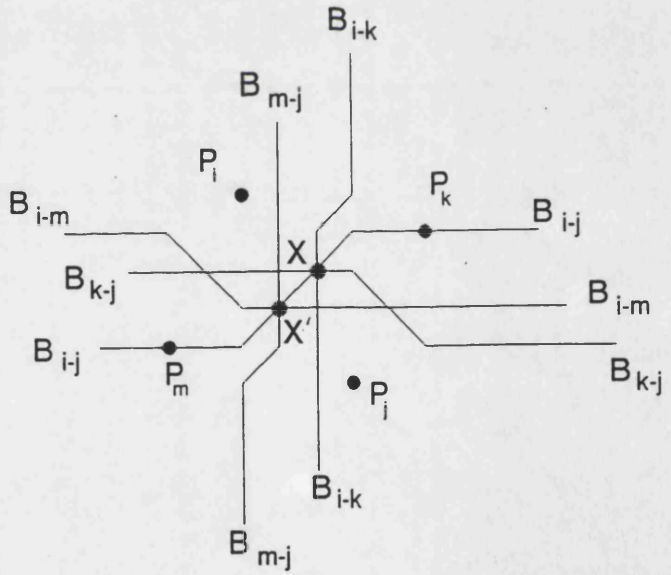

Figure 2.10: *Globally optimal edge*

Equivalently, there must be a fourth demand point, P_m in figure 2.10 preventing us from moving along B_{i-j} in a way that increases the distance from all demand points. Each endpoint of the edge is defined by the intersection of three bisectors and corresponds to a basic solution of problem P. Point X in the figure is equidistant from P_i , P_j and P_k and X′ is equidistant from P_i , P_j and P_m .

The third lemma refers to case 1 of theorem 2.2. To simplify the notation we assume, without loss of generality, that vertex V (x_v , y_v) is the intersection of the first two constraints, and that demand point P_1 is nearest to V at distance L_v .

Lemma 2.3

There is no local solution at V if:

$$(s_1 a_2 - r_1 b_2)/\Delta < 0 \text{ or } (r_1 b_1 - s_1 a_1)/\Delta < 0,$$

where: $\Delta = a_1 b_2 - a_2 b_1$,

$r_1 = w_1$ if $x_v \le x_1$ and $r_1 = -w_1$ if $x_v > x_1$,

and $s_1 = w_1$ if $y_v \le y_1$ and $s_1 = -w_1$ if $y_v > y_1$.

Proof:

A local solution to P must satisfy the following three equalities for dual feasibility:

$$q_1 = 1,\ r_1 + a_1 p_1 + a_2 p_2 = 0 \text{ and } s_1 + b_1 p_1 + b_2 p_2 = 0$$

in such a way that $p_1 \ge 0$ and $p_2 \ge 0$, with $r_i = w_i q_i$ if $zx_i = 0$ etc. Consequently, $p_1 = (s_1 a_2 - r_1 b_2)/\Delta \ge 0$, and $p_2 = (r_1 b_1 - s_1 a_1)/\Delta \ge 0$ which proves the lemma.

Lemma 2.3 implies that although the distance from the nearest point to any vertex can be used as a lower bound to the objective function, there may be no local solution at a given vertex. Example 2.1 presents an extreme case where there is no local solution at any vertex.

Example 2.1

Consider n=4 demand points P_1 to P_4 with coordinates (3, 8.5), (9.5, 8.5), (10, 4.5) and (3, 1.5) respectively. Also consider a feasible polygon S defined by vertices A to E with coordinates (0, 8), (9, 11), (12, 5), (4, 1) and (0, 2) respectively (see figure 2.11).

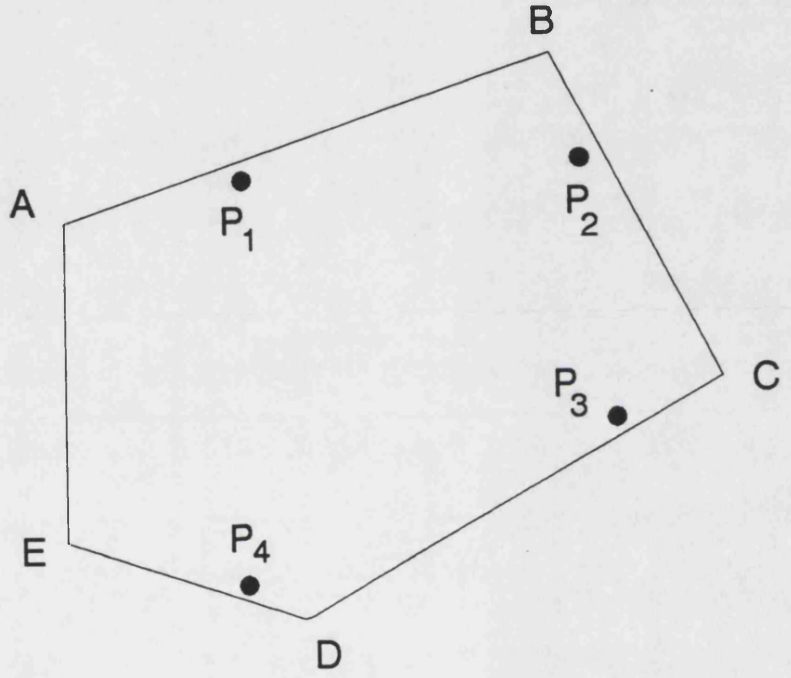

Figure 2.11: *Example 2.1*

It can be checked that vertex A, for instance, cannot provide a local maximum since we can move towards E along AE and increase the distance from its nearest point, i.e. P_1.

The fourth lemma refers to case 2 of theorem 2.2. Without loss of generality let us assume that demand points P_1 and P_2 are the nearest clients to a possible local solution at X (x, y) on an edge defined by $ax + by = c$.

Lemma 2.4

X (x, y) is not a local solution if

either $|x_1 - x_2| < |y_1 - y_2|$ and $|b/a| > 1$,

or $|x_1 - x_2| > |y_1 - y_2|$ and $|b/a| < 1$.

Proof:

Consider the case where $|x_1 - x_2| < |y_1 - y_2|$. Clearly, if P_1 and P_2 are in S, X (x, y) must satisfy:

either (case 1) $x < x_1 < x_2$ and $y_1 < y < y_2$

or (case 2) $x < x_1 < x_2$ and $y_2 < y < y_1$

either (case 3) $x > x_1 > x_2$ and $y_1 < y < y_2$

or (case 4) $x > x_1 > x_2$ and $y_2 < y < y_1$.

The dual conditions to be satisfied for case 1 are:

$q_1 + q_2 = 1$, $r_1 + r_2 + ap = 0$, $s_1 + s_2 + bp = 0$, $p \geq 0$, $q_1 \geq 0$, $q_2 \geq 0$.

Note that $x < x_1$ implies that $zx_1 = 0$ and, as shown in the proof of lemma 1, $r_1 = w_1 q_1$. By a similar analysis we can infer that $r_2 = w_2 q_2$, $s_1 = -w_1 q_1$ and $s_2 = w_2 q_2$. Substituting in the first three equations of (18) we get:

$p = -2w_1 w_2/\Delta$, $q_1 = w_2(a+b)/\Delta$ and $q_2 = w_1(a-b)/\Delta$

where $\Delta = (a-b)w_1 + (a+b)w_2$. For p to be non-negative, $\Delta \leq 0$. Consequently, $q_1 \geq 0$ and $q_2 \geq 0$ imply $a+b \leq 0$ and $a-b \leq 0$ respectively or, equivalently $|b/a| \leq 1$.

Cases 2 to 4 can be proven similarly.

Lemma 2.4 can be interpreted graphically as follows. We call an edge of the boundary of S a "tall edge" if its end vertices form a tall box, and a "long edge" if they form a long one. Then lemma 2.4 states that the bisector of two demand points forming a tall box cannot define a local solution on a long edge while the bisector of two points forming a long box cannot define a local solution on a tall edge. See figure 2.12, where the bisector B_{i-j} of P_i and P_j intersects edge MK of S at X, where

no local solution is possible since a slight movement towards M increases the distance from both demand points.

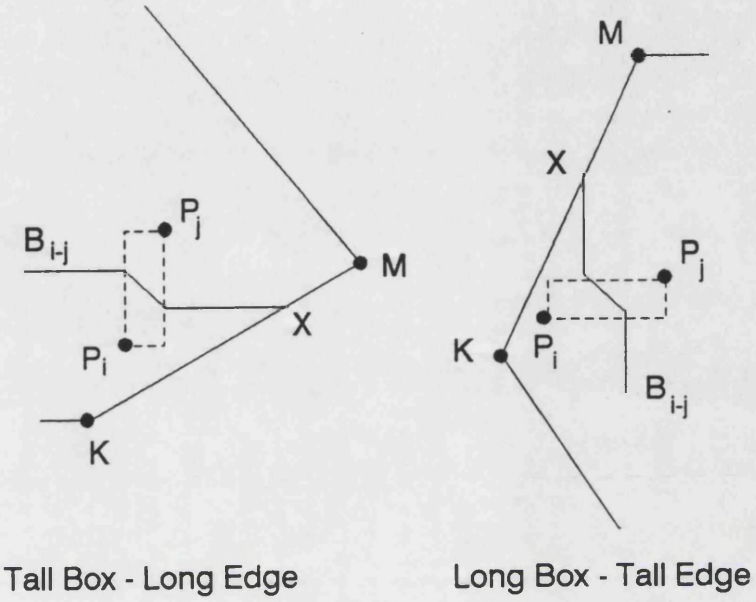

Figure 2.12: *Graphical interpretation of lemma 2.4*

2.6 Summary

In this chapter we presented various formulations of the single undesirable facility problem and focused our attention on the MAXIMIN version of the problem. We also stated the properties of the optimal solution, both in the Euclidean and the rectilinear distance metric. Finally, we used the concept of the bisectors to interpret these properties geometrically, and showed how duality, a seemingly unrelated

technique, can be utilized to derive several interesting results about the nature of local solutions to the problem.

CHAPTER THREE

PREVIOUS APPROACHES TO THE MAXIMIN PROBLEM

3.1 Introduction

In the previous chapter we stated the single undesirable facility problem and emphasised the MAXIMIN version of the problem, both with the Euclidean and the rectilinear distance metric. We also analyzed the properties of the optimal solution and showed that they can be viewed from various equivalent perspectives.

In this chapter we will outline several existing solution methods which exploit these properties to find the optimal location. We will also discuss other approaches which use special geometrical structures to solve the problem.

Section 3.2 reviews solution techniques for the Euclidean problem. Most of them are based on the properties presented in the previous chapter and enumerate all local optima in order to select the best one. Other methods transform the problem into an equivalent MINIMAX problem which is then solved to yield the optimal solution to the original problem. Finally, there exist graphical solution techniques which obtain an approximate solution to the problem. Their main advantage is that they are applicable in realistic problems since they do not require the feasible region to be convex or even connected.

Section 3.3 discusses the existing approaches to the rectilinear problem, which has not been as popular as the Euclidean one. Three major approaches will be discussed. The first one searches for the optimal location on the boundary and then in the interior of the feasible region. The second is essentially a complete enumeration technique using the properties of the optimal solution whereas the third approach starts by dividing the feasible region into rectangular areas and then solves a linear programming problem (LP) for each of them.

Section 3.4 briefly discusses the complexity of these approaches and concludes that the number of constraints as much as the number of demand points affects the overall performance of any method.

Finally, section 3.5 refers to the Voronoi diagram, a very elegant geometrical structure which has been used very efficiently to solve the unweighted problem, where all demand points are considered equally important.

3.2 Approaches to the Euclidean Problem

3.2.1 Complete Enumeration Approaches

Dasarathy and White (1980) are the first to consider the unweighted MAXIMIN problem in k dimensions where the feasible region is a bounded, non-empty convex polyhedron in R^k. More specifically, given n demand points P_i for i=1,...,n and a feasible region S, the aim is to find the largest hypersphere centred in S, whose interior is free of points P_i. If R is the radius of that hypersphere and X a k-dimensional vector representing its centre, the problem can be formulated as follows:

$$\begin{aligned} & \max L && && (P1) \\ \text{s.t.} \quad & d_E(X, P_i) \geq L && i=1,\ldots,n & \\ & X \in S && & \end{aligned}$$

Dasarathy and White use the Kuhn-Tucker conditions to prove that the properties derived in chapter 2 for the two-dimensional problem can be extended in k dimensions as well. They then propose a complete enumeration algorithm which is described below:

Algorithm A1

1. (Initialization)
   Let L be the radius of the best hypersphere found so far. Initialize it to 0.
2. (Search in the interior of S)
   Generate all local maxima in the interior of the convex hull by considering all combinations of the P_i's taken k+1 at a time. For each such combination, if the radius of the corresponding hypersphere is greater than L and its interior free of points P_i, update L. If not, discard the combination.
3. (Search on the faces of S)
   Consider all d-dimensional faces F of S where $d \leq k$.
   For each F consider all combinations of P_i's taken d+1 at a time and for each of them check if it can define a local solution on F which is better than L.
   If so, update L.

Clearly, the worst case complexity of the above algorithm is $O(n^{k+2})$ since in step 2 for each of the $\binom{n}{k+1}$ combinations, n demand points have to be checked for inclusion in the corresponding hypersphere. In order to improve the performance of the algorithm, Dasarathy and White use upper and lower bounds on R, details of which can be found in Dasarathy and White (1980). As a result, they report that the average complexity of their algorithm in three dimensions is $n^{3.85}$, much less than n^5.

Melachrinoudis (1985) addresses the problem when the feasible region S is a two-dimensional convex polygon and, basically, uses the properties of the optimal solution to solve it. He states the problem as follows:

$$\max \; L \qquad\qquad (P2)$$

$$\text{s.t.} \quad g_i(x, y) \leq 0 \qquad i=1,\ldots,n \qquad (1)$$

$$h_j(x, y) \leq 0 \qquad j=1,\ldots,m \qquad (2)$$

where

- X (x, y) is the location of the undesirable facility,
- $g_i(x, y) = L^2 - [w_i d_E(X, P_i)]^2 \leq 0$ and
- $h_j(x, y) = a_j x + b_j y + c_j$

The best value L_0 of the objective function at the vertices of S can be used as a lower bound for the global optimum, i.e.

$$L_0 = \max_j \{ \min_i w_i d_E(V_j, P_i) \} \qquad (3)$$

The solution method is essentially a complete enumeration of local optima, very similar to the one presented in Dasarathy and White (1980) and can be described as follows:

Algorithm A2

1. (Initialization)
   Start with the lower bound L_0.
   (Clearly, this step requires nm calculations and nm-1 comparisons).
2. (Search inside the convex hull)
   Take all combinations of three non-collinear points P_u, P_v and P_w, solve the system of three simultaneous equations $g_i(x, y) = 0$ for i=u, v, w and check each solution for feasibility of (1) and (2). If a solution is feasible and yields a value of L greater than the best solution found so far, update the current best.
3. (Search on the boundary)
   Assume that L_0 in (3) is achieved for demand point i=k and vertex j=p. Move along the p-th side of S and find all the points where two

constraints from set (1) are binding. If the value of L at any such point is greater than the current solution, update the current best. Move in the same fashion until all sides of S have been considered.

At the end of the process the current best solution is the global optimum which might not be unique, since the problem is not convex. The worst case complexity of the algorithm is $O(n^4)$ since $\binom{n}{3}$ combinations are considered in step 2 and each of them has to be tested for feasibility of constraint set (1).

A heuristic approach based on algorithm A2 is presented in Melachrinoudis and Cullinane (1985a) where step 2 is slightly modified to ignore combinations of points located far apart from each other since it is unlikely that a local maximum will result from such a combination.

Melachrinoudis and Cullinane (1986b) formulate the problem as a MINIMAX model assuming that the undesirable effect of the new facility to any demand point is inversely proportional to the distance between them. The solution method is almost identical to the one given by Melachrinoudis (1985) with a worst case complexity of $O(n^4)$.

A slightly different model is presented in Melachrinoudis and Cullinane (1985b) where the new facility must be at distance at least r_i from demand point P_i and the feasible region S can be a non-convex polygon represented by a clockwise sequence of its vertices Q_j. In this case the problem is formulated as follows:

$$\max L \qquad \text{(P3)}$$

$$\text{s.t.} \quad w_i d_E(P_i, X) \geq L \qquad i=1,\ldots,n \qquad (4)$$

$$d_E(P_i, X) \geq r_i \qquad " \qquad (5)$$

$$X \in S$$

The properties of the optimal solution are almost identical to the ones presented in the previous chapter, the exception being that a vertex of S cannot be a

local solution if it is associated with an interior angle greater than π. Algorithm A3 is slightly modified to cater for constraints (5) as explained below:

Algorithm A3

1. Relax constraints (5). Solve the resulting problem using algorithm A2.
2. If the solution from step 1 satisfies all constraints (5), stop. Otherwise, add into the problem the constraints of set (5) which are violated.
3. Solve the new problem using algorithm A2 and go to step 2.

As reported in Melachrinoudis (1985) and Melachrinoudis and Cullinane (1986a), the computation times of algorithms A2 and A3 are increasing with a power of n approximately equal to 3.

An interesting variation of problem (P1) is introduced by Karkazis and Karagiorgis (1986, 1987) who outline an algorithm for locating an obnoxious facility within a closed polygon S that contains a number of polygonal regions characterised as either restricted or protected. Restricted regions are areas where the facility may not be located whereas protected ones are areas for which care should be taken to locate the facility as far away as possible from their perimeters. The problem, as stated by Karkazis and Karagiorgis, is to find a feasible location in S, i.e. a point in S neither restricted nor protected, that maximizes the minimum distance between that point and the perimeter of the protected regions. Equivalently, find the maximum circle centred inside a free area and intersecting none of the protected regions.

Karkazis and Karagiorgis introduce the notion of a local maximum circle following a feasible course, i.e. intersecting no protected regions, while "rolling" around the perimeter of protected regions. They prove that as the circle is "rolling" its centre forms a trajectory consisting of first or second order segments (straight line

segments, circular arcs, parabolic, elliptic or hyperbolic segments). They then develop a method for scanning this trajectory until the optimal location is found.

Although this version of the problem is much more realistic, no computational results are provided and, to the best of our knowledge, no actual implementation of the method has been reported.

3.2.2 Bisection Approaches

Drezner and Wesolowsky (1980) introduce the weighted single facility problem in a two-dimensional convex region. The optimal location should maximize the weighted distance of the undesirable facility from its nearest demand point. At the same time the facility must be within a pre-specified distance r_i from each client P_i for i=1,...,n. Hence, the feasible region is the intersection of n circles of radius r_i, each centred at a facility point P_i and representing a maximum distance constraint with respect to the point in question.

Drezner (1983) presents a unified approach to solving single facility MINIMAX as well as MAXIMIN problems in the plane and on the sphere. According to this approach the system's effectiveness is measured either by the Euclidean distances themselves, or by a general function of these distances. More simply, a function $f_i [d_E (P_i , X)]$ is associated with demand point P_i ; for regular Euclidean distances $f_i [d_E (P_i , X)] = w_i d_E (P_i , X)$ where w_i is the weight corresponding to P_i . There are two sets of constraints, set S_1 , limiting the solution to lie inside given circles, and S_2 , limiting it to lie outside given circles. Consequently, the MINIMAX problem is formulated as follows:

$$\begin{aligned} & \min F(X) && \text{(P4)} \\ \text{s.t.} \quad & S_1 \text{ and } S_2 , \end{aligned}$$

where $F(X) = \max_i \{f_i[d_E (P_i, X)]\}$.

If the problem is MAXIMIN, $f_i [d_E (P_i , X)]$ is simply replaced by $-f_i [d_E (P_i , X)]$.

For a given value F_0 , $F(X) \leq F_0$ is equivalent to $f_i [d_E (P_i , X)] \leq F_0$ for $i=1,...,n$. When $f_i (d)$ is strictly monotonic, there exists an inverse function g_i such that $f_i [g_i (f)] = f$. If $f_i (d)$ is increasing, then $f_i [d_E (P_i , X)] \leq F_0$ is inside the circle with centre P_i and radius R_i , and if $f_i (d)$ is decreasing then it is outside that circle where $R_i = g_i (F)$.

A lower bound F_{min} and an upper bound F_{max} on the objective function are calculated as explained in Drezner (1983) and the optimal solution F^* is found by a bisection method, which can be outlined as follows:

Algorithm A4

1. Find F_{min} and F_{max} .
2. Let $F_0 = (F_{min} + F_{max})/2$. Consider all circles C_i satisfying the inequality $f_i [d_E (P_i , X)] \leq F_0$ for $i=1,...,n$. If $f_i (d)$ is increasing, add C_i to S_1 , otherwise add it to S_2 . Let these extended groups of constraints be S_1' and S_2' respectively.
3. Find a point satisfying S_1' and S_2'.
4. If such a point exists, then update F_{max} to F_0 . Otherwise, update F_{min} to F_0 .
5. If $F_{max} - F_{min} \leq \epsilon$, where ϵ is a pre-specified tolerance, $F^* = F_{max}$ and the optimal location is the last feasible solution found in step 3. Otherwise, go to step 2.

Drezner generalizes a procedure used in Drezner and Wesolowsky (1980) to find a feasible point in step 3. Briefly, if there is a point in the interior of some circles and the exterior of some others, then there must be a point which is *on* one of the circles. Therefore, we can consider each circle in turn and cut off the parts of its circumference which are infeasible to other circles one by one. If the intersection of the circumference of that circle with all others is not empty, we have

found a feasible point. On the other hand, if the intersection of every circle with all others is empty, no feasible point exists. The complexity of this procedure, as stated in Drezner (1983), is $O(n^3)$. The number of iterations of algorithm A2 is independent of n, when all circles are bounded in the same area. Hence, the overall complexity of the algorithm is $O(n^3)$.

The above algorithm cannot cater for linear constraints, unless they are approximated by circles with large radii. However, the transformation of a set of linear constraints into a circle is not obvious at all.

3.2.3 Graphical Methods

Most of the approaches referred to in the previous sections assume that the feasible region is a connected polygon. However, this assumption makes it impossible to model realistic situations, since this is rarely the case in real life problems. Consequently, a number of researchers propose graphical solution procedures to obtain a number of near-optimal locations.

The main idea behind these procedures is very simple: circles of increasing radius are drawn around each demand point and the final area not covered by any circle indicates the optimal location. The fact that the graphical methods do not require the feasible region to be convex or even connected implies that natural barriers, such as mountains or lakes, can be dealt with easily. Moreover, the forbidden regions can have any shape, not necessarily circular, as assumed in Melachrinoudis and Cullinane (1985b). Details on the implementation of interactive graphical approaches can be found in Melachrinoudis and Cullinane (1985a) and Melachrinoudis (1985), where it is reported that the graphical method was not applicable to large scale problems, with more than 10 clients, due to the rapid increase in computation time as the number of demand points increased. However, in chapter 5 we show that with suitable modifications the graphical approach can be used to solve problems with more than 500 demand points in reasonable time.

Hansen, Peeters and Thisse (1981) present what is perhaps the most comprehensive approach to the single undesirable facility problem. They consider a general distance metric and a feasible region defined by the union of m convex polygons P_j which might even be disjoint. The cost associated with each demand point is a decreasing function D_i of distance, not necessarily linear or quadratic. The objective is to minimize the maximum of these costs. The algorithm, called *Black and White*, can be described as follows:

Algorithm A5

1. (Initialization)
   Represent the points P_i and the feasible region S on a map. Choose a few feasible points $s \in S$ and compute the corresponding values of the objective function. Let L_{opt} denote the smallest cost and s_{opt} the corresponding point.
2. (Elimination of Regions)
   Compute $R_i = G_i(L_{opt})$ for each i, where G_i is the inverse function of D_i. Trace the corresponding iso-cost curves on the map and shade the interior of each of them.
3. (Improvement of Solution)
   Consider the unshaded regions. If all of them have diameter smaller than a pre-specified tolerance, terminate with s_{opt} the optimal solution. Otherwise, select a central point s_h in each unshaded region S_h, compute the objective function for all s_h and let L_{opt} be the minimum of all L_h and s_h the corresponding point. Then return to step 2.

However, as Hansen et al. admit, the method requires significant user intervention in step 3 since the user has to determine the feasible areas S_h and to find points $s_h \in S_h$ where the function is to be evaluated. Hence, the algorithm above

difficult to implement on a computer as an automated process and as a result, no large scale application of this method has been reported.

3.3 Approaches to the Rectilinear Problem

3.3.1 A Boundary and Segment Search Approach

The first attempt to address the rectilinear problem is by Drezner and Wesolowsky (1983) who state the problem as follows:

$$\max \; F(X) \qquad (P5)$$

$$\text{s.t.} \quad a_j x + b_j y \leq c_j \qquad j=1,\ldots,m \qquad (6)$$

where:

- $F(X) = \min_i \{ w_i d_r(X, P_i) \}$
- P_i are n demand points and w_i their corresponding weights
- X (x, y) is the facility to be located.

Drezner and Wesolowsky propose a boundary and segment search technique for solving (P5). More simply, they search for the optimal location first on the boundary B and then in the interior I of the feasible region S. In order to search along B, they pre-process the constraint set (6) and obtain a description of B as a list of closed segments B_j for $j=1,\ldots,m$. Having done that, they then calculate F_j^U, an upper bound on F (X) on segment B_j. Since $w_i d_r(X, P_i)$ is convex on the segment, its maximum u_i must occur at either end of B_j. Consequently, an upper bound is given by $F_j^U = \min_i \{ u_i \}$.

These upper bounds are then sorted in descending order. Starting from the segment with the largest upper bound, Drezner and Wesolowsky use the value of the objective function at any point on the segment as a lower bound F_j^L and then perform a binary search to find the optimal solution on this segment. At each stage of this binary search they check whether a given value $f = (F_j^L + F_j^U)/2$ for F(X) is feasible for the segment B_q in question by considering a diamond with semi-diagonal f/w_i

around each P_i and discarding the part of B_q which is inside the diamond. If one or more subsegments of B_q are left, then a feasible solution with value f exists on B_q and $F_j^L = f$. Otherwise, $F_j^U = f$.

This search yields the optimal solution on B_q. The process is repeated until the next largest upper bound is less then the current optimal solution.

Having found the optimal solution F_0 on B, Drezner and Wesolowsky search the interior I next. They acknowledge that if the optimal solution to the whole problem is in I, it must be equidistant from at least two clients, say P_i and P_j and the optimal distance will be $F^* = H_{i-j} = (w_i w_j/(w_i + w_j))d_r(P_i, P_j)$. Moreover, they state that in the non-degenerate case the locus of points which are at distance H_{i-j} from both P_i and P_j is a 45° segment D_{i-j} as explained in the previous chapter. Consequently, they consider all pairs of P_i's and use F_0 as a lower bound on the optimal solution, as explained below:

Algorithm A6

1. Find the optimal solution F_0 on the boundary as explained above.
2. Consider all pairs of demand points P_i and P_j. If $H_{i-j} \leq F_0$, then discard the pair.
   Otherwise, for all P_k for $k \neq i,j$ discard the points of D_{i-j} whose distance from P_k is less than H_{i-j}.
   If some point(s) of D_{i-j} remain then $F_0 = H_{i-j}$.
   Otherwise, discard the pair.
3. $F^* = F_0$.

Drezner and Wesolowsky do not give complexity bounds for their algorithm. However, it can be seen that the calculation of the upper bound F_j^U for each segment j of S involves computing the u_i's for $i=1,\ldots,n$ and finding their minimum. Hence, computing the upper bounds for all segments requires O(mn) calculations. Moreover,

the search in the interior I requires $O(n^3)$ calculations since n(n-1)/2 pairs of points are considered and n-2 feasibility checks are performed for each of them.

3.3.2 A Complete Enumeration Approach

Apart from the Euclidean problem, Melachrinoudis and Cullinane (1986a) discuss the rectilinear version as well. They formulate the problem as follows:

$$\max L \qquad \text{(P6)}$$

$$\text{s.t.} \quad L \le w_i d_r(X, P_i) \quad i=1,\ldots n \qquad (7)$$

$$a_j x + b_j y \le c_j \quad j=1,..,m \qquad (8)$$

The solution technique, based on the properties of the optimal solution presented in the previous chapter, is very similar to algorithm A2, as outlined below:

Algorithm A7

1. (Search on the vertices of S).
   Evaluate the objective function at all the vertices of S. Let L_0 be the maximum of these values.
2. (Search in the interior of S).
   For all pairs of points P_i and P_j find the value of L where the two corresponding constraints from set (7) are binding.
   If $L > L_0$ then find the feasible part of the corresponding 45° segment.
   If such a part exists then $L_0 = L$.
3. (Search on the boundary).
   For each boundary equation of (8) and each pair of constraints (7) let L be the solution of the system of these three equations and X the corresponding location.
   If $L > L_0$ and X is feasible then $L_0 = L$.

At the end of the process L_0 is equal to the global optimum. By the same argument that was used for the search in the interior for algorithm A6, the worst case

complexity of algorithm A7, according to Melachrinoudis and Cullinane, is O (n^3) as well.

3.3.3 Linear Programming Based Approaches

Apart from the binary and segment search algorithm, Drezner and Wesolowsky (1983) present also a method based on linear programming (LP). Its basic idea is quite simple: given n demand points $P_i (x_i , y_i)$ and a convex feasible region S, defined by m linear constraints, construct the smallest rectangle encasing S and then draw one horizontal and one vertical line through each demand point. As a result, the rectangle is divided into at most $(n+1)^2$ rectangles some of which may be entirely outside S, as shown in figure 3.1.

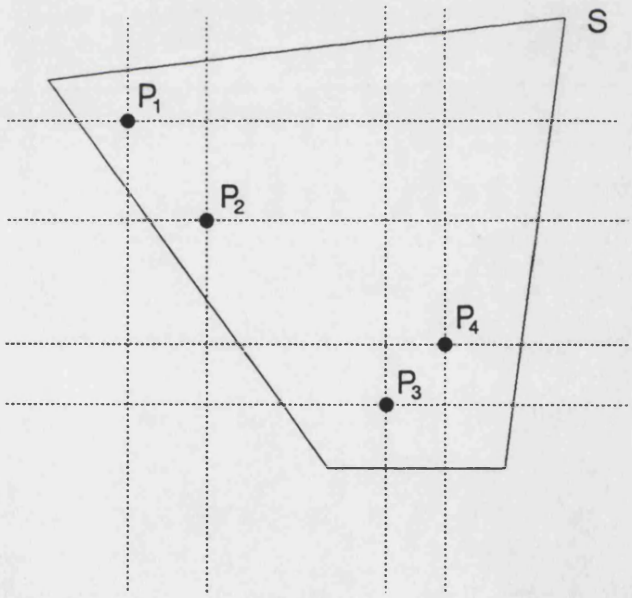

Figure 3.1: *Relevant rectangles for n = 4*

Clearly, the optimal location must lie inside S in one of these rectangles. However, as Drezner and Wesolowsky observe, a location X (x, y) inside a rectangle determines uniquely for each P_i the sign of $x-x_i$ and $y-y_i$. Consequently, the problem for each rectangle can be formulated as an LP as follows:

$$\max L \qquad \text{(P7)}$$

$$\text{s.t.} \quad \pm(x - x_i) \pm(y - y_i) \geq L \qquad i=1,\ldots,n \qquad (9)$$

$$x_{min} \leq x \leq x_{max},\ y_{min} \leq y \leq y_{max} \qquad (10)$$

$$a_j x + b_j y \leq c_j \qquad j=1,\ldots,m \qquad (11)$$

where x_{min} , x_{max} , y_{min} and y_{max} are the upper and lower bounds on x and y respectively, defining the rectangle in question, and the plus (minus) sign in (9) is used when the succeeding expression is positive (negative). Problem (P7) can be solved separately for each rectangle and the solution with the largest L over all rectangles is then the solution to the original problem.

Since this approach results in $O(n^2)$ LP problems, Drezner and Wesolowsky develop special techniques in order to improve the performance of the algorithm.

Firstly, they check each rectangle for feasibility before performing linear programming and discard the rectangles which are outside S. More specifically, they define the segments $[y_1, y_2]$ and $[y_3, y_4]$ as the parts of the lines $x=x_{min}$ and $x=x_{max}$ respectively, which are inside the feasible region (see figure 3.2).

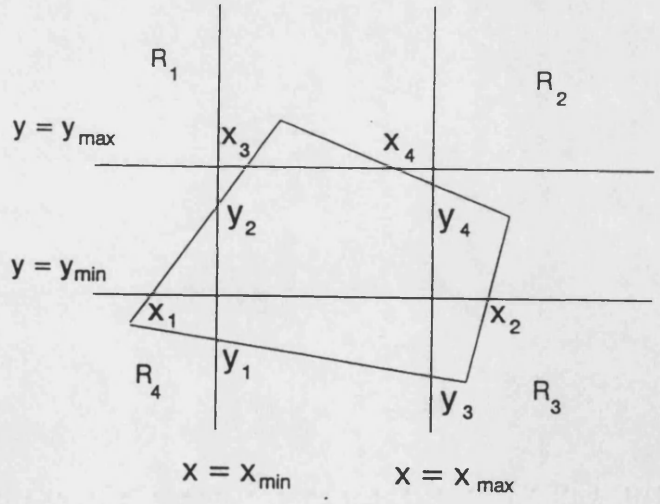

Figure 3.2: *Identifying infeasible rectangles*

They also define $[x_1, x_2]$ and $[x_3, x_4]$ similarly, although they do not explain how they derive these segments from the m original constraints describing S.

Clearly, there is no feasible point inside a given rectangle if all of the following conditions hold:

$$y_1 > y_{max} \text{ or } y_2 < y_{min}$$
$$y_3 > y_{max} \text{ or } y_4 < y_{min}$$
$$x_1 > x_{max} \text{ or } x_2 < x_{min}$$
$$x_3 > x_{max} \text{ or } x_4 < x_{min}$$

Secondly, they calculate an upper bound UB_k on L inside rectangle k by evaluating the objective function on the four vertices V_1 to V_4 of k and setting:

$$UB_k = \min_i \{w_i \max_{V_j} \{d_r(P_i, V_j)\}\} \qquad (12)$$

Hence, the complete algorithm can be outlined as follows:

Algorithm A8

1. Eliminate all infeasible rectangles using the conditions described above.
2. Calculate the upper bounds UB_k for all remaining rectangles using (12) and sort them in descending order.
3. Solve the dual of (P7) for all rectangles, starting from the one with the largest upper bound. Stop when the next largest upper bound is not greater than the best LP solution found so far.

According to Drezner and Wesolowsky algorithm A8 was generally faster than algorithm A6 (the boundary and segment search approach). However, it requires much more memory since up to $(n+1)^2$ upper bounds and their associated coordinates may have to be stored. Moreover, the preprocessing stage required for the elimination of infeasible rectangles is not very obvious.

Melachrinoudis (1988) adopts essentially the same approach to the problem, by constructing the grid of rectangles and then solving the dual of (P7) for each of

them. If the dual solution is degenerate, a dual Simplex pivot is performed yielding another optimal point and, thus defining a multiple solution edge within the rectangle in question. In other words, Melachrinoudis recognizes that solutions in the interior of S are equidistant from two demand points and are multiply optimal.

Mehrez, Sinuany-Stern and Stulman (1986) suggest an interesting improvement of Drezner and Wesolowsky's LP-based method, for the unweighted version of the problem. They observe that by the construction of each rectangle k, every demand point must fall into one of the four regions R_1 to R_4 (see figure 3.3) and that if a particular region is empty, only one demand point in that region, defined as the closest point, will form an active constraint of type (9) for (P7). All other clients in that region must be further away from the entire rectangle, thus forming redundant constraints.

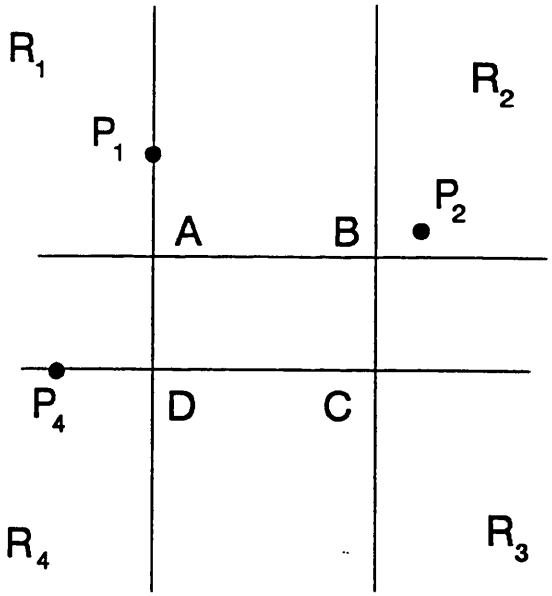

Figure 3.3: *Four regions around a rectangle*

Consequently, the calculation of the upper bounds is simplified and the dimension of the LP's is reduced. More specifically, let us define regions R_i and R_j to be opposite each other when $|i-j| = 2$, and neighbours of each other when

$|\, i-j \,| \neq 2$; let also P_i be the closest point corresponding to region R_i for a particular rectangle k. According to Mehrez et al. the upper bound UB_k for rectangle k can be calculated as follows:

Case 1

Suppose that only two neighbouring regions, say R_1 and R_4, are non empty. Mehrez et al. implicitly consider the bisector defined by P_1 and P_4, as shown in figure 3.4 where C and B are the corners of the rectangle opposite P_1 and P_4 respectively and M the point where the bisector between P_1 and P_4 intersects the line $x = x_{max}$, if that intersection exists. If $L = d_r(M, P_1) = d_r(M, P_4)$, $L_1 = d_r(C, P_1)$ and $L_4 = d_r(B, P_4)$ then $UB_k = \min \{ L, L_1, L_4 \}$.

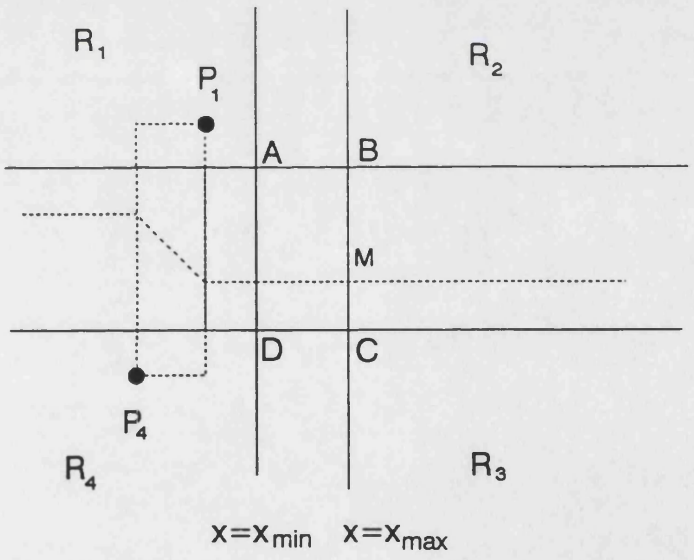

Figure 3.4: *Two neighbouring non empty regions*

Case 2

In all other cases, if say region R_1 is not empty let L_1 be either the half distance of P_1 from P_3 if region R_3 is not empty, or its distance from the opposite corner of k i.e. point C in figure 3.4 if R_3 is empty. The remaining L_i 's are calculated similarly. The upper bound in this case is the minimum of L_i .

Given these conditions for the calculation of the upper bounds the closest point algorithm can be described by the following iterative process:

Algorithm A9

1. Create the grid of rectangles.
2. Eliminate any infeasible rectangles, as explained in Drezner and Wesolowsky (1983).
3. Calculate the upper bounds UB and sort them in descending order.
4. If the point producing the largest upper bound falls within its rectangle, then it is the exact solution to the problem. If not, solve (P7).
5. Continue step 4 for rectangles with progressively smaller upper bounds and stop when the next largest upper bound is not greater than the best exact solution found so far.

Clearly, algorithm A9 is a significant improvement of the LP-based method by Drezner and Wesolowsky, since it results in considerably smaller LP's and reduces the number of LP's that have to be solved. Moreover, the upper bounds proposed by Mehrez et al. are tighter than the ones given by Drezner and Wesolowsky and can lead to more efficient solutions. In fact, algorithm A9 seems to be the most efficient LP based method to date and will be discussed in greater detail in the following chapter, where some improvements of the algorithm will be suggested.

3.4 Comments on the Complexity of the Previous Approaches

Apart from the *Black and White* algorithm, the algorithmic complexity for most of the methods presented so far, both for the Euclidean and the rectilinear problem, is reported in terms of the number of demand points n. However, all of them at some stage perform a search along the boundary B of the feasible region,
(a) either to calculate a lower bound on the objective function, or
(b) to identify candidate solutions on the boundary.

This search implies that for each boundary constraint j,
(a) at least n computations are performed to find a lower bound, or
(b) it is checked whether each of the $O(n^2)$ pairs of clients can yield a local solution on j.

Hence, if B is defined by m constraints, there may be $O(mn^2)$ locations which have to be checked for feasibility. The search along B may require a total of $O(mn^2)$ computations. The methods presented in the previous sections seem to ignore this factor and merely give the complexity as a function of n.

However, if m is sufficiently large, say $m > n$, the search along B may become more costly than the search in the interior. In these cases, the complexity given in terms of n can be misleading and it does not make much sense to compare alternative methods on this basis.

Hence, it should be kept in mind that the number of clients is not the only critical aspect in location problems. The number m of boundary constraints may be equally important and if the complexity of any method is to be taken as a measure of its efficiency, it should be expressed in terms of m as well.

3.5 The Voronoi Diagram Approach

3.5.1 Definitions and Properties

The Voronoi diagram of a given set of points N is a well known geometric structure which contains proximity information about the members of N. It has been used very efficiently in a wide scope of applications ranging from physics to archaeology. Shamos (1975) and Shamos and Hoey (1975) introduce the Voronoi diagram to computational geometry and use it to solve a variety of closest-point problems in the Euclidean metric.

Given n points P_i on the plane (i=1,...,n), they define the **Voronoi polygon** associated with P_i , denoted by V_i , as the locus of points closer to P_i than to any other point.

In the previous chapter we defined the unweighted bisector B_{i-j} between two points P_i and P_j either in the Euclidean or in the rectilinear metric and stated that it divides the plane into two half-planes each containing one of the two points. Obviously, if h (i, j) is the half-plane containing P_i , then $V_i = \bigcap_{i \neq j} h(i,j)$, i.e. the Voronoi polygon is the intersection of all half-planes containing P_i . The entire set of these polygons, some of which may be unbounded, is referred to as the **Voronoi diagram** of the given set of points.

An edge shared by two Voronoi polygons V_i and V_j is called a **Voronoi edge** and is a portion of the bisector B_{i-j} . The intersection of two or more such edges is called a **Voronoi vertex** and is equidistant from at least three points.

Lee (1980) contains a very detailed discussion of the Voronoi diagram in the generalised p-metric where the distance d_p between two points $P_i (x_i , y_i)$ and $P_j (x_j , y_j)$ in that metric is:

$$d_p (P_i, P_j) = (| x_i - x_j |^p + | y_i - y_j |^p)^{1/p} \qquad \text{for } 1 \leq p < \infty$$

and

$d_\infty (P_i, P_j) = \max \{ | x_i - x_j | , | y_i - y_j | \}$

Clearly, the rectilinear and the Euclidean metric correspond to p=1 and p=2 respectively. See figure 3.5 for an example of the unweighted Voronoi diagram of five points in the rectilinear metric.

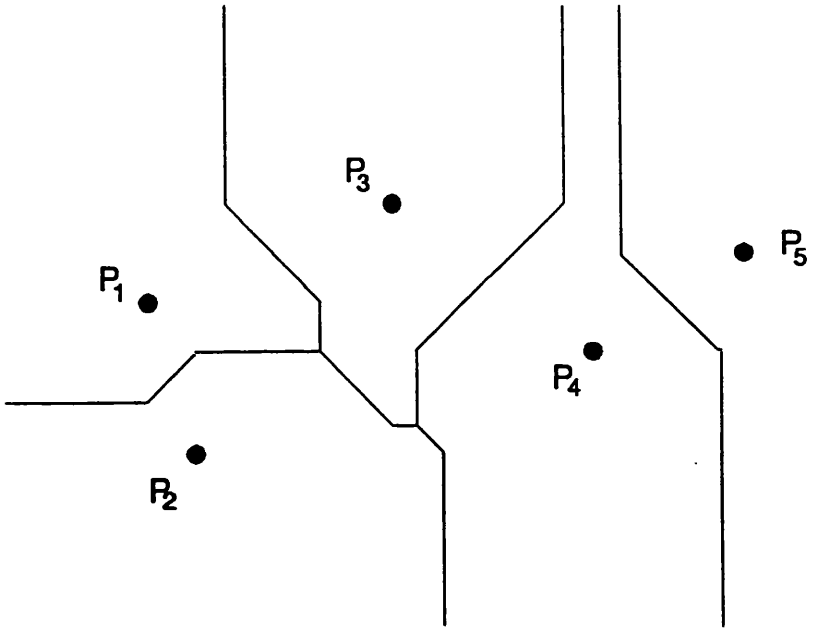

Figure 3.5: *Rectilinear unweighted Voronoi diagram with 5 points*

Lee states that, given n points on the plane, the number of Voronoi edges as well as the number of Voronoi vertices are linear functions of n. Furthermore, he presents an optimal divide-and-conquer algorithm for the construction of the Voronoi diagram with worst case complexity of O (nlogn) (see Lee (1980) for details).

3.5.2 The Unweighted Problem

Shamos (1975) and Shamos and Hoey (1975) address the Euclidean MAXIMIN problem when the feasible region S is taken to be the convex hull of the demand points P_i. They observe that the problem is equivalent to finding the largest circle centred in the interior of S and containing no demand points and that the centre

of that circle must lie at a Voronoi vertex or at the intersection of a Voronoi edge and the boundary of the convex hull of the P_i's. They then prove that linear time suffices to examine all candidate solutions and conclude that the overall complexity is O(nlogn) since the running time is dominated by the time required for the construction of the Voronoi diagram.

Dasarathy and White (1980) modify that algorithm so as to cater for the general case when the feasible region S is any convex polygon, described by k linear constraints. Based on the properties of the optimal solution and the definition of the Voronoi diagram, they confine the set of candidate solutions to the Voronoi vertices, the intersections of the Voronoi edges with the edges of S and the vertices of S. The O(n) Voronoi vertices can be generated in O(nlogn) time and checked for inclusion in S in O(nlogk) time using a construction presented in Shamos and Hoey (1975). The number of intersection points is a linear function of n since each Voronoi edge can intersect the boundary of S at most twice. Hence, these O(n) intersections can be generated in O(nlogk) time by a technique given in Dasarathy and White (1980). Finally, the vertices of S can be checked for optimality in $O(klog^2n + nlogn)$ time, as explained in Shamos (1975). Hence, the overall complexity of the algorithm is $O(klog^2n + nlogn + nlogk)$.

This algorithm can be applied to the unweighted rectilinear problem as well, since the non-diagonal segment of a bisector can still intersect the boundary at most twice and the set of candidate solutions is the same as in the Euclidean case.

3.5.3 The Weighted Problem

Although the Voronoi diagram can be applied very efficiently to the unweighted problem, it has several very unpleasant properties when each demand point is assigned a positive weight. In the previous chapter we illustrated that in the weighted case the bisector between two points is a circle, in the Euclidean metric, or a closed polygon in the rectilinear one.

Let us define the region of dominance of a demand point P_i as the set of points which are closer to P_i than to any other demand point, and denote it by reg_i. Since the weighted bisector in the Euclidean case is a circle, reg_i consists of circular edges and the weighted Voronoi diagram (WVD) is a subdivision of the plane with such edges. See figure 3.6 showing the Euclidean WVD of five points P_i with weights $w_1 = w_3 = 2$, $w_2 = 1$ and $w_4 = w_5 = 2$.

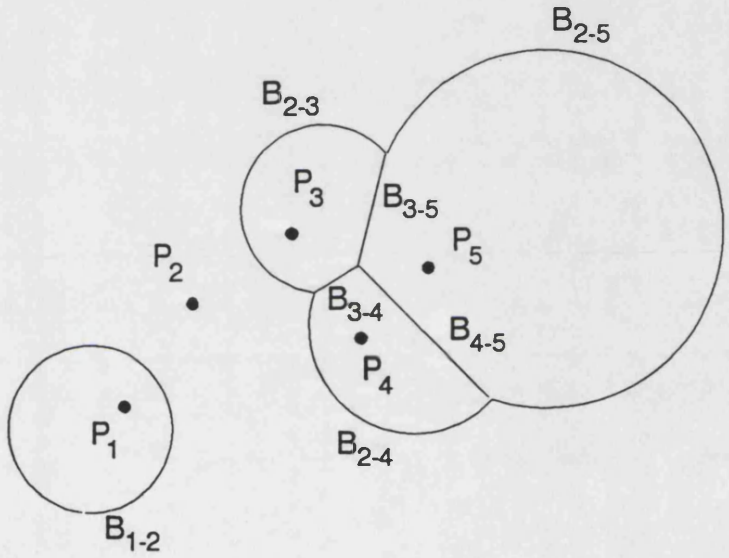

Figure 3.6: *Euclidean weighted Voronoi diagram with five points*

Aurenhammer and Edelsbrunner (1984) prove that the WVD of n points has at most $O(n^2)$ edges and at most $O(n^2)$ vertices. Since each circular Voronoi edge can intersect each boundary edge at most twice, there exist $O(mn^2)$ such intersections to be checked for optimality in addition to the $O(n^2)$ vertices. Hence, it seems that the WVD can provide an efficient solution technique for the weighted problem as well. Aurenhammer and Edelsbrunner outline a theoretical algorithm which constructs it in $O(n^2)$ time. However, they do not provide enough details on the implementation of their algorithm which seems to require extremely complex data structures. Hence,

as Angell and Moore (1986) conclude, a practical implementation of this algorithm is "out of the question".

As for the rectilinear problem, to the best of our knowledge there exists no polynomially bounded algorithm for the construction of the WVD. Hence, the results given in Dasarathy and White (1980) and Lee (1980) for the unweighted case cannot be directly utilised in the weighted version of the problem.

As a result, to the best of our knowledge, no attempt has been made to solve the weighted problem using Voronoi diagrams.

3.6 Summary

In this chapter we reviewed various alternative approaches to the MAXIMIN problem and outlined several existing solution methods for both the Euclidean and the rectilinear version. We observed that the complexity of these methods, given in terms of the number of clients, can often be misleading. Finally, we discussed the use of the Voronoi diagram to solve the unweighted problem efficiently and explained why it cannot yield equally efficient solutions to the weighted version.

CHAPTER FOUR

LINEAR PROGRAMMING BASED APPROACHES TO THE RECTILINEAR PROBLEM

4.1 Introduction

Most of the literature on the single facility MAXIMIN problem refers to the Euclidean distance metric. As evidenced by the review of the existing methods given in the previous chapter, the dominant procedure for this problem is the enumeration of local optima using the Kuhn-Tucker conditions. Although this analysis can be extended to the rectilinear problem as well, the dominant approach there seems to be one of dividing the feasible region into rectangular segments and solving a sequence of LPs.

In this chapter we will focus our attention on the closest point algorithm proposed by Mehrez et al. (1986) which seems to be the most efficient LP-based method to date. Section 4.2 discusses the theoretical significance of this algorithm and illustrates that although it is, in general, very efficient, it is not entirely correct.

Section 4.3 introduces PROFLAWLP, an alternative method based on the closest point algorithm, which does not require linear programming at all.

Finally, section 4.4 presents some computational results comparing the two techniques and showing that in all the test problems used PROFLAWLP was faster.

4.2 The Closest Point Algorithm

4.2.1 Description of the Algorithm for the Weighted Problem

In the previous chapter we gave a brief description of the closest point algorithm which is used in Mehrez et al. (1986) to solve the unweighted MAXIMIN problem. As explained below, we have slightly modified the method to cater for the weighted version as well.

The first step is to construct the grid of rectangles in exactly the same way as in the unweighted problem, namely by drawing one horizontal and one vertical line through each demand point. Since at most four clients are relevant for each rectangle k, the optimal solution for k is given by the following LP :

$$\max L \qquad \text{(P1)}$$

$$\text{s.t.} \quad w_1(x - x_1 + y_1 - y) \geq L \qquad \text{(1a)}$$

$$w_2(x_2 - x + y_2 - y) \geq L \qquad \text{(1b)}$$

$$w_3(x_3 - x + y - y_3) \geq L \qquad \text{(1c)}$$

$$w_4(x - x_4 + y - y_4) \geq L \qquad \text{(1d)}$$

$$a_j x + b_j y \leq c_j \qquad j=1,\ldots,m \qquad \text{(2)}$$

$$x_{min} \leq x \leq x_{max} \qquad \text{(3a)}$$

$$y_{min} \leq y \leq y_{max} \qquad \text{(3b)}$$

where $P_i(x_i, y_i)$ is the closest point corresponding to region R_i, constraints (2) are the m linear constraints defining the feasible region S and constraints (3) define the rectangle in question.

Mehrez et al. use the conditions developed in Drezner and Wesolowsky (1983) to eliminate all rectangles which are entirely outside the feasible region. The method seems to imply taking each constraint in turn and using it to cut off the part of each of the lines $x=x_{min}$, $x=x_{max}$, $y=y_{min}$ and $y=y_{max}$ which is outside the feasible region (see figure 4.1). A particular rectangle is infeasible if the feasible part of each of

these four lines is outside the rectangle or, equivalently, if all the conditions of section 3.3.3 are satisfied.

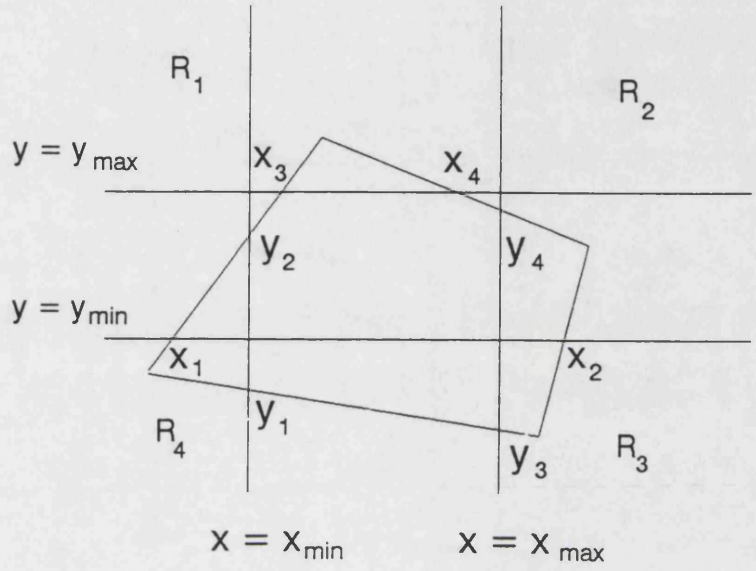

Figure 4.1: *Drezner and Wesolowsky's method for eliminating infeasible rectangles*

In the weighted version of the problem the upper bound UB_k for a particular rectangle k can be calculated as follows :

Case 1

Suppose that two neighbouring regions, say R_1 and R_4, are non empty and that $w_1 = 2$ and $w_4 = 1$. Consider the weighted bisector defined by P_1 and P_4 and the points where that bisector intersects the boundary of k and let M be the intersection whose weighted distance from both points is maximized, if such intersections exist (see figure 4.2). Let also C and B be the corners of the rectangle opposite P_1 and P_4 respectively. If $L = w_1 d_R(M, P_1) = w_4 d_R(M, P_4)$, $L_1 = w_1 d_R(C, P_1)$ and $L_4 = w_4 d_R(B, P_4)$ then $UB_k = \min \{ L, L_1, L_4 \}$.

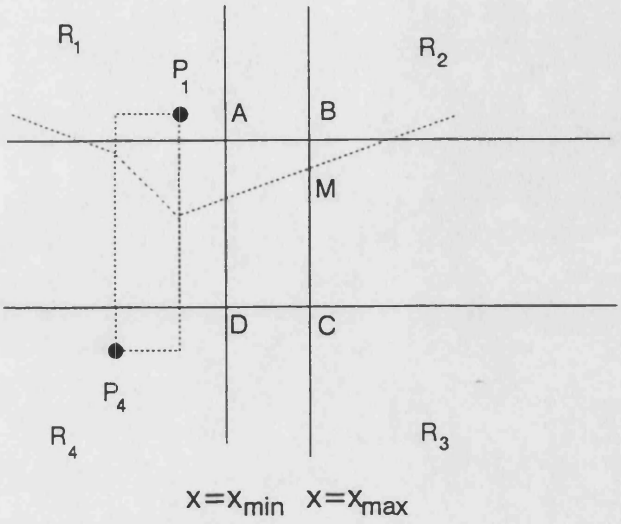

Figure 4.2: *Example illustrating case 1*

Case 2

In all other cases, if say region R_1 is not empty let L_1 be either

(a) the weighted half distance of P_1 from P_3 if R_3 is not empty, namely

$L_1 = d_R(P_1, P_3) * (w_1 w_3)/(w_1 + w_3)$, or

(b) its weighted distance from the opposite corner of k, namely point C in the figure, if R_3 is empty.

The remaining L_i's are calculated similarly. The upper bound in this case is $UB_k = \min \{ L_i \}$ for $i=1,\ldots,4$.

The upper bounds are then sorted in descending order and if the point producing the largest upper bound falls within the feasible part of its corresponding rectangle and satisfies the distance constraints of set (1), it is the global optimum. If not, an LP is solved to obtain the optimal location inside that rectangle. The process is repeated until the next largest upper bound is less than the best exact solution found so far.

4.2.2 Feasibility Checks

Obviously, each upper bound point must be checked for feasibility of constraint set (2) even when it is within its rectangle and an LP must be solved whenever the upper bound point is in the infeasible part of the rectangle. In the next section we will present a technique which exploits the structure of the grid of rectangles to identify the infeasible ones as well as the ones which are intersected by one or more linear constraints. In this way the feasibility check can be avoided for rectangles which are inside S.

In addition to this, each upper bound solution must be checked for feasibility of constraint set (1) as well, as proven in Appa and Giannikos (1993a). The following counter example, shown in figure 4.3, demonstrates a case where the application of the closest point algorithm, as given by Mehrez et al. would lead to the wrong solution.

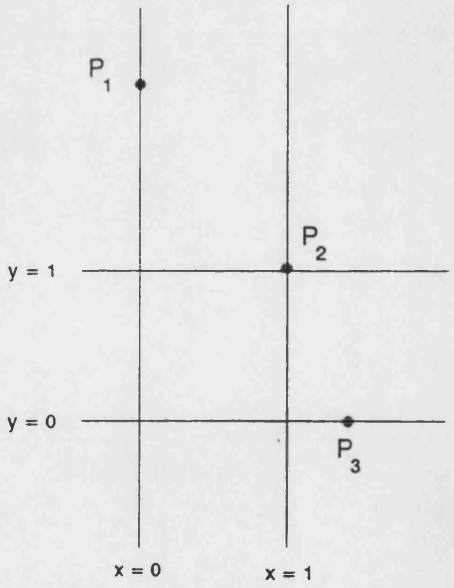

Figure 4.3: *Counter example*

Counter Example

Without loss of generality assume that a particular cell is defined by the lines $x=0$, $x=1$, $y=0$ and $y=1$ and that three of the regions around the cell are occupied

by the points P_1 (0, Y), P_2 (1, 1) and P_3 (X, 0) with X > 1 and Y > 1 (see figure 4.3). The optimal distance for this cell, according to Mehrez et al., is min {2,(X+Y)/2}; if X+Y > 4 Mehrez et al. indicate that the optimal location will be (0, 0) with value 2. However, if Y < 2 or X < 2, this will not be correct since point P_1 (0, Y) or P_3 (X, 0) respectively will be at distance less than 2 from (0, 0).

4.2.3 Characteristics of the Method

We described how the closest point algorithm can be modified to address the weighted single facility MAXIMIN problem in a feasible region defined by m linear constraints. The main idea behind the method is that at most four demand points are relevant for each rectangular area. As a result the size of the LP, when one is required, is reduced significantly, especially in large problems. Moreover, the simple form of constraint set (1) of (P1) simplifies the calculation of the upper bound for each rectangle. In fact the upper bounds, as calculated in Mehrez et al. (1986), are tighter than the ones originally suggested by Drezner and Wesolowsky (1983) and can lead to more efficient solutions.

In addition to this, fewer LP's might have to be solved since each upper bound point is checked for inclusion in the corresponding rectangle, unlike the Drezner-Wesolowsky algorithm where an LP is solved instead.

Apart from the inclusion test, each upper bound point is checked for feasibility of all m linear constraints, although none or only a few of them might be relevant for the rectangle in question. In the following section we will present an alternative technique, that identifies which constraints, if any, intersect each rectangle and checks the upper bound point for feasibility of these constraints only.

The first step of the closest point algorithm is the elimination of all infeasible rectangles which implies considering each constraint with each horizontal or vertical line passing through each client. The alternative technique referred to in the previous paragraph exploits the structure of the grid of rectangles to identify infeasible

rectangles and implies considering each constraint with only the horizontal lines it intersects.

Finally, the properties of the optimal solution, as stated in chapter 2, are not used at all. As a result, upper bounds are calculated even for rectangles which cannot yield local optima e.g. ones that are entirely inside the feasible region and have only two non empty regions.

Hence, although the closest point algorithm is a significant improvement of the original Drezner-Wesolowsky algorithm, there are still ways it can be further improved.

4.3 An Alternative Method Based on the Closest Point Algorithm

4.3.1 Aspects of the Method

In the previous section we outlined the characteristics of the closest point algorithm and discussed aspects of the method that could be improved. By combining the main ideas of this algorithm and the properties of all local optima, as stated in Melachrinoudis (1988) and Appa and Giannikos (1992), we develop a Properties-based Rectilinear Obnoxious Facility Location Algorithm Without Linear Programming (PROFLAWLP). As mentioned in Appa and Giannikos (1993b), PROFLAWLP has the following features :

1. It identifies rectangles which cannot contain a local solution satisfying the properties mentioned above and eliminates them from the subsequent calculations.
2. It exploits the structure of the grid of rectangles to identify the infeasible ones and, at the same time, mark the constraints intersecting each rectangle.
3. It uses the properties of local solutions to solve (P1) for each rectangle without using linear programming.

More specifically, as stated in chapter 2, a local solution to problem (P1) satisfies the following properties :

1. At a local solution on a vertex of the feasible region S, at least one constraint from set (1) of (P1) is binding.
2. A local solution on the boundary of S (but not on a vertex) is equidistant from at least two demand points.
3. A local solution in the interior of S occurs along a multiple solution edge which is equidistant from two demand points. However, there exists at least one point on such an edge corresponding to a basic solution of (P1) which is equidistant from three demand points.

Combining these properties with the fact that at most four points are relevant for each rectangle, we can reduce the work required by the closest point algorithm in two ways. Firstly, we can eliminate all rectangles in the interior of S which cannot contain a point equidistant from three clients. Secondly, we can derive algebraic rules for finding the optimal location in all other rectangles, so that the global optimum can be found without using linear programming.

4.3.2 Finding Local Solutions

Definition 4.1

A rectangle is called **infeasible** if it is entirely outside S, **partly feasible** if some points of the rectangle are on the boundary of S, and **feasible** otherwise. Note that feasible rectangles are always in the interior of S, as shown in figure 4.4. Also note that there may exist partly feasible rectangles where all points are feasible; these are rectangles where a boundary edge coincides with a side of the rectangle.

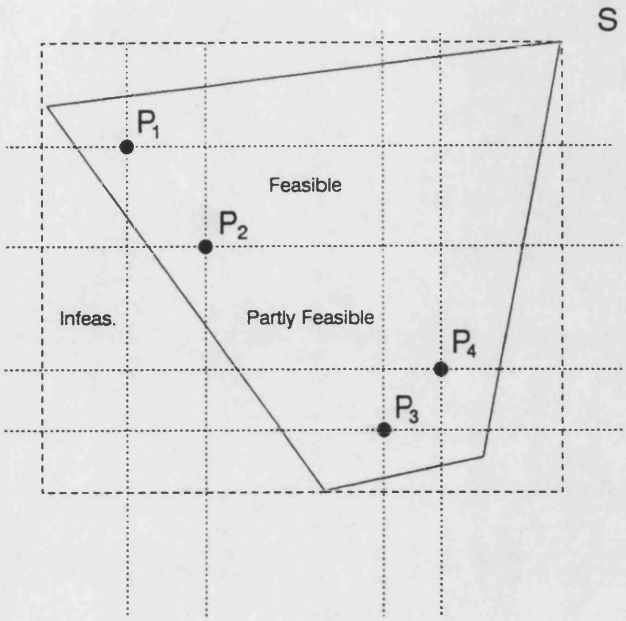

Figure 4.4: *Feasible, Infeasible and Partly feasible rectangles*

4.3.2.1 Eliminating Some Feasible Rectangles

Since local solutions in the interior of S correspond to basic solutions of (P1) where three constraints from set (1) are effective, we can discard feasible rectangles with one or two non empty regions without even calculating their upper bound. As a result the number of upper bound values that have to be stored and then sorted is less than the total number of rectangles, namely $(n+1)^2$.

4.3.2.2 Finding Solutions in Feasible Rectangles

For simplicity of the subsequent calculations we will assume that $w_i = 1$ for $i=1,\ldots,n$. The following results can be extended to the weighted problem without any theoretical difficulty.

Case 1: Three non empty regions.

Suppose that regions R_1, R_2 and R_3 are non empty. A local solution (x, y) at distance L from P_1, P_2 and P_3 must satisfy the following system of equations :

$$(x - x_1) + (y_1 - y) = L$$
$$(x_2 - x) + (y_2 - y) = L$$
$$(x_3 - x) + (y - y_3) = L$$

Solving this 3x3 system with respect to x, y and L we get the coordinates of the location which is equidistant from P_1, P_2 and P_3. If this location is inside the rectangle, it defines a local solution.

The coordinates of candidate local solutions in all possible cases are given in the following table.

Table 4.1: Possible Locations in Feasible Rectangles

	NON-EMPTY REGIONS	2 * x	2 * y	2 * L
I.1	R_1,R_2,R_3	$x_1-y_1+x_2+y_2$	$x_2+y_2-x_3+y_3$	$y_1-x_1+x_3-y_3$
I.2	R_1,R_2,R_4	$x_1-y_1+x_2+y_2$	$y_1-x_1+x_4+y_4$	$x_2+y_2-x_4-y_4$
I.3	R_1,R_3,R_4	$x_3-y_3+x_4+y_4$	$y_1-x_1+x_4+y_4$	$y_1-x_1+x_3-y_3$
I.4	R_2,R_3,R_4	$x_3-y_3+x_4+y_4$	$x_2+y_2-x_3+y_3$	$x_2+y_2-x_4-y_4$

If $x_{min} \leq x \leq x_{max}$ and $y_{min} \leq y \leq y_{max}$ (I.5)

then the best location in the rectangle is (x, y) at distance L from all three points. If not, there is no local solution in the rectangle.

Case 2: Four non empty regions.

For a feasible rectangle k with all regions R_i non empty, the optimal distance is :

$$L = \min \{ L_{1\text{-}3}, L_{2\text{-}4} \}$$

where $L_{1\text{-}3} = (-x_1 + y_1 + x_3 - y_3)/2$

with x and y coordinates from I.1 or I.3, and

$$L_{2\text{-}4} = (x_2 + y_2 - x_4 - y_4)/2$$

with x and y coordinates from I.2 or I.4.

If $L = L_{1\text{-}3} \leq L_{2\text{-}4}$, we check if the location equidistant from P_1, P_2 and P_3 or from P_1, P_3 and P_4 is within k. More simply, we check whether the coordinates given by either I.1 or I.3 (table 4.1) satisfy I.5. If so, this is the optimal location for k. Note that if the optimum is defined by, say P_1, P_2 and P_3 then the distance of this location to the remaining demand point, viz. P_4, will be at least L since $L_{1\text{-}3} \leq L_{2\text{-}4}$. Similarly, if $L_{1\text{-}3} > L_{2\text{-}4}$ we check whether the coordinates given by I.2 or I.4 satisfy I.5, thus defining the optimal solution for k.

4.3.2.3 Finding Solutions in Partly Feasible Rectangles

The properties of local optima indicate that in a partly feasible rectangle k we can have three types of local solution: solutions at a vertex of S, solutions on a boundary edge of S or solutions in the interior of $(S \cap k)$. Note that there can be more than one type of local solution in a partly feasible rectangle k. Rules 1 to 3 below check for the existence of all three types of local optima in k.

Rule 1: If there is more than one constraint passing through k, we evaluate the objective function at each of the corresponding vertices of S, i.e. we calculate the distance of each vertex to the nearest client and store the largest of these values.

Rule 2: To cater for solutions on the boundary of S, we consider each constraint which intersects k with all possible pairs of closest points corresponding to k. Note that there are at most three such pairs for each constraint. For each such pair (P_u, P_v) and each constraint H: $ax + by \leq c$, we check whether there exists a local solution on H equidistant from both P_u and P_v which is inside k. If u=2 and v=4, such a solution must satisfy the following system of equations:

$$\begin{aligned} w_2 (x_2 - x + y_2 - y) &= L \\ w_4 (x - x_4 + y_4 - y) &= L \\ a * x + b * y &= c \end{aligned}$$

If there is more than one such intersection, we choose the one with the largest value for L.

Rule 3: If there are three non empty regions around k, we use the results of table 4.1 to check whether the corresponding closest points define a local solution inside k, equidistant from all three of them.

The location with the largest objective function from rules 1 to 3 is the optimal solution for rectangle k. However, in order to calculate the optimal solution for any partly feasible rectangle k we need to know which constraints from set (2) of (P1) intersect k. In appendix B we give a representation method for S and an algorithm to identify the infeasible and partly feasible rectangles. The main idea behind this method is to take each constraint in turn and mark which rectangles it intersects. As a result, for each partly feasible rectangle k we have a list of the relevant constraints from set (2). Hence, when checking each candidate solution for feasibility, we only consider the constraints that are relevant for the corresponding rectangle rather than all m constraints which define S.

4.3.3 Description of the Algorithm

Assuming that the feasible region S is defined by a set of vertices V_1 to V_m given in clockwise order, PROFLAWLP can be outlined as follows :

STEP 1 Data Preparation

1.1 Construct the grid of rectangles.

1.2 Mark each rectangle as feasible to start with. For each constraint V_j to V_{j+1} mark the rectangles intersected by that edge as partly feasible and the ones on the infeasible side of $V_j \rightarrow V_{j+1}$ as infeasible. (See appendix B for details). At the end of the process for each rectangle k we have its status (Feasible, Partly Feasible or Infeasible) and if Partly Feasible a list of consecutive edges intersecting k.

STEP 2 Finding Upper Bounds

For each rectangle k in turn :

(i) if k is infeasible, discard it;

(ii) if not, check if it is feasible and has at most two non empty regions; if so, discard it;

(iii) in all other cases, calculate the upper bound UB_k using the rules suggested by Mehrez et al. and store it in the array UArray of upper bounds.

STEP 3 Sort

Sort the elements of UArray in descending order.

STEP 4 Finding Local Solutions

If the point producing the largest upper bound falls within its rectangle, it is the global optimum. If not, find the optimal solution for that rectangle as explained in section 4.3.2.

STEP 5 Termination Rule

Repeat Step 4 for rectangles with progressively smaller upper bounds. Stop when the next largest upper bound is less than the best exact solution found so far.

PROFLAWLP tests each rectangle for the existence of a local solution and discards the rectangle if it does not contain one, whereas other LP-based methods solve an LP instead. The following example illustrates the algorithm.

Example 4.1

Consider five demand points within a convex polygon S bounded by five vertices (see figure 4.5).

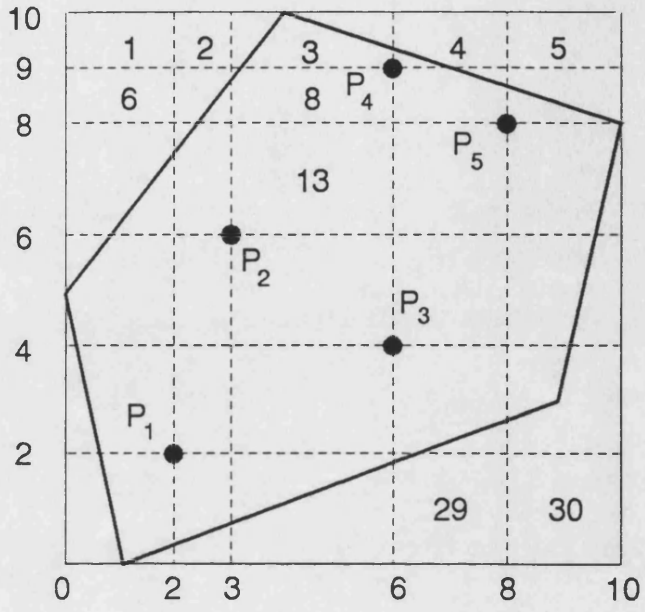

Figure 4.5: *Demand points and feasible region for example 4.1*

The x and y-coordinates of the demand points as well as the vertices of S are given in the following tables :

Table 4.2i: Demand Points for Example 4.1

Point P_i	1	2	3	4	5
x_i	2	3	6	6	8
y_i	2	6	4	9	8

and

Table 4.2ii: Vertices of S for Example 4.1

Vert. V_j	1	2	3	4	5
x_j	0	4	10	9	1
y_j	5	10	8	3	0

The upper bound as well as the status (Feasible, Partly Feasible or Infeasible) for each rectangle are given in table 4.3. (See appendix B for details). Note that 6 rectangles have been discarded either because they are infeasible (rectangles 1, 2, 5, 6 and 30) or because they are feasible but have only two non-empty regions (rectangle 22).

Since the upper bound point for rectangle 25 (which has the largest upper bound) is infeasible, i.e. not within the feasible region, we calculate the best local solution within that rectangle as explained in section 4.3.2. The optimal location satisfying the necessary properties is point (9, 3) at distance $L_{25}=4$ from its nearest point. Since L_{25} is less than the next largest upper bound, i.e. 6 for rectangle 29, we consider that rectangle next. It turns out that there is no local solution inside rectangle 29 satisfying any of the known properties, hence we continue with rectangle 11 whose upper bound point is also infeasible. This rectangle does not contain any appropriate local solution either, so we check rectangle 20 next. The upper bound point for this rectangle is also infeasible; however, there exists a local solution at (9.4, 5) at distance $L_{20}=4.4$ from its nearest demand points. In fact this location is the best exact solution found so far. Similarly, the optimal solution for rectangle 28 is $L_{28}=3.5$ at (5, 1.5) and for rectangle 16 $L_{16}=4.4$ at (0.1, 4.5). Since the next largest upper bound, namely 4 for rectangle 7, is not better than the best current solution i.e. 4.4 the process is terminated and the global optimum is point (9.4, 5) (or (0.1, 4.5)) at distance L=4.4.

Table 4.3: Upper Bound and Status for each Rectangle

Rect i	Status	Up. Bound	Up. Bound Point
25	PART. FEASIBLE	6	(10, 2)
29	PART. FEASIBLE	6	(8, 0)
11	PART. FEASIBLE	5	(0, 8)
20	PART. FEASIBLE	5	(10, 5)
28	PART. FEASIBLE	5	(5, 0)
16	PART. FEASIBLE	4.5	(0, 4.5)
7	PART. FEASIBLE	4	(2, 9)
3	PART. FEASIBLE	4	(3, 10)
15	PART. FEASIBLE	4	(10, 6)
24	PART. FEASIBLE	4	(8, 2)
26	PART. FEASIBLE	4	(0, 0)
21	PART. FEASIBLE	4	(0, 4)
12	PART. FEASIBLE	3	(2, 8)
13	FEASIBLE	3	(4.5, 7.5)
14	FEASIBLE	3	(7, 6)
19	FEASIBLE	3	(7, 6)
23	FEASIBLE	3	(4, 3)
10	PART. FEASIBLE	3	(10, 9)
8	PART. FEASIBLE	3	(4.5, 7.5)
27	PART. FEASIBLE	3	(3, 0)
4	PART. FEASIBLE	2.5	(7.5, 10)
17	FEASIBLE	2.5	(2.5, 4)
18	FEASIBLE	2.5	(4.5, 5)
9	PART. FEASIBLE	1.5	(7, 8.5)
1	INFEASIBLE	-	-
22	FEASIBLE	-	-
6	INFEASIBLE	-	-
5	INFEASIBLE	-	-
2	INFEASIBLE	-	-
30	INFEASIBLE	-	-

4.4 Computational Results

A series of test problems were randomly generated to assess the relative performance of PROFLAWLP and the correct version of the closest point algorithm. For each combination of n and m, 5 random problems were generated and the results were averaged over these 5 problems. The problems were tested on a PS\2 with an 80386 processor. Some of the average execution times are given in table 4.4. In all cases PROFLAWLP was significantly faster than the closest point algorithm.

Table 4.4: A Comparison of Computation Times in Seconds

Dem. Points	Constraints	PROFLAWLP	Closest point algorithm
20	6	2.77	4.77
40	6	6.03	8.97
60	6	9.96	16.47
80	6	11.77	18.27
100	6	15.35	22.17
20	15	3.53	4.90
40	15	6.50	9.90
60	15	8.80	18.67
80	15	9.26	19.93
100	15	11.50	23.26
20	50	3.38	11.25
40	50	5.68	16.30
60	50	7.10	19.54
80	50	8.63	22.31
100	50	9.90	26.65

The n demand points were generated by a uniform probability distribution on [0, 10] on each coordinate. The m constraints were constructed as a convex polygon circumscribed around a circle of given radius r centred at (5, 5). For example, with five constraints and r=5 the feasible region would be a pentagon with coordinates (10, 8.63), (3.09, 10.88), (-1.18, 5), (3.09, -0.88) and (10, 1.37). The weights were also randomly generated by a uniform distribution on [1, 5].

The description of both algorithms given in the previous sections explains the significant reduction in computation time achieved by PROFLAWLP. Firstly, the closest point algorithm identifies and eliminates the infeasible rectangles by considering each rectangle k and then using each constraint j to cut off the infeasible parts of the boundary of k. In other words, in the worst case, each of the m constraints is considered for each of the $(n+1)^2$ rectangles. On the other hand, PROFLAWLP takes each constraint j in turn, checks which particular rectangles it intersects and exploits the structure of the grid to identify the infeasible rectangles. As explained in appendix B, for each j at most n checks have to be made, thus resulting in a significant reduction in computation time.

Secondly, PROFLAWLP uses the properties of local optima to eliminate feasible rectangles which cannot contain proper local solutions. As a result, fewer upper bounds have to be stored and then sorted and computation time is saved, especially as the problem size increases.

Thirdly, whenever the upper bound point corresponding to a particular rectangle k does not fall within the feasible part of k, the closest point algorithm performs an LP. On the other hand, PROFLAWLP performs at most six simple calculations to find the optimum for k, as explained in section 4.3. Mehrez et al. claim that their method hardly requires LP. When the feasible region S is rectangular it is indeed difficult to find cases where LP would be performed. However, when S has five or more vertices LP is almost always necessary as indicated by table 4.5 which presents the average number of LP's required for various problem sizes.

Table 4.5: Average Number of LP's for the Closest Point Algorithm

Dem. Points	Constraints	No. of LP's
20	5	3.7
40	5	6.3
60	5	17.3
80	5	25.8
100	5	32.6

Hence, the execution time is reduced considerably since PROFLAWLP does not require linear programming at all. Moreover, in order to find the optimum for a partly feasible rectangle k, PROFLAWLP only considers the constraints intersecting k, as given by the corresponding list of edges referred to in the description of the algorithm. On the other hand, the closest point method solves an LP for k using all m constraints.

One could argue that although PROFLAWLP is significantly faster than the closest point method, it requires much more memory since it needs to store the constraints which intersect each rectangle k. However, in most problems many of these lists are empty anyway since the corresponding rectangles are feasible. Furthermore, the upper bounds stored by PROFLAWLP are, in general, fewer than the ones required by the closest point method since the rectangles which do not contain proper local solutions are discarded.

4.5 Summary

In this chapter we described how the closest point algorithm, as introduced by Mehrez et al., can be modified to solve the weighted version of the single facility MAXIMIN problem. After illustrating that it constitutes an efficient improvement

of the original Drezner and Wesolowsky algorithm, we pointed out some logical shortcomings of the method as well as ways of further improvement. We also established an alternative solution method, based on the same theoretical principles, which exploits the properties of local optima and does not require linear programming at all. Finally, we presented some computational results indicating that this alternative approach is significantly faster than the closest point method.

CHAPTER FIVE

AN INTERACTIVE GRAPHICAL APPROACH

5.1 Introduction

Most of the models discussed in the previous three chapters assume that the feasible region is a convex polygon. However, in real life problems the permissible area, say a city or a county, is usually a union of non convex and disjoint areas; it may even have non permissible regions such as rivers or parks where the undesirable facility cannot be located. Hence, the assumptions of convexity and connectivity are quite restrictive when it comes to realistic situations.

Consequently, several researchers have introduced graphical models with general feasible regions that can cater for real life problems. These models can be used to obtain approximate solutions to problems which cannot be solved by existing analytical methods. However, these approaches require extensive user intervention and have only been applied to small scale problems with fewer than 10 demand points.

We present an interactive graphical method, based on existing approaches, that can solve realistic problems with up to 1000 demand points in reasonable time. This method allows for totally flexible feasible regions that can even contain non permissible subregions. It requires minimal user intervention and can easily be modified to find the exact solution to the problem whereas previous techniques find only approximate solutions. It can even produce a list of candidate solutions which can then be assessed using various criteria.

Section 5.2 briefly discusses previous graphical approaches and introduces an enhanced graphical model which reflects more aspects of the real world than any analytical model reviewed so far. This model has been implemented on a microcomputer as an interactive process.

Section 5.3 gives a detailed description of LOCOBNOX, the software implementing the above model, and explains the options available to the user at each stage of the interactive process.

Section 5.4 introduces a stochastic termination rule whose objective is to minimize user intervention in the first steps of the iterative process.

Section 5.5 presents some computational results regarding the performance of the termination rule as well as the overall performance of LOCOBNOX. These results indicate that the stochastic termination rule is effective even in large instances of the problem.

Section 5.6 presents a parametric version of the single facility problem where all the weights are raised by a parameter q. The speed and efficiency of the graphical approach allows us to solve the problem for different values of q and study the effect of the parameter on the optimal solution. This parametric investigation often reveals interesting information about the structure of the problem since in some examples the solution is quite sensitive to changes in the value of q while in others it is not affected at all.

Finally, section 5.7 discusses the advantages of the graphical method in comparison to the analytical approaches. It also suggests ways of enhancing the method so that it may be used in real life applications effectively.

5.2 Description of the Graphical Model

The rapid developments in computer software and hardware and especially the advent of computer graphics has enabled several researchers to use geometric insight and develop graphical solution methods to location problems. The first interactive graphical procedure in the location literature is by Brady and Rosenthal (1980) who solve the weighted Euclidean MINIMAX problem for a single facility on an arbitrary feasible region. A typical application of this problem is the location of a radio

receiver to monitor a given set of transmitters. The main idea behind the method is to draw circles of radius r around each transmitter and visually inspect whether there exist feasible locations within reach of all transmitters, i.e. within all these circles. If so, the value of r can be reduced until this condition is violated. More simply, the optimal distance is the smallest value of r for which the intersection of all circles is not empty.

In the literature of the MAXIMIN problem there are two major attempts to solve the problem graphically. Melachrinoudis (1985) outlines a graphical technique which involves drawing circles of increasing radii around the demand points and selecting the final point not covered by any circle. However, he reports that the method is not applicable to problems with more than 10 demand points due to the rapid increase in computation time.

As mentioned in chapter 3, the *Black and White* algorithm by Hansen, Peeters and Thisse (1981) is the most general approach to the problem. Assuming that a cost function D_i is associated with each demand point P_i, the objective is to minimize the maximum of these costs. However, as explained in chapter 3, the method assumes that the feasible region is the union of a finite number of convex polygons, requires significant user intervention and is not easy to implement on a computer as an automated process.

The method we will outline in this section was originally suggested to us by Professor Ailsa Land and does not make any assumption regarding the feasible region. Given any permissible region S and n demand points P_i for $i=1,\ldots,n$ the objective is to locate a new undesirable facility at X, so that its distance to the nearest demand point is maximized. In other words the optimal location is the solution to the following problem :

$$\max \; L \qquad\qquad (P1)$$

$$\text{s.t.} \quad w_i d_i(P_i, X) \geq L \qquad i=1,\ldots,n$$

$$X \in (S - N)$$

where :

- S is the union of m feasible polygons S_j, not necessarily convex, each represented by a list of vertices given in clockwise order

- N is the union of q non permissible areas N_k, assumed to be bounded polygons where the facility cannot be located; each N_k is also given as a list of vertices in clockwise order

- $d_i(P_i, X)$ is the distance of demand point i to the new facility, either in the Euclidean or in the rectilinear metric

- w_i is the positive weight associated with demand point i; note that the smaller the weight the more important the corresponding demand point.

The method is simply based on the fact that the locus of points which are at weighted distance less than or equal to L from a given point P is :

(a) the circle with centre P and radius L/w_i in the Euclidean case, or

(b) the diamond with centre P and semi-diagonal equal to L/w_i, in the rectilinear case (see figure 5.1).

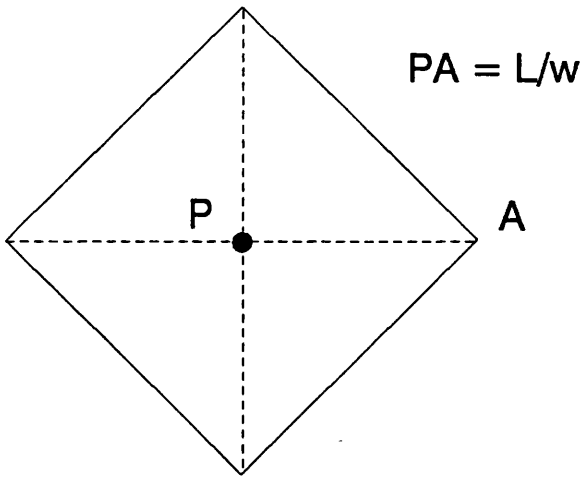

Figure 5.1: *Locus of points at rectilinear distance L from P*

Clearly, for any value L_0 of the objective function of (P1) the feasible points, if any, must be outside shapes (circles or diamonds depending on the distance metric)

having centres P_i and sizes L_0/w_i. The existence of such points inside S and outside the non-permissible regions N_k indicates that the objective function may still be increased. Hence, starting from an initial size L_0 we can go on increasing the size of these shapes by a predetermined quantity ΔL until the feasible area not covered by them is sufficiently small to be considered a point. This area is a close approximation of the optimal location.

During the early stages of development it was discovered that the overall performance of the algorithm is highly dependent on the initial size of the shapes. If the initial size is poorly chosen it may take a great number of iterations before the optimal location is found. Two relatively simple heuristics are used to obtain a good initial size and speed up the whole process.

Firstly, let L_V be the value of the objective function at vertex V of S i.e. the distance of V to its nearest demand point. Similarly, let L_W be the value of the objective function at vertex W of N. Clearly, the maximum L_1 of all the L_V's and the L_W's can be used as a lower bound since we will need shapes of size at least L_1 to cover all the vertices of S and all vertices of N.

Secondly, supposing the feasible area is covered by non-overlapping shapes, the area of each shape can be calculated simply by dividing the total area by n. Knowing its area, the size of each shape can be easily computed and this estimate gives a second lower bound L_2. The total area is given by the following formula:

$$Total\ Area = \sum_j AS_j - \sum_k AN_k$$

where AS_j is the area of the j-th feasible polygon and AN_k the area of the k-th non-permissible one. Supposing that AS_j has m_j vertices with coordinates (x_u, y_u) for $u=1,\ldots,m_j$ its area is given by

$$AS_j = \frac{1}{2} \left| \sum_{u=0}^{m_j-1} x_u (y_{u+1} - y_{u-1}) \right|$$

where indices are taken modulo m_j, as proven in Shamos (1975). The area of AN_k is calculated similarly.

The largest of L_1 and L_2 is used as the initial size of the shapes. After each iteration the user may terminate the process if he/she is satisfied with the current solution or continue by increasing the size of the diamonds or circles. The process stops when the user feels that the feasible area(s) still uncovered are sufficiently small.

Apart from its obvious simplicity the main advantage of the method is its flexibility and the ability to represent more aspects of the real world. In fact it is not limited to any distance metric. For example, one might wish to use the Chebyshev distance metric where $d_C = \max \{ |x_i - x_j|, |y_i - y_j| \}$ is the Chebyshev distance between points P_i and P_j. It can be seen that in this case instead of circles or diamonds one would draw squares with centre P_i and sides $2L/w_i$. The method has been implemented as an interactive graphical process on a microcomputer. A detailed description of the software is given in the following section.

5.3 LOCOBNOX: An Interactive Graphical Optimization Procedure

The model presented in the previous section was originally implemented using Borland's Turbo Pascal Graphics Toolbox, a graphics environment accompanied by a library of predefined routines. Quite surprisingly, polygon or curve filling was not one of them. Consequently, we had to write our own area filling routines, which were obviously much slower than the predefined ones. Since speed was a major concern we decided to develop our own graphics environment from scratch and exploit the predefined filling routines of Turbo Pascal which, unfortunately, were not accessible through the Toolbox.

The resulting software, called LOCOBNOX, runs under DOS version 3.2 or higher on an IBM or compatible microcomputer and produces an approximate solution interactively. It can solve problems with up to 1000 demand points, 500 boundary constraints and 20 non-permissible areas. A brief outline of the program is given in the flow diagram of figure 5.2.

Input consists of the coordinates of the demand points and the details regarding the feasible region which may be the union of up to 20 disjoint areas, each approximated by a closed polygon. The user may supply the coordinates of the vertices of these polygons or use the mouse to draw them interactively on the screen. Similarly, he/she may represent each of the non-permissible areas, if any. These forbidden areas are filled beforehand and are thus eliminated as candidates for the optimal location.

The distance metric can be Euclidean, rectilinear or Chebyshev depending on the particular application. The user is then asked to supply the weights corresponding to the existing facilities or set them all equal to one if he/she wishes to solve the unweighted version of the problem. Finally, he/she has to select a step size ΔL by which the objective function is to be increased if there are still uncovered feasible points. Clearly, in the weighted case the size of the shape (circle or diamond) corresponding to demand point P_i should increase by $\Delta L/w_i$ after each iteration.

Given the description of the feasible region and the coordinates of the demand points the program calculates a lower bound on the objective function using the heuristics of the previous section. The user may then keep this bound as the initial size L_0 of the shapes or change it if he/she wishes to. Table 5.1 presents some experimental results showing that for most randomly generated problems L_0 was on average at least 40% of the approximate optimal distance L^*.

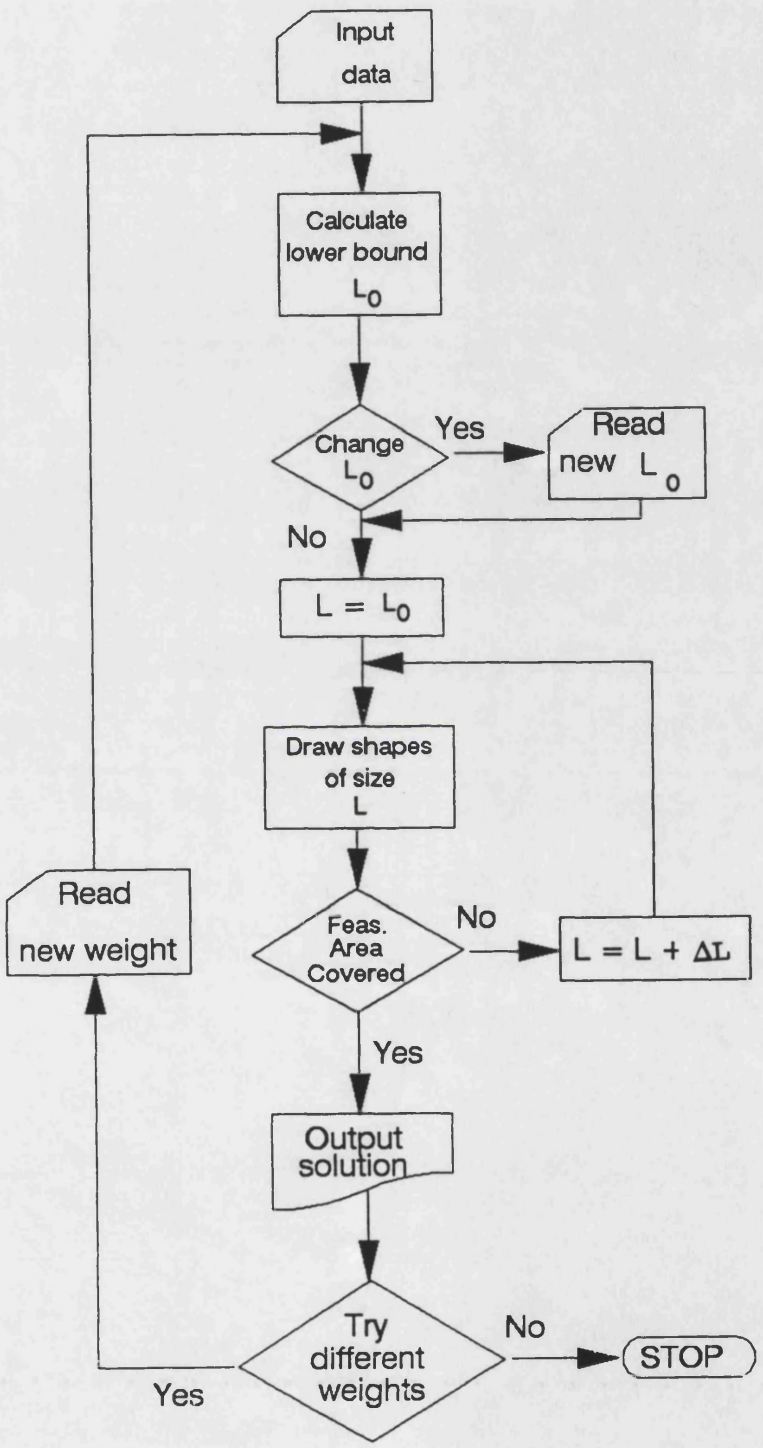

Figure 5.2: *Flow diagram for LOCOBNOX*

Table 5.1: L_0/L^* (%)

Number of dem. points	Number of Vertices of S 10	50	100
20	86	91	92
60	70	82	82
100	58	72	84
200	61	74	71
300	70	69	41
400	45	41	44
500	48	54	32

The next step is the implementation of the method itself; shapes of size L_0 are drawn around each demand point. The user is then asked whether he/she is satisfied with the current solution. If so, he/she may use the mouse to click the uncovered area(s) and obtain an approximate solution. If not, he/she can increase the objective function until most of the feasible area is covered. At the end of the program, the user may choose to solve the same problem using different weights in order to study the effect of the new weight set on the optimal location. Figures 5.3 to 5.5 demonstrate a small realistic example.

Example 5.1

Suppose that we wish to locate an undesirable facility, such as a nuclear factory, somewhere in the British Isles. The facility must be as far away as possible, in the Euclidean sense, from twenty major population centres, represented by black

dots in figure 5.3. In addition to this it cannot be located in any of six forbidden regions (mountains and lakes) represented by hatched areas in the same figure.

Figure 5.3: *Example with 20 population centres and 6 non permissible areas*

If all population centres are considered equally important the optimal location turns out to be in the north of Scotland, as shown in figure 5.4. However, if each population centre is assigned a weight relative to its size then the optimal area is the south-western corner of Cornwall (see figure 5.5).

A brief description of the program is given in appendix C. The program itself accompanied by several test problems is available in the final appendix of this thesis.

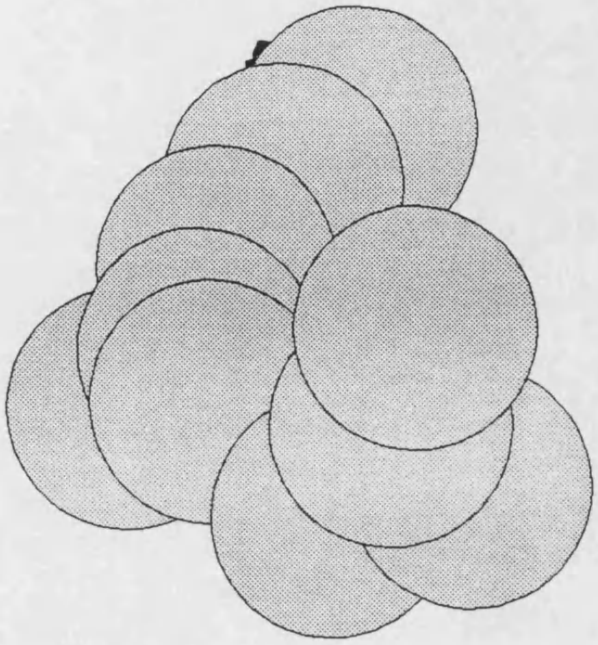

Figure 5.4: *Solution when all weights are equal*

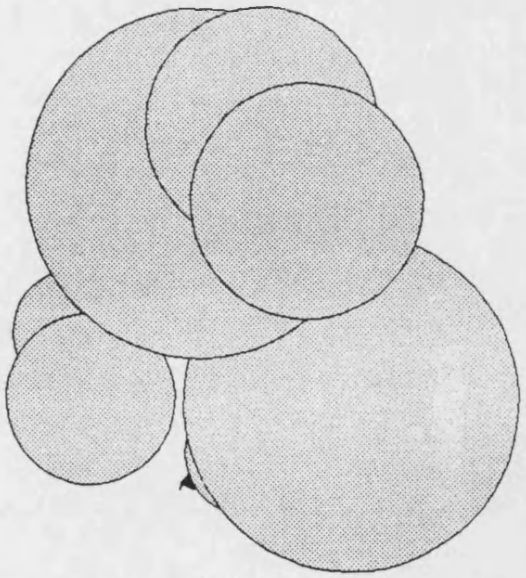

Figure 5.5: *Solution when weights are proportionate to the population of each centre*

5.4 A Stochastic Termination Rule

A comparison of the graphical approaches to the single facility MAXIMIN problem reveals that they all have the same theoretical principle: they are based on the sequential construction of iso-cost curves of value L_t and the elimination of all locations which are within these curves. The *Black and White* algorithm by Hansen et al. (1981) selects several feasible locations, calculates the objective function at each of them, sets L_t equal to the largest of these values, and draws the iso-cost curves of value L_t thus eliminating the locations inside them. On the other hand, LOCOBNOX, as well as similar methods, try increasing values L_t of the objective function and check whether there are still feasible locations outside the corresponding iso-cost curves.

Both approaches rely on the human user to identify whether the termination criterion has been satisfied after each iteration of the process. However, in the early stages it is rather tedious for the user to be asked whether he\she wishes to continue the process, when large parts of the feasible region are still uncovered by the iso-cost curves. Hence, we developed a stochastic termination rule to minimize user intervention and speed up the whole procedure.

The rule is very simple but effective in the two dimensional space which is relevant for most location problems anyway. It makes use of the predefined **GetColor** function in Turbo Pascal which returns the colour of a specified point on the screen. Given that the iso-cost curves (diamonds or circles) in our case are filled in a predetermined colour, we can use the above function to check whether a randomly chosen point is covered or not. Repeating the test for a sufficiently large number of points, say 100, we can thus get an estimate of the proportion of the feasible area which is already covered.

If this estimate exceeds a certain limit, say 95%, we repeat the test 10 times to reduce the probability of sampling error. If for each of the 10 random tests the

proportion of the area that is covered is still more than 95%, control is passed on to the user who can decide whether the process should be continued. At this stage the user can increase as well as decrease the value of the objective function until he/she is satisfied with the solution. If the estimate in any test is less than the limit, the process continues automatically. Note that the number of randomly chosen points, the limit of the proportion as well as the number of times the test is repeated can all be changed by the user.

Obviously, there is no guarantee that the stochastic rule will never fail. One can always come across the odd case where the automatic process terminates, passing the control to the user whereas the objective function can still be increased considerably. However, in most examples the random rule performs satisfactorily as indicated by the following experimental results.

5.5 Computational Results

The overall performance of LOCOBNOX was assessed using a number of randomly generated examples. The vertices of the feasible region were generated by a uniform probability distribution on [0, 1000] on each coordinate subject to the constraint that these vertices formed a simple polygon, not necessarily convex. The demand points were also generated by a uniform distribution on the same range under the condition that they were all within the feasible region. Finally, the weights were also randomly generated by a uniform distribution on [1, 5]. In all randomly generated problems a step size $\Delta L = 10$ was chosen.

All experiments were performed on a PS\2 with an 80386 processor. The computational results for different problem sizes are given in tables 5.2 to 5.4. In each example we recorded :

(a) the average time required for the calculation of the lower bound (column 1),
(b) the average time required for the main part of the algorithm (column 2), namely drawing shapes around the clients until the stochastic rule decided that control should be passed on to the user,
(c) the average total computation time (column 3) i.e. the sum of (a) and (b) and finally
(d) the average ratio L_t/L^* (column 4) where L_t is the objective function when the termination condition is satisfied and L^* the approximate optimal solution.

Table 5.2: Computation Times in Seconds
(Number of vertices of S = 10)

Number of demand points	Time required for Lower Bound (1)	Draw. Shapes (2)	Total Time (3)	L_t/L^* (%) (4)
20	0.56	5.88	6.44	94
60	0.84	17.50	18.34	96
100	1.68	15.82	17.50	92
200	3.08	29.96	33.04	90
300	4.76	24.99	29.75	86
400	5.88	23.52	29.40	87
500	7.56	29.12	36.68	93

Table 5.3 contains the results for problems where the feasible polygon has 50 vertices.

Table 5.3: Computation Times in Seconds
(Number of vertices of S = 50)

Number of demand points	Time required for Lower Bound (1)	Draw. Shapes (2)	Total Time (3)	L_t/L^* (%) (4)
20	1.40	5.74	7.14	96
60	3.92	13.02	16.94	94
100	6.44	16.68	23.12	90
200	9.24	21.84	31.08	93
300	20.16	14.00	34.16	79
400	26.32	29.68	56.00	97
500	33.04	35.56	68.60	90

Finally, table 5.4 presents the results for even larger problems, where S is defined by 100 vertices.

Table 5.4: Computation Times in Seconds
(Number of vertices of S = 100)

Number of demand points	Time required for Lower Bound (1)	Draw. Shapes (2)	Total Time (3)	L_t/L^* (%) (4)
20	6.16	5.18	11.34	96
60	18.20	5.60	23.80	88
100	30.24	9.31	39.55	93
200	60.48	10.22	70.70	82
300	90.44	10.92	101.36	91
400	120.96	12.18	133.14	88
500	151.20	15.96	167.16	90

As evidenced by the results above, the stochastic termination rule is very effective even in large instances of the problem. In all cases but one, when the iterative process terminated, the objective function was on average more than 80% of the optimal distance. User intervention is thus minimized and is only required at the final stage when the user is asked to click the uncovered area(s) and obtain the approximate solution(s).

The results regarding the total computation time indicate that LOCOBNOX can indeed be used as a site-generation tool even in large scale applications. Problems with 500 demand points lying in feasible regions with 100 vertices can be solved in less than 3 minutes on a microcomputer. Given that geographical restrictions (e.g. non-permissible areas) can easily be catered for, this implies that the method is applicable to realistic situations.

In addition to this, the analysis of the total computation time reveals that the number of vertices of S as much as the number of demand points affects the complexity of the algorithm. Although the iterative stage, as indicated by column (2) of the tables above, is not affected significantly, the calculation of the lower bound is much slower as the number of vertices of S increases. These results confirm the analysis of existing algorithms in chapters 3 and 4 where it was stated that the complexity of these algorithms should be expressed in terms of both the number n of demand points and the number m of vertices of S rather than just in terms of n alone.

In fact, as far as LOCOBNOX is concerned, for feasible regions with more than 100 vertices it is worthwhile omitting the calculation of the lower bound and starting with an initial size $L_0 = 0$. Execution time in this case is less than the total time required for the calculation of the lower bound and the iterative step of the method, as shown in table 5.5.

Table 5.5: Computation Times in Seconds
(Number of vertices of S = 300)

Number of demand points	Time required for Lower Bound Draw. Shapes Total Time (1)	(2)	(3)	Time required starting from $L_0=0$ (4)
20	9.1	7.3	16.40	30.5
60	27.6	7.9	35.50	44.6
100	45.0	13.1	58.10	55.8
200	87.8	14.6	102.40	65.2
300	135.4	15.2	150.60	78.7
400	180.8	12.4	193.20	115.6
500	216.0	22.6	238.60	135.6

For such large scale problems the program suggests a lower bound of zero, although the user may ask for the lower bound to be calculated if he\she wishes to do so.

5.6 A Parametric Version of the Model

In section 5.3 we presented a small real life application of the single facility MAXIMIN problem where an undesirable facility had to be located in the British Isles. We also demonstrated how the optimal location may be affected by the weights representing the relative incompatibility between a given population centre and the facility to be located.

Although several researchers have studied the weighted version of the problem, most of them do not investigate the effect of the weight set on the optimal solution. Erkut and Öncü (1991) present a parametric version of the problem and

observe how the optimal location changes as the effect of the weights is cancelled out systematically. More specifically, they study the following version of the problem:

$$\max L \qquad \text{(P2)}$$
$$\text{s.t. } (w_i)^{1/q} d_E (X, P_i) \geq L \qquad i=1,\ldots,n$$
$$X \in S$$

where $1 \leq q < \infty$, S is the feasible region and w_i is the positive weight corresponding to demand point P_i. Note that $q = 1$ corresponds to the ordinary weighted version of the problem and also that as q tends to infinity (P2) tends to the unweighted version.

Erkut and Öncü prove that for $q = 2$ (P2) is equivalent to a MINIMAX problem first introduced by Melachrinoudis and Cullinane (1986a). They then give two small numerical examples and graphically display the trajectory of the optimal locations as a function of the parameter q. Their results demonstrate that the optimal solution may be quite sensitive to changes in the value of q. Consequently, selecting the "correct" weights is of vital importance since the solution is highly affected by them.

The graphical approach, as described in the previous sections, can be used very efficiently to perform a parametric investigation in larger, more realistic problems. This investigation may provide the decision maker with useful information regarding the structure of a particular problem. In the examples presented by Erkut and Öncü the solution is indeed very sensitive to changes in the value of q. However, this is not always the case; certain problems may have a more "stable" structure where the optimal location is less affected. In the example of section 5.3, where the feasible region are the British Isles and the weights are proportionate to the population of each city, the optimal location is in Cornwall at location A for $1 \leq q < 1.55$, and in Scotland at location B for $q \geq 1.56$ as shown in figure 5.6. Hence, the decision maker in this case can be fairly confident that regardless of the weights there are just two candidate locations which can be assessed with respect to other criteria like transportation or maintenance costs.

Figure 5.6: *Parametric investigation for example 5.1*

If rectilinear distances are used, the results are very much the same. There exist problems where the optimal solution is highly dependent on the value of q. Figure 5.7 shows example 6.1 with 6 clients where S is defined by 5 vertices.

Example 5.2

Consider 6 demand points $P_i(x_i, y_i)$ and a feasible region S bounded by 5 vertices. The coordinates and the weights of the demand points as well as the coordinates of the vertices of S are given in the following tables:

Table 5.6i: Demand Points for Example 5.2

Point i	1	2	3	4	5	6
x_i	2	2	4	5	7	9
y_i	1	4	8.5	6	2	8
w_i	3.75	4.5	3.0	3.21	2.25	1

Table 5.6ii: Vertices of S for Example 5.2

Vert. j	1	2	3	4	5
x_j	0	3	10	8	2
y_j	4	10	9	1	0

The optimal location is along segment (a) for $1 \leq q < 1.12$, segment (b) for $1.12 \leq q < 5.5$ and segment (c) for $5.5 \leq q < \infty$. The arrows in the figure indicate the movement of the optimal location as q is increased.

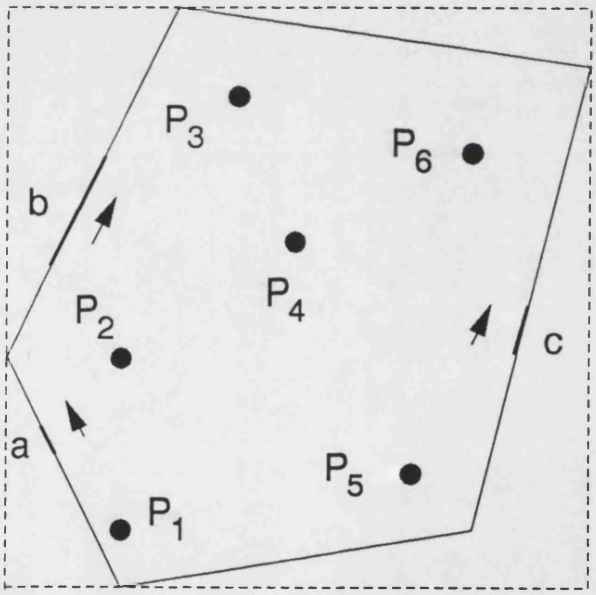

Figure 5.7: *Parametric investigation for example 5.2*

On the other hand, it is easy to construct problems where the solution is not affected by q at all. Figure 5.8 shows another example where the optimal location is at vertex V regardless of the value of the parameter.

Example 5.3

Consider 6 demand points within the feasible region of example 5.2. Their coordinates are given in the following table:

Table 5.7i: Demand Points for Example 5.3

Point i	1	2	3	4	5	6
x_i	2	2	5	6	7	9
y_i	2	5	4	1	9	8
w_i	3.75	4.5	3.0	3.21	2.25	1

The optimal location is at vertex V regardless of the value of the parameter, as shown in figure 5.8.

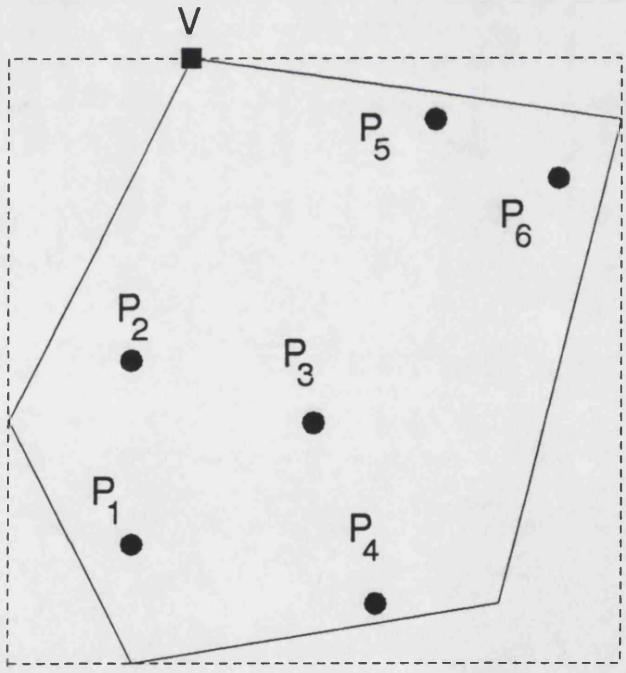

Figure 5.8: *Parametric investigation for example 5.3*

These results indicate that the parametric investigation described in this section is always worth attempting in real life problems since it can reveal valuable

information about the structure of a given problem. It should be kept in mind that it does by no means free the decision maker from setting the weights themselves. It does, however, allow him/her to investigate the "stability" of a particular problem, i.e. the effect on the optimal location as the weights become less important. The graphical approach seems to be the most appropriate method for this investigation since it can be applied in realistic problems where this analysis is most required.

5.7 Applications and Extensions of the Graphical Model

It goes without saying that the real world problem of locating obnoxious facilities is extremely complex. According to Erkut and Neuman (1989) it is even more complex than the one of locating desirable facilities. They feel that the perceived disutility associated with an undesirable facility is higher than the perceived utility associated with a desirable one. Consequently they suggest that the decision process for locating an obnoxious facility should consist of two stages: (a) identifying a small set of candidate locations (site-generation) and (b) selecting the final location (site-selection).

Most of the methods reviewed in chapter 3 can be used as site-generation tools since they find all local optima of the problem. However, the interactive graphical approach seems to be the most appropriate although its theoretical principles are very simple. In fact its simplicity and ease of implementation are major advantages since most decision makers feel much more comfortable with methods they can understand rather than with ones they know very little of. In addition to this, since the method is interactive the user may use his/her experience to identify potential locations which he/she can then evaluate using a number of often conflicting criteria.

Moreover, the graphical method is much more flexible than any analytical technique and can represent many more aspects of the real world. We have already

mentioned that it is not limited to the Euclidean or the rectilinear distance metric. In fact it can cater for all L_p-norms depending on the particular application. It is even possible to use one metric for some points and another for others.

Specific distance constraints related to a particular problem can be modelled fairly easily. A lower bound of L_{min} on the distance between a client and the new facility can be represented by a forbidden region of size L_{min} around this client. An upper bound L_{max} can be incorporated by not allowing the size of the shape (diamond, circle or whatever) around the client to grow above L_{max}. Regions of varying environmental compatibility with the facility to be located can be represented by varying shades of background colour on the screen.

Another useful extension of the graphical approach would be to allow the user to "zoom in" on a subregion of the graphics screen when the vicinity of the optimal location has been identified. Hence, accuracy can be improved by giving the user a cleaner picture to analyze.

Having mentioned these possible enhancements one could argue that, though flexible and versatile, the graphical method is not very useful since it only yields approximate solutions. However, especially in complex large scale problems, where the primary objective is to select several candidate areas, which will then be assessed on the basis of other criteria, good approximate solutions are more than satisfactory. Moreover, in appendix D we demonstrate how the graphical approach can be combined with the properties of the optimal solution to obtain the exact solution to the problem. In general, given the approximate solution X^a, we find the demand points that are nearest to it, three in the Euclidean problem or four in the rectilinear one. We then examine whether these points can determine a local solution in the interior of the feasible region S, on the boundary of S, or on the boundary of a non permissible area. In the Euclidean problem, for instance, we check whether the point equidistant from all three clients is uncovered, thus defining a solution in the interior of S. If it is covered, we then check whether each pair of clients can determine a solution along a side of S or along a side of a non permissible area. More simply,

we check whether the intersection of the corresponding bisector with each side of S (or each side of the non permissible regions) is uncovered. If it is, then it must be the optimal location. For more details of the method see appendix D.

Undoubtedly there are many issues associated with obnoxious facility location which the graphical approach does not consider. Like all other methods it does not discuss economic factors at all. There are certain operating, transportation or maintenance costs associated with each location. For particular applications safety issues may also be important. It would be too risky, for instance, to locate a nuclear factory in an earthquake prone area. These issues which may be of great importance do not come into the analysis. However, one should always keep in mind that the graphical method should primarily be used as a site-generation tool, simply to identify several candidate areas rather than find the optimum. These candidates should then be evaluated considering the issues mentioned above, which cannot be represented by analytical models anyway.

5.8 Summary

In this chapter we have presented an interactive graphical approach to the single facility problem and explained that its main advantage is flexibility and the ability to represent aspects of the real world which other approaches do not cater for. In addition to this we introduced a stochastic termination rule aimed at reducing user intervention and presented computational results regarding the overall performance of the algorithm. These results indicate that the method can be used to address real life problems efficiently. We also demonstrated how the graphical model can be used to perform a parametric investigation which may reveal useful information about the underlying structure of a given problem. However, we pointed out that, though flexible and effective, the graphical model is far from a precise representation of the

real world. Hence, it should be seen as a site-generation tool, used to identify potential locations rather than as a rigorous technique trying to achieve optimality.

CHAPTER SIX

THE MULTIPLE FACILITY PROBLEM

6.1 Introduction

Although the single obnoxious facility problem has been studied by several researchers, very little research has been done on the multiple facility problem in the two dimensional plane. This may be due to the fact that there is no unique way of defining the multifacility problem. The introduction of more undesirable facilities raises a number of issues regarding the interaction of the demand points with the facilities to be located as well as the interaction of the undesirable facilities with each other. As a result a number of multifacility location models have been generated depending on the way we address these issues.

Some of these models deal with problems where there are no demand points and the objective is to maximize some function of distance between the new facilities whereas other models consider existing facilities as well. In addition to this, there exist minimization models with minimum distance constraints ensuring that the distances between the undesirable facilities and the demand points exceed specified values. Section 6.2 presents several of these models and discusses the situations where each of them is appropriate.

Throughout this chapter we will focus our attention on models involving existing facilities as well. Most of the published approaches refer to problems where the feasible locations are on a network. Section 6.3 contains a review of the published approaches on the MAXIMIN problem where the feasible region is a bounded area in the two-dimensional plane.

Section 6.4 introduces TWOPROFLAWLP, a new analytical algorithm for the

rectilinear two-facility problem in the presence of demand points. The method is essentially a bisection technique exploiting the ideas of the LP-based approaches to the single facility problem. It is a generalization of the PROFLAWLP algorithm introduced in chapter 4 and solves the two-facility problem without using linear programming.

The new algorithm was tested on problems with up to 60 demand points. Section 6.5 presents some computational results which invoked a slight modification of the algorithm in order to improve its overall performance.

6.2 Alternative Models for the Multifacility Problem

6.2.1 Definitions

Let P_i for $i=1,\ldots,n$ be n existing facilities (clients) in a feasible region S and w_i be the corresponding weights expressing the relative importance of each client. Let also $d(P_i, X)$ express the distance between client i and facility location X in a given distance metric. The feasible region S may be :

(a) A network in which case the P_i's are restricted to be points on the network and $d(P_i, X)$ denotes any distance norm defined on the network, usually the shortest path between P_i and X. Moreover, X can only be on a node of the network (discrete problem) or is allowed to be on an edge as well (continuous version).

(b) A bounded area in the two-dimensional plane in which case the P_i's are points in the plane and $d(P_i, X)$ represents either the Euclidean or the rectilinear distance between P_i and X.

The problem is to locate r identical obnoxious facilities at X_j (for $j=1,\ldots,r$) as far away as possible from all clients. The existence of more than one undesirable facility poses several issues regarding the interaction between the clients and the facilities to be located.

Firstly, we need to decide which of these interactions are undesirable. This depends on the kind of facilities to be located as well as the nature of the feasible region. When locating several dump sites for waste disposal for example, the only interactions that matter are between the dump sites and the population centres served by these sites. On the other hand, when locating missile depots in an unpopulated area, it usually suffices to consider the interactions between depots. Finally, when locating a number of nuclear power plants in a country, the interactions between the population centres and the plants as well as the ones between plants are important.

Secondly, we need to clarify how to measure the distance between a client and the solution set. In some applications a client interacts with all obnoxious facilities, in which case the sum of the distances to the solution set is required. However, in other cases a client is affected only by the nearest undesirable facility, so the minimum distance to the solution set is used.

The issues outlined above give rise to a number of alternative formulations some of which are presented below.

6.2.2 Models without Existing Facilities

The existing literature in this area deals with models where there are no demand points since only the interactions between the facilities to be located are considered important. All the models in this category address problems where the feasible region is a network and the distance metric is the shortest path between two points on the network. Clearly, such models are useful when locating mutually undesirable facilities. A typical example is the dispersion of military installations to prevent damage of several facilities by the same attacker. Another example is the location of franchises in a populated area by the same company. Such franchises are mutually undesirable since they may compete for the same customer base; consequently, a deeper market penetration may be achieved through spatial dispersion of the franchises.

Assuming that each franchise is only affected by its nearest "competitor" the

problem is to maximize the minimum distance between any two franchises, known as the **MAXIMIN-MIN** or **dispersion** problem:

$$\begin{aligned} & \max L && && (P1) \\ \text{s.t.} \quad & L \leq v_j && j=1,\ldots,r \\ & X_j \in S && " \end{aligned}$$

where $v_j = \min_{k \neq j} \{ d(X_j, X_k) \}$ is the distance of facility j to its nearest facility.

(P1) seems to be by far the most popular problem in the area. Shier (1977) and Tansel et al. (1982) establish a duality relationship between the r-dispersion problem and the (r-1)-centre problem on a tree network. The objective of the latter problem is to locate r-1 new facilities so that the maximum distance between them and a given set of clients is minimized. Shier and Tansel et al. prove that on a tree network the MAXIMIN distance solution to the r-dispersion problem is exactly twice as large as the MINIMAX distance solution to the (r-1)-centre problem. Chandrasekaran and Daughety (1981) use this duality relationship and present an algorithm which involves solving a finite number of anti-cover problems for the dispersion problem and a finite number of cover problems for the centre problem. The cover problem minimizes the number of facilities to be located under the restriction that each demand point is covered by at least one facility i.e. its distance from the facility is smaller than a specified constant. On the other hand, the anti-cover problem locates the maximum number of facilities under the restriction that no two are closer than a specified distance from each other. For more details on these problems see Moon and Chaudhry (1984).

Kuby (1987) studies the discrete dispersion problem on a network and presents a mixed integer programming formulation but does not propose an algorithm to solve realistic instances of the problem.

The dispersion problem is based on the assumption that each facility is only affected by its nearest facility. However, there are situations where each facility interacts with all others. When locating several radio transmitters, for instance, each of them may be affected by all the others. Hence, in order to minimize the maximum

interference we should maximize the minimum sum of distances between the transmitters (**MAXIMIN-SUM** or **dispersion-sum** model) also discussed in Kuby (1987).

$$\begin{aligned} & \max L && && \text{(P2)} \\ \text{s.t.} \quad & L \le v_j && j=1,\ldots,r \\ & X_j \in S && " \end{aligned}$$

where

$$v_j = \sum_{k=1}^{r} d(X_j, X_k)$$

Adopting the MAXISUM rather than the MAXIMIN criterion in (P1) leads to a **MAXISUM-MIN** or **defence** problem, discussed by Moon and Chaudhry (1984), who present an integer programming (IP) formulation for the discrete version of the problem without actually solving it. Such a model is applicable when locating strategic installations to protect them from simultaneous enemy attacks.

$$\begin{aligned} & \max L && && \text{(P3)} \\ \text{s.t.} \quad & X_j \in S && j=1,\ldots,r \end{aligned}$$

where $L = \sum_j v_j$ and $v_j = \min_{k \neq j} \{d(X_j, X_k)\}$.

Similarly, maximizing the total sum of distances between the new facilities gives rise to a **MAXISUM-SUM** or **defence-sum** problem, stated in Erkut and Neuman (1991). Locating chairs in an examination hall in order to minimize communication between participating students is an example of such a situation.

$$\max \sum_{j=1}^{r} \sum_{k=j+1}^{r} d(X_j, X_k) \qquad \text{(P4)}$$

$$\text{s.t. } X_j \in S \quad j=1,\ldots r.$$

Although, the above models have only been applied to problems on networks, their logic can be extended to problems on the plane as well. However, to the best of our knowledge, no application in planar problems has been reported.

6.2.3 Models Considering Existing Facilities

Although dispersion models can be useful when locating mutually undesirable facilities which are not themselves obnoxious, they cannot be used in obnoxious facility location. Undoubtedly, the undesirability of a chemical factory or a nuclear power plant is considered with respect to a number of existing facilities (e.g. population centres). Consequently, any multiple obnoxious facility model should take into account the interactions between the undesirable facilities as well as the ones between these facilities and the existing demand points. Supposing that each demand point interacts with all undesirable facilities and that the MAXISUM criterion is adopted, the problem of locating r such facilities can be formulated as a **MAXISUM-SUM** model as follows:

$$\max L \qquad \text{(P5)}$$

$$\text{s.t.} \quad X_j \in S \qquad j=1,\ldots,r$$

where

$$L = \sum_{i=1}^{n}\sum_{j=1}^{r} w_i * d(P_i,X_j) + \sum_{j=1}^{r}\sum_{k=1}^{r} d(X_j,X_k) \qquad (1)$$

S is the set of permissible locations (either a network or a bounded area on the plane) and the distance metric is defined accordingly. If the interactions between the new facilities are considered negligible the second term of (1) can be omitted in which case extra restrictions e.g. minimum distance constraints, have to be added to ensure that the facilities to be located are not placed on the same location.

On the other hand, using the MAXIMIN criterion and assuming that each demand point is affected by its nearest undesirable facility leads to a **MAXIMIN-**

MIN model:

$$\max L \qquad \text{(P6)}$$

$$\text{s.t.} \quad w_i\, d(P_i, X_j) \geq L \qquad i=1,\ldots,n \text{ and } j=1,\ldots,r \qquad (1)$$

$$d(X_j, X_k) \geq L \qquad j,k=1,\ldots,r \qquad (2)$$

$$X_j \in S \qquad j=1,\ldots,r \qquad (3)$$

Alternatively, constraints (2) of (P6) can be replaced by minimum distance constraints ensuring that the facilities to be located will not be placed on top of each other. More simply, if facilities j and k must be at least c_{jk} away from each other, the optimal locations are given by the following model:

$$\max L \qquad \text{(P7)}$$

$$\text{s.t.} \quad w_i\, d(P_i, X_j) \geq L \qquad i=1,\ldots,n \text{ and } j=1,\ldots,r \qquad (1)$$

$$d(X_j, X_k) \geq c_{jk} \qquad j,k=1,\ldots,r \qquad (2)$$

$$X_j \in S \qquad j=1,\ldots,r \qquad (3)$$

Drezner and Wesolowsky (1985) propose a model similar to (P7) where the objective is to maximize the nearest client-to-facility weighted distance under the restriction that every client is "within reach" of the nearest facility. More simply,

$$\max L \qquad \text{(P8)}$$

$$\text{s.t.} \quad \min_j \{ d(P_i, X_j) \} \leq b_i \qquad i=1,\ldots,n \qquad (1)$$

$$X_j \in S \qquad j=1,\ldots,r \qquad (2)$$

where $L = \min_{ij} \{ w_i\, d(P_i, X_j) \}$ and b_i denotes the "within reach" distance from client i.

In addition to this, Drezner and Wesolowsky present a modification of the r-centre problem where the objective is to minimize the maximum client-to-facility distance. If the facilities are obnoxious, minimum distance constraints are added to ensure that each new facility must be at least c_i away from client i. Formally stated, the model is:

$$\min D \qquad \text{(P9)}$$

$$\text{s.t.} \quad d(P_i, X_j) \geq c_i \qquad i=1,\ldots,n \text{ and } j=1,\ldots,r \qquad (1)$$

$$X_j \in S$$

where $D = \max_i \{ \min_j \{ u_i d(P_i, X_j) \} \}$ and u_i is a positive weight expressing the relative importance of client i.

Drezner and Wesolowsky establish certain duality relationships between problems (P8) and (P9) and propose an algorithm for the one-dimensional problem where S is a line. Their findings are discussed in greater detail in the following section. To the best of our knowledge, problems (P6) and (P7) have not been addressed yet for polygonal feasible regions on the plane, although they seem to be the natural generalisation of the single facility models given in chapter 2.

It should be kept in mind that the models presented in this section are by no means equivalent to each other. They are merely some alternative ways of modelling the problem of locating multiple undesirable facilities. Each model is based on different assumptions and is only applicable in certain situations for which these assumptions are valid.

6.3 Previous Approaches to the Multifacility Problem on the Plane

6.3.1. An Interactive Graphical Approach

Although model (P9) of the previous section is essentially a MINIMAX model, more appropriate for locating desirable facilities, the minimum distance constraints (1) guarantee that all new facilities will be at least c_i away from demand point i. Hence, provided that the c_i 's are sufficiently large, (P9) can be used to locate facilities that serve the given set of clients but also have certain undesirable effects.

In chapter 5 we described a single facility algorithm by Brady and Rosenthal (1980). This technique involves drawing circles of decreasing radii around each demand point and continuing while the intersection of all the circles is not empty i.e. stop when two circles are tangent.

Brady, Rosenthal and Young (1983) develop an interactive graphical

technique, based on the above algorithm, for the MINIMAX problem with general constraints. More specifically, using the notation of the previous section, they define the problem as follows:

$$\min \ \max_i \ \{ \min_j \{ w_i \, d(P_i , X_j) \} \} \qquad \text{(P10)}$$

$$\text{s.t.} \quad X_j \in S' \qquad j=1,\ldots,r$$

where w_i is the weight corresponding to P_i and S' is the set of feasible locations. Clearly, (P10) is a more general version of (P9).

The multi-facility algorithm is basically a recursive application of the original Brady and Rosenthal algorithm and can be described as follows:

Brady, Rosenthal and Young Multifacility Algorithm

1. Represent the demand points P_i and the feasible region S' on a map.
2. Set $j \leftarrow 1$.
3. If $j=r$ stop.
   Otherwise, use the original Brady-Rosenthal algorithm to locate the j-th undesirable facility.
4. Delete the demand points served by the j-th facility, i.e. the ones whose circles yield the first tangency.
5. Set $j \leftarrow j + 1$.
   Return to step 3.

Hence, we can use the above algorithm to solve problem (P9) if we define S' as the intersection of S and constraint set (1) of (P9).

6.3.2 A Duality-Based Approach

As mentioned in the previous section, Drezner and Wesolowsky (1985) present two alternative formulations for the multifacility problem in the k-dimensional space.

The first one is a MAXIMIN model aiming to maximize the minimum client-to-facility distance, constraining every client to be "within reach" of the nearest

facility. Thus, the problem is stated as follows:

$$\max L \qquad \text{(P11)}$$

$$\text{s.t.} \quad \min_j \{ d(P_i, X_j) \} \leq b_i \qquad i=1,\ldots,n \qquad (1)$$

$$X_j \in S \qquad j=1,\ldots,r \qquad (2)$$

where $L = \min_{i,j} \{ w_i \, d(P_i, X_j) \}$ is the minimum distance in the system, w_i is a positive weight and b_i denotes the "within reach" distance from client i. Recall that this is model (P8) discussed earlier.

The second model, (P9) in the previous section, is a modification of the well-known r-centre problem, aiming to minimize the maximum client-to-facility distance under the restriction that all undesirable facilities must be at least c_i away from client i. More simply:

$$\min D \qquad \text{(P12)}$$

$$\text{s.t.} \quad d(P_i, X_j) \geq c_i \qquad i=1,\ldots,n \text{ and } j=1,\ldots,r \qquad (1)$$

$$X_j \in S$$

where $D = \max_i \{ \min_j \{ u_i \, d(P_i, X_j) \} \}$ and u_i is a positive weight expressing the relative importance of client i. Note that in general $w_i \neq u_i$. A large u_i implies a relatively important demand point whereas a large w_i implies a relatively trivial one.

Drezner and Wesolowsky define (P11) and (P12) to be **dual to each other for a value f_0** if $w_i \, c_i = u_i \, b_i = f_0$ for i=1,...n. They then use this definition to establish several useful properties for both problems.

Firstly, supposing that $S_1 (f_0)$ is the set of all feasible locations for (P11) for which $L \leq f_0$ and $S_2 (f_0)$ the set of all feasible locations for (P12) for which $D \geq f_0$, they prove that if (P11) and (P12) are dual then $S_1 (f_0) = S_2 (f_0)$.

Secondly, they prove that if X_j^* for j=1,...,r are the optimal locations for (P11) and L^* the optimal distance, then the X_j^*'s are also optimal for the dual created by $w_i \, c_i = u_i \, b_i = L$ and that $D^* = L^*$.

The significance of this result is illustrated by the following example, also presented in Drezner and Wesolowsky. Suppose that model (P11) is used to locate several dump sites as far away as possible from n cities under the restriction that each

city is less than 10 miles away from the nearest site. Suppose, also, that the optimal distance for (P11) is 5 miles and that at least one city is 5 miles away from a dump site. Then, the dual is a problem where we aim to minimize the maximum city-to-site distance and require each city to be at least 5 miles away from the nearest site. The result given by Drezner and Wesolowsky states that the optimal distance to this dual problem will turn out to be 10 miles.

Thus, starting from (P11) and a given value f_0 we can construct the dual by setting $c_i = f_0 / w_i$ and $u_i = f_0 / b_i$ and solve it using the Brady, Rosenthal and Young algorithm. If a feasible solution exists to this dual then we know from the results discussed above that a feasible solution also exists for (P11), consequently $L^* \geq f_0$. On the other hand, if the dual is not feasible $L^* < f_0$. Hence, the duality results by Drezner and Wesolowsky can be combined with the interactive graphical algorithm by Brady, Rosenthal and Young to yield a bisection solution method for the original maximization problem (P11) on the two-dimensional plane.

Apparently unaware of the graphical algorithm, Drezner and Wesolowsky report that finding a feasible solution for (P12) is complicated by the combinatorics involved in the allocation of facilities to demand points. Consequently, they only present an algorithm for the one-dimensional rather than the two-dimensional version of the problem.

6.3.3 A One-Dimensional Algorithm

Let P_i for $i=1,\ldots,n$ be the locations of n given demand points as measured from an arbitrary origin in the one-dimensional space. Let also w_i be the corresponding weights. Without loss of generality, assume that $P_i \leq P_{i+1}$ for $i=1,\ldots,n-1$. Problem (P11) now becomes:

$$\max L \qquad \text{(P13)}$$

$$\text{s.t.} \quad \min_j \{ |P_i - X_j| \} \leq b_i \qquad i=1,\ldots,n.$$

where $L = \min_{i,j} \{ w_i |P_i - X_j| \}$.

Figure 6.1 shows an example with three clients.

Example 6.1

Consider three demand points located at $P_1 = 2$, $P_2 = 4$ and $P_3 = 7$ with weights $w_1 = 0.25$, $w_2 = 1$ and $w_3 = 1.5$ and "within reach" distances $b_1 = 1$, $b_2 = 1.5$ and $b_3 = 2$ respectively.

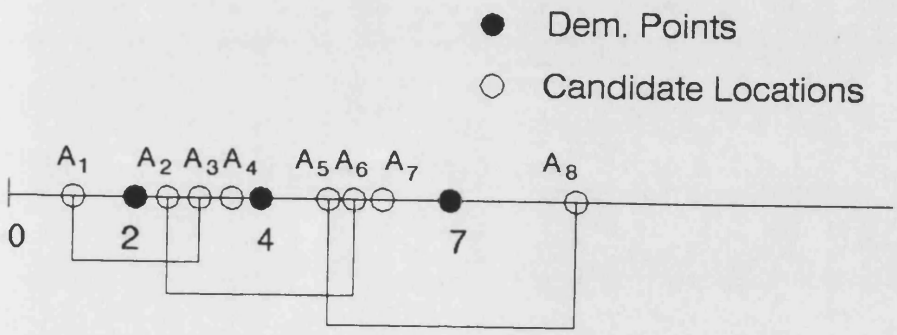

Figure 6.1: *Example 6.1*

Figure 6.1 also illustrates the "weighted mid-points" for each range i.e. the points X for which $w_i |P_i - X| = w_{i+1} |P_{i+1} - X|$. These are locations A_4 and A_7 in the figure. Drezner and Wesolowsky state that a maximum for L of (P13) will occur at such a "weighted mid-point" or at a point $P_i \pm b_i$. Thus, they identify the set of candidate locations denoted by A_k such that $A_k \leq A_{k+1}$ for $k = 1,\ldots,N-1$ where N is the number of these locations. It can be seen that for the particular example $N = 8$ with $A_1 = 1$, $A_2 = 2.5$, $A_3 = 3$, $A_4 = 3.6$, $A_5 = 5$, $A_6 = 5.5$, $A_7 = 5.8$ and $A_8 = 9$.

Drezner and Wesolowsky exploit the structure of the problem and develop a heuristic to find which r of the A_k's constitute a feasible solution to (P13). They then describe their algorithm as follows:

One-Dimensional Algorithm

1. Find a set of r locations for an initial feasible solution using the heuristic mentioned above.
2. Evaluate the objective function L for the current feasible solution.
3. Reject all A_k (k=1,...,N) for which $w_i |P_i - A_k| \leq L$ for some i.
4. Try to find a feasible solution for the reduced set of sites.
5. If no feasible solution exists, the previous feasible solution is optimal. Otherwise, go to step 2.

A detailed description of the heuristic is given in appendix E. It can be checked that if r=2 the optimal solution for example 6.1 is $L^*=0.25$ with $X_1^* = A_1 = 1$ and $X_2^* = A_5 = 5$. Note that this solution is not unique; e.g. $X_1^* = A_1$ or $X_1^* = A_3$ and $X_2^* = A_k$ for $k \geq 3$ are also optimal.

Clearly, the fact that the new facilities have to be located along a line is a rather unrealistic assumption. In the next section, we will introduce an algorithm to locate two obnoxious facilities when the feasible region is a two-dimensional polygon.

6.4 Solving the Two-Facility Rectilinear Problem on the Plane

6.4.1 Problem Formulation

Suppose that two undesirable facilities are to be located as far away as possible from a given set of n demand points $P_i (px_i, py_i)$ with corresponding weights w_i for i=1,...,n. Suppose, also, that each client is affected only by its nearest undesirable facility and that we wish to maximize the minimum client-to-facility rectilinear distance. The optimal locations $X_j (x_j, y_j)$ under these assumptions are the solutions to problem (P6) for r=2. If the feasible region S is a convex polygon defined by the intersection of m linear constraints, problem (P6) can be rewritten as follows:

$$\max L \qquad \text{(P14)}$$

$$\text{s.t.} \quad w_i d_r(P_i, X_j) \geq L \qquad i=1,\ldots,n \text{ and } j=1,2 \qquad (1)$$

$$d_r(X_1, X_2) \geq L \qquad (2)$$

$$a_q x_j + b_q y_j \leq c_q \qquad j=1,2 \text{ and } q=1,\ldots,m \qquad (3)$$

A naive way to address (P14) would be to solve the single facility problem using any of the LP-based algorithms presented in chapter 4, consider the optimal location as a new demand point and solve the enhanced problem to find the location of the second facility. However, this sequential approach does not guarantee optimality as illustrated by the following example:

Example 6.2

Consider six demand points within a convex polygon S defined by $0 \leq x \leq 10$ and $0 \leq y \leq 10$ (see figure 6.2) and suppose that $w_i = 1$ for $i=1,\ldots,6$.

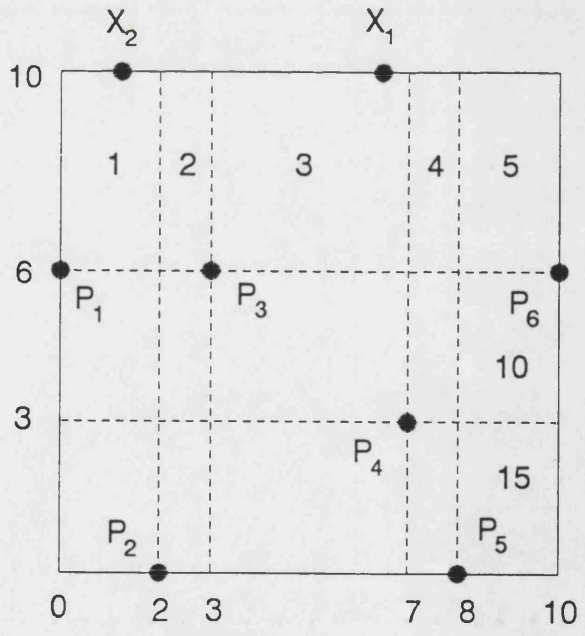

Figure 6.2: *Example 6.2*

The coordinates of the demand points are given in the following table:

Table 6.1: Data for Example 6.2

Point P_i	1	2	3	4	5	6
px_i	0	2	3	7	8	10
py_i	6	0	6	3	0	6

Using PROFLAWLP, the single facility algorithm established in chapter 4, we get X_1 (6.5, 10) as the optimal location at distance L_1 =7.5 from P_3 and P_6 . Considering X_1 as an extra client and solving the new problem produces X_2 (1.25, 10) as the optimal location at distance L_2 =5.25 from P_1 and X_1 . Hence, the solution to the original problem is L=5.25 since this is the minimum client-to-facility distance and $d_r(X_1, X_2)$=5.25 as well. However, this solution is not optimal since locating the two facilities at X_1 (6.5, 8) and X_2 (1.5, 10) yields L=5.5 as the objective function value.

6.4.2 A Bisection Approach

Obviously, the solution to the single facility problem provides an upper bound L_{max} on the optimal distance L^* of (P14) whereas the sequential approach provides a feasible solution to (P14) and, consequently, a lower bound L_{min} on L^* . These bounds form the basis of a bisection algorithm which solves (P14) optimally. At each stage of the process we check whether there exists a feasible solution to (P14) with objective function value L. If so, $L^* \geq L$. Otherwise, $L^* < L$.

Hence, we need a method to check whether there exists a feasible solution with value L, namely whether there exist two locations X_1 and X_2 which are at least L away from all demand points and L away from each other.

Let us consider the loci of points which are at least L away from all demand points. These are the areas outside diamonds drawn around each client with semi-

diagonal L/w_i. For the data of example 6.2 and L=5 these loci are the hatched areas in figure 6.3.

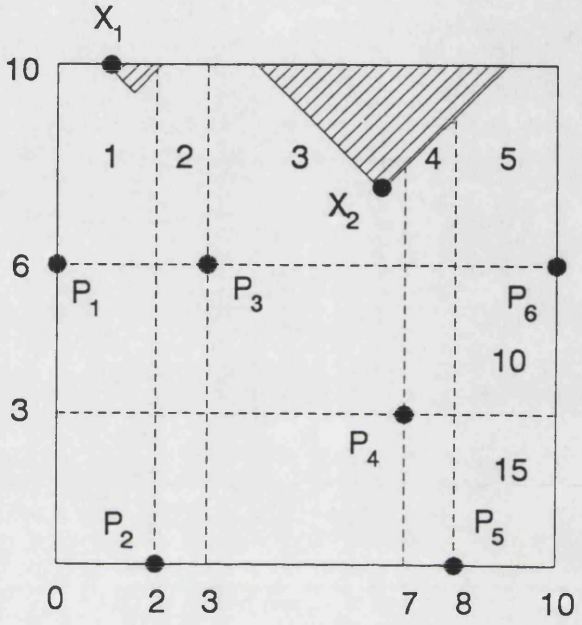

Figure 6.3: *Hatched areas in example 6.2 for L = 5*

It can be seen that these areas must be simple polygons S_j formed by the intersection of linear constraints, namely the sides of the diamonds and the boundary constraints defining the feasible region.

The problem is to find two locations in these polygons which are at least L away from each other. If two such locations exist they constitute a feasible solution to the whole problem. The method for finding a feasible solution, if it exists, is based on the following two theorems.

Theorem 6.1

Let A, B and $X_0(x_0, y_0)$ be three points in the two-dimensional plane. Let also X(x, y) be any point along the line AB. The function f with $f(X) = d_r(X_0, X)$

is convex on AB.

Proof:

Let $X_1 (x_1 , y_1)$ and $X_2 (x_2 , y_2)$ be two points along AB. Let also $0 \leq \lambda \leq 1$. By the definition of f we have

$$\begin{aligned} f[\lambda X_1 + (1-\lambda)X_2] &= |\lambda x_1 + (1-\lambda)x_2 - x_0| + |\lambda y_1 + (1-\lambda)y_2 - y_0| \\ &= |\lambda(x_1 - x_0) + (1-\lambda)(x_2 - x_0)| + \\ &\quad |\lambda(y_1 - y_0) + (1-\lambda)(y_2 - y_0)| \\ &\leq \lambda|(x_1 - x_0)| + (1-\lambda)|(x_2 - x_0)| + \\ &\quad \lambda|(y_1 - y_0)| + (1-\lambda)|(y_2 - y_0)| \\ &= \lambda[\,|(x_1 - x_0)| + |(y_1 - y_0)|] + \\ &\quad (1-\lambda)[\,|(x_2 - x_0)| + |(y_2 - y_0)|\,] \\ &= \lambda f(X_1) + (1-\lambda)f(X_2) \end{aligned}$$

Since $f[\lambda X_1 + (1-\lambda)X_2] \leq \lambda f(X_1) + (1-\lambda)f(X_2)$ for every λ such that $0 \leq \lambda \leq 1$, f is convex on AB.

Theorem 6.2

The maximum rectilinear distance between two simple polygons is realised by two vertices.

Proof:

Let S_1 and S_2 be two simple (non self-intersecting) polygons. Suppose that the maximum distance between S_1 and S_2 is given by points X_0 and X where X is in the interior of S_2 (see figure 6.4). Since X is an interior point of S_2 there must be at least two points A and B on the boundary of S_2 such that X is between them. However, by theorem 6.1 $f(X)=d_r (X_0 , X)$ is convex on AB; therefore it must achieve its maximum at an extreme point of AB i.e. one of its endpoints, say A. Suppose that A is on edge UV of the boundary of S_2 but not on a vertex. However, f(A) is convex on UV, hence the maximum is achieved at one of the endpoints i.e. at a vertex of S_2 . By the same argument we can show that X_0 must be at a vertex of S_1 .

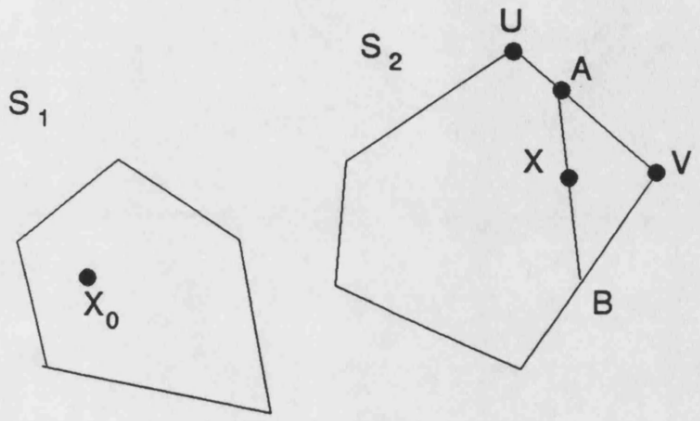

Figure 6.4: *Maximum distance between two polygons*

Theorem 6.2 implies that it suffices to check the vertices of the S_j's to see if any two of them provide a feasible solution with value L. If no two such vertices exist or, equivalently, if the maximum distance between all these vertices is less than L, such a solution cannot exist. Hence, we need to identify these vertices first. By the definition of the S_j's it can be seen that each vertex of such a polygon is:
(a) either the intersection of a boundary constraint with a diamond side, i.e. a location on the boundary at distance L from a demand point e.g. location X_1 in figure 6.3, or
(b) the intersection of two diamond sides i.e. a location at distance L from two clients e.g. location X_2 in the figure.

Having used PROFLAWLP to obtain the upper and the lower bound on the optimal distance, we can exploit the technique further in order to identify the vertices of the S_j's. As explained in chapter 4, PROFLAWLP divides the feasible region into rectangular areas by drawing one horizontal and one vertical line through each demand point. For each rectangle M it then calculates an upper bound UB_M on the

optimal distance and finds the boundary constraints, if any, passing through M. The upper bounds for the data of example 6.2 are given in table 6.2.

Table 6.2: Upper Bounds for Example 6.2

Rectangle	Upper Bound
1	5.5
2	5
3	7.5
4	7
5	6
6	4
7	3.5
8	3.5
9	3
10	3
11	4
12	3.5
13	4
14	2
15	4

After sorting the upper bounds in descending order the algorithm starts from the rectangle with the largest upper bound and solves the problem for the rectangle in question. It then continues with the rectangle corresponding to the next largest upper bound until this bound is less than the best solution found so far. These ideas are used to identify the list VS of vertices of the S_j's for a given value L as explained below.

Finding the Vertices of the S_j's

1. (Initialization).
   Set VS $\leftarrow \varnothing$.
   Start from the rectangle M with the largest upper bound.
2. (Termination Criterion).
   If $UB_M < L$ then stop.
   Otherwise, go to step 3.
3. (Identifying locations at distance L from two clients).
   For each pair of closest points P_i and P_k around M find a location V at distance L from both of them by solving the system of two simultaneous equations: $d_r(P_i, V) = L$ and $d_r(P_k, V) = L$.
   If V is within M and is feasible then add it to the list of vertices (VS $\leftarrow$ VS $\cup$ { V }).
4. (Identifying locations on the border at distance L from one client).
   For each closest point P_i and each constraint H: $ax + by \leq c$ passing through M solve the system of equations: $d_r(P_i, V) = L$ and $au + bv = c$. If V(u, v) is within M then add it to the list of vertices (VS $\leftarrow$ VS $\cup$ { V }).
5. Let M be the rectangle with the next largest upper bound.
   Return to Step 2.

At the end of the process we have a list VS of locations which are at least L away from all demand points. If any two of these locations are at least L away from each other they provide a feasible solution. For the data of example 6.2 and L=5 it can be checked that there exist seven such locations V_1 to V_7 with coordinates (1, 10), (1.5, 9.5), (2, 10), (4, 10), (6.5, 7.5), (7, 8) and (9,10) respectively. For L=5.5 there exist five locations V_1 to V_5 with coordinates (1.5, 10), (4.5, 10), (6.5, 8), (7, 8.5) and (8.5, 10) respectively as shown in figure 6.5.

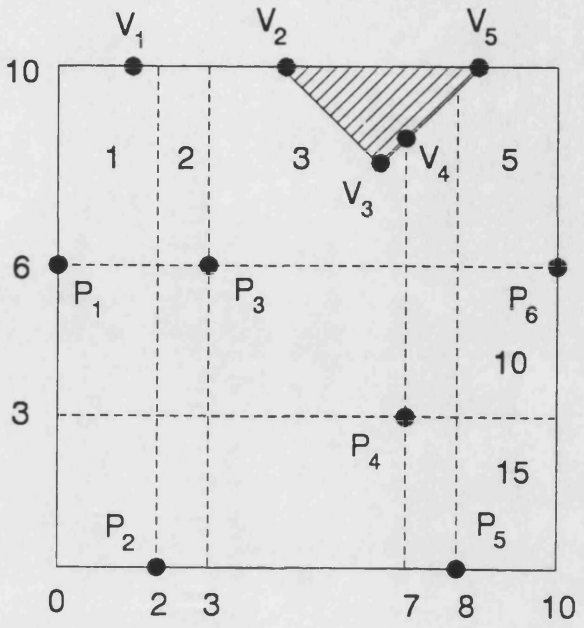

Figure 6.5: *Hatched areas in example 6.2 for L = 5.5*

Having established the process for finding a feasible solution to problem (P14) or (P15) the whole algorithm can be described as follows:

TWOPROFLALP: A Two-Facility Algorithm

1. Use the PROFLAWLP algorithm to obtain an upper bound L_{max} and an initial feasible solution as well as a lower bound L_{min} on the global optimum.
2. If $L_{max} - L_{min} < \epsilon$, where ϵ is the specified tolerance, $L^* = L_{min}$ and the optimal solution is the last feasible solution.
   Otherwise, set $L = (L_{min} + L_{max})/2$.
3. Check whether there exists a feasible solution to (P14) with objective function value L.
   If so, then set $L_{min} = L$.
   Otherwise, set $L_{max} = L$.
4. Return to Step 2.

It turns out that the maximum distance for the data of example 6.2 is $L^*=5.5$ and the optimal locations are V_1 and any of V_3 , V_4 or V_5 , all of which are at distance L^* from their nearest clients (see figure 6.5). Considering one of these multiply optimal solutions, say V_1 and V_3 , we can see that any movement away from V_1 will decrease the distance to its nearest clients, namely P_1 and P_3 , and thus decrease the value of the objective function. On the other hand, it is possible to move away from V_3 , along the vertical bisector defined by P_3 and P_6 and increase the distance from both clients until we get at distance $L^*=5.5$ from V_1 . Location V_1 is, in this sense, critical whereas V_3 is non-critical. Although there is nothing we can do about the critical facility, it is reasonable to try and locate the non-critical facility as far away as possible from its nearest demand points under the restriction that it remains at least L^* away from the critical one. Even though this alternative solution yields the same value of the objective function as the one given by the two-facility algorithm, it is more satisfactory for realistic situations since the non-critical facility can, in general, be more than L^* away from its nearest client(s). This philosophy is known as **lexicographic optimization** (see Brady, Rosenthal and Young (1983)). A method which uses the solution given by the two-facility algorithm to achieve lexicographic optimization for problem (P14) is described below.

6.4.3 An Enhancement of the Two-Facility Algorithm

Let L^* be the optimal solution to problem (P14). Let also X_1^* and X_2^* be the optimal locations of the undesirable facilities and VS the list of vertices of S_j's as given by TWOPROFLAWLP.

Definition 6.1

An undesirable facility located at X^* is called **critical** if any movement in the ϵ-neighbourhood of X^* away from X^* will decrease L^* , and **non-critical** otherwise.

Note that if the maximum distance between the vertices of VS is equal to L^*

then both facilities are critical since any movement away from their current locations will bring them either closer to a demand point or closer to each other.

The process of improving the solution given by TWOPROFLAWLP to achieve lexicographic optimization is described below:

Method for Improving the Solution

1. Based on definition 6.1 find the critical facility(ies).
2. If both facilities are critical then stop.
   Otherwise, let X_C^* be the location of the critical facility and X_N^* the location of the non-critical one and go to Step 3.
3. Let M be the rectangle with the largest upper bound.
   Set $D^*=L^*$ and $X^*=X_N^*$.
4. If $UB_M \leq D^*$ stop.
   Otherwise go to Step 5.
5. Use the procedure for Partly Feasible cells given in chapter 4 to find the optimal location X(x, y) inside M under the additional constraint:
   $d_r(X, X_C^*) \geq L^*$ (1)
   Let D_0 be the optimal solution.
6. If $D_0 \geq D^*$ then set $D^*=D_0$ and $X^*=X$.
7. Let M be the rectangle with the next largest upper bound.
   Return to Step 4.

Constraint (1) is a simple linear constraint depending on the location of the critical facility with respect to a particular rectangle M. The critical facility in example 6.2 is located at (1.5, 10). Consequently, constraint (1) for rectangle 3 which has the largest upper bound becomes: $(x - 1.5) + (10 - y) \geq 5.5$.

It can be seen that the optimal locations for rectangle 7 under the new constraint are all points along the segment E_1E_2 in figure 6.6, at distance $D^*=7$ from the nearest demand point. Since $UB_M \leq 7$ for $M \neq 3$ this is, in fact, the largest

distance we can achieve for the non-critical facility. Hence, the improved optimal solution to (P14) is $X_1^* = (1.5, 10)$, $X_2^* = X$ with $L^* = 5.5$ where X is any point along E_1E_2.

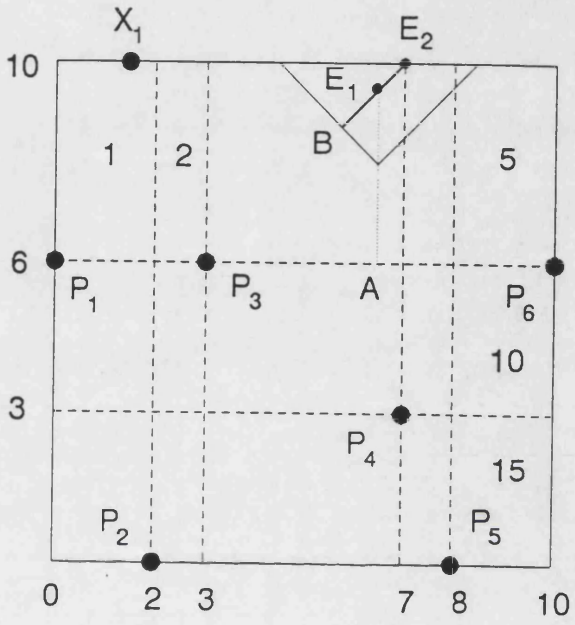

Figure 6.6: *Lexicographic optimization in example 6.2*

6.4.4 An Alternative Version of the Problem

Alternatively, constraint (2) of (P14) can be replaced by a minimum distance constraint in which case the problem becomes:

$$\max L \qquad \text{(P15)}$$

$$\text{s.t.} \quad w_i d_r(P_i, X_j) \geq L \qquad i=1,\ldots,n \text{ and } j=1,2$$

$$d_r(X_1, X_2) \geq c_{12}$$

$$a_q x_j + b_q y_j \leq c_q \qquad j=1,2 \text{ and } q=1,..,m$$

where c_{12} is the minimum distance between the two facilities to be located. A feasible solution to (P15) is a pair of vertices of VS which are at least c_{12} away from each other. With that slight modification TWOPROFLAWLP can be used to solve

problem (P15) as well.

Moreover, lexicographic optimization for (P15) can be achieved using the ideas described in the previous paragraph. The critical facility is identified based on definition 6.1. Note that in the case of (P15) both facilities are critical when the maximum distance between the vertices of VS is c_{12}. Having identified the critical facility(ies) we can improve the solution with respect to the non-critical one simply by replacing the additional constraint (1) by: $d_r(X, X_C^*) \geq c_{12}$.

6.5 Computational Experience

A series of 45 test problems were randomly generated to assess the performance of TWOPROFLAWLP. All problems were created using the ideas discussed in section 4.4. More specifically, the demand points were generated by a uniform probability distribution on [0, 10] on each coordinate whereas the boundary constraints were constructed as a simple polygon circumscribed around a circle of given radius centred at (0.5, 0.5). Finally, the weights were generated by a uniform distribution on [1, 5].

The problems were tested on a PS\2 with an 80386 processor. The first computational results revealed that a considerable amount of time was spent on the calculation of the initial feasible solution i.e. when the problem was solved sequentially to obtain a lower bound on the optimal distance. This was due to the fact that the construction of the grid of rectangles and the calculation of the upper bounds, which were quite costly in terms of computational complexity, had to be repeated when the problem was solved for the second time.

Consequently, rather than solving two single facility problems (method 1) we used an alternative technique (method 2) which solves only one such problem and exploits the solution to find an initial feasible solution for the two-facility problem.

Method 2 for Finding an Initial Feasible Solution

1. Solve the single facility problem. Let L_1^* be the optimal distance, X_1^* the optimal location and M_1 the rectangle containing the optimal location.
2. Let M be the rectangle following M_1 in the list of sorted rectangles.
3. Solve the problem for M, let L′ be the optimal distance and X′ the optimal location.
4. If $d_r(X_1^*, X') \geq L'$ then stop, setting $L_{min} = L'$.
   Otherwise, let M be the rectangle with the next largest upper bound and return to Step 3.

The second method was indeed significantly faster than the sequential approach especially in larger problems, as indicated by the average execution times given in the following table:

Table 6.3: Average Execution Times in Seconds

Demand Points	Constraints	Average Time using Method 1	Average Time using Method 2
20	6	6.1	4.3
40	6	43.8	27.5
60	6	163.3	103.8
20	15	6.5	5.7
40	15	46.3	29.1
60	15	180.4	109.0
20	50	7.3	6.3
40	50	56.2	36.2
60	50	194.6	114.7

6.6 Summary

In this chapter we discussed the problem of locating several undesirable facilities. We presented alternative formulations of the problem and explained the situations where each of them is more appropriate. We observed that most of the existing literature refers to problems where the feasible region is a network although problems in the two-dimensional plane are more realistic. Finally, we introduced a two-facility algorithm based on the single facility method given in chapter 4 and explained how experimental results led to a modification of the method to improve its overall performance.

CHAPTER SEVEN

CONCLUDING REMARKS

7.1 Summary

The problem of locating obnoxious facilities in a way that minimizes their undesirable effects on a given set of demand points was discussed in this thesis. Typical applications include locating industrial plants, dump sites for waste disposal or even placing pieces of hazardous equipment within a working environment. Assuming that the effect of such a facility on a given demand point is a decreasing function of the distance between them, the problem was to locate the new facilities within a specified two dimensional feasible region as far away as possible from the given set of demand points. Distances were measured either in the Euclidean or in the rectilinear metric, the latter being applicable in situations where movement is possible along a grid of roads or channels. Since an obnoxious facility can even have disastrous effects on a demand point, we considered MAXIMIN models whose objective is to maximize the minimum rather than the average distance between a facility and its nearest demand point. These models are also applicable in the location of desirable facilities which, for some reason, must be kept apart from a set of demand points or from each other (e.g. locating military installations in order to minimize the effects of an enemy attack). The objective of this thesis was to analyze the properties of this problem and exploit them to establish new, more efficient solution techniques.

Most of the literature on obnoxious facility location in the two-dimensional plane refers to single facility models. In chapter 2 we defined the single facility problem formally and discussed the properties of the optimal solution both in the

Euclidean and in the rectilinear case. We demonstrated how these properties can be viewed from different, seemingly unrelated perspectives. More specifically, we used simple geometric arguments to verify properties that have been proven algebraically in the literature. We also introduced a mixed integer programming formulation for the rectilinear single facility problem and used duality results to prove known and establish new properties of the optimal solution.

In chapter 3 we analyzed previous approaches to the single facility MAXIMIN problem. We observed that the most popular method in the Euclidean case seemed to be the enumeration of local optima whereas in the rectilinear case the dominant approach was based on linear programming (LP). We concluded that the most efficient technique in terms of computational complexity was the use of Voronoi diagrams to find the optimal solution to either the Euclidean or the rectilinear problem when all demand points are equally important. However, we did not find any application of this method in the weighted problem where each demand point is assigned a positive weighting factor, expressing its relative importance. Apparently this is due to the lack of an efficient algorithm for constructing the weighted Voronoi diagram (WVD) in the two-dimensional plane. The most flexible and efficient of all methods seem to be interactive graphical techniques which, unlike other methods, do not make any assumptions regarding the feasible region. In our opinion these techniques, which only provide approximate solutions, have not been fully exploited although they are the only ones that can be applied to realistic problems.

Chapter 4 contained a detailed discussion of the LP based approaches to the rectilinear problem. The most efficient of them seems to be the closest point algorithm introduced by Mehrez et al. (1986) for the unweighted problem where all demand points are equally important. After pointing out its theoretical significance, we modified this algorithm in order to solve the weighted problem as well. We then proved that the method has certain logical errors and as a result does not always produce the correct solution. Combining the main ideas of this algorithm with the properties of the optimal solution we developed PROFLAWLP, an algorithm which

solves the problem without using linear programming at all as opposed to all other methods. Computational results indicated that PROFLAWLP by far outperforms the closest point method in all test problems.

Chapter 5 established LOCOBNOX, a graphical model which has been implemented on a computer as an interactive process. Like previous graphical approaches, LOCOBNOX is applicable to realistic problems where the feasible region may be nonconvex or even disconnected. Since previous methods require a significant amount of user intervention, we introduced a stochastic termination rule to keep this intervention minimal. As a result we developed a realistic and efficient method which has been used effectively even in large scale problems with up to 1000 demand points. We then outlined how real life aspects, like areas of varying environmental importance, can be incorporated into the model. Hence, we argued that the flexibility of the method makes it an ideal site-generation tool, aiming to identify potential solutions which the decision maker can then assess based on a number of possibly conflicting criteria.

In chapter 6 we discussed the problem of locating more than one obnoxious facility. Most of the literature on this problem refers to models where a finite set of candidate solutions has already been specified. Although in most realistic situations the feasible region is a two dimensional area very little research has been done on this problem. We gave alternative formulations of the multifacility problem in the plane and discuss the situations where each of them is more appropriate. We then introduced TWOPROFLAWLP, an algorithm which solves the two-facility rectilinear problem without using linear programming. This algorithm is based on PROFLAWLP and exploits the geometry of the problem to find the optimal solution. To the best of our knowledge it is the only attempt to locate more than one facility within a planar area.

7.2 Future Work

It has to be said that the models discussed in this thesis are by no means precise representations of the real life problem of locating obnoxious facilities. Consequently, there are many more interesting avenues of research some of which are outlined below.

1) Although the single facility problem on the plane is well documented there is a severe lack of published methods on the multifacility problem. We plan to investigate the geometry of this problem and try to extend the two-facility algorithm of chapter 6 for more facilities.

2) Clearly, the problem of locating undesirable facilities is a multiobjective one. Apart from the distance considerations, economic and social issues are also relevant. Ignoring these issues may lead in unrealistic solutions. For example, a purely distance maximization model may locate an undesirable facility at the top of Mount Everest. Hence, single objective models are not satisfactory. We feel that emphasis should be given to multiobjective models where maximizing the distance between demand points and facilities is only one of the relevant objectives.

3) At the moment the graphical model presented in chapter 5 can be used to generate candidate solutions to the problem. We wish to exploit the ideas introduced by Banerjee et al. (1992) and develop this model into a decision support system where candidate solutions will be evaluated on the basis of pre-determined quantitative or qualitative criteria which may be associated with certain demand points or subsets of the feasible region. An object-oriented programming environment seems to be the most appropriate setting for such a model.

4) In chapter 2 we discussed the use of Voronoi diagrams in the single facility problem. Recall that the only published method for constructing the weighted Euclidean Voronoi diagram (WVD) is extremely complex and has not been implemented. It may be possible to design and implement an efficient algorithm for constructing the WVD and extend it to the rectilinear metric as well. Such an algorithm would form the basis of a solution method for the weighted problem which would be of lower computational complexity than all existing techniques.

5) The single facility algorithm of chapter 4 involves the division of the feasible region into rectangular areas and the calculation of an upper bound for each of them. Since this process can be done in parallel, it may be worth investigating the use of parallel algorithms to solve the problem efficiently.

6) Some of the multifacility problems presented in chapter 6 have been formulated as mixed integer programming (MIP) models. Recall that in chapter 2 we used duality results from an MIP to prove known and reveal new properties of the single facility problem. A natural extension is to examine whether similar results can be derived for the multifacility problem as well.

7) Finally, it is worth examining whether the problem can be formulated and solved using the concept of **minimax** algebra as defined by Cunninghame-Green (1991). Minimax algebra is the system $M=\{\mathbf{R}\cup\{-\infty,+\infty\}, \otimes, \oplus, \otimes', \oplus'\}$ where $x\oplus y=\max\{x, y\}$, $x\otimes y=x+y$, $x\oplus' y=\min(x, y\}$ and $x\otimes' y=x+y$.

Whether these avenues of research will be explored or not remains to be seen. One thing that is certain, though, is our desire to follow them, preferably in the near future.

APPENDICES

APPENDIX A

EUCLIDEAN WEIGHTED BISECTORS

Let $P_1 (x_1 , y_1)$ and $P_2 (x_2 , y_2)$ be two points in the two-dimensional plane with weights w_1 and w_2 respectively. Without loss of generality assume that $w_2 > w_1$ .The locus of points X(x, y) which are at weighted distance L from both P_1 and P_2 is given by the following equation:

$$w_1 * d_E (X, P_1) = w_2 * d_E (X, P_2) = L, \quad \text{or}$$

$$w_1^2 * [(x-x_1)^2 + (y-y_1)^2] = w_2^2 * [(x-x_2)^2 + (y-y_2)^2] \quad <=>$$

$$w_1^2 * [(x-x_1)^2 + (y-y_1)^2] * (w_2-w_1) = w_2^2 * [(x-x_2)^2 + (y-y_2)^2](w_2-w_1) \quad <=>$$

$$w_2^4 * [(x-x_2)^2 + (y-y_2)^2] + w_1^4 * [(x-x_1)^2 + (y-y_1)^2] = w_1^2 w_2^2 * [(x-x_1)^2 + (y-y_1)^2 + (x-x_2)^2 + (y-y_2)^2] \quad <=>$$

$$w_2^4 * [(x-x_2)^2 + (y-y_2)^2] + w_1^4 * [(x-x_1)^2 + (y-y_1)^2] = w_1^2 w_2^2 * (x_1^2+x_2^2-2x_1x_2+y_1^2+y_2^2-2y_1y_2+2x^2-2x_1x -2x_2x+2x_1x_2+2y^2-2y_1y-2y_2y+2y_1y_2) \quad <=>$$

$$w_2^4(x-x_2)^2+w_1^4(x-x_1)^2-2w_1^2w_2^2(x-x_1)(x-x_2) + w_2^4(y-y_2)^2+w_1^4(y-y_1)^2-2w_1^2w_2^2(y-y_1)(y-y_2) = w_1^2w_2^2 * [(x_1-x_2)^2 + (y_1-y_2)^2] \quad <=>$$

$$[w_2^2(x-x_2)-w_1^2(x-x_1)]^2 + [w_2^2(y-y_2)-w_1^2(y-y_1)]^2 =$$
$$w_1^2w_2^2 * [(x_1-x_2)^2 + (y_1-y_2)^2] \qquad <=>$$

$$(w_2^2x-w_1^2x-w_2^2x_2+w_1^2x_1)^2 + (w_2^2y-w_1^2y-w_2^2y_2+w_1^2y_1)^2 =$$
$$w_1^2w_2^2 * [(x_1-x_2)^2 + (y_1-y_2)^2] \qquad <=>$$

$$[x-(w_2^2x_2-w_1^2x_1)/(w_2^2-w_1^2)]^2 + [y-(w_2^2y_2-w_1^2y_1)/(w_2^2-w_1^2)]^2 =$$
$$[(x_1-x_2)^2 + (y_1-y_2)^2] * [w_1w_2/(w_2^2-w_1^2)]^2 \qquad (1)$$

If $x_0 = (w_2^2x_2-w_1^2x_1)/(w_2^2-w_1^2)$, $y_0 = (w_2^2y_2-w_1^2y_1)/(w_2^2-w_1^2)$ and
$r = d_E(P_1 , P_2) * w_1w_2/(w_2^2-w_1^2)$ then (1) is equivalent to:

$$(x-x_0)^2 + (y-y_0)^2 = r^2$$

which is the equation of a circle with centre (x_0 , y_0) and radius r.

APPENDIX B

AN ALGORITHM FOR IDENTIFYING INFEASIBLE AND PARTLY FEASIBLE RECTANGLES FOR LP-BASED METHODS

Suppose that the feasible region S is described by a list of extreme points V_j, given in clockwise order rather than as a set of linear constraints. Let also XL be the smallest and XU the largest x-coordinate of all V_i. YL and YU are defined similarly.

In order to construct the grid of $(n+1)^2$ rectangles we need a list of the x-coordinates of the P_i's in ascending order and a similar list of their Y-coordinates. In general these lists are:

XList : XL, x_1, x_2, ..., x_n, XU

YList : YL, y_1, y_2, ..., y_n, YU

where X_j and Y_j do not necessarily correspond to the same demand point j.

Consider a particular constraint G: $V_s \rightarrow V_t$ where $X_s \leq X_t$ and let the line between V_s and V_t be given by H: $ax + by = c$. Constraint G is called "upper" ("lower") if the feasible region S is below (above) G. Equivalently, G is "upper" ("lower") if for every demand point $P_i(x_i, y_i)$, y_i is less (greater) than the y-value of H for $x=x_i$. This can be found very simply by checking whether any demand point is above or below G.

In order to find the rectangles which are intersected by G we start from the vertex V_s (X_s, Y_s) with the smallest x-coordinate and progress towards the end-point V_t (X_t, Y_t) calculating the values of the corresponding constraint for each value in the XList between X_s and X_t. More simply, suppose that the coordinates of V_s and V_t are in the following ranges of the XList and the YList:

..., x_i, x_s, x_{i+1}, ..., x_{i+k}, x_t, x_{i+k+1}, ...

..., y_j, y_s, y_{j+1}, ..., y_{j+m}, y_t, y_{j+m+1}, ...

Clearly the constraint $V_s \rightarrow V_t$ intersects the cell with

$x_{min} = x_i$, $x_{max} = x_{i+1}$ and $y_{min} = y_j$, $y_{max} = y_{j+1}$

If the value of H for $x = x_{i+1}$ is y' such that $y_{j+u} \leq y' \leq y_{j+u+1}$, then the constraint in question also passes through the following rectangles :

$x_{min} = x_i$, $x_{max} = x_{i+1}$ and $y_{min} = y_{j+1}$, $y_{max} = y_{j+2}$

$x_{min} = x_i$, $x_{max} = x_{i+1}$ and $y_{min} = y_{j+2}$, $y_{max} = y_{j+3}$

. . . .

. . . .

. . . .

$x_{min} = x_i$, $x_{max} = x_{i+1}$ and $y_{min} = y_{j+u}$, $y_{max} = y_{j+u+1}$

and also through

$x_{min} = x_{i+1}$, $x_{max} = x_{i+2}$ and $y_{min} = y_{j+u}$, $y_{max} = y_{j+u+1}$

The rectangles which are infeasible because of constraint G can be identified as follows:

Case 1

If G is "upper" then the following rectangles are above G and, hence, are infeasible:

$x_{min} = x_i$, $x_{max} = x_{i+1}$ and $y_{min} = y_{j+u+1}$, $y_{max} = y_{j+u+2}$

$x_{min} = x_i$, $x_{max} = x_{i+1}$ and $y_{min} = y_{j+u+2}$, $y_{max} = y_{j+u+3}$

. . . .

. . . .

. . . .

$x_{min} = x_i$, $x_{max} = x_{i+1}$ and $y_{min} = y_n$, $y_{max} = YU$

Case 2

If G is "lower" the following rectangles are infeasible since they are below G:

$x_{min} = x_i$, $x_{max} = x_{i+1}$ and $y_{min} = y_j$, $y_{max} = y_{j-1}$

$x_{min} = x_i$, $x_{max} = x_{i+1}$ and $y_{min} = y_{j-3}$, $y_{max} = y_{j-2}$

. . . .

. . . .

. . . .

$x_{min} = x_i$, $x_{max} = x_{i+1}$ and $y_{min} = YL$, $y_{max} = y_1$

Continuing the same process for all X-values in the XList up to X_t we can mark all the rectangles which are intersected by that particular constraint.

For the data of example 4.1:

XL = 0, XU = 10, YL = 0 and YU = 9.

Consequently, the XList and the YList for this example are:

XList : 0, 3, 5, 8, 9, 10

YList : 0, 2, 4, 6, 8, 9

Suppose we want to find the rectangles which are intersected by constraint $V_1 \rightarrow V_2$ whose equation is y = 1.25 * x + 5 (see figure 4.4). Since the starting point V_1 is within rectangle 16, that rectangle is intersected by the constraint in question. For x = 2 which is the next largest in the XList, y = 7.5 on the edge $V_1 \rightarrow V_2$, which is between 6 and 8 in the YList. Hence this constraint intersects cell 11 which has x_{min} =0, x_{max} =2 and y_{min} =6, y_{max} =8 as well as cell 12 with x_{min} =2, x_{max} =3 and y_{min} =6, y_{max} =8. Moreover, since it is an "upper" constraint it leaves rectangles 1 and 6 infeasible since they are above it. The next largest x value is 3; for x = 3, y = 8.75 on the edge, which is between 8 and 9 in the YList, hence the constraint intersects rectangles 7 and 8 and leaves rectangle 2 infeasible. Finally, since the end point V_2 is in rectangle 3 that rectangle is also partly feasible.

APPENDIX C

A BRIEF DESCRIPTION OF LOCOBNOX

This appendix outlines the main modules of LOCOBNOX, the computer program implementing the interactive graphical approach discussed in chapter 5. The program consists of a number of units written in Turbo Pascal and can run on any IBM PC or compatible with a VGA graphics card.

The user is first asked to supply the details regarding the feasible region which may consist of up to 20 disjoint polygonal areas. The coordinates of their vertices can be entered either using the keyboard or from a text file; alternatively these areas may be drawn on the screen using the mouse. The same options are available for the description of the non-permissible areas. Finally, the coordinates of the demand points are either randomly generated or specified by the user interactively or from a text file.

Having read the input data the program applies the iterative method of chapter 5 to find an approximate solution to the problem. More specifically, it draws progressively larger shapes (circles or diamonds) around each demand point and performs the stochastic test to estimate the proportion of the area that has been covered. If this estimate exceeds the specified limit the test is repeated a certain number of times and if in every one of them the estimate is higher than the limit, the approximate solution has been found. The program then uses this solution to find the exact optimal location if the user wishes to do so.

We would like to stress at this point that this appendix is by no means a complete documentation of LOCOBNOX. It merely intends to explain the structure of the program and provide a broad outline of the most significant modules.

We start by describing the global data type definitions and variable declarations given in listing C.1 below.

Listing C.1: Definitions and Declarations

```
unit Declar ;
interface
const
   Epsilon = 1e-3 ;
   MaxPoints=1000 ;
   MaxSize=1000 ;
   ManyPoints=200 ;

   MaxPolys = 20 ;
   MaxNonPerm=20 ;

   MaxTimes=10 ;

   Limit=95 ;

type

   PointArray=array [ 1..MaxPoints ] of real ;
   Points=record
          X, Y : integer
         end ;
   PlotArray=array [ 1..MaxPoints ] of Points ;

   Polys = array [ 1..ManyPoints ] of Points ;

   FeasScreens = array [ 1..MaxPolys ] of Polys ;

   NonPermShapes= array [ 1..MaxNonPerm ] of Polys ;

   Metrics = ( Euclid, Rect, Chebyshev ) ;

   PolyArray = array [ 1..MaxPolys ] of integer ;


var
   NrPoints : integer ;
   DemCoord : PlotArray ;
   Weight : PointArray ;

   Metric: Metrics ;

   RandCoord : boolean ;

   SameWeight : boolean ;

   NewWeight : boolean ;

   UseMouse : boolean ;
```

```
    Lower : integer ;

    NrFeas : integer ;
    NrVertices : PolyArray ;
    FeasRegion : FeasScreens ;

    NrRestr : integer ;
    NonPermArea : NonPermShapes ;
    Vert : array [ 1..MaxNonPerm ] of integer ;

    XCoord, YCoord : integer ;

    XOpt, YOpt, LOpt : real ;

implementation
end.
```

Interpretation of Constants

MaxSize	: Maximum size of the x and y axis
MaxTimes	: Maximum times the test for the termination of the algorithm is repeated
Limit for	: Proportion of the screen that must be covered the algorithm to terminate

Interpretation of Variables

NrPoints	: Number of demand points
DemCoord	: Array containing the coordinates of the demand points.
Weight	: Weights corresponding to demand points
Metric	: Kind of distance metric
RandCoord	: TRUE when the demand points coordinates are randomly generated
SameWeight	: TRUE when all demand points have equal weights
NewWeight	: TRUE when the user wishes to solve the problem again using a different set of weights
UseMouse	: TRUE when the user wishes to use the mouse
Lower	: Lower bound on the optimal distance

NrFeas	: Number of feasible areas
NrVertices	: Array containing the number of vertices of each feasible area
FeasRegion	: Array of feasible polygons
NrRestr	: Number of non-permissible areas
Vert	: Array containing the number of vertices of each non-permissible area
NonPermArea	: Array of non-permissible areas
XCoord, YCoord	: x and y coordinate of approximate solution
XOpt, YOpt	: x and y coordinate of exact solution
LOpt	: Exact optimal distance

The main program body is given in listing C.2 followed by an explanation of the most important subprograms.

Listing C.2: Main Program Body

```
program LOCOBNOX ;
uses
  Dos, Crt, Graph, MousePac, Declar, Screen, InData, GenFunctions ;

begin { Main }

 InputData ( NrPoints, RandCoord, DemCoord, NrFeas, NrVertices, FeasRegion, NrRestr,
             NonPermArea, Metric, Weight, Step ) ;

 Setup ( NrPoints, RandCoord, DemCoord, NrFeas, NrVertices, FeasRegion, NrRestr,
             NonPermArea ) ;

 repeat

  MainProcedure ( NrPoints, DemCoord, XCoord, YCoord ) ;
  AskNewWeight ( NewWeight ) ;
  if NewWeight then
   begin
    FileWeights ( NrPoints, Weight ) ;
     RedrawScreen ( NrPoints, DemCoord, NrFeas, NrVertices, FeasRegion,
                    NrRestr, NonPermArea )
   end

 until not NewWeight ;
```

```
 repeat until Keypressed ;
 CloseGraph
end.
```

Unit *Graph* contains the predefined graphics routines available in Turbo Pascal whereas *MousePac* is a collection of subprograms related to the mouse. *Declar* includes the global definitions and declarations of the program. *Screen* contains subprograms that actually draw on the graphics screen whereas *InData* consists of routines associated with data input. Finally, *GenFunctions* includes several arithmetic and logical functions used in the program.

Explanation of Subprograms

Setup initialises the graphics environment and draws the feasible region on the screen. It then reads the coordinates of the demand points or generates them randomly. The whole procedure is given in listing C.3 below.

Listing C.3: Procedure Setup

```
procedure Setup ( NrPoints : integer ; RCoord : boolean ;
                  var FCoord : PlotArray ; NrFeas : integer ;
                  var NrVert : WHAT ; var FeasReg :FeasScreens;
                  NrRestr : integer ; var RestrArea : NonPermShapes ) ;
 begin { Setup }
  Initialise ( NrFeas, NrVert, FeasReg, NrRestr, RestrArea ) ;
  MakeCoord ( NrPoints, RCoord, DemCoord )
end ; { Setup }
```

If the user has already input the description of the feasible region and the non-permissible areas, in the *InputData* procedure, *Initialise* draws the corresponding polygons on the screen. On the other hand, if the user wishes to draw these polygons interactively, *Initialise* offers the option to do so using the mouse.

MakeCoord reads the coordinates of the demand points or generates them randomly if requested by the user.

MainProcedure in listing C.2 basically implements the graphical method of chapter 5. The procedure is given in listing C.4 below.

Listing C.4: Main Procedure

```
procedure MainProcedure ( NrPoints : integer var Coord : PlotArray ) ;
var
  WhichPoly : 1..MaxPolys ;
  Inside, OnNonPerm, OnBound, Exact : boolean ;
begin { MainProcedure }

  CalcBound ( Lower ) ;

  FindRegion ;

  FindCoordinates ( XCoord, YCoord ) ;

  DisplayCoordinates ( XCoord, YCoord, Exact ) ;

  if Exact then
    begin
      FindNearest ( Metric, XCoord, YCoord, Nearest ) ;
      CheckInterior ( Metric, Nearest, Inside, Xopt, YOpt ) ;
      if not Inside then
        begin
          CheckNonPerm ( Metric, XCoord, YCoord, Nearest,
                         OnNonPerm, XOpt, YOpt ) ;
          if not OnNonPerm then
            begin
              WhichPoly := FindPoly ( XCoord, YCoord ) ;
              CheckBoundary ( Metric, XCoord, YCoord,
                        Nearest, WhichPoly, OnBound, XOpt, YOpt )
            end
        end ;
      DisplayExact ( XOpt, YOpt )
    end
end ; { MainProcedure }
```

CalcBound calculates the lower bound *Lower* on the optimal distance. *FindRegion* draws the progressively larger shapes and performs the stochastic termination test. *FindCoordinates* allows the user to move the mouse and click the uncovered area to find the approximate optimum (*XCoord*, *YCoord*).

If the user wishes the exact solution the program applies the method described in appendix D. It finds the relevant demand points (*FindNearest*) depending on the distance metric and then checks whether the optimal location (*XOpt*, *YOpt*) is in the interior of the feasible region (*CheckInterior*), on a boundary edge (*CheckBoundary*) or on the boundary of a non-permissible area (*CheckNonPerm*).

Finally, *AskNewWeight* asks the user whether he/she wishes to solve the problem using a different set of weights. If so, the new weight set is read (*FileNewWeight*) and the problem is solved again.

The Pascal code for all the units of listing C.2 can be found in the final appendix of this thesis.

APPENDIX D

USING THE GRAPHICAL APPROACH TO OBTAIN THE EXACT SOLUTION TO THE SINGLE FACILITY PROBLEM

In chapter 5 we explained how the interactive graphical approach can be used as a site generation tool to identify potential locations for the single facility problem. We pointed out that the solutions produced by this method are approximate, since the process terminates when the user regards the uncovered area(s) sufficiently small to be considered as points. However, especially in small problems, an approximate solution may not be satisfactory; the exact optimum may be required. In this section we will demonstrate how the graphical approach can be combined with the properties of the optimal solution, as given in chapter 2, to produce the exact solution to the single facility problem. Note that the technique presented below is by no means the best way of finding the exact optimum. It merely shows that it is possible to obtain the exact solution from a graphical method.

Consider problem (P1) of chapter 5 and let X^* (x^*, y^*) be the exact optimal location and L^* the exact optimal distance. Let also X^a (x^a, y^a) be the approximate solution produced by LOCOBNOX and L^a the corresponding distance. As explained in chapter 2, X^* will be :

(i) On a vertex of a feasible polygon S_j, at distance L^* from at least one demand point, or

(ii) On the boundary of an S_j, but not on a vertex, at distance L^* from at least two demand points, or

(iii) In the interior of an S_j. In the Euclidean metric such a location must be equidistant from at least three demand points. In the rectilinear metric it will occur along a $\pm 45^o$ segment, both ends of which are equidistant from at least three clients, while all points in between are equidistant from at least two.

Finally, the optimal location may turn out to be :

(iv) On the boundary of a non-permissible area N_k (either on a vertex or on a side). If X^* is on a vertex V of N_k there will be at least one client at distance L^* from V. If X^* is along edge w of N_k, but not on a vertex, at distance L^* from only one client P_i we can always move it towards one of the endpoints of w and increase its distance from P_i. Hence, X^* on an edge w must be equidistant from at least two demand points both in the Euclidean and the rectilinear case.

If the optimal location was on a vertex of S or N, it would have been identified at the first step of LOCOBNOX, namely the calculation of the lower bound L_0, where the objective function is evaluated at all vertices of S and N. Consequently, the whole of the feasible region would be covered after the first iteration and there would be no uncovered areas. In other words, the existence of uncovered areas implies solutions in the interior or along the boundary of S or even on the boundary of a non-permissible region N_k.

Without loss of generality, suppose that the rectilinear metric is adopted and that the optimal location is in the interior of a feasible polygon S_j. See figure D.1 in the following page where ABCD is the area left uncovered after the process has terminated and X^a is the approximate optimal location, as indicated by the user clicking the mouse. Clearly, there are four "closest" demand points P_1 to P_4 corresponding to four "walls" that define the optimal area, one in the north-western, one in the north-eastern, one in the south-eastern and one in the south-western direction of X^a. Note that it would not be possible to have a situation like the one shown in figure D.2, where five walls form an L-shaped figure, because we could increase the size of the diamonds until we are left with only four walls. However, in some large scale problems with more than 100 demand points the final uncovered area is so small that it is not visually clear whether it is defined by four walls. In this cases, the analysis presented below is not valid. However, in large problems one is usually content with an approximate solution whereas in smaller problems one expects exact optima.

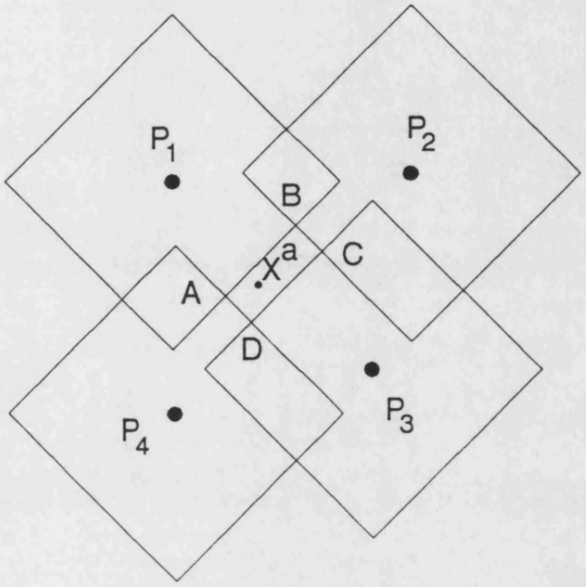

Figure D.1: *Solution in the interior (rectilinear case)*

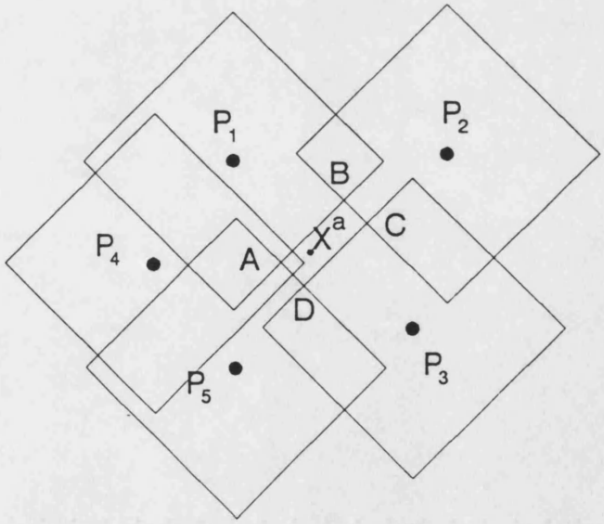

Figure D.2: *Example where the objective function may still be increased*

If the global optimum is on the boundary of an S_j (or an N_k) it can be seen that the uncovered area will have to be a triangle formed by a boundary constraint and two diamonds, as illustrated in figure D.3, where the uncovered area ABC is bounded by the constraint $Q_r \rightarrow Q_{r+1}$ and the diamonds corresponding to demand points P_1 and P_2.

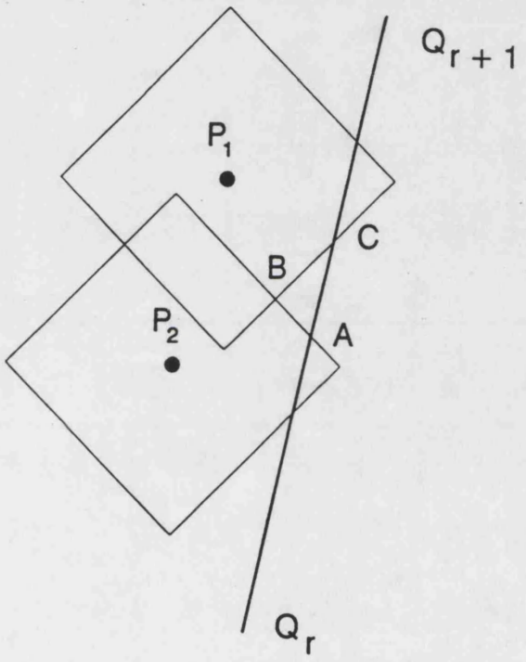

Figure D.3: *Solution on the boundary (rectilinear case)*

In the Euclidean case a solution in the interior is defined by three closest demand points, as proven in chapter 2. On the other hand, a solution on the boundary is equidistant from two clients.

Hence, given the approximate solution X^a we can find the relevant demand points in each case and compute the exact solution analytically, as explained below.

(I) Solution in the Interior

(a) Rectilinear Case

Let P_1 , P_2 , P_3 and P_4 be the nearest demand points in the north-west, north-east, south-east and south-west direction of X^a respectively.

Let also $L_{13} = d_R(P_1, P_3)(w_1 * w_3)/(w_1 + w_3)$ and

$L_{24} = d_R(P_2, P_4)(w_2 * w_4)/(w_2 + w_4)$

In chapter 2 we explained that if X^* is in the interior of S it must lie along a multiple solution edge X_1X_2 . If $L_{13} < L_{24}$ then all points of this edge will be equidistant from P_1 and P_3 as shown in figure D.4 where it is assumed, without loss of generality, that X_1 is equidistant from P_1 , P_3 and P_2 whereas X_2 from P_1 , P_3 and P_4 .

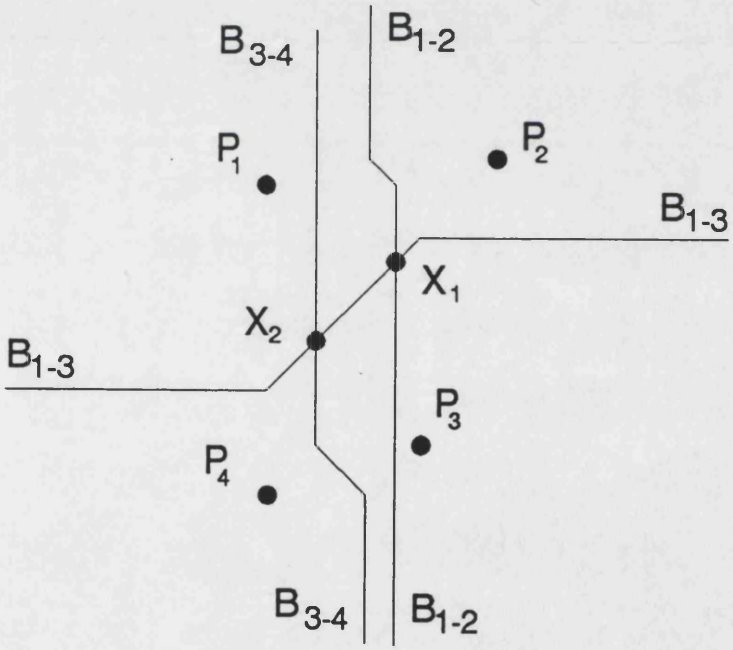

Figure D.4: *Multiple optimal edge X_1 X_2*

The coordinates of X_1 and X_2 are given by the following two systems of linear equations.

$$w_1 d_R (P_1, X_1) = L$$
$$w_3 d_R (P_3, X_1) = L$$
$$w_2 d_R (P_2, X_1) = L$$

and

$$w_1 d_R (P_1, X_2) = L$$
$$w_3 d_R (P_3, X_2) = L$$
$$w_4 d_R (P_2, X_2) = L$$

On the other hand, if $L_{13} \geq L_{24}$, X_1 will be equidistant from say, P_2, P_4 and P_1, in which case X_2 will be equidistant from P_2, P_4 and P_3 and their coordinates will be the solutions of two systems of equations similar to the above.

(b) Euclidean Case

Given X^a let P_1, P_2 and P_3 be the three nearest demand points. The optimal location X^* will be given by the following system of quadratic equations.

$$w_1 d_E (P_1, X^*) = L$$
$$w_2 d_E (P_2, X^*) = L$$
$$w_3 d_E (P_3, X^*) = L$$

(II) Solution on Edge u of S_j

Let $ax + by = c$ be the equation of the u-th side of feasible polygon S_j. If P_1 and P_2 are the two relevant demand points X^* is given by the following system of equations.

$$w_1 d(P_1, X) = L$$
$$w_2 d(P_2, X) = L$$
$$ax + by = c$$

where the distance is expressed either in Euclidean or in rectilinear terms.

(III) Solution on Edge w of N_k

If ax + by = c is the equation of the w-th side of N_k and P_1 and P_2 the relevant demand points, the optimal location X^* is given by a system of equations similar to case 1.2 above.

Hence, the whole process of using the approximate solution X^a to obtain the exact optimum X^* can be described as follows:

1. Given X^a, find the relevant demand points.

(i) rectilinear case : find the four demand points nearest to X^a as explained in (I)(a) above.

(ii) Euclidean case : find the three clients nearest to X^a.

2. Check for a solution in the interior.

(i) rectilinear case

2.1 If the clients found in step (1) are less than four then go to step 3 (solution on boundary of feas. area).
Otherwise, go to step 2.2

2.2 If at least one point X of $X_1 X_2$ is uncovered then $X^* = X$.
Otherwise, go to step 4.

(ii) Euclidean case

2.1 If point X from (I)(b) is uncovered then $X^* = X$
Otherwise, go to step 3 (solution on boundary of feas. area)

3. Check for a solution on the boundary of S.

3.1 Find which polygon S_j contains X^a.
(Use a technique described in Akl (1989). The main idea is to draw a vertical line through X^a and count the number of intersections

between this line and the edges of each S_j. If the number of such intersection points above X^a is odd, then X^a is inside S_j; otherwise it is outside. See figure D.5 for an illustration).

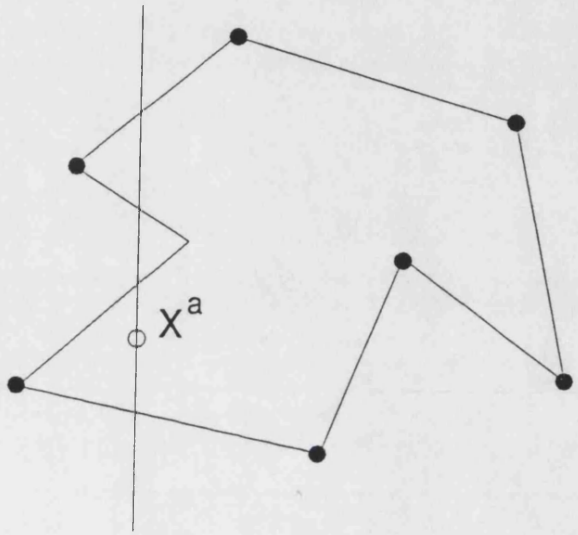

Figure D.5: *Test for inclusion of a point in a simple polygon*

3.2 For each pair of relevant clients and each edge of S_j consider point X from (II).
If X is uncovered then $X^* = X$

3.3 If no solution was found in 3.2 then go to step 4

4. Check for a solution on the boundary of an N_k

4.1 For each non-permissible area N_k repeat 4.2

4.2 For each pair of relevant clients and each edge of N_k consider point X from (III)
If X is uncovered then $X^* = X$

Hence, the main idea behind the method is to identify the closest demand points to the uncovered area and then use the properties of the optimal solution to check all candidate optima. Consequently, the process is guaranteed to find the global optimum.

However, in all experimental problems the approximate solution L^a given by LOCOBNOX was more than satisfactory since it was at least 95 % of the exact optimal distance L^*. Hence, although it can be argued that it is not necessary, the program offers the option to calculate the exact optimum if the user wishes to do so for a particular application.

APPENDIX E

FINDING A FEASIBLE SOLUTION TO THE ONE-DIMENSIONAL MULTI-FACILITY PROBLEM

Finding a feasible solution to problem (P13) of chapter 6 is essentially a set covering problem in the sense that each demand point must be covered by at least one new facility. Maintaining the notation of chapter 6, P_i is covered by candidate location A_k when $|P_i - A_k| \leq b_i$. Let $h_{ik} = 1$ if P_i is covered by A_k and 0 otherwise. Define variables $v_k = 1$ if A_k is chosen and $v_k = 0$ otherwise. The set covering problem is to find v_k for $k=1,\ldots,N$ such that $\sum_k h_{ik} v_k \geq 1$ for $i=1,\ldots,n$ and

$\sum v_k = r$.

Drezner and Wesolowsky observed that the matrix $H = \{ h_{ik} \}$ has a very interesting property. Consider the matrix for example 6.1:

Table E.1 : Matrix H for example 6.1

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
P_1	1	1	1	0	0	0	0	0
P_2	0	1	1	1	1	1	0	0
P_3	0	0	0	0	1	1	1	1

Note that in each row the sequence of 1's is uninterrupted, since the A_k's are ordered along the line. Consider columns 1 and 2 of the above table. Every 1 in column 1 corresponds to a 1 in column 2; moreover, column 2 has more 1's than column 1. Hence, site A_2 can cover all sites covered by A_1 and even more. Site A_2

by setting $v_1 = 0$. Using the concept of dominance, the procedure for finding a feasible solution can be described as follows:

Procedure for Selecting a Feasible Set of r Sites

1. Find site A_k, for the smallest k, such that A_{k+1} does not dominate it.
2. Set $v_k = 1$.
3. Delete all rows of H that correspond to demand points covered by A_k. Also, delete columns 1 to k.
4. Repeat steps 1 to 3 until all clients are covered by r or fewer columns (if fewer, place the remaining facilities on the last chosen site).

Supposing that in example 6.1 we wanted to locate 2 facilities, it can be seen that the above process would give sites A_2 and A_5 as an initial feasible solution with objective function value L=0.125.

APPENDIX F

A BRIEF DESCRIPTION OF TWOPROFLAWLP

This appendix gives a short description of TWOPROFLAWLP, the software implementing the two-facility algorithm introduced in chapter 6. The program is written in Turbo Pascal and can run on any IBM PC or compatible. It accepts as imput the coordinates of the demand points and the coordinates of the vertices of the feasible polygon and locates two obnoxious facilities in a way that maximizes the minimum distance between the demand points and the facilities to be located. This program can be used to solve the single facility problem as well since TWOPROFLAWLP starts by solving this problem to obtain an upper bound on the global optimum of the two-facility problem. After finding the optimal distance the program improves the solution with respect to the noncritical facility to achieve lexicographic optimization as explained in chapter 6.

It should be kept in mind that this appendix is by no means a complete documentation of the program. It merely provides an outline of the most significant modules.

The most significant data type definitions and variable declarations are given in listing F.1 below.

Listing F.1: Definitions and Declarations

```
unit Declar ;
interface

type
    Points = record
               X, Y : real
             end ;

    Longitudes = ( North, South ) ;
    Latitudes = ( West, East ) ;
```

```
    DemPoints = array [ 1.. MaxPoints ] of Points ;

    Weights = array [ 1..MaxPoints ] of real ;

    ExtrPoints = array [ 1..MaxVertices ] of Points ;

    State = ( Feasible, Partly, Infeasible ) ;

    RealArray = array [ 0..MaxPoints ] of real ;

    BoundArray = array [ 1..MaxRects ] of real ;

    Indexes = array [ 1..MaxRects ] of integer ;

    OptArray = array [ 1..MaxObnox ] of real ;
    LocArray = array [ 1..MaxObnox ] of Points ;

    ConLists = ^ConRec ;
    ConRec = record
               Con : integer ;
               Next : ConLists
             end ;

    Lists = ^ListRec ;
    ListRec = record
                Cell : integer ;
                ConList : ConLists ;
                Next : Lists
              end ;

var
    NrDem :  integer ;
    NrVert : integer ;
    NrRectangles : integer ;
    NrObnox : integer ;

    OptDist : OptArray ;
    OptLocation : LocArray ;

    L0, LU, L, LMin, LMax  : real ;

    Status : array [ 1..MaxRects ] of State ;

    DemCoord : Facilities ;
    Weight : Weights ;

    Vertices : ExtrPoints ;

    XArray, YArray : RealArray ;
```

```
    UArray : BoundArray ;

    OptPoint : Points ;

    IArray : Indexes ;

    CList : Lists  ;

implementation
end.
```

Interpretation of Variables

NrDem	: Number of demand points
NrVert	: Number of vertices of feasible polygon
NrRectangles	: Number of rectangles
NrObnox	: Number of undesirable facilities
OptLocation	: Array of optimal locations
Status	: Status for each rectangle as far as feasibility is concerned
DemCoord	: Details regarding demand points
Weight	: Array of weights
Vertices	: Details regarding the vertices of the feasible region
XArray	: Array of the x-ccordinates of the demand points given in ascending order after values that appear more than once have been omitted
YArray	: Same as above for y-coordinates
UArray	: Array of upper bounds

IArray	: Indices of rectangles given in descending order of the corresponding upper bound.
CList	: List of partly feasible rectangles each accompanied by a list of boundary constraints passing through it.
L	: Optimal distance
L0	: Lower bound on L
LU	: Upper bound on L

The main body of the program is given in listing F.2.

Listing F.2: Main Program Body

```
program MULTIPLE ;
uses
  Dos, Crt, Declar, GenFunct, FindOptimum ;

begin { Main }

  InputData ( NrDem, NrVert, NrObnox, DemCoord, Vertices, XArray, YArray ) ;

  PreProcessData ( NrFac, DemCoord, XArray, YArray ) ;

  FindStatus ( NrRectangles, Status ) ;

  ProcessData ( NrRectangles, UArray, IArray ) ;

  FindSolution ( LU, OptLocation [ 1 ], Rect ) ;

  FindFeasSolution ( NrObnox, Rect, L0 ) ;

  MainProcedure ( NrObnox, Rect, OptLocation, L ) ;

  ImproveSolution ( L, OptLocation ) ;

  PrintSolution ( output, L, OptLocation ) ;

end. { Main }
```

Unit *Declar* contains the global declarations of the program. *GenFunct* consists of some general arithmetic and logical functions whereas *FindOptimum* contains the subprograms which actually solve the problem.

Explanation of Subprograms

PreprocessData accepts as input the x and y coordinates of the demand points, discards values that appear more than once and sorts the remaining ones in ascending order.

ProcessData calculates the upper bound for each rectangle using the ideas of chapter 4 and sorts the array of upper bounds (*UArray*) in descending order. (NB. Actually it sorts the array of indices *IArray* rather than the upper bounds themselves so that *IArray [1]* contains the rectangle with the largest upper bound).

FindSolution essentially finds the optimum of the single facility problem as explained in chapter 4, thus providing an upper bound *LU* on the global optimum *L*. *Rect* is the rectangle containing the optimal location.

FindFeasSolution finds a feasible solution to the two-facility problem using method 2 of section 6.5. This yields a lower bound *L0* on *L*.

MainProcedure implements the bisection technique for finding the optimal solution *L* using *LU* and *L0* above. The procedure is given in listing F.3.

Listing F.3: Main Procedure

```
procedure MainProcedure ( NrObn, Rect: integer;
    var OptLoc: LocArray; var Dist: real ) ;
var
  LMin, LMax: real ;
begin { MainProcedure }
  LMin := L0 ;
  LMax := LU ;
  if NrObn = 1 then
    L := LMax
  else
    while LMax - LMin > Epsilon do
      begin
```

```
      L := ( LMin + LMax )/2.0 ;
      if ExistSol ( L ) then
        LMin := L
      else
        LMax := L
    end
end ; { MainProcedure }
```

ExistSol returns TRUE when a feasible solution with value L exists. The method for checking that is described in section 6.4.

Finally, *ImproveSolution* in listing F.2 identifies the critical facility and then improves the solution with respect to the noncritical one, thus achieving lexicographic optimization.

The remaining subprograms are fairly self explanatory. The program itself can be found in the final appendix of this thesis.

APPENDIX G

PROGRAM DISK

(Attached to the back cover of this thesis)

REFERENCES

Akl, S. (1989), *"The design and analysis of parallel algorithms"*, Prentice-Hall International Editions.

Angell, I. O. and Moore, R.E.M. "A Quad-Tree Algorithm for Displaying a 2-Dimensional Slice of an N-Dimensional Weighted Voronoi Tesselation", *Eurographics '86*, Elsevier Science Publishers BV (North-Holland), 19-27.

Appa, G. and Giannikos, I. (1992) "Using Duality Information from a Mixed Integer Program to Derive New and Old Results for the Rectilinear Obnoxious Facility Location Problem", *Working Paper Series in Operational Research*, LSE OR 92.2.

Appa, G. and Giannikos, I. (1993a), "A good enhancement with logical weaknesses", *Journal of the Operational Research Society*, 44/1, 103-106.

Appa, G. and Giannikos, I. (1993b), "Is linear programming necessary for single facility location with maximin of rectilinear distance?", to appear in the *Journal of the Operational Research Society*.

Aurenhammer, F. and Edelsbrunner, H. (1984), "An optimal algorithm for constructing the weighted Voronoi diagram in the plane", *Pattern Recognition* 17/2, 251-257.

Banerjee, P., Montreuil, B., Moodie, C.L. and Kashyap, R.L. (1992), "A modelling of interactive facilities layout reasoning using qualitative patterns", *International Journal of Productions Research* 30/3, 433-453.

Brady, S.D. and Rosenthal, R.E. (1980), "Interactive computer graphical solutions of constrained minimax location problems", *AIIE Transactions* 12, 241-248.

Brady, S., Rosenthal, R. and Young, D. (1983), "Interactive graphical minimax location of multiple facilities with general constraints", *IIE Transactions* 15/3, 242-253.

Chandrasekaran, R. and Daughety, A. (1981), "Location on tree networks: p-centre and n-dispersion problems", *Mathematics of Operations Research* 6/1, 50-57.

Cunninghame-Green, R.A. (1991), "Minimax algebra and applications", *Fuzzy Sets and Systems* 41/3, 251-267.

Dasarathy, B. and White, L.J. (1980), "A maximin location problem", *Operations Research* 28/6, 1385-1401.

Drezner, Z. (1983), "Constrained location problems in the plane and on a sphere", *IIE Transactions* 15/3, 300-304.

Drezner, Z. and Wesolowsky, G.O. (1980), "A maximin location problem with maximum distance constraints", *AIIE Transactions* 12, 249-252.

Drezner, Z. and Wesolowsky, G.O. (1983), "The location of an obnoxious facility with rectangular distances", *Journal of Regional Science* 23/2, 241-248.

Drezner, Z. and Wesolowsky, G.O. (1985), "Location of multiple obnoxious facilities", *Transportation Science* 19/3, 193-202.

Erkut, E. and Neuman, S. (1989), "Analytical models for locating undesirable facilities", *European Journal of Operational Research* 40, 275-291.

Erkut, E. and Neuman, S. (1991), "Comparison of four models for dispersing facilities", Infor 29/2, 68-85.

Erkut, E. and Öncü (1991), "A parametric 1-maximin location problem", *Journal of the Operational Research Society* 42/1, 49-55.

Hansen, P., Peeters, D. and Thisse, J.-F. (1981), "On the location of an obnoxious facility", *Sistemi Urbani* 3, 299-317.

Karkazis, J. and Karagiorgis, P. (1986), "A method to locate the maximum circle(s) inscribed in a polygon", *Belgian Journal of Operations Research, Statistics and Computer Science* 26, 4-36.

Karkazis, J. and Karagiorgis, P. (1987), "The general problem of locating obnoxious facilities in the plane", *Proceedings of the 11th IFORS Conference*, Buenos Aires, 703-717.

Kuby, M.J. (1987), "Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems", *Geographical Analysis* 19, 315-329.

Lee, D.T. (1980), "Two-dimensional Voronoi diagrams in the L_p metric", *Journal of the ACM* 27/4, 604-618.

Mehrez, A., Sinuany-Stern, Z. and Stulman, A. (1986), "An enhancement of the Drezner-Wesolowsky algorithm for single facility location with maximin of rectilinear distance", *Journal of the Operational Research Society* 37/10, 971-977.

Melachrinoudis, E. (1985), "Determining an optimum location for an undesirable facility in a workroom environment", *Applied Mathematical Modeling* 9, 365-369.

Melachrinoudis, E. (1988), "An efficient computational procedure for the rectilinear maximin location problem", *Transportation Science* 22/3, 217-223.

Melachrinoudis, E. and Cullinane, T.P. (1982), "A maximin approach to the location of an undesirable facility in a nonconvex region", *Modeling and Simulation* 13, 533-538.

Melachrinoudis, E. and Cullinane, T.P. (1985a), "A heuristic approach to the single facility maximin location problem", *International Journal of Productions Research* 23/3, 523-532.

Melachrinoudis, E. and Cullinane, T.P. (1985b), "Locating an undesirable facility within a geographical region using the maximin criterion", *Journal of Regional Science* 25/1, 115-127.

Melachrinoudis, E. and Cullinane, T.P. (1986a), "Locating an obnoxious facility within a polygonal region", *Annals of Operations Research* 6, 137-145.

Melachrinoudis, E. and Cullinane, T.P. (1986b), "Locating an undesirable facility with a minimax criterion", *European Journal of Operational Research* 24, 239-246.

Moon, I.D. and Chaudhry, S.S. (1984), "An analysis of network location problems with distance constraints", *Management Science* 30/3, 290-307.

Shamos, M.I. (1975), "Geometric complexity", *Proc. of the Seventh Ann. ACM Symposium on Theory of Comp.*, 224-233.

Shamos, M.I. and Hoey, D. (1975), "Closest-point problems", *16th Ann. Symposium on Foundation of Computer Science*, 151-162.

Shier, D.R. (1977), "A min-max theorem for p-center problems on a tree", *Transportation Science* 11/3, 243-252.

Tansel, B.C., Franscis, R.L., Lowe, T.J. and Chen, M.I. (1982), "Duality and distance constraints for the nonlinear p-center problem and covering problem on a tree network", *Operations Research* 30/4, 725-744.

TIME magazine January 2 1989, 20-24.

Wainwright M., "Heseltine rejects toxic waste plant", *The Guardian*, November 13 1991, page 5.

BIBLIOGRAPHY

Angell, I. and Griffith (1988), *"High Resolution Graphichs using Pascal"*, Macmillan Education, Macmillan Computer Science Series.

Borland's *Turbo Pascal reference manual.*

Chatterji, M. (1987), *"Hazardous Material Disposal: Siting and Management"*, Gower Publishing Co., Brookfield, VT.

Church, R.L. and Garfinkel, R.S. (1978), "Locating an obnoxious facility on a network", *Transportation Science* 12/2, 107-118.

Current, J., Min, H. and Schilling, D. (1990), "Multiobjective analysis of facility location decisions", *European Journal of Operational Research* 49, 295-307.

Elzinga, J. and Hearn, D. (1972), "Geometrical solutions for some minimax location problems", *Transportation Science* 4, 172-181.

Erkut, E. (1990), "The discrete p-dispersion problem", *European Journal of Operational Research* 46, 48-60.

Erkut, E., Baptie, T. and von Hohenbalken, B. (1990), "The discrete p-maxian location problem", *Computers in Operations Research* 17/1, 51-61.

Francis, R.L. and White, J.A. (1974), *"Facility layout and location: an analytic approach"*, Prentice-Hall, New Jersey.

Hsu, W.-L. and Nemhauser, G.L. (1979), "Easy and hard bottleneck location problems", *Discrete Applied Mathematics* 1, 209-215.

Mehrez, A., Sinuany-Stern, Z. and Stulman, A. (1983), "The one-dimensional single facility maximin distance location problem", *Journal of Regional Science* 23/2, 233-239.

Mehrez, A., Sinuany-Stern, Z. and Stulman, A. (1985), "A single facility location problem with a weighted maximin-minimax rectilinear distance", *Computers in Operations Research* 12/1, 51-60.

Minieka, E. (1983), "Anticenters and antimedians of a network", *Networks* 13, 359-364.

Moon, D. and Papayanopoulos, L. (1991), "Minimax location of two facilities with minimum separation: interactive graphical solutions", *Journal of the Operational Research Society* 42/8, 685-694.

Saaty, T.L. and Alexander, J.M. (1981), *"Thinking with models"*, Pergamon Press, Oxford.