UNDESIRABLE AND SEMI-DESIRABLE FACILITY LOCATION PROBLEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DENİZ NADİRLER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JULY 2004

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Çağlar GÜVEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Esra KARASAKAL
Supervisor

Examining Committee Members:

Prof. Dr. Murat KÖKSALAN (METU, IE)

Assist. Prof. Dr. Esra KARASAKAL (METU, IE)

Assoc. Prof. Dr. Levent KANDİLLER (METU, IE)

Assoc. Prof. Dr. Canan SEPİL (METU, IE)

Dr. Feyzan ARIKAN (GU, IE)

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Deniz NADİRLER

Signature :

ABSTRACT

UNDESIRABLE AND SEMI-DESIRABLE FACILITY LOCATION PROBLEMS

Nadirler, Deniz
M.S., Department of Industrial Engineering
Supervisor: Assist. Prof. Dr. Esra Karasakal

July 2004, 170 pages

In this thesis, single undesirable and semi-desirable facility location problems are analyzed in a continuous planar region considering the interaction between the facility and the existing demand points. In both problems, the distance between the facility and the demand points is measured with the rectilinear metric. The aim in the first part where the location of a pure undesirable facility is considered, is to maximize the distance of the facility from the closest demand point. In the second part, where the location of a semi-desirable facility is considered, a conflicting objective measuring the service cost of the facility is added to the problem of the first part. For the solution of the first problem, a mixed integer programming model is used. In order to increase the solution efficiency of the model, new branch and bound strategies and bounding schemes are suggested. In addition, a geometrical method is presented which is based on upper and lower bounds. For the biobjective problem, a three-phase interactive geometrical branch and bound algorithm is suggested to find the most preferred efficient solution.

Keywords: Location, Undesirable, Semi-desirable, Multiobjective Decision Making, Interactive Approach

ÖZ

İSTENMEYEN VE YARI-İSTENEN TESİS YERLEŞİM PROBLEMLERİ

Nadirler, Deniz
M.S., Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Y. Doç. Dr. Esra Karasakal

Temmuz 2004, 170 sayfa

Bu çalışmada, istenmeyen ve yarı istenen tesis yerleşim problemleri, sürekli bir düzlemde, tesisin varolan talep noktaları ile etkileşimi göz önüne alınarak incelenmiştir. Her iki problemde de tesis ve talep noktaları arasındaki uzaklık rektilineer metrik ile ölçülmüştür. İstenmeyen tesis yerleşiminin ele alındığı ilk kısmın amacı, tesisin en yakın talep noktasından uzaklığını en çoklamaktır. Yarı-istenen tesis yerleşiminin ele alındığı ikinci kısımda, ilk probleme, servis maliyetini ölçen ama ilk amaçla çelişen bir amaç eklenmiştir. İlk problemin çözümü için karışık tamsayılı doğrusal bir model kullanılmıştır. Modelin çözüm verimliliğini artırmak amacı ile yeni dal-sınır algoritma stratejileri ve sınırlama teknikleri kullanılmıştır. Buna ek olarak, alt ve üst sınırlara dayanan geometrik bir metod önerilmiştir. İki amaçlı problemde, en çok tercih edilen etkin noktanın bulunması için, üç fazlı etkileşimli bir geometrik dal-sınır algoritması önerilmiştir.

Anahtar Kelimeler: Yerleşim Problemi, İstenmeyen, Yarı-istenen, Çok Amaçlı Karar Verme, Etkileşimli Yaklaşım

To My Precious Family

ACKNOWDLEGMENTS

I would like to thank to my supervisor Assist. Prof. Dr. Esra Karasakal for her guidance, advices, criticism, patience, encouragements and insight throughout the research. Her understanding throughout the whole study is gratefully acknowledged.

I would like to express my deepest gratitude to my family for their endless moral support, especially to my sister without her this study would never be possible.

I also would like to thank to all of my colleagues in ACT Logistics for their understanding and encouragement throughout the thesis study.

TABLE OF CONTENTS

PLAGIARISM....................iii
ABSTRACT....................iv
ÖZ....................v
ACKNOWLEDGEMENTS....................vi
TABLE OF CONTENTS....................vii
LIST OF TABLES....................ix
LIST OF FIGURES....................xi
LIST OF ABBREVIATIONS....................xii

CHAPTER
1. INTRODUCTION....................1
2. LITERATURE REVIEW....................6
2.1. Undesirable Facility Location Problems....................6
2.1.1. 1-Maximin Problem with Euclidean Metric....................10
2.1.2. 1-Maximin Problem with Rectilinear Metric....................12
2.1.3. 1-Maxisum Problem....................14
2.1.4. A General Solution Methodology....................15
2.2. Semi-Desirable Facility Location Problems....................16
3. UNDESIRABLE FACILITY LOCATION PROBLEM....................21
3.1. Problem Definition and Assumptions....................21
3.1.1 Feasible Region....................22
3.1.2 Distance Metric....................23
3.1.3 Objective Function ‘Measuring the Social Cost’....................24
3.2.Mathematical Model....................25

3.3. Computational Experiments on *(Maximin-l^1)* with Default Branch and Bound Strategies of CPLEX..29
3.4. Solution Approaches...30
3.4.1. Investigation of the Different Branch and Bound Strategies of CPLEX..30
3.4.2. Upper and Lower Bounding Strategies..40
3.4.3. Cut and Prune Method..47
4. SEMI-DESIRABLE SINGLE FACILITY LOCATION PROBLEM.............55
4.1. Problem Definition..55
4.2. Interactive Big Square Small Square (IBSSS) Algorithm.......................71
5. CONCLUSION AND DIRECTIONS FOR FUTURE RESEARCH..........116
REFERENCES...120
APPENDICES
A. EXPERIMENTS WITH DEFAULT CPLEX STRATEGIES........................124
B. COMPARISON OF CPLEX STRATEGIES..128
C. EFFECT OF BOUNDING...156
D. INTERACTIVE BIG SQUARE SMALL SQUARE (IBSSS) ALGORITHM...161

LIST OF TABLES

Table 3.1. Computational Results…………………………………………….30

Table 3.2. Tested CPLEX Strategies and Their Default Values……………...33

Table 3.3. Results of Tested Strategies for 100 and 500 Demand Points…….35

Table 3.4. Results of Tested Strategies for 1000 and 3000 Demand Points….36

Table 3.5. Tested Combination of Strategies…..……………………………38

Table 3.6. Results of Tested Strategy Combinations...………………………39

Table 3.7. Results of the 1st Set of Problems…………………………………43

Table 3.8. Deviation of Bounds from the Optimal……………………………44

Table 3.9. Effect of Bounds…………………………………………………...46

Table 3.10. Percent Reduction in the Solution Time with Bounds…………...47

Table 3.11. Coordinates of Demand Points……...……………………………50

Table 3.12. Bounds Obtained in Subregions…...……………………………..51

Table 3.13. Bounds Obtained in Subregions..………………………………..52

Table 3.14. Subsquares Remained After Elimination..………………………53

Table 3.15. Results…...………………………………………………………54

Table 4.1. Computational Results…………….………………………………67

Table 4.2. Demand Points of Example 4.1…….……………………………..86

Table 4.3. *LCES* at the End of Phase 1……….………………………………...87

Table 4.4. *LIFV* at the End of Phase 1………..……………………………...87

Table 4.5. Elimination in the *LCES* at Phase 2………..……………………...89

Table 4.6. Percent Elimination Achieved.……………………………………90

Table 4.7. Region's Data…….…………………………………………………91

Table 4.8. *LCES* at the End of Phase 1……….……………………………...95

Table 4.9. *LIFV* at the End of Phase 1………..……………………………...96

Table 4.10. 1st Elimination in *LCES* at Phase 2……………………………98

Table 4.11. Percent Elimination Achieved…..………………………………100

Table 4.12. *LCES* at the End of Phase 2………..………………………...100

Table 4.13. Region's Data………..……………………………………………102

Table 4.14. Problem Data……….…………………………………………..105
Table 4.15. *LCES* at the End of Phase 1…………..………………………...107
Table 4.16. *LIFV* at the End of Phase 1…………..………………………...107
Table 4.17. Percent Elimination Achieved…..………………………………109
Table 4.18. Region's Data…………..…………………………………………110

LIST OF FIGURES

Figure 3.1. Finding the Smallest Distance …………………………………..…..50

Figure 3.2. Remaining Regions after the Elimination………………………...53

Figure 4.1. Altering the Reference Point……………………………………...70

Figure 4.2. Reduced Feasible Region at the End of Phase 1………………….88

Figure 4.3. Reduced Feasible Region at the End of Phase 2………………….89

Figure 4.4. Reduced Feasible Region at the End of Phase 1………………….97

Figure 4.5. Reduced Feasible Region at the End of Phase 2
(First Elimination)……………………………………………………………99

Figure 4.6. Reduced Feasible Region at the End of Phase 2
(Second Elimination)……………………………………………………...101

Figure 4.7. Jutland Area…………………………………………………..106

Figure 4.8. Reduced Feasible Region at the End of Phase 2………………...109

LIST OF ABBREVIATIONS

ASP	: Achievement Scalarizing Program
ASPLP	: Achievement Scalarizing Parametric Program
BSSS	: Big Square Small Square
DM	: Decision Maker
GBSSS	: Generalized Big Square Small Square
IBSSS	: Interactive Big Square Small Square
K-K-T	: Karush-Kuhn-Tucker
LB	: Lower Bound
LCES	: List of Candidate Efficient Squares
LCLP	: List of Candidate Location Points
LFDP	: List of Filtered Demand Points
LIFV	: List of Incumbent Function Values
LCNV	: List of Candidate Nondominated Vectors
LP	: Linear Programming
MCDM	: Multicriteria Decision Making
MIP	: Mixed Integer Programming
UB	: Upper Bound

CHAPTER 1

INTRODUCTION

Location problems are among the first optimization problems ever studied and evaded many researchers' attention for many centuries. The essence of the traditional location problems has been the location of service facilities with minimization objectives where the speed and the cost of service are the main concerns. This type of problems has been studied in a great extent in the theory of location because they are generally easy to state and understand but not that easy to solve. Warehouses, fire stations, blood centers, post offices, courier service centers, police stations and ambulance facilities can be given as examples of such facilities that provide services to population centers where the interaction generally occurs through travel distances. The main objective in the location of such traditional facilities is to locate them so that some distance function is minimized for providing the least cost service. For example, in the location of a distribution system, the main goal is generally to minimize the total distance of the facility from the existing demand points. On the other hand, in the location of facilities like fire stations or ambulance facilities, the main aim is to minimize the distance of the facility from the farthest demand point that receives the lowest quality of service.

Since the second half of the last century, recent advances and innovations in technology and industry created facilities like chemical plants, nuclear reactors, wastewater treatment plants and solid waste disposal areas having strong negative externalities on the surrounding population centers which are generally attributed with long term polluting effects. In parallel with the ever growing technology, environmental concern is increasing everyday with new regulations and stringent requirements. With these changes, researchers' interest shifted towards the location of this type of facilities starting from the early 1980's.

This new research area created a new terminology in the location theory in which the facilities with a disservice to existing population centers are called 'undesirable facilities'. The term is generally used for both 'noxious' facilities which are detrimental or hazardous to human beings and 'obnoxious' facilities that possess a threat to lifestyle through discomfort. Meanwhile, it should be noted that the solution methodologies suggested for undesirable facilities are generally valid for both types.

An interesting point is that, in spite of the undesirable effects, most of these facilities have become a vital part of our lives and the only alternative is to find ways to locate them in a way which minimizes the negative externalities on the people living around them. Considering that the undesirable effect of such facilities is a decreasing function of the travel distances, the simplest way is to locate them as far as possible from the population centers to minimize the disservice cost.

From another point of view, if undesirable facilities are of concern somehow and if there are many studies attempted to find ways to locate them, they are desirable in some extent through their service to the society, otherwise there would be no incentive to locate them. In other words, they are necessary but the long term disservice cost generally outweighs their service cost. On the other hand, it is recently realized that many facilities that have been considered as desirable so far have also some underestimated undesirable effects on people and the environment in real life situations.

As an example, recently, there is an ever-growing problem of garbage disposal and location of dump sites considering that a person living in a big city produces approximately one ton of garbage annually. According to Environmental Protection Agency (EPA) reports, there are many topographical, climatic, geological and hydrological factors that should be considered while locating a solid waste disposal area to a city. However, besides all these, keeping the cost of garbage collection low through transportation is one of the most important factors that municipalities often consider alone. On the other hand, no one wants to live close to any solid waste disposal area that is often located far from city centers because of the danger of garbage gases, unpleasant odors and noise.

The above mentioned facts triggered the definition of a new problem i.e. 'semi-desirable facility location' which balances public concerns and environmental requirements with the needs of facility planners in a sense that both nearness to a facility and protection from them are valued simultaneously.

In this thesis, we study the location of semi-desirable facilities with the motivation that inclusion of this type of facilities to our lives will continue to rise. Besides, to our knowledge, there are very few studies in this very important research area.

With this motivation, we suggest an interactive approach in which we use multiobjective decision making tools that (we believe) have been underutilized in this research area until now. Before going into details of the semi-desirable facility location problem, we first study pure undesirable facilities in order to have necessary insights that will construct a base for our solution approach to semi-desirable facility location problem.

In addition to all of the above, it should be noted that any solution methodology developed for undesirable facility location problems may apply to any problem in which a dispersed set of points is to be generated (White, 1996). For example, in multicriteria decision making, generation of a discrete set of efficient solutions that represents efficient faces (see e.g., Steuer and Harris, 1980; Karasakal and Koksalan, 2001; Sayin, 2003) is desirable with the motivation that presentation of all the efficient faces or efficient extreme points may cause information overload on the decision maker (Steuer, 1986, p.245). Sayin (2003) generated representative efficient points by solving a mixed integer programming (MIP) model iteratively. In another study, Sayin (2000) measured the quality of the representative set which is called coverage error using the same MIP model. In a recent work, Sayin (2000) illustrated that the MIP model used to generate representative points and measure the coverage error can be adapted to the problem of locating a single undesirable facility. In this thesis, we aim to improve the computational performance of the MIP model proposed by Sayin (2000, 2003). Hence, the improvements achieved in this study will be useful in finding the coverage error of a discrete representative set and generating representative efficient points.

A different example for the use of undesirable facility location models outside the location area belongs to Erkut and Neuman (1989). They mentioned that the models in undesirable facility location area can be used when a new product is to be positioned into the market, since planners generally try to design new products as different as possible from the existing products in the market.

The organization of the thesis is as follows: After a brief review of the existing literature in Chapter 2, we go into details of pure undesirable single facility location problem in Chapter 3. In this chapter, we present our assumptions that constitute the base of the mathematical model, explain the model in detail and present some computational results. We illustrate the solution approaches on some example problems.

In Chapter 4, we present our assumptions for the biobjective model. The chapter is finalized with the presentation of an interactive geometrical branch and bound algorithm based on the findings of Chapter 3. The algorithm is presented on some example problems including a real life one from the literature.

Final remarks, conclusions and directions for future research are given in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

The aim of this chapter is to briefly review the existing studies in the literature suggested for the solution of undesirable and semi-desirable facility location problems.

2.1 UNDESIRABLE FACILITY LOCATION PROBLEMS

Before presenting the literature review for undesirable facility location problems, we would like to give the general classification of the problems according to four criteria as seen below:

(1) Number of facilities to be located

- ❖ single facility
- ❖ multiple facilities

(2) Feasible region

- ❖ discrete
- ❖ continuous
- ❖ network

(3) Distance metric

- euclidean
- rectilinear

(4) Objective function

- maximin
- maxisum

Regarding the first criterion, because of the solution complexity of the multiple facility location models, most of the studies in the literature are dedicated to single undesirable facility location, several of which we overview in this chapter.

The second criterion is the type of the feasible region. There are mainly three types that can be observed in the models of this area, namely; discrete, continuous and network.

- ***Discrete*** location models are used when a facility is to be located to a site chosen among a discrete set of predetermined alternative sites. The solution methodologies for these problems are generally based on integer and combinatorial optimization techniques.

- ***Continuous*** location models are the ones where a facility is located in a m-dimensional space, R^m. The solution methodologies are mainly based on geometrical and mathematical analysis, linear and non-linear programming and global optimization methods.

- ***Network*** location models try to site a facility to a node or an edge of a graph, the edges representing the transportation links. The solution methodologies mainly rely on graph theory.

The third criterion is the distance metric used. Early models in this area generally measure the distance by using euclidean metric with the idea that pollution spreads continuously over a region. After the 1980s rectilinear metric has been introduced to the literature. In the location theory, this measure of distance is generally used when the travel between points is assumed to happen through network of streets that it is often termed Manhattan distance.

The objective function used is the fourth classification criterion. As mentioned in the previous chapter, the simplest way of locating an undesirable facility is to site it as far as possible from the population centers (demand points) to minimize its disservice cost based on the fact that undesirable effects are decreasing function of travel distances. Different objective functions have been used in the literature for this purpose; which are all based on two main types. The first is the well-known ***maximin*** objective which maximizes the distance of the facility from the closest demand point. Indeed, this objective provides the highest protection on the demand point that is most influenced by the undesirable effects of the facility. The second is the ***maxisum*** objective which considers the aggregate effect of the facility on the entire set of demand points by maximizing the total distance between them. This objective can be thought as the minimization of the disservice cost of the facility on the whole society.

A more detailed classification of the problems can be found in Erkut and Neuman (1989).

In relation to our focus in the thesis, we would like to concentrate on the part of the literature dedicated to the problem of locating a single undesirable facility in a continuous feasible region in which distances are measured either with euclidean or rectilinear metrics. Throughout this chapter, the terms ***1-maximin*** and ***1-maxisum*** are used to refer to this problem along with the objective pointed. It should be noted that the solution complexity of these problems depends on the objective function defined and the distance metric used.

Since ***1-maximin*** problem is nonlinear and nonconvex with both euclidean and rectilinear distances, it is difficult to solve and several local optima exist. On the other hand, ***1-maxisum*** problem is convex with any l_p distance, the solution of which is simpler than the former one. Hence, most of the research has been focused on the solution approaches of several versions of ***1-maximin*** problem. As for the distance metric used, rectilinear distance is piecewise linear, hence any model including this distance can be linearized in several ways. Therefore, the solution techniques are based on the extraction and solution of LP subproblems. However, this is not the case for euclidean distance since the nonlinearity cannot be removed from the model by common techniques. Hence, the approaches for this problem are generally based on geometrical means and the general theorems valid for nonlinear programming problems.

Depending on the above, we would like to review the studies on ***1-maximin*** problem briefly for euclidean and rectilinear cases first. After this, we turn our attention to the literature dedicated to ***1-maxisum*** problem. The section ends with the presentation of a general study that is applicable to all types of single facility location problems and can be adapted to the undesirable facility location problem.

2.1.1 1-MAXIMIN PROBLEM WITH EUCLIDEAN METRIC

The first related studies are Shamos (1975) and Shamos and Hoey (1975). These two studies considered the problem in one and two-dimensional space and attempted to find the optimal point in the convex hull of n demand points. In two dimensions, they constructed Voronoi polygons. These polygons, centered at each demand point, are formed by the lines that are composed of feasible points equally spaced from the demand point pairs (i.e. the bisectors of demand point pairs). They developed an algorithm based on the properties of Voronoi polygons.

Dasarathy and White (1980) considered the 1- maximin problem within a convex polyhedron. They proved the existence of finite candidate solutions in m-dimension and suggested algorithms for two and three dimensions. They attempted to find the largest hypersphere in the feasible region that does not contain any demand point, with the idea that the center point of this hypersphere is the optimal point with the objective function equal to the radius of that hypersphere. For finding this hypersphere, they used a nonconvex, nonlinear programming formulation, where the local optima were searched by an algorithm based on Kuhn-Tucker (K-T) conditions. A Lagrangian upper bound and an iteratively developed lower bound on the radius of the largest hypersphere were used to increase the efficiency of the algorithm. For the two-dimensional case they constructed Voronoi polygons, and searched the optimal point with a simpler algorithm.

Drezner and Wesolowsky (1980) dealt with 1-maximin problem within a two-dimensional continuous region. The feasible region is the intersection of the circles representing the prespecified maximum distance constraints of each demand point. In other words, the feasible region is defined by the constraints ensuring the location of the facility to be in prespecified distance of the demand points. A numerical bisection method was presented up to a prespecified precision. The aim of the study was to find the last feasible point that is not covered by any circle drawn from each demand point iteratively. They have used upper and lower bounds updated in each step of the algorithm, the difference of which determines whether to cease the algorithm.

Hansen et al. (1981) studied 1-maximin problem with a general metric in a union of finite number of polygons in R^2. They assumed a decreasing and continuous nuisance cost function. The models suggested were constructed to minimize the total nuisance cost and the maximum nuisance cost. These problems were defined as Anti-Weber and Anti-Rawls problems respectively the properties of which are exactly the same as those of the 1-maxisum and 1-maximin problems if the nuisance cost function is linear. For Anti-Rawls problem they suggested a simple geometrical method which they term 'Black and White Method' based on the elimination of parts of the feasible region with the help of incumbent values of some selected points.

Melachrinoudis and Cullinane (1985) studied the problem in a polygon in R^2, where the existing demand points were assumed to have forbidden regions around them. This assumption actually provides realism to the problem, which can be the case in most real life location problems considering the geographical barriers lying on the surface of the earth. They presented some properties of the possible location of the optimal point in the resulted nonconvex feasible region relying on K-T conditions. Based on these properties they have constructed an algorithm, which they presented on a real life application.

In a later study, Melachrinoudis and Cullinane (1986) studied the problem on a polygon in R^2. They have proved that the optimal point is either at the boundary of the feasible region or in the convex hull of demand points. In the latter case they proved that the optimal point is equidistant from at least three points, which was first proved by Dasarathy and White (1980) for the same problem in R^m. With this fact, they constructed a geometrical algorithm to search the feasible region; the average complexity of which is O (n^3), where n is the number of demand points.

Erkut and Neuman (1989) and Plastria (1996) reviewed the literature on the location of undesirable facilities, enlightening the unexplored areas of the problem and the possible areas for future studies.

Fernandez et al. (1997) have studied the same problem with Melachrinoudis and Cullinane (1985), where the feasible region was a convex polyhedron and there were protected zones surrounding each demand point. The enumeration of candidate locations for the optimal point was made possible by (K-T) optimality conditions. However, since the cardinality of candidate locations is too large, a geometrical approach was proposed, and shown on a real life example.

2.1.2 1-MAXIMIN PROBLEM WITH RECTILINEAR METRIC

To our knowledge the first study on the problem was carried out by Drezner and Wesolowsky (1983). They presented two approaches on a convex planar region based on the fact that optimal solution is either at the boundary or located in the interior points which lie on equirectilinear line of two demand points. Based on this finding, they have constructed their first algorithm based on a boundary search followed by an interior search. The idea of the second algorithm was to partition the feasible region by horizontal and vertical lines passing through each demand point. With this idea, they came up with a great number of LP's, which should be solved for each subregion resulting in O(n^2) problems, where n is the number of demand points. They proposed the use of an upper bound for each region to decrease the number of LP's to be solved in some extent.

Melachrinoudis and Cullinane (1986) studied the problem on a polygon in R^2 along with the euclidean version mentioned above. They presented three properties of the problem which enlightens the possible locations of the optimal point. Based on these properties, they suggested a geometrical algorithm the average complexity of which is O (n^2) where n is the number of demand points.

In another study, Melachrinoudis (1988) proved the properties suggested in the former research. A similar algorithm with Drezner and Wesolowsky's (1983) was presented. The same linearization technique was used which is partitioning of the feasible region from each demand point with two perpendicular lines parallel to the x and y axes. The upper bound that was used is the same as the one proposed by Drezner and Wesolowsky (1983). The only difference in their geometrical algorithm was that they solved the dual of the generated LP's with a great reduction in the size of the constraint set.

The above mentioned approaches were improved by Mehrez et al. (1986) and Appa and Giannikos (1994). In the former study, an interesting improvement was that the number of LP's constructed for each region is reduced by an approach called 'closest point approach'. They used a new upper bound calculated with the closest points which decreases the number of LP's by the elimination of some subregions. In the latter study, the authors showed that the enhancement suggested by Mehrez et al. (1986) can be further improved by exploiting the possible locations of the optimal solution. With this way, some regions can be eliminated, removing the need for storing data for each region and the optimal point for the remaining regions can be found without even using linear programming. However, the search for the optimal point is carried out in all the bisectors of demand point couples, therefore the complexity of the algorithm is directly related with the number of demand points.

In the above studies, the problem has been well studied assuming that the feasible region is in R^2. White (1996) removed the necessity for the feasible region to be a polygon and the studied the problem in R^m. He suggested the use of an algorithm that finds partial optimal solutions. The deficiency of the algorithm was due to the dependence of the final solution to the initial seed point. Besides, he presented a new upper bound valid in R^m which was proved to be better than the upper bound suggested by Drezner and Wesolowsky (1983). He noted that the upper bound can be used to check the result of the algorithm and to control initial seed values.

Sayin (2000) suggested a new solution approach to the problem with a MIP model which can be solved by any standard MIP solver. She found out that the model resulted in affordable computational times for problems of decent size. She also suggested the use of an upper bound which was the dual version of the bound suggested by White (1996).

2.1.3 1-MAXISUM PROBLEM

In their study, Hansen et al. (1981) suggested a geometrical branch and bound algorithm called 'Big Square Small Square Method' (BSSS) that deals with Anti-Weber problem where summation of some function of distances of the facility from the demand points measured with a general metric is minimized. Since the idea in this simple algorithm has been used in several studies further in the literature for different types of problems, we would like to explain the algorithm briefly. The branching consists of partitioning a square covering the feasible region with sides parallel to the axes into four equal subsquares. They suggested the use of a bound calculated with the distance of the farthest feasible points to the existing points. The elimination was made possible by the comparison of the bound with the function value of an incumbent point which is improved in each iteration. The algorithm stops when the side length reached a prespecified stopping value. Although they proved that the solution to the Anti-Weber problem is either at the convex hull of the existing points or the points of the feasible region remote from the convex hull, they have not incorporated this finding into their algorithm. In the case of a linear nuisance cost function, they proved that the optimal solution should be investigated at the extreme points of the feasible region remote from the convex hull of the existing points.

Melachrinoudis and Cullinane (1986) studied the problem on a polygon in R^2. Since they have not used any social cost function, and directly used the maxisum formulation, it was simple to prove that the optimal point should be searched at the extreme points of the feasible region both for euclidean and rectilinear case.

2.1.4 A GENERAL SOLUTION METHODOLOGY

Plastria (1992) has presented a modified version of the BSSS algorithm, named as 'Generalized Big Square Small Square Algorithm' (GBSSS) that generalizes the application of BSSS algorithm to all types of planar single facility location problems by assuming that the objective function is continuous and boxwise optimizable (i.e. 'both the minimal and maximal value of the objective function on any box can be determined without too much effort') without concerning whether the objective is a maximization or minimization in nature.

The main differences between GBSSS and BSSS algorithm are as follows: BSSS algorithm stops when the squares have a prespecified side length while GBSSS is a two-phase algorithm, based on finding the optimal value up to a prespecified relative precision in Phase 1, and a region of near optimality in Phase 2. Phase 2 of the algorithm is completely new compared to BSSS. The BSSS algorithm cuts all the feasible regions at hand into four simultaneously to simplify the list keeping operation leading the storage of unnecessary data. On the other hand, in GBSSS a single region with the best bound is divided at each step similar to the best-bound search in the standard branch and bound algorithm. It should be realized that GBSSS has several advantages over BSSS. However, the bounds that is used in the algorithm is exactly the same as in BSSS. We believe that the bounds should be improved depending on the objective function used.

2.2 SEMI-DESIRABLE FACILITY LOCATION PROBLEMS

Semi-desirable facility location problems are the ones that balance the desirable and undesirable aspects of any facility on some existing demand points. This type of problems is rarely studied in the literature, since the definition of semi-desirable facility is relatively new compared to that of undesirable facility. These problems have been defined as biobjective problems including the two conflicting objectives, where the complexity of the solution methodologies are obviously based on the objective function pairs selected and the distance metric used. As explained in the beginning of the chapter in detail, there are two types of objective functions that represent the undesirable aspects of the facility, namely, ***maximin*** and ***maxsum***. For centuries, desirable facilities are located so as to minimize the total or the maximum distance of the facility from the demand points which are referred to as ***minsum*** and ***minmax*** respectively.

The first study we have found is by Mehrez et. al. (1983). They defined a problem on a square feasible region with maximin and minmax objectives using the rectilinear distances. They assumed that the decision maker's objective is to find the location which minimizes the weighted combination of the maximin and minmax objectives. Based on this assumption, they suggested an algorithm to find an optimal solution to the weighted maximin-minmax rectilinear distance problem. They found out that the optimal points are either on the intersection points of bisectors or on the boundary of the feasible region, which was proved for any polygonal region in this study. It should be noted that a bisector of two demand points is a line formed by the points equally spaced from these two demand points. Their algorithm specifies the intersection points of concern with the help of some geometrical findings. Some points can be eliminated from further consideration as a result of the comparison with the other points. They pointed out that the algorithm has significant advantage of providing ranges for optimal solution values for all possible ranges of weights. However, they have not mentioned the efficiency of the algorithm in the existence of big sample of demand points in which case the enumaration of the candidate optimal points requires great effort.

Morales et.al. (1997) developed a global optimization approach with a global objective function including two cost functions, the first of which is nonincreasing function of distances measuring for the social cost of the facility and the second is a nondecreasing function measuring the transportation cost. These two functions are based on the total distance of the facility to all the demand points. Actually, the social cost function is equivalent to the function of Anti-Weber problem defined by Hansen et. al (1981) and the transportation function is the function used in the well known Weber problem. Their solution algorithm is based on the BSSS algorithm suggested by Hansen et. al. (1981) with an improvement in bounding scheme. They obtained an upper bound with the Lagrangian Relaxation of some constraints. They have not calculated the optimal Lagrangian multiplier; instead, they have conducted a few iterations with sub-gradient method. They have performed computational tests to see the effect of the new bound, and come up with the fact that it requires much less computational effort. However, we believe that as the number of demand points increases, the number of Lagrangian multipliers is expected to increase, making even small number of sub-gradient iterations computationally prohibitive.

Brimberg and Juel (1998) studied the problem with the following two cost functions. The one for transportation cost is the weighted sum of all the distances to the facility as in the Weber problem, while the social cost function is the minsum objective, where euclidean distance raised to a negative power. This latter objective is a new type used for the undesirable facility location problem. The authors stated that this function minimizes the combined effect of the facility on the demand points, while the decreasing marginal rates of return of distances are also taken into account, which they claimed more realistic compared to the general objectives. For this social cost function, they have proved that the optimal point is either in the convex hull of the demand points, or at the boundary, and elimination of some regions based on this fact were suggested. Since the social cost function is indeed neither convex, nor concave, they argued that several local minima exist. Their solution approach is based on the minimization of the weighted sum of the two functions as in Mehrez et al. (1983). A trajectory of efficient points is defined by a system of differential equations. The deficiency of the method is in the detection of the discontinuities of the efficient trajectory, which requires some effort.

In another study, Brimberg and Juel (1998) proposed a different approach for the solution of the unconstrained version of the problem. They used the minimization of the total weighted distance of the facility from the demand points for measuring the transportation cost by using a general metric. The social profit is maximized by maximizing the weighted euclidean distance of the facility from the closest demand point. For finding the efficient set they used two formulations the first of which is a parametric model where the sum of the weighted distances is minimized subject to constraints ensuring that the distance from the demand points must exceed some value. The minimum of this value is found by calculating the social profit of the point which minimizes the transportation cost function. They proved that the parametric solution of the model yields the efficient set. The second formulation was based on a standard procedure in multi-criteria analysis which is the construction of the weighted sum of the functions and minimizing it. They suggested a geometrical method for the solution.

The maximin-minsum objective pair with rectilinear distance metric first appeared in the study of Melachrinoudis (1999). He assumed that the feasible region is rectangular. His solution method for the nonlinear nonconvex biobjective problem was based on partitioning the feasible region into n^2 subregions where n is the number of demand points as suggested by Drezner and Wesolowsky (1983). By this partitioning, one can eliminate the nonlinearity for each objective and have n^2 LP's. Additionally, the closest point approach for the maximin objective considering the four neighboring region of each subregion suggested by Mehrez et al. (1986) was used in this study. Indeed, Drezner and Wesolowsky (1983), Melachrinoudis (1988), Mehrez et al. (1986) and Appa and Giannikos (1994) used this linearization technique, considering a single maximin objective. In this study Melachrinoudis (1999) included a conflicting objective in the same approach which is the minimization of transportation cost with minsum objective. For the adaptation of this objective, small LP's generated by partitioning the feasible region are solved as biobjective problems. When the closest point approach was used, the number of constraints for maximin objective decreased dramatically which triggered the author to use Fourier-Motzkin Elimination. With this method, n^2 rectangular subregions are generated, so that the LP's can be constructed in $O(n^2)$ time. Although, the size of the LP's are very small, and they can be solved efficiently, the number of subregions is too large which we believe makes the algorithm affordable just for decent size problems.

Carrizosa and Plastria (1999) have presented a literature review on the models suggested for the solution of semi-desirable facilities.

Skriver and Andersen (2003) studied the same problem with Brimberg and Juel (1998) where the social cost function is the sum of the weighted euclidean distances powered with a negative integer, and the transportation cost is represented as the sum of the weighted euclidean distances. They studied the problem in both planar and network cases. They followed the suggestions of Brimberg and Juel (1998) for planar problem and proposed the biobjective adaptation of the BSSS algorithm. In every branching lower bounds are found for each objective corresponding to the subregions. Subregions are eliminated whenever these lower bound pairs are dominated by any incumbent point. They presented that in some cases the optimal minsum value for a subregion can be found by checking the negative gradient at each corner point which is valid only when the direction of steepest decent points is away from the square. This finding supported the BSSS algorithm. However, they stated that most of the time this approach does not work, and in this case the bounds suggested by Hansen et. al. (1981) are used. They illustrated their solution approaches both for planar and network models on a real life example.

Melachrinoudis et. al. (2003) is the most recent study we found in the literature. They studied the maximin-minsum objective pair in a planar region with euclidean distances. In their study, they partitioned the feasible region into Voronoi polygons which was first suggested by Shamos and Hoey (1975) and developed by Dasarathy and White (1980). The complete trajectory of efficient solutions was obtained by using the Karush-Kuhn-Tucker (K-K-T) conditions along with the geometrical properties of Voronoi diagrams. They reduced the search region for efficient solutions to the points lying on the bisectors which maximize distance from a demand point to minsum contours, parts of Voronoi edges and parts of the edges of the feasible region.

CHAPTER 3

UNDESIRABLE FACILITY LOCATION PROBLEM

In this chapter, we define the problem of locating a single undesirable facility and explain the basic assumptions. We present our solution approaches and report the results of the computational experiments.

3.1 PROBLEM DEFINITION AND ASSUMPTIONS

Undesirable facilities are the facilities having negative externalities on the people living in the vicinity. Undesirable facility location problems attempt to locate such facilities as far from the population centers as possible so as to minimize their social cost.

Single undesirable facility location problems can be differentiated according to the basic assumptions regarding the underlying feasible region, the distance metric used and the objective function defined as highlighted in Section 2.1 in detail. We will explain our assumptions below.

3.1.1 FEASIBLE REGION

As mentioned in Section 2.1, there are mainly three types of feasible regions used in this area, namely; discrete, continuous and network.

Erkut and Neuman (1989) claimed that any solution methodology suggested for the undesirable facility location problem should both include a ‘site generating’ and a ‘site selection’ step. In discrete location problems one first needs to screen the candidate sites considering some factors such as geography and economy, or first needs to generate a near-optimality region from which some discrete points can be selected. Hence, discrete models can be used for site selection after the generation of several candidate sites.

Network models are appropriate when there is an existing road network eliminating the need of construction of any when a facility is to be located. However, when any undesirability is of concern, the network models are not realistic, with the idea that pollution does not spread through road networks.

Although the right choice of the feasible region actually depends on the type of the facility to be located, we believe that interaction between an undesirable facility and a community generally happens in continuous regions, hence this assumption is the most realistic in a general modeling framework.

In this study, we assume that, feasible region is continuous, and defined as a convex polygon in R^2 with k constraints.

$$S = \left\{ \boldsymbol{x} \in R^2 : \; e_j x_1 + f_j x_2 \leq g_j \quad \text{for } j = 1, \ldots, k \right\}$$

This is a realistic assumption considering that any particular shape of convex region can be approximated by a convex polygon, and facility planners in real life situations are generally faced with project layouts which happen in two dimensional space.

3.1.2 DISTANCE METRIC

It is a general claim that the distance metric should be determined considering the continuous spread of the undesirable effects; therefore network or rectilinear metrics may directly be ruled out. With this idea, euclidean metric is selected as the most appropriate measure by many researchers besides its solution complexity.

Although the general trend is the utilization of euclidean metric, we believe that this is also not realistic to model the undesirable effects like noise and air pollution. Hence, the solution complexity caused by this metric may not be worth studying in many situations considering the various factors involved in the spread of such effects like wind, geographical barriers etc. On the other hand, from our point of view, the use of rectilinear metric should not to be ruled out directly, because it can be realistic depending on the application of concern. For example, as indicated in Melachrinoudis (1999), the unpleasant effects of a facility generally spread through rectangular isles in a factory including walls. Indeed, rectilinear metric is quite widely used in the literature along with the Euclidean metric. For instance 1-maximin problem with rectilinear distance has been studied recently by White (1996) and Sayin (2000). In this study, we assume that the distance metric is rectilinear, and the distance between any two points $\boldsymbol{x}$ and $\boldsymbol{y}$ is calculated as follows:

$$d(\boldsymbol{x},\boldsymbol{y}) = |x_1 - y_1| + |x_2 - y_2|$$

3.1.3 OBJECTIVE FUNCTION 'Measuring the Social Cost'

As indicated in Section 2.1, the most famous objective functions that are used to measure the social cost associated by an undesirable facility are maximin and maxisum objectives. The first objective maximizes the protection on the demand point which is the most effected, while the latter maximizes the aggregate protection on a whole community. However, maxisum objective may result in a solution which is in the immediate neighborhood of any demand point, making this objective not preferable in most of the situations. Hence, many studies are dedicated to location models with maximin objective.

However, measurement of the social cost is too difficult that sometimes people may see the effects of a nuclear power plant after tens of years after the first interaction. As also indicated by Erkut and Neuman (1989), for a more accurate representation, the decreasing marginal rate of social cost should be considered which may level off to zero at some sufficiently far point from the demand points. With this claim, objectives consisting of linear functions of distances turned out to be unrealistic.

We believe that efforts in the utilization of objective functions measuring decreasing marginal rates of return and long term effects are justifiable only when the problem involves a facility with specific type of undesirability since the character of the social cost changes from one facility to another. Hence, from our point of view, well- known maximin or maxisum objectives are appropriate for a general modeling frame. Knowing that maxisum objective may result in an optimal location in the immediate neighborhood of a demand point which is not preferred in real life situations, we will use the most generally used one, maximin objective function.

3.2 MATHEMATICAL MODEL

Let $\boldsymbol{b^i} = (b_1^{\ i}, b_2^{\ i})$ for i=1,...,N be the coordinates of the existing demand points and $F(x_1, x_2)$ be the rectilinear distance between a facility located at (x_1, x_2) and the closest demand point to it.

$$F(x_1, x_2) = \min_{i=1,\ldots,N} \left\{ \left| x_1 - b_1^{\ i} \right| + \left| x_2 - b_2^{\ i} \right| \right\}$$

The problem then is to maximize this distance within a given convex polygon.

(M-1)

$$\begin{aligned} & Max\ F(x_1, x_2) \\ & subject \text{ to} \\ & e_j x_1 + f_j x_2 \le g_j \quad \text{for } j = 1, \ldots, k \end{aligned}$$

where e_j, f_j and g_j are constants that define the linear constraints. Since the objective function is not concave, the global optimal point cannot be guaranteed.

The model can be reformulated as follows;

(M-2)

$$\begin{aligned} & \mathrm{L}^* = Max\ \mathrm{L} \\ & subject \text{ to} \\ & L \le (\left| x_1 - b_1^{\ i} \right| + \left| x_2 - b^i{}_2 \right|) \quad \text{for } i = 1, \ldots, N \quad (1) \\ & \mathrm{e}_j x_1 + f_j x_2 \le g_j \qquad\qquad \text{for } j = 1, \ldots, k \quad (2) \end{aligned}$$

where L is the distance of the facility from the closest demand point.

Besides its nonconvexity, the model is nonlinear. The nonlinearity is because of the absolute values in the first constraint set. As explained in Chapter 2 in detail, Drezner et al. (1983) and Melachrinoudis (1988) proposed the partitioning of the feasible region into rectangular segments, by drawing vertical and horizontal lines from each demand point for the purpose of linearization. Then, to find the optimal point, $(N+1)^2$ linear programs should be solved for each subrectangle.

To linearize the problem, we use the mixed integer mathematical model proposed by Sayin (2000) in which rectilinear distance is calculated by a set of constraints controlled by integer variables.

Let $d(\boldsymbol{x},\boldsymbol{b^j})$ be the rectilinear distance between the two points $\boldsymbol{x},\boldsymbol{b^j} \in R^2$

$$d(\boldsymbol{x},\boldsymbol{b^j}) = \left|x_1 - b_1{}^j\right| + \left|x_2 - b_2{}^j\right| \quad \boldsymbol{x},\boldsymbol{b^j} \in R^2$$

Let $a_i = \left|x_i - b_i{}^j\right|$ for i = 1, 2 is the i^th^ component of $d(\boldsymbol{x},\boldsymbol{b^j})$ and can be calculated as;

$$a_i = \max\left\{(x_i - b_i{}^j), (-x_i + b_i{}^j)\right\} \text{ for i = 1, 2}$$

Then it is true that

$$a_i \geq x_i - b_i{}^j$$
$$a_i \geq -x_i + b_i{}^j$$

Since the objective is maximization, these two constraints have to be controlled somehow to guarantee that one of the inequalities holds as equality; otherwise the program would be unbounded. For this purpose, the surplus variables u_i and o_i are introduced one of which should be forced to zero.

$$a_i - u_i = x_i - b_i^j$$
$$a_i - o_i = -x_i + b_i^j$$

When the following constraints including binary variables are added to the above set, the rectilinear distance can be measured. M is a sufficiently big number.

$$t_i + z_i \le 1$$
$$u_i - Mt_i \le 0$$
$$o_i - Mz_i \le 0$$

Below model is the two-dimensional version of the MIP model suggested by Sayin (2000) which uses the set of constraints presented above for the calculation of rectilinear distance.

(Maximin - l^1)

$$L^* = Max\ L \qquad (1)$$
$$subject\ to$$
$$L \le d_j \qquad for\ j = 1,\ldots,N \qquad (2)$$
$$d_j = a_1^{\ j} + a_2^{\ j} \qquad for\ j = 1,\ldots,N \qquad (3)$$
$$a_i^j - u_i^j = x_i - b_i^j \qquad for\ j = 1,\ldots,N \quad i = 1,2 \qquad (4)$$
$$a_i^j - o_i^j = b_i^j - x_i \qquad for\ j = 1,\ldots,N \quad i = 1,2 \qquad (5)$$
$$u_i^j \le M t_i^j \qquad for\ j = 1,\ldots,N \quad i = 1,2 \qquad (6)$$
$$o_i^j \le M z_i^j \qquad for\ j = 1,\ldots,N \quad i = 1,2 \qquad (7)$$
$$t_i^j + z_i^j \le 1 \qquad for\ j = 1,\ldots,N \quad i = 1,2 \qquad (8)$$
$$e_j x_1 + f_j x_2 \le g_j \qquad for\ j = 1,\ldots,k \qquad (9)$$
$$d_j, a^j, u^j, o^j \ge 0 \qquad for\ j = 1,\ldots,N$$
$$t^j, z^j \in \{0,1\} \qquad for\ j = 1,\ldots,N$$

Parameters:

b_i^j	:	Coordinate of the j[th] demand point in i[th] dimension
M	:	Sufficiently big number
e_j, f_j, g_j	:	Constants that define the feasible region

Decision Variables:

d_j	:	Rectilinear distance of the facility from the j[th] demand point
a_i^j	:	i[th] component of d_j where $d_j = a_1{}^j + a_2{}^j$
(x_1, x_2)	:	Coordinates of the facility
u_i^j, o_i^j	:	Surplus variables associated with the j[th] demand point in i[th] dimension
t_i^j	:	$\begin{cases} 0 & \text{if } x_i > b_i^j \\ 1 & \text{otherwise} \end{cases}$
z_i^j	:	$\begin{cases} 0 & \text{if } b_i^j > x_i \\ & \text{otherwise} \end{cases}$

Above mixed integer program maximizes the distance of the facility from the closest demand point which is found by the constraint set (2). Constraint set (3) ensures that absolute distance is calculated as: $d_j = a_1{}^j + a_2{}^j = \left|x_1 - b_1{}^j\right| + \left|x_2 - b_2{}^j\right|$. Constraint sets (4)-(8) guarantee the calculation of rectilinear distance as explained before. Constraint set (9) defines the feasible region.

(Maximin-l^1) has (3 + 7N) continuous variables, (4N) binary variables and (k + 12N) constraints. Clearly, the solution complexity increases with increasing number of demand points.

3.3 COMPUTATIONAL EXPERIMENTS ON *(Maximin-l^1)* WITH DEFAULT BRANCH AND BOUND STRATEGIES OF CPLEX

In this part, *(Maximin-l^1)* model is tested to see the rate of increase of the solution time with increasing number of demand points. The feasible region is a 100 x 100 square in R^2 defined by the constraints $0 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. The location of demand points was generated according to uniform distribution in the interval [0,100]. The demand points were assumed to have equal weights based on the results of the computational experiments given in Sayin (2000) because the solution time of the weighted version of *(Maximin-l^1)* was found out to be shorter compared to the unweighted one (i.e. equal weighted version). We have used seven different problem sizes and 10 randomly generated problems were solved for each category.

The computational experiments were conducted on Pentium IV personal computer with 256 MB random access memory (RAM). The optimization models were solved in GAMS Version 20.2. The computer code that calls optimization programs was written in BORLAND C++ BUILDER Version 3. CPLEX Version 7.5 operating under GAMS Version 20.2 was used as the MIP solver. Table 3.1 illustrates the average number of branch and bound nodes and the average CPU time for each problem category. The results of all runs can be seen in **Appendix-A**.

As the problem size gets bigger, solution time of the model increases exponentially as observed from Table 3.1.The following numbers give an idea about the increase in the number of variables and constraints with the increasing number of demand points. In the case of 3000 demand points; there are 21,003 continuous variables, 12,000 binary variables and 36,002 constraints.

Table 3.1- Computational Results

Number of Demand Points	Number of Branch and Bound Nodes	CPU Time (sec)
25	44.50	0.77
50	77.70	1.78
100	138.30	5.04
500	692.30	132.98
1000	1112.50	494.61
2000	1728.10	1779.67
3000	2779.40	4969.56

As stated in Sayin (2000) and as evident from Table 3.1, *(Maximin-*l^1*)* can be solved using a standard MIP solver in reasonable computation times for small size problems; however, when the number of demand points increases, the solution time increases exponentially; which weakens the model compared to the ones suggested in the literature.

3.4 SOLUTION APPROACHES

3.4.1 INVESTIGATION OF THE DIFFERENT BRANCH AND BOUND STRATEGIES OF CPLEX

Although *(Maximin-*l^1*)* has a poor performance for big sample of demand points, we believe that the model is very practical and useful and it can also be adapted to other objective functions.

As mentioned before, *(Maximin-l^1)* is not only used in the location literature but also used in multiobjective mathematical programming context to measure the coverage error of a discrete set, which is a representation of continuous efficient faces, or used iteratively in the generation of this representative set. However, its poor computational performance for large problems causes a serious problem in the application of the model.

Moreover, in the second part of this study, we will add another objective to *(Maximin-l^1)* to model the semi-desirable facility location problem. Hence, being able to solve *(Maximin-l^1)* fast even for big size problems is very important for the generation of efficient points when the biobjective problem is of concern.

As mentioned before, we used CPLEX 7.5 as the MIP solver operating under GAMS 20.2. The default MIP strategy settings intend to solve a vast majority of MIP models with the minimum solution time. However, difficult models exist which may benefit from revision of performance measures of branch and bound algorithm (CPLEX, 1998). At this point, we attempt to see the effects of different branch and bound strategies on the solution time of *(Maximin-l^1)*.

There are several CPLEX strategies that directly affect the solution performance of the model. The most basic and well-known strategies of branch and bound algorithm are the selection of a node to branch on (NODESEL) and the selection of a variable to fix at each branching (VARSEL). Depending on the relative degradation of the objective function value at any node compared to the parent node, the solver backtracks to the pool of unexplored nodes. At this point, it can either select the last processed candidate node or any node with best bound or with best estimate function value. Once the branching node is selected, to decide on the variable to branch on is also important. Maximum or minimum infeasibility, pseudo costs or strong branching are the options for this strategy.

Gregory (2003) stated that although default settings of simplex method are difficult to be improved, in models with excessive iterations, pricing method can make a difference with parameter DPRIIND.

In addition, at each subproblem, CPLEX MIP solver generates and adds several types of cuts to restrict the feasible region to eliminate finding noninteger solutions that would otherwise be the solution of the subproblems. The solver repeats the process of adding cuts at a node until it finds no further effective cuts. Depending on the structure of the problem, sometimes adding cuts takes a long time without a considerable improvement in the solution efficiency (CPLEX, 1998).

On the other hand, there is a very important parameter, MIPEMPHASIS which decides on the orientation of the solver towards either to find succession of improving integer feasible solutions or to work toward a proof of optimality. Although the suggested default value for the majority of the models is the use of the latter one, emphasis on feasibility eliminates frequent backtracking within the tree producing faster sequence of integer solutions. Gregory (2003) stated that this may save time eliminating the need for various analysis steps performed early in the optimization.

When MIPEMPHASIS parameter is set in conjunction with other CPLEX parameters like selection of down or upward branch with command BRDIR, selection of pricing strategy with command DPRIIND or turning off all cuts with command CUTS, the resulting performance may be more productive or counter-productive (CPLEX, 1998). In most cases, the effect of the MIPEMPHASIS parameter increases when used with one of the above.

We first conducted a preliminary analysis to select strategies which may increase the computational performance of *(Maximin-l^1)* and then performed experiments with the selected strategies. The brief explanation of the tested strategies and their default values in CPLEX solver can be seen in Table 3.2.

Table 3.2 – Tested CPLEX Strategies and Their Default Values

STRATEGY	EXPLANATION
VARSEL	**Sets the rule for selecting the branching variable at the branching node** ***Default = 0 CPLEX selects the best rule based on the problem and its progress***
-1	Branch on minimum infeasibility
1	Branch on maximum infeasibility
3	Strong branching, CPLEX solves a number of subproblems to see which branch is most promising
CUTS	**Turns off the generation of all cuts at once** ***Default = YES Cut generation is allowed***
No	Turn off all types of cuts
DPRIIND	**Defines the pricing strategy for dual simplex method** ***Default = 0 CPLEX selects the best rule based on the problem and its progress***
1	Standard dual pricing
2	Steepest edge pricing
MIPEMPHASIS	**Controls the tactics of the solution, whether emphasize feasibility or optimality** ***Default = 0 Emphasize finding a proven optimal solution as quickly as possible***
1	Emphasize finding feasible solutions at the expense of spending the time required to find a proven optimal solution
BRDIR	**Decides which branch should be processed first** ***Default = 0 CPLEX selects the best rule based on the problem and its progress***
-1	Down branch selected first
1	Up branch selected first

Table 3.2 – (cont'd)

NODESEL	**Sets the rule for selecting the next node to process when backtracking occurs** ***Default = 1 Best-bound search***
0	Depth-first search. Chooses the most recently created node
2	Best-estimate search. Chooses the node with best estimate integer objective
3	Alternate best-estimate search
PRIORITY	**Gives a priority to integer variables**

Ref: (CPLEX, 1998).

At this stage, we used four different problem sizes, and for each problem size, 10 randomly generated problems were solved using each CPLEX strategy to test the effect of the strategies on the solution time of *(Maximin-l^1)*. Problem parameters and the computation environment were as in the previous tests. Average of the 10 runs can be seen in Table 3.3 and Table 3.4. The details of all these runs are presented in **Appendix- B**.

According to the results obtained, changing some of the CPLEX default strategies resulted in considerable saving in the solution time of *(Maximin-l^1)*. Although strong branching (VARSEL=3) reduced the problem size in a great extent, it increased the solution time because it partially solved a number of subproblems for variable selection at each branching node (CPLEX, 1998). Setting the variable selection strategy to 1 (VARSEL=1) worked very well in all of the samples. Actually, this rule causes larger changes in the branch and bound tree which produces faster overall solution times (CPLEX, 1998).

Table 3.3 – Results of Tested Strategies for 100 and 500 Demand Points

	Number of Demand Points			
	100		**500**	
CPLEX Strategy Used	***Number of Branch and Bound Nodes***	***CPU Time (sec)***	***Number of Branch and Bound Nodes***	***CPU Time (sec)***
DEFAULT	138.30	5.04	692.30	132.98
VARSEL –1	140.90	2.50	686.30	100.20
VARSEL 1	170.40	3.20	937.30	95.50
VARSEL 3	86.20	6.00	542.00	164.30
CUTS NO	143.90	3.20	523.30	63.30
DPRIIND 1	139.50	4.50	820.30	81.80
DPRIIND 2	165.30	7.90	768.80	246.90
MIPEMPHASIS 1	117.00	2.40	498.20	42.80
BRDIR –1	154.50	5.30	556.60	128.70
BRDIR 1	131.60	5.20	682.70	143.80
NODESEL 0	128.10	4.50	449.40	95.10
NODESEL 2	123.50	4.70	721.30	135.30
NODESEL 3	108.90	4.50	367.00	91.20
PRIORITY	135.10	4.10	653.80	108.80

It seems that changing the tactic of the branch and bound algorithm to find better integer solutions with (MIPEMPHASIS=1), turning off cut constraints with (CUTS=NO), fixing the pricing strategy with (DPRIIND=1), changing the node selection strategy with (NODESEL=0, 2, 3), branching on maximum infeasibility (VARSEL= 1) and giving a priority order to one of the integer variables with PRIORITY bring considerable improvement.

Table 3.4 – Results of Tested Strategies for 1000 and 3000 Demand Points

	Number of Demand Points			
	1000		3000	
CPLEX Strategy Used	*Number of Branch and Bound Nodes*	*CPU Time (sec)*	*Number of Branch and Bound Nodes*	*CPU Time (sec)*
DEFAULT	1112.50	494.61	2779.40	4969.56
VARSEL –1	1714.20	557.60	3356.78	3592.11
VARSEL 1	1720.50	356.50	3507.78	2750.56
VARSEL 3	1190.30	720.40	2425.11	5358.67
CUTS NO	687.10	255.50	1303.67	3037.78
DPRIIND 1	1221.60	270.30	2284.22	1828.44
DPRIIND 2	1318.00	1179.90	1900.56	8836.78
MIPEMPHASIS 1	829.00	184.30	2078.89	1944.67
BRDIR –1	1227.80	529.90	2324.00	4569.89
BRDIR 1	1213.90	517.80	2389.56	4466.22
NODESEL 0	985.50	404.10	1876.78	3073.00
NODESEL 2	1060.00	475.10	2246.33	4081.56
NODESEL 3	867.40	397.80	1606.89	3037.89
PRIORITY	1310.00	466.30	2518.22	3557.44

In contrast with strong branching, it is observed that when some strategies were used, the number of branch and bound nodes increased although the solution time decreased compared to the default case. For instance, when variable selection was based on maximum or minimum infeasibility (VARSEL=-1 and 1), the number of branch and bound nodes increased in the case of 3000 demand points, while the solution time decreased. The same situation can be observed in some other cases.

The reason may be that some strategies make the branch and bound algorithm more productive by fixing some decisions that would otherwise cost CPLEX to make some more iterations or solve partial problems at the default settings.

Indeed, CPLEX tries to make the most promising decisions to decrease the branch and bound tree size and the solution time together. However, in our case, since the branch and bound tree size is very big, making extra iterations prior to the decisions results in an increase in the computation time.

For instance, CPLEX does not solve any partial problem to select a variable at each node with (VARSEL=-1 and 1), which means it does not try to find out the most promising branch. This increased the number of branch and bound nodes. On the other hand, since the problem size is very big, when the number of demand points increases, elimination of the need for solving partial problems decreased the solution time considerably.

Although the results improved when the above mentioned strategies were used alone, their combined effect should also be tested. However, combining the strategies at once may be misleading to test this effect due to the fact that some strategies may honor the others while some of them may be counter productive together. Hence, we have tested all of the above mentioned strategies by combining them one by one. The tested combinations can be seen in Table 3.5

Table 3.5 – Tested Combination of Strategies

	COMBINATION 1	COMBINATION 2	COMBINATION 3	COMBINATION 4	COMBINATION 5	COMBINATION 6	COMBINATION 7
MIPEMPHASIS 1	√	√	√	√	√	√	√
CUTS NO	√	√	√	√	√	√	√
DPRIIND 1		√	√	√	√	√	√
VARSEL 1			√	√	√	√	√
NODESEL 0				√			
NODESEL 2					√		
NODESEL 3						√	
PRIORITY							√

10 randomly generated problems were solved for each problem type and strategy combination to test the effect of the combinations of strategies (given in Table 3.5) on the computational performance of *(Maximin-l^1)*. The results are reported in Table 3.6. Parameters and computational environment were as in the previous runs.

As seen in Table 3.6, combined strategies 2-7 decreased the solution time of (*Maximin-l^1*) model with the default strategies approximately 10 times. For example, in the case of 3000 demand points; we solved *(Maximin-l^1)* model with 21,003 continuous variables, 12,000 binary variables and 36,002 constraints in approximately 6 minutes. These combinations make the Sayin's proposed model practical even for very large problems. Since the effects of these combinations, especially combinations 3-7, were close to each other, we could select any combination among them. We have decided to use the strategy combination 3 and in the rest of the thesis we solved *(Maximin-l^1)* using this combination.

Table 3.6 – Results of Tested Strategy Combinations

	Number of Demand Points			
	100		500	
CPLEX Strategy Used	*Number of Branch and Bound Nodes*	*CPU Time (sec)*	*Number of Branch and Bound Nodes*	*CPU Time (sec)*
DEFAULT	138.30	5.04	692.30	132.98
Combination#1	107.70	1.10	476.50	27.20
Combination#2	122.00	1.10	462.90	13.70
Combination#3	101.90	1.10	405.50	11.50
Combination#4	100.40	1.00	405.50	12.00
Combination#5	101.90	1.00	405.50	11.80
Combination#6	100.40	1.00	403.90	11.40
Combination#7	97.80	1.00	396.10	12.20
	Number of Demand Points			
	1000		**3000**	
CPLEX Strategy Used	*Number of Branch and Bound Nodes*	*CPU Time (sec)*	*Number of Branch and Bound Nodes*	*CPU Time (sec)*
DEFAULT	1112.50	494.61	2779.40	4969.56
Combination#1	766.00	118.90	1828.60	1518.20
Combination#2	735.30	49.10	1609.50	429.50
Combination#3	671.10	41.30	1643.40	342.40
Combination#4	665.40	41.20	1766.80	355.10
Combination#5	671.10	40.80	1643.40	342.20
Combination#6	679.00	41.00	1641.00	333.60
Combination#7	683.40	41.70	1727.90	366.10

3.4.2 UPPER AND LOWER BOUNDING STRATEGIES

It is of common knowledge that application of bounds to any MIP model generally decreases the solution time. Keeping this in mind, we have gone through literature with the aim of finding bounds to the optimal value of *(Maximin-l^1)*, but could only find three types of upper bounds.

1) Upper Bound Suggested by Drezner and Wesolowsky (1983)

The optimal value of *(Maximin-l^1)* can be written as

$$L^* = \max_{(x_1,x_2)\in S}\left\{\min_{i=1,\dots,N}(\left|x_1-b_1^i\right|+\left|x_2-b_2^i\right|\right\}$$

where S is the feasible region and (b_1^i, b_2^i) is the coordinates of i^{th} demand point. It follows that

$$L^* \le \min_{i=1,\dots,N}\left\{\max_{(x_1,x_2)\in S}(\left|x_1-b_1^i\right|+\left|x_2-b_2^i\right|\right\}$$

Since any lp distance is convex, it attains its maximum at some vertex of S.

Let V denote the set of vertices.

$$\overline{L}^* = \min_{i=1,\dots,N}\left\{\max_{(x_1,x_2)\in V}(\left|x_1-b_1^i\right|+\left|x_2-b_2^i\right|)\right\}$$

where $\overline{L}^*$ is the upper bound on L^*

This upper bound was used by Drezner and Wesolowsky (1983) and Melachrinoudis and Cullinane (1986).

2) Upper Bound Suggested by White (1996)

$$L^* \le \min_{\lambda \in \wedge}\left\{ \max_{(x_1,x_2)\in S}\left(\sum_{i=1,\ldots,N} \lambda_i \left(\left|x_1 - b_1^i\right| + \left|x_2 - b_2^i\right|\right)\right)\right\} \le \min_{i=1,\ldots,N}\left\{ \max_{(x_1,x_2)\in S}\left(\left|x_1 - b_1^i\right| + \left|x_2 - b_2^i\right|\right)\right\}$$

where $\wedge = \left\{ \lambda \in R_+^N : \sum_{i=1..N} \lambda_i = 1 \right\}$

The proof of the above inequality can be found in White (1996). This upper bound is tighter compared to the upper bound suggested by Drezner and Wesolowsky (1983).

3) Upper Bound Suggested by Morales at al. (1997)

Morales et al. (1997) used the Lagrangian relaxation of (M-1) (see Section 3.2) as an upper bound. However, the optimal Lagrangian multiplier was not calculated, instead, a fixed number of iterations have been performed.

From the above upper bounds, the first is used in many related studies. The second upper bound suggested is tighter than the first one as proved by White (1996). However, neither its quality was tested nor it has been used in any of the algorithms in the literature. Recently, Sayin (2000) has suggested the same bound as White (1996), by giving the proof for the dual of the upper bound formulation.

Provided that the coordinates of the extreme points and the demand points are known, White's upper bound can be found by the following model, *UB(Maximin-l^1)* regardless of the distance metric used (Sayin, 2000).

$UB(Maximin - l^1)$

$$\overline{L}^* = Max\ \overline{L}$$

$$subject\ to$$

$$\overline{L} \le \sum_{j=1}^{t} \lambda_j d(v^j, b^i) \quad \text{for } i = 1,\dots,N \qquad (1)$$

$$\sum_{j=1}^{t} \lambda_j = 1 \qquad (2)$$

$$\sum_{j=1}^{t} \lambda_j v_i^{\ j} - x_i = 0 \quad \text{for } i = 1,2 \qquad (3)$$

$$e_j x_1 + f_j x_2 \le g_j \quad \text{for } j = 1,\dots,k \qquad (4)$$

$$\lambda_j \ge 0 \quad \text{for } i = 1,\dots,t$$

where

v^j : j^{th} extreme point of S

λ_j : weight associated to each extreme point.

Since the coordinates of the extreme points are known, $d(v^j, b^i)$ is just a parameter. Constraint set (3) holds provided that the feasible region is convex. Constraint set (4) defines the feasible region.

As mentioned above, to our knowledge the quality of White's upper bound has not yet been tested in the literature though it is tighter than the upper bound suggested by Drezner and Wesolowsky (1983).

In this study, we tested the performance of White's upper bound, with the idea that we could reduce the solution time of our model *(Maximin-l^1)* further by the application of an upper bound to the branch and bound tree in addition to the revised strategies. Along with questioning the quality and effect of an upper bound on our model, another aim here was to test the effect of a lower bound.

For finding a lower bound, we picked T random points from the feasible region, and selected the function value of the one whose distance to the closest demand point was the maximum.

$$LB = \max_{i=1,\dots,T} \left\{ \min_{j=1,\dots,N} (\left|x_1^i - b_1^j\right| + \left|x_2^i - b_2^j\right|) \right\}$$

where x_j^i is the j[th] coordinate of $\boldsymbol{x^i}$.

We used nine different problem sizes and 10 randomly generated problems were solved for each category. Problems were solved with and without bounds using the strategy combination 3. For finding lower bounds, 100 points were picked from each region. The results of the first set of problems are presented in Table 3.7.

Table 3.7- Results of the 1st Set of Problems

Number of Points	**Optimal Value**	**Optimal Point**	**LB**	**UB**
25	43.86	[100,0]	43.86	100
50	30.79	[83.99,50.87]	29.42	100
100	22.42	[31.17,100]	22.27	100
500	10.15	[0,40.79]	9.55	100
1000	8.97	[0,39.61]	8.65	100
2000	6.09	[0,36.73]	5.32	100
3000	4.97	[0,37.85]	4.88	100
4000	4.56	[0,100]	4.56	100
5000	3.97	[24.17,100]	3.93	100

Evident from the above table, *UB(Maximin-l^1)* model gave the same value, the side length of the feasible region, regardless of the number and the location of demand points. *UB(Maximin-l^1)* generally found the center point as the optimal.

In the remaining 9 problem sets, the upper bound was exactly the same as above. It was very loose and in all the problems it was found same as the side length of the feasible region.

Table 3.8 summarizes the results of 10 problem sets and reports the average percent deviation of both bounds from the optimal objective function value. As depicted in the table, the maximum average percent deviation of the lower bound is 8.84, while maximum average percent deviation is 96.11 for the upper bound.

Table 3.8 – Deviation of Bounds from the Optimal

Number of Demand Points	Average Percent Deviation of Lower Bound from the Optimal	Average Percent Deviation of Upper Bound from the Optimal
25	0.85	59.73
50	1.73	69.33
100	2.03	76.77
500	7.18	89.18
1000	4.77	92.31
2000	8.84	94.27
3000	7.12	95.23
4000	8.78	95.70
5000	6.12	96.11

Although the quality of the White's upper bound did not seem good, we conducted our runs to find out the effect of this bound on our model *(Maximin-l^1)* along with the lower bound.

The effect of bounds on the number of branch and bound nodes, iterations and on the CPU time can be observed from Table 3.9. 10 randomly generated problems were solved for each problem type. The detailed results of all test problems are given in **Appendix-C**. As seen from the table, incorporation of the lower bound into the model decreased the number of branch and bound nodes as well as number of iterations. On the other hand, when the upper bound was utilized, the number of branch and bound nodes increased in the samples of 1000, 2000, 4000 and 5000 demand points. After the examination of the outputs of branch and bound trees of some test problems, it was found out that the initial node changed with the utilization of the upper bound. It showed that when the upper bound was defined, the initial solution and consequently the sequence of nodes visited in the branch and bound tree has changed. This may sometimes result in an increase in the number of branch and bound nodes when the bound is not tight. It should be noted that in spite of the increase in the number of nodes in the above mentioned problems with the utilization of the upper bound, the overall number of simplex iterations decreased. When the CPU times were examined, it can be observed that the usage of both bounds had a decreasing effect on the solution time. Since the upper bound was loose in all of the runs, its additional effect was small compared to the sole effect of the lower bound. However, when the problem size got bigger (e.g. 3000, 4000 and 5000), we observed that even this loose upper bound had a considerable effect on the solution time.

Table 3.9 – Effect of Bounds

	Number of Branch and Bound Nodes			Number of Branch and Bound Iterations			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	38	29	24	795	621	551	0.47	0.40	0.37
50	61	50	30	1844	1458	1158	0.56	0.51	0.43
100	102	87	62	4486	3576	3188	1.18	0.88	0.87
500	406	383	370	37273	32926	29541	11.98	10.39	9.21
1000	671	650	685	87974	79949	69432	40.07	36.61	33.88
2000	1142	1106	1232	217171	198546	173141	150.32	134.42	129.45
3000	1643	1589	1568	407596	374645	296019	342.57	305.45	260.38
4000	1867	1832	2000	552080	519485	439273	547.32	502.16	459.08
5000	2166	2153	2267	778154	751155	600135	864.11	795.66	688.67

Below table shows the effects of bounds on the solution time of *(Maximin-l^1)* as percent reduction in CPU time. Although the quality of the upper bound was very low and it was very loose in all test problems, its additional effect on the solution time was considerable.

Table 3.10 – Percent Reduction in the Solution Time with Bounds

Number of Demand Points	Sole Effect of Lower Bound (%)	Additional Effect of Upper Bound (%)	Combined Effect (%)
25	15.68	6.57	22.25
50	9.61	14.59	24.20
100	25.59	1.27	26.86
500	13.29	9.82	23.11
1000	8.64	6.81	15.46
2000	10.58	3.31	13.89
3000	10.84	13.16	23.99
4000	8.25	7.87	16.12
5000	7.92	12.38	20.30

For example, the additional effect of the upper bound was higher than the sole effect of the lower bound in the case of 3000 demand points where the lower bound was 7.12 % deviated and the upper bound was 95.23 % deviated from the optimal value on the average. These results showed that even a very loose upper bound had an effect on the solution time and results in considerable savings.

3.4.3 CUT AND PRUNE METHOD

In this section, we tried to solve *(Maximin-l^1)* with a geometrical approach which is called 'Cut and Prune Method'. The idea is the division of the smallest rectangle covering the feasible region into subsections and elimination of some regions with the help of upper and lower bounds. Every time the region is divided, upper bound decreases, while the lower bound increases in return. The elimination of the regions occurs when the worst possible function value of any region is better than the best possible value of some others. Indeed, the efficiency of the approach totally depends on the quality of the bounds.

In the previous section we came up with the fact that in contrast with the lower bound, White's upper bound was very loose. However, we think that this performance may have depended on some problem parameters. After having solved many problems, it was observed that this condition was observed most probably due to the uniformity of the generated demand point samples. Because, even if the sample size was very big, if there was a nonuniformity in the demand points, *UB(Maximin-*l^1*)* program found other points than the center as the optimal point, and therefore found other objective function values than side length which were generally much closer to the optimal value.

Based on the above, we use White's upper bound in the 'Cut and Prune Method'. In cases where this bound fails to be tight, we propose the use of a supplementary upper bound for the subregions which is found by solving *(Maximin-*l^1*)* only with the internal demand points. This does not seem as a practical upper bound at first glance. However, as shown in Section 3.4.1 the solution time of *(Maximin-*l^1*)* decreases considerably with the new branch and bound strategies; therefore when the internal points are taken into consideration alone, the calculation of the upper bound is not expected to take much time. It should be noted that, in the proposed method, both of the above mentioned upper bounds are calculated and the smallest of them is used. The lower bound is calculated as in Section 3.4.2.

When the feasible region is divided, we eliminate some of the subregions by using above mentioned bounds. After this elimination, some subregions are left that may contain the optimal point. It should be noted that at this stage of the method, we can also select some feasible points from the remaining regions and try to further eliminate regions. At this point, the optimal solution can be found solving the model *(Maximin-l^1)* for the remaining regions. It should be noted that if the remaining regions are partly feasible then the feasibility constraints of the initial region should be added to *(Maximin-l^1)*. When we iterate further, the solution time of our MIP model is reduced. Indeed, when the special combined branch and bound strategies are used, the solution time is expected to reduce further. We claim that with this approach the overall solution time will decrease compared to the case where the problem is solved with *(Maximin-l^1)* directly.

In addition to all these, we believe that in order to find the optimal of a subregion, all the demand points need not be involved. In order to increase the efficiency of the approach, they can be filtered. The idea is that the demand points having their smallest distance to the feasible region grater than the upper bound are obviously redundant to *(Maximin-l^1)* because they just increase the number of binary, continuous variables and constraints without having any effect on the problem.

If a general l_p norm is used, it has been shown in Hansen et al. (1981) that the smallest distance can be calculated by extending the sides of a rectangle into straight lines, cutting the plane into 9 regions, namely, N(orth), S(outh), W(est), E(ast), NW, NE, SW, SE and the rectangle itself (see Figure 3.1). If a demand point is in the north side of a rectangle as in b^1 in the below figure, then the smallest distance is calculated by projecting the point onto the rectangle. Symmetrically, this is valid for South, East and West parts. If a demand point is located in the corner regions as in b^2 in the figure, then the smallest distance is calculated with the corner point of the rectangle in that region.

In order to illustrate the approach, two example problems are solved, the first of which has a small, nonuniform sample of demand points, while the other has a big, uniform one.

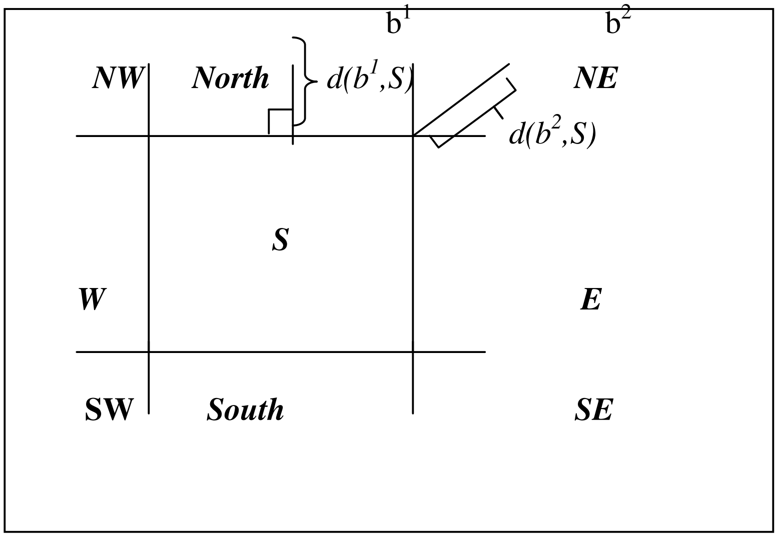

Figure 3.1-Finding the Smallest Distance

Example.1

Consider the undesirable single facility location problem with 7 demand points. The feasible region is a 30 x 30 square.

Table 3.11 – Coordinates of Demand Points

j	**1**	**2**	**3**	**4**	**5**	**6**	**7**
b^j	(5,20)	(18,8)	(2,16)	(14,17)	(7,2)	(5,15)	(12,4)

When the feasible region was divided into 4, the upper and lower bounds of the 4 subregions are calculated. (see Table 3.12)

Table 3.12 – Upper and Lower Bounds Obtained in Subregions

Square No	White's Upper Bound	Upper Bound with Interior Points	Lower Bound (100 Points)
1	20	15	11
2	20	20	20
3	16	18.33	15.50
4	29	-	29

Squares 1,2 and 3 can be eliminated since the lower bound of Square 4 is greater than the upper bound of these regions. The optimal point is in Square 4 which is the corner point: (30,30) with an objective function value of 29.

<u>**Example 2**</u>

Consider the undesirable single facility location problem with 2000 demand points generated uniformly in the interval (0,100). The feasible region is a 100x100 square.

In this problem when the feasible region was divided into 4, no elimination occurred; therefore we further divided the region into 16. With this division, the subregions have the following bounds. It should be noted that the upper bounds are calculated by solving *(Maximin-l^1)* model with the interior demand points since the White's upper bound gave the side length for all subregions (see Table 3.13).

Table 3.13 – Bounds Obtained in Subregions

Square No	White's Upper Bound	Upper Bound with Interior Points	Lower Bound (100 points)
1	25	5.35	5.17
2	25	5.61	4.78
3	25	6.11	3.63
4	25	4.40	4.32
5	25	6.10	6.04
6	25	4.80	3.58
7	25	5.47	3.85
8	25	5.75	5.64
9	25	6.45	5.34
10	25	6.56	4.67
11	25	4.84	4.06
12	25	5.37	4.32
13	25	5.39	5.21
14	25	4.84	4.83
15	25	5.56	4.60
16	25	6.90	4.60

The execution time of this step is recorded as 24.57 seconds. After elimination with bounds we were left with 5 squares (see Table 3.14), as can be seen from Figure 3.2.

Table 3.14 – Subsquares Remained After Elimination

Square No	Upper Bound	Lower Bound
3	6.11	3.63
5	6.10	6.04
9	6.45	5.34
10	6.56	4.67
16	6.90	4.60

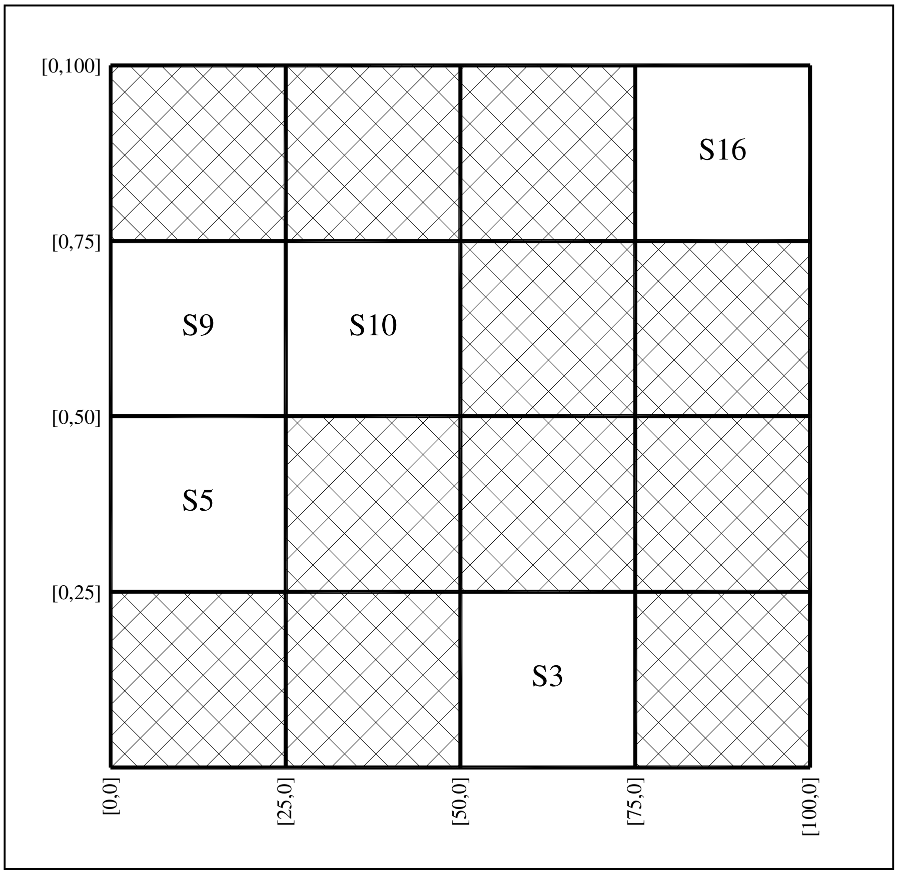

Figure 3.2 – Remaining Regions after the Elimination

For the remaining squares, the demand points are filtered as mentioned before. *(Maximin-l^1)* model is solved with the filtered demand points to find the optimal point in each subregion. The details of this step can be seen from the below table.

Table 3.15 – Results

Square No	Upper Bound	Lower Bound	Number of Demand Points after Filtration	Optimal Point	Optimal Value	CPU Time (sec)
3	6.11	3.63	229	[52.95,0]	3.95	1.86
5	6.10	6.04	214	[0,36.73]	6.09	1.02
9	6.45	5.34	220	[0,50]	5.34	1.39
10	6.56	4.67	271	[25.28,51.18]	4.87	2.00
16	6.90	4.60	196	[76.15,80.81]	4.89	0.97
				Total Execution Time		7.24

The optimal point is found to be located in Square 5. Along with the previous step the overall execution time is 31.81 seconds.

For comparison, for the same 2000 demand points, *(Maximin-l^1)* model with the default strategies was solved in 1779.67 seconds; with the special combined strategy it was solved in 150.32 seconds. Incorporation of upper and lower bounds further decreased the solution time to 129.45. Now, with the suggested approach of this section, the optimal could be found in 31.81 seconds.

CHAPTER 4

SEMI-DESIRABLE FACILITY LOCATION PROBLEM

The second part of our study focuses on the problem of locating a single semi-desirable facility. In the following sections, problem definition is given; an interactive solution algorithm is presented and illustrated on some example problems.

4.1 PROBLEM DEFINITION

A facility can be defined as semi-desirable if it has both undesirable and desirable effects to the people living in the vicinity. Although need for such facilities has been increasing rapidly, there has been not much work on the semi-desirable facility location in the literature.

The assumptions on the feasible region and the distance metric remain the same as in the undesirable facility location problem (see Chapter 3). An objective that measures the desirable aspects of the facility to be located is defined in addition to the objective used in the undesirable facility location problem. To summarize, in this problem we assume that;

❖ Feasible region S is a convex polygon in R^2 defined by k constraints

$$S = \left\{ \boldsymbol{x} \in R^2 :\ e_j x_1 + f_j x_2 \le g_j \quad \text{for } j = 1,...,k \right\}$$

❖ The distance metric is rectilinear,

$$d(\boldsymbol{x}, \boldsymbol{y}) = |x_1 - y_1| + |x_2 - y_2|$$

❖ There are N demand points with coordinates,

$$\boldsymbol{b^i} = (b_1^{\ i}, b_2^{\ i}) \quad \text{for i} = 1,..., N$$

❖ The first objective function used to model the undesirable effects is maximin which maximizes the distance of the facility from the closest demand point.

$$L^*(S) = \max_{(x_1,x_2)\in S} \left\{ \min_{i=1,...,N} (\left|x_1 - b_1^{\ i}\right| + \left|x_2 - b_2^{\ i}\right| \right\}$$

❖ The second objective function used to model the desirable effects is minsum which minimizes the total distance of the facility from the demand points.

$$W^*(S) = \min_{(x_1,x_2)\in S} \left\{ \sum_{i=1}^{N} (\left|x_1 - b_1^{\ i}\right| + \left|x_2 - b_2^{\ i}\right|) \right\}$$

The assumptions of the problem had already been discussed in Chapter 3. In this section however, a new objective is added for measuring the service cost. For this purpose, there are two objective functions used in the literature, namely minimax and minsum. The minimax function, which minimizes the maximum distance of the facility from the demand points, is applied specifically to emergency facilities like fire stations and hospitals that should be sited as close as possible to the point receiving the lowest quality of service. The minsum function on the other hand, is the most widely used in the literature for general service facilities and minimizes the sum of the distances between the facility and the demand points. It measures the service cost as the cost of locating the facility far from the set of demand points which is generally called transportation cost. Service cost is generally measured with minsum objective in semi-desirable facility location problems, e.g. Morales et al. (1997), Brimberg and Juel (1998), Melachrinoudis (1999), Skriver and Andersen (2003), Melachrinoudis et. al. (2003) have all used minsum objective. In this study, we use the minsum objective, since we believe that for most of the facilities service cost generally occurs through transportation cost.

The mathematical model for the undesirable facility location problem with maximin objective has already been presented along with the solution approaches in Chapter 3. Before focusing on the biobjective problem, we would like to construct a mathematical model with minsum objective based on the mentioned assumptions.

The model (M-1) of Section 3.2 can be adapted for minsum objective as follows.

(M-3)

$$Min \sum_{i=1}^{N} L_i$$

$$subject\ to$$

$$L_i = \left|x_1 - b_1^i\right| + \left|x_2 - b^i{}_2\right| \qquad for\ i = 1,...,N$$

$$e_j x_1 + f_j x_2 \le g_j \qquad for\ j = 1,...,k$$

Parameters:

b_j^i	:	Coordinate of the i[th] demand point in the j[th] dimension
e_j , f_j , g_j	:	Constants that define the feasible region

Decision Variables:

L_i	:	Rectilinear distance of the facility from the i[th] demand point
x_i	:	i[th] coordinate of the facility

Absolute values in the distance constraints make the model nonlinear. The model is linearized as shown below. The rectilinear distance is calculated as follows.

Let

$$d(\boldsymbol{x}, \boldsymbol{b^j}) = \left|x_1 - b_1{}^j\right| + \left|x_2 - b_2{}^j\right| \quad \boldsymbol{x}, \boldsymbol{b^j} \in R^2$$

$a_i = \left| x_i - b_i^{\,j} \right|$ for i = 1, 2 is the i[th] component of $d(\boldsymbol{x}, \boldsymbol{b^j})$ and can be calculated as;

$a_i = \max\left\{(x_i - b_i^{\,j}), (-x_i + b_i^{\,j})\right\}$

Then it is true that

$$a_i \ge x_i - b_i^{\,j}$$
$$a_i \ge -x_i + b_i^{\,j}$$

Since the objective is minimization, there is no need to control the above two constraints with integer variables as in the case of *(Maximin-l^1)* while constructing the linear model. Hence, when these two constraints are used along with a minimization objective, it is guaranteed that one of them holds as equality. Thus, our mathematical program is as follows:

$(Minsum - l^1)$

$$Min \sum_{j=1}^{N} d_j \qquad (1)$$

subject to

$$d_j = a_1^{\,j} + a_2^{\,j} \qquad for\ j = 1,\ldots,N \qquad (2)$$
$$a_i^j \ge x_i - b_i^j \qquad for\ j = 1,\ldots,N\ \ i = 1,2 \qquad (3)$$
$$a_i^j \ge b_i^j - x_i \qquad for\ j = 1,\ldots,N\ \ i = 1,2 \qquad (4)$$
$$e_j x_1 + f_j x_2 \le g_j \qquad for\ j = 1,\ldots,k \qquad (5)$$
$$d_j, a^j \ge 0 \qquad for\ j = 1,\ldots,N$$

Parameters:

b_i^j	:	Coordinate of the j[th] demand point in the i[th] dimension
e_j , f_j , g_j	:	Constants that define the feasible region

Decision Variables:

d_j	:	Rectilinear distance of the facility from the j[th] demand point
a_i^j	:	i[th] component of d_j where $d_j = a_1^j + a_2^j$
x_i	:	i[th] coordinate of the facility

Minimization of objective function (1) ensures that a facility which minimizes the total distance of the facility from the demand points is selected as the optimal solution. Constraint set (2) ensures that absolute distance is calculated as $d_j = a_1^j + a_2^j = \left|x_1 - b_1^j\right| + \left|x_2 - b_2^j\right|$. Constraint sets (3) and (4) guarantee the calculation of rectilinear distance as explained before. Constraint set (5) defines the feasible region. Since in the semi-desirable facility location problem we consider maximin and minsum objective functions, the following biobjective mathematical model is formulated by combining models *(Maximin-l^1)* and *(Minsum-l^1.)*.

$(Biobjective - l^1)$

$$L^* = Max\ L \qquad (1)$$

$$W^* = Min\ W \qquad (2)$$

subject to

$$W = \sum_{j=1}^{N} d_j \qquad (3)$$

$$L \le d_j \qquad for\ j = 1,...,N \qquad (4)$$

$$d_j = a_1^{\ j} + a_2^{\ j} \qquad for\ j = 1,...,N \qquad (5)$$

$$a_i^j - u_i^j = x_i - b_i^j \qquad for\ j = 1,...,N\ \ i = 1,2 \qquad (6)$$

$$a_i^j - o_i^j = b_i^j - x_i \qquad for\ j = 1,...,N\ \ i = 1,2 \qquad (7)$$

$$u_i^j \le Mt_i^j \qquad for\ j = 1,...,N\ \ i = 1,2 \qquad (8)$$

$$o_i^j \le Mz_i^j \qquad for\ j = 1,...,N\ \ i = 1,2 \qquad (9)$$

$$t_i^j + z_i^j \le 1 \qquad for\ j = 1,...,N\ \ i = 1,2 \qquad (10)$$

$$e_j x_1 + f_j x_2 \le g_j \qquad for\ j = 1,...,k \qquad (11)$$

$$d_j, a^j, u^j, o^j \ge 0 \qquad for\ j = 1,...,N$$

$$t^j, z^j \in \{0,1\} \qquad for\ j = 1,...,N$$

where,

L	:	Distance of the facility from the closest demand point.
W	:	Total distance between the facility and the demand points

The other variables and parameters have already been defined in *(Maximin-l^1)* in Section 3.2.

Let

$$L(\boldsymbol{x^i}) = \left\{ \min_{j=1,\dots,N} (\left|x_1^{\ i} - b_1^{\ j}\right| + \left|x_2^{\ i} - b_2^{\ j}\right|\right\}$$

$$W(\boldsymbol{x^i}) = \left\{ \sum_{j=1}^{N} (\left|x_1^{\ i} - b_1^{\ j}\right| + \left|x_2^{\ i} - b_2^{\ j}\right|) \right\}$$

In accordance with Multicriteria Decision Making (MCDM) terminology, $\boldsymbol{x} = (x_1, x_2)$ can be referred to as a ***decision vector***. The vector of objective function values z(***x***) = (L(***x***), W(***x***)) belonging to ***x*** is named as an ***objective*** (***criterion) vector***. Feasible region, S, containing decision vectors is called the ***feasible decision region.*** It is a subset of the ***decision space***, R^2. ***Feasible objective (criterion) region*** is defined as the image of the feasible decision region in the two objective functions. It is a subset of the ***objective (criterion) space***, R^2. Throughout the thesis, words "feasible region" and "feasible decision region" will be used interchangeably.

In the presence of two objectives, the main concern is to find the set of efficient solutions the definition of which is given below.

Definition 4.1

A feasible solution ***x*** to *(Biobjective-l^1)* is ***efficient*** if and only if there does not exist another feasible solution $\boldsymbol{x^i}$ such that L($\boldsymbol{x^i}$) $\geq$ L(***x***), W($\boldsymbol{x^i}$) $\leq$ W(***x***) and (L($\boldsymbol{x^i}$),W($\boldsymbol{x^i}$)) $\neq$ (L(***x***),W(***x***)). A feasible solution ***x*** is ***weakly efficient*** if and only if there does not exist another feasible solution $\boldsymbol{x^i}$ such that L($\boldsymbol{x^i}$) > L(***x***), W($\boldsymbol{x^i}$) < W(***x***).

The set of all efficient solutions is called the efficient frontier.

Definition 4.2

A solution is ***approximately efficient*** if and only if it is efficient with respect to a large set of known solutions.

Efficiency is defined in the decision space; image of any efficient solution in the objective space is a nondominated objective vector.

Let ***Z*** define the feasible objective region;

$$\boldsymbol{Z} = \left\{ z(\boldsymbol{x}) \in R^2 \,\middle|\, z(\boldsymbol{x}) = (L(\boldsymbol{x}), W(\boldsymbol{x})),\ \boldsymbol{x} \in \mathrm{S} \right\}$$

Definition 4.3

$z(x) \in \boldsymbol{Z}$ is a ***nondominated*** objective vector if and only if $\boldsymbol{x}$ is an efficient solution to *(Biobjective-l^1)*. Otherwise, it is dominated.

In parallel with Skriver and Andersen (2003) we use a geometrical branch and bound algorithm to solve *(Biobjective-l^1)*. The difference is that we try to adapt the new version of the BSSS method, 'Generalized Big Square Small Square Method' (GBSSS) suggested by Plastria (1992) to the semi-desirable facility location problem and we develop an algorithm in which a Decision Maker (DM) is involved interactively.

Indeed, the main idea of the methods (BSSS and GBSSS) is to eliminate some parts of the feasible region up to a prespecified precision. The elimination occurs when the inefficiency of a subregion is proved with the help of bounds. To our knowledge, this idea is used for the biobjective problem only in Skriver and Andersen (2003) in which the feasible region is reduced by an algorithm until it reaches a predetermined size.

In fact, any solution approach to semi-desirable facility location problem should be supported with a multiobjective decision aid. For instance, we may consider the problem of determining the location of a landfill in a city. Hence, the approaches that end up with the complete trajectory of efficient points or the areas that may contain efficient points may not help the DM in real life situations.

In this framework, we suggest an additional phase to GBSSS, in which we guide the DM in selecting a single location at the end, based on her/his preferences. The aim of this phase is to search the subregions that we cannot prove as inefficient interactively with the involvement of the DM.

Typically multiobjective optimization methods assume that a multiobjective problem is converted into a parametric single-objective problem whose solution provides an efficient point. Different conversions can be observed in the literature; reference point approach introduced by Wierzbicki (1980) is the most well-known approach.

For the interactive search phase of our solution approach, basically we propose the use of the reference point approach, which projects any point in the objective space to the efficient frontier of the region of concern.

Let ***R(S)*** represents the ideal objective vector of region S and ***Q(S)*** denotes the nadir objective vector corresponding to the efficient solutions of S. It should be noted that throughout the thesis, an approximation to the nadir objective vector will be used, since nadir objective vectors are difficult to obtain.

$R_1(S) = \text{Max}_{\boldsymbol{x} \in S} \{L(\boldsymbol{x})\}$, $R_2(S) = \text{Min}_{\boldsymbol{x} \in S} \{W(\boldsymbol{x})\}$

$Q_1(S) = \text{Min}_{\boldsymbol{x} \in \boldsymbol{E}} \{L(\boldsymbol{x})\}$, $Q_2(S) = \text{Max}_{\boldsymbol{x} \in \boldsymbol{E}} \{W(\boldsymbol{x})\}$

where E represents the set of efficient solutions in S.

The model that adapts our problem to Wierzbicki's (1980) reference point idea is called 'Achievement Scalarizing Program' *(ASP)*. The *(ASP)* operates in the objective space and minimizes the maximum deviation of objectives from the levels specified with a reference point. In other words, the program finds the closest efficient point to the reference point in the Tchebycheff metric. The *(ASP)* is presented below.

(ASP)

$$Min\left[\alpha - \rho(L - W)\right] \quad (1)$$

$$subject\ to$$

$$\alpha \geq w_1^0 \left(\frac{G_1^0 - L}{R_1(S)}\right) \quad (2)$$

$$\alpha \geq w_2^0 \left(\frac{W - G_2^0}{Q_2(S)}\right) \quad (3)$$

$$W = \sum_{j=1}^{N} d_j \quad (4)$$

$$L \leq d_j \qquad for\ j = 1, \ldots, N \quad (5)$$

$$d_j = a_1^j + a_2^j \qquad for\ j = 1, \ldots, N \quad (6)$$

$$a_i^j - u_i^j = x_i - b_i^j \qquad for\ j = 1, \ldots, N \quad i = 1, 2 \quad (7)$$

$$a_i^j - o_i^j = b_i^j - x_i \qquad for\ j = 1, \ldots, N \quad i = 1, 2 \quad (8)$$

$$u_i^j \leq M t_i^j \qquad for\ j = 1, \ldots, N \quad i = 1, 2 \quad (9)$$

$$o_i^j \leq M z_i^j \qquad for\ j = 1, \ldots, N \quad i = 1, 2 \quad (10)$$

$$t_i^j + z_i^j \leq 1 \qquad for\ j = 1, \ldots, N \quad i = 1, 2 \quad (11)$$

$$e_j x_1 + f_j x_2 \leq g_j \qquad for\ j = 1, \ldots, k \quad (12)$$

$$d, a, u, x, o \geq 0$$

$$t, z \in \{0, 1\}$$

where,

$\boldsymbol{G^0}$	:	Reference point
α	:	Maximum deviation of the solution objective vector from the reference point
ρ	:	Sufficiently small constant
$\boldsymbol{w}^0 = (w_1^0, w_2^0)$	:	Weights associated with each objective

Minimization of the objective function (1) ensures that a point which minimizes the maximum deviation from the levels specified with a reference point $\boldsymbol{G^o}$ is determined as the optimal solution. The second term in (1) prevents the program finding weakly efficient solutions by giving a slight slope to the contours of the objective function with a sufficiently small positive constant, ρ. Constraints (2) and (3) calculate the weighted Tchebycheff distance between the reference point and the solution vector which are normalized with the ideal value of maximin objective and the approximate nadir value of minsum objective. Both objectives are given weights by the DM considering their relative importance. Constraint (4) and constraint set (5) calculate the sum and the minimum of the rectilinear distance of the facility from the demand points. Constraint set 6 ensures that the absolute distance is calculated as: $d_j = a_1^{\,j} + a_2^{\,j} = \left|x_1 - b_1^{\,j}\right| + \left|x_2 - b_2^{\,j}\right|$. Constraint sets (6)-(11) guarantee the calculation of rectilinear distance as explained in Section 3.2. Constraint set (12) defines the feasible region.

However, as in *(Maximin-l^1)*, there are binary variables in the *(ASP)* to control the calculation of the rectilinear distance. The number of binary variables increases with increasing number of demand points, which directly increases the solution time exponentially. This fact, as elaborated in Chapter 3, makes the *(ASP)* inefficient for big sample of demand points. Since the binary structure of *(ASP)* is the same as that of *(Maximin-l^1)*, we believe the use of the suggested strategy combination for *(Maximin-l^1)* in Section 3.4.1 will increase the efficiency of the model. With this idea, we have conducted an experimental study. The feasible region was defined in R^2, which is a 100x100 square defined by the constraints $0 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. We conducted three experiments using 1000, 3000 and 5000 demand points (10 randomly generated problems were solved in each experiment). The locations of the demand points were generated according to uniform distribution in the interval [0,100]. We assumed that both objectives were attributed with equal weights. The reference point was assumed to be the ideal point in each problem. The runs were conducted on Pentium IV personal computer with 256 MB random access memory (RAM). The optimization models were solved in GAMS Version 20.2. The computer code that calls optimization programs was written in Borland C++ Builder Version 3. CPLEX Version 7.5 operating under GAMS Version 20.2 was used as the MIP solver. Table 4.1 reports the average CPU for each problem set.

Table 4.1- Computational Results

	CPU Time(sec)		**% Reduction in CPU Time**
Number of Demand Points	**Default Strategies**	**Strategy Combination 3**	
1000	27.7	10.2	63
3000	164.5	50.5	69
5000	430	129	70

As evident from Table 4.1, the solution time obtained for different sample sizes decreased considerably with the strategies as estimated. Although a great saving is achieved in the solution time of the *(ASP)* with the new branch and bound strategies, we also use the idea suggested by Karaivanova et al. (1995) for the solution of multiple objective integer linear programs. They proposed the use of a two-phase continuous/integer method in their study. In the first phase, the method operates in the relaxed continuous space parametrically to find a number of nondominated continuous solutions iteratively. Once the most preferred continuous solution is determined, the closest integer solution is found with the help of the *(ASP)*. The rationale behind this hybrid approach is that in mixed integer linear programs, computation time increases exponentially with the number of integer variables. Therefore, the number of mixed integer linear programs to be solved should be decreased. In fact, the logic is that it is not reasonable to generate precise nondominated integer solutions in the early iterations, when the DM is searching regions far from the most preferred solution.

For the search in the nondominated continuous objective region, basically there are two alternative approaches used in this study. First of them is the reference point - reference direction approach, in which, we solve the LP relaxation of the *(ASP)* to find an initial nondominated continuous solution $\boldsymbol{y}^{o}$. Here, it should be noted that the continuous solution is very sensitive to the choice of M. Although, minsum objective forces constraints (7) and (8) to measure the true rectilinear distance even in the LP relaxation, when the integrality requirements are relaxed, both $\boldsymbol{u}$ and $\boldsymbol{o}$ are free to take positive values which directly depend on the value of M by constraints (9) and (10). Therefore, value of M should be selected with care at its possible minimum level. It should be noted that in all our examples and test problems throughout the thesis, it is chosen as small as possible.

Once the initial nondominated continuous solution is found, the nondominated continuous objective region is searched by a parametric linear program similar to the *(ASP)*. Suggested by Korhonen and Laakso (1986), this method iteratively projects a line segment in the objective space onto the nondominated surface of the feasible objective region. This method is adapted to our problem with the below model that we call ‘Achievement Scalarizing Parametric Linear Program’ *(ASPLP)*.

$(ASPLP)$

$$Min\left[\alpha - \rho(L - W)\right] \quad (1)$$

subject to

$$\alpha \geq w_1\left(\frac{y_1^{o} + p\Delta d_1 - L}{R_1(S)}\right) \quad (2)$$

$$\alpha \geq w_2\left(\frac{W - (y_2^{0} + p\Delta d_2)}{Q_2(S)}\right) \quad (3)$$

$$W = \sum_{j=1}^{N} d_j \quad (4)$$

$$L \leq d_j \qquad for\ j = 1,\ldots,N \quad (5)$$

$$d_j = a_1^{\,j} + a_2^{\,j} \qquad for\ j = 1,\ldots,N \quad (6)$$

$$a_i^j - u_i^j = x_i - b_i^j \qquad for\ j = 1,\ldots,N \quad i = 1,2 \quad (7)$$

$$a_i^j - o_i^j = b_i^j - x_i \qquad for\ j = 1,\ldots,N \quad i = 1,2 \quad (8)$$

$$u_i^j \leq M t_i^j \qquad for\ j = 1,\ldots,N \quad i = 1,2 \quad (9)$$

$$o_i^j \leq M z_i^j \qquad for\ j = 1,\ldots,N \quad i = 1,2 \quad (10)$$

$$t_i^j + z_i^j \leq 1 \qquad for\ j = 1,\ldots,N \quad i = 1,2 \quad (11)$$

$$e_j x_1 + f_j x_2 \leq g_j \qquad for\ j = 1,\ldots,k \quad (12)$$

$$d, a, u, x, o, t, z \geq 0$$

In this model, all parameters and variables are as in the *(ASP)* except $\boldsymbol{y^0}$ and $\boldsymbol{\Delta d}$ which stands for the reference point and the reference direction respectively. After foundation of $\boldsymbol{y^0}$ by the solution of the relaxed version of the *(ASP)*, $\boldsymbol{\Delta d}$ is determined by the DM based on his direction of preferences. p is a scalar which decides the number of points projected onto the efficient frontier in the determined direction.

The second approach for searching the nondominated continuous objective region is based on the perturbation of the initial reference point (Wierzbicki, 1980). In this approach, the first continuous solution (point A in Figure 4.1) is found by projecting the reference point (point B) to the efficient frontier with the LP relaxation of the *(ASP)* as in the previous approach. Then, by using the percent deviation of Point A from Point B, we perturb the reference point in each objective while generating a number of vectors decided by the DM. The perturbation amount is determined with the deviation of the reference point from the continuous solution found. The projection of the perturbed reference points (Points C and D) generates points E and F and these points give the DM a chance to perceive the efficient frontier better since she/he finds the opportunity to see a number of continuous solutions that differ from each other as the reference point gets far from the efficient frontier.

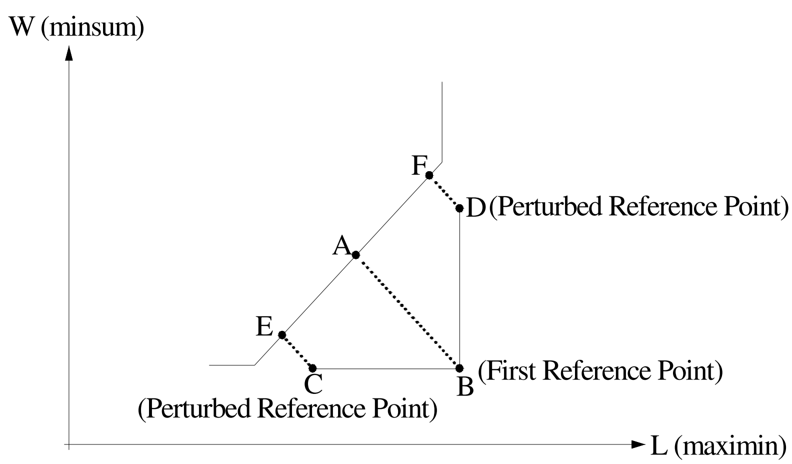

Figure 4.1- Altering the Reference Point

After the interactive search in the candidate efficient regions, the DM is most probably left with a number of alternative solutions. At this point, we propose the use of an outranking method when she/he can not decide between the alternatives.

We believe that our algorithm will serve as a decision support system to real life semi-desirable facility location problems. The detailed version of our algorithm is presented in the next section.

4.2 INTERACTIVE BIG SQUARE SMALL SQUARE (IBSSS) ALGORITHM

In this section, we present an interactive geometrical branch and bound algorithm for the solution of a single semi-desirable facility location problem. Our algorithm consists of three main phases which are highlighted below.

In the first two phases of the algorithm, we try to prove that some parts of the feasible region are inefficient and can be fathomed from further consideration with the help of incumbent points. In the last phase, we search the reduced feasible region with the involvement of the DM.

Phase 1: Rough Cut Phase

In this phase, the upper bound on the optimal maximin objective value and the optimal minsum objective value are used with the idea that this pair of values for a region is always better than the ideal point of that region and if any incumbent point dominates this pair, obviously there is no point in the region that is better than this incumbent point. Hence the region is proved to be inefficient and can be discarded from further consideration. Indeed, the purpose of this rough phase is to get rid of some subregions that can be eliminated with the help of an upper bound on the maximin objective without requiring the optimal value, since it is time consuming to find the optimal solution to *(Maximin-l^1)* especially for big sample of demand points.

The idea of branching in the algorithm is to improve the bounds of the regions, since after every division, the optimal maximin objective value decreases so does the upper bound; while the optimal minsum objective increases. At the same time, new incumbent points are added in each branching. Considering these two facts, it is obvious that the chance to eliminate regions increases in every branching.

The algorithm is obviously a geometrical branch and bound algorithm. As in the standard branch and bound algorithm, one of the most important parameters determines how to select the next region to branch on. This decision actually changes the character of the algorithm totally. In our algorithm, we use the best bound search. Every time we look for a region to branch on, the region with the highest upper bound / optimal maximin objective value or the region with the lowest optimal minsum objective value is selected to be divided. The rationale behind is that the regions with best bounds are difficult to eliminate with incumbent points because of its good objective values; hence it is reasonable to branch them first. In addition, every time we divide a region, a number of points are picked from it. Therefore, by best bound search, priority in branching is given to the regions whose elimination is difficult and from where good incumbent points can be obtained. By doing this, we increase the chance to eliminate regions with worse objective values without branching them much. Since, every branching means storing more new data; it directly affects the efficiency of the algorithm.

Skriver and Andersen (2003) use the branching style developed in Hansen et. al. (1981) which proposes the division of all the regions at the same time into four equal subregions which makes all the regions identical at each iteration. Our claim is that, this style of branching results in unnecessary large amount of data to be stored and it actually does not utilize the possibility of eliminating big regions at once.

Phase 2: Precise Cut Phase

After branching up to a predetermined size in Phase 1, the optimal maximin objective values are found for the remaining subregions in Phase 2. For finding the optimal maximin values, we again use the idea that the demand points, whose shortest distance to the region of concern is greater than the upper bound of that region on maximin objective, have no effect in finding the optimal solution with *(Maximin-l^1)*, thus should be filtered. It should be noted that every time *(Maximin-l^1)* is solved; the strategy combination suggested in Section 3.4.1 is used.

By finding the optimal value on the maximin objective, we have the chance to compare the ideal objective vector of the regions with the incumbent points for any possible further elimination. In this phase, we allow the DM to branch the remaining squares further with the optimal values. Since the regions are already divided up to a prespecified side length in Phase 1, branching with optimal values is expected to take reasonable time.

With Phase 1 and Phase 2, some parts of the feasible region that are proved to contain only inefficient points are eliminated. The remaining subregions after these phases may contain efficient points together with inefficient points.

Phase 3: Interactive Search Phase

This phase of the algorithm is the beginning of the interactive search with the DM. In this part, we develop two procedures; one is exact and the other is an approximate procedure.

The exact procedure is based on the reference point approach which guarantees to find an efficient point at the end. In this procedure, the remaining subregions after the first two phases are presented to the DM along with ideal and nadir objective vectors. Each time a region is selected to be searched, the DM is asked to specify aspiration levels in both objectives (i.e. reference point) based on her/his preferences and the *(ASP)* is solved to project this reference point to the efficient frontier of the selected subregion with the selected weight set. The solution found is efficient with respect to the region from which it is generated, but it may be dominated by the solutions in the other regions (i.e. it is approximately efficient). Therefore, each time a reference point is projected onto the efficient frontier of the selected region, we need to check whether there exist solutions in the other subregions which dominate the one at hand. For this check, we again use the idea of the *(ASP)* with which the solution at hand is projected onto the efficient frontier of the other regions with the initial weight set. If there is no solution dominating the one at hand, then it is proved to be nondominated. If any dominating or approximately efficient solution is found, then the DM is given the chance to pass to the region from which it is produced and continue the search from that region. The DM continues to generate solutions in the same manner from the subregions she/he selected by identifying reference points. When the DM stops searching the subregions, he is asked to select the most preferred solution from the resulting nondominated objective vectors according to her/his preferences.

The approximate procedure is also based on the reference point approach. This procedure can be used when the DM wants to see both efficient and approximately efficient solutions instead of finding guaranteed efficient solutions which is computationally cumbersome. In this procedure, the interactive search is carried out in the subregions that the DM selects. The procedure guarantees to find efficient solutions for the selected subregion. However, the solution may be inefficient with respect to the other regions. Hence, each time a solution is found, it is compared to the other solutions already generated (i.e. solutions in the List of Candidate Location Points (*LCLP)* and the List of Incumbent Function Values

(*LIFV)*). Hence, the chance to obtain an inefficient final solution decreases considerably with these comparisons and the approach finds approximately efficient solutions.

In the approximate procedure, the remaining subregions after the first two phases are presented to the DM along with ideal and nadir objective vectors. Each time a region is selected to be searched, the DM is asked to specify aspiration levels in both objectives (i.e. reference point). Similar to Phase 1 and Phase 2, a rough approach is used in the beginning and continuous solutions are generated. The findings of this approach are used to generate integer solutions. In the beginning, the LP relaxation of the *(ASP)* is solved which minimizes the maximum deviation from the reference point. This program with an augmentation constant guarantees to find a nondominated continuous solution for the region under search. With this solution at hand, we present two ways to the DM. She/he may find the closest integer solution to the continuous one, or she/he can continue to search for other continuous solutions.

If the latter is selected, alternative continuous solutions are found with two different ways. In the first one, we use the percent deviation of the first continuous solution from the reference point, and we perturb the reference point in each objective. While doing this, we produce a number of solutions which is determined by the DM. Then, we project the perturbed reference points to the efficient frontier again with the LP relaxation of the *(ASP)*. We claim that the DM has a better understanding about the efficient frontier with this approach. Once the DM is satisfied with a continuous solution, an integer solution closest to it is found with the *(ASP)* and kept as an objective vector of a candidate location point.

In the second approach, the DM is asked to specify a reference direction. The continuous nondominated solutions closest to the points on this direction are found with solving the *(ASPLP)*. Once the DM is satisfied with a continuous solution, an integer solution closest to it is found with the *(ASP)* and kept as an objective vector of a candidate location point.

When the algorithm is completed and if the DM cannot decide between the candidate location alternatives, they are ranked with an outranking method (e.g. Promethee II, (Brans and Philippe Vincke, 1985)). Promethee II is based on ranking of the alternatives based on the entering and leaving flows calculated according to the indifference and preference thresholds determined by the DM. This method yields a unique, complete preorder.

The short version of the algorithm is presented below. The detailed version can be seen from **Appendix D.**

4.2.1 THE ALGORITHM

FINDING CANDIDATE EFFICIENT SQUARES

Phase 1: Pruning with *UB(Maximin-l^1)* and *(Minsum-l^1)*

Branching the Initial Square

- Find a square approximation of the feasible region.

- Ask the DM to specify a stopping side length.

- Pick T feasible points $\{x^1, x^2, .., x^T\}$ from the divided region.

❖ Evaluate maximin and minsum function values for these T points.

$$L(\boldsymbol{x}^{i}) = \left\{ \min_{j=1,\dots,N} (\left|x_1^{\,i} - b_1^{\,j}\right| + \left|x_2^{\,i} - b_2^{\,j}\right|\right\}$$

$$W(\boldsymbol{x}^{i}) = \left\{ \sum_{j=1}^{N} (\left|x_1^{\,i} - b_1^{\,j}\right| + \left|x_2^{\,i} - b_2^{\,j}\right|) \right\} \text{ for i=1,...,T}$$

❖ Add these points to the *LIFV* after a dominance check.

Add (L($\boldsymbol{x}^{i}$), W($\boldsymbol{x}^{i}$)) to the *LIFV* if and only if there does not exist another objective vector (L($\boldsymbol{x}^{j}$), W($\boldsymbol{x}^{j}$)) ∈ *LIFV* ∋ L($\boldsymbol{x}^{j}$) ≥ L($\boldsymbol{x}^{i}$), W($\boldsymbol{x}^{j}$) ≤W($\boldsymbol{x}^{i}$) and (L($\boldsymbol{x}^{i}$), W($\boldsymbol{x}^{i}$)) ≠ (L($\boldsymbol{x}^{j}$), W($\boldsymbol{x}^{j}$)).

If any element of the *LIFV* (L($\boldsymbol{x}^{j}$), W($\boldsymbol{x}^{j}$)) is dominated by the newly added objective vector (L($\boldsymbol{x}^{i}$), W($\boldsymbol{x}^{i}$)) ∋ L($\boldsymbol{x}^{i}$) ≥ L($\boldsymbol{x}^{j}$), W($\boldsymbol{x}^{i}$) ≤W($\boldsymbol{x}^{j}$) and (L($\boldsymbol{x}^{i}$), W($\boldsymbol{x}^{i}$)) ≠ (L($\boldsymbol{x}^{j}$), W($\boldsymbol{x}^{j}$)) delete (L($\boldsymbol{x}^{j}$), W($\boldsymbol{x}^{j}$)) from *LIFV*.

❖ Divide the initial square into 4 equal subsquares by two perpendicular lines passing from the center.

❖ In order to find the upper bound on maximin objective value for each subsquare, solve,

- *UB(Maximin-*l^1*)* with all existing demand points

- *(Maximin-*l^1*)* with demand points located inside the square

Choose the smallest of the above as the upper bound on maximin objective.

- In order to find the optimal minsum objective value for each subsquare, solve *(Minsum-l^1)*.

- Compare the recently generated squares with the *LIFV*.

- If any incumbent point has a maximin objective greater than the upper bound of some subsquare and has a minsum value less than the minsum optimal value of that square at the same time, then obviously, the ideal point of the square is dominated by this incumbent solution. Square is clearly inefficient and can be deleted.

- Keep the nondominated squares in the List of Candidate Efficient Squares *(LCES)*.

- If the stopping side length is not reached, then select where to branch next, else stop Phase 1 and go to Phase 2.

Selecting Where to Branch Next

- Check the *LCES*. Choose the square either with the highest upper bound on maximin or with the lowest optimal minsum objective value to branch on.

- Pick T feasible points from the selected region.

- Update the *LIFV* after every branching.

- Update the *LCES* after every branching by comparing the upper bound on optimal maximin objective and optimal minsum objective value of the squares with the recently generated incumbent value vectors.

- Check each recently generated square. Add the squares which are not dominated by any incumbent value vector of the *LIFV* to the *LCES*.

- Repeat the process until the maximum side length of the squares reaches the stopping side length.

<u>Phase 2: Pruning with *(Maximin-l^1)* and *(Minsum-l^1)*</u>

Finding Optimal Maximin Objective with Filtered Demand Points

For all squares in the *LCES*;

- Check the smallest distance of the demand points to the square as shown in Figure 3.1.

- Keep the demand points with the smallest distance to the square is smaller than the upper bound on the optimal maximin objective value of that square in the List of Filtered Demand Points *(LFDP)*.

- Solve *(Maximin-l^1)* by using the filtered demand points. Obtain the ideal objective vector.

- Compare the ideal objective vector of the square with the *LIFV*.

- If there is any square whose ideal objective vector is dominated by an element of the *LIFV* then delete this from the *LCES*.

Stopping or Dividing Further

- Ask the DM if she/he wants to divide the regions further, if so, determine a stopping side length for Phase 2.

- Divide all the squares into 4 simultaneously until the stopping side length is reached. Prune with comparing ideal objective vectors with the *LIFV*.

- If the DM wants to stop Phase 2, then combine the remaining squares in appropriate regions.

SEARCH IN THE CANDIDATE EFFICIENT REGIONS

<u>Phase 3: Interactive Search</u>

- Ask the DM which procedure she/he wants to use: Exact or approximate.

A. Exact Procedure

- Present each region to the DM with its ideal and nadir objective vector.

- Ask the DM to choose a region for starting the search.

- Ask the DM to set her/his aspiration levels in both objectives. Let this vector be the reference point in the objective space.

- Ask the DM which objective is more important and how much. Set the weights accordingly.

- Solve the *(ASP)* for the selected region. If the solution at hand is feasible, check whether it is nondominated or not. For this, compare it with the *LIFV* and the *LCLP*. If it is not proved to be dominated then project it to the other regions by solving the *(ASP)* with the same weight set. If the solution at hand is not dominated by one of the produced solutions than add it to the *LCLP*. Also check whether the produced solutions are approximately efficient. Add the approximately efficient ones to the List of Candidate Nondominated Vectors *(LCNV)*.

- If it is dominated, check the resulting dominating and approximately efficient solutions. Add these to the List of Candidate Nondominated Vectors *(LCNV)*. Ask the DM if he wants to select one of the solutions in *LCNV* and check whether it is a nondominated solution. If so repeat this step for the selected solution. If not, the DM either can pass to other regions or stop the search.

- Present the *LCLP* to the DM. Ask the DM if she/he can select the most preferred solution among the solutions, if so stop Phase 3, if not rank the alternatives with Promethee II.

- In each step, every time a solution is found, check whether the ideal point of any region is dominated by it, if so delete the region from further consideration.

B. Approximate Procedure

Starting the Search

- Present each region to the DM with its ideal and nadir objective vector.

- ❖ Ask the DM to choose a region for starting the search.

Finding a Starting Efficient Continuous Solution

- ❖ Ask the DM to set her/his aspiration levels in both criteria. Let this vector be the reference point in the objective space.

- ❖ Ask the DM which objective is more important and how much. Set the weights accordingly.
- ❖ Solve the LP relaxation of the *(ASP)* to find a starting continuous solution closest to the reference point.

- ❖ If the DM likes the solution, find the closest integer solution by the *(ASP)*. Otherwise, search the nondominated continuous objective region using one of the below approaches.

Generating Alternative Nondominated Continuous Solutions

Approach 1: Better Perception of the Efficient Frontier with Perturbed Reference Points

- ❖ Find the total percent deviation of the reference point from the starting continuous solution in both objectives.

- ❖ Ask the DM to determine the number of solutions that she/he wants to see.

- ❖ Generate the perturbed reference points by changing the reference point in each objective by using the percent deviation found.

- Solve the LP relaxation of the *(ASP)* with perturbed reference points to find additional continuous nondominated solutions.

- Present the nondominated solutions to the DM. If she/he does not like the solutions then ask her/him to change his first reference point. Else ask him to select the most promising continuous solution for which she/he wants to see the closest integer solution.

Approach 2: ***Reference Direction Approach***

- Ask the DM to specify a reference direction.

- Ask the DM the number of solution that she/he wants to see.

- Solve the *(ASPLP)* to find the closest nondominated continuous solutions in the region to the initial continuous solution and to the solutions that lie in the determined reference direction.

- Present the nondominated solutions to the DM. If she/he does not like the solutions then ask her/him to change his first reference point. Else ask him to select the most promising continuous solution for which she/he wants to see the closest integer solution.

Finding an Approximately Efficient Integer Solution

- Find the closest integer solution to the selected continuous solution by solving the *(ASP).*

- Check the integer solution if it is infeasible; dominated by any incumbent objective vector of the *LIFV* or the *LCLP*. If it is dominated, ask the DM if she/he likes to continue searching this region further. If so, then ask her/him to change his first reference point. Otherwise, she/he either can pass to other regions or stop the search. If the solution is not dominated and preferred by the DM, add it to the *LCLP*.

- Every time an integer solution is found, check whether the ideal point of any region is dominated by it, if so delete the region from further consideration.

Continue to search the regions until the DM is satisfied with the location alternatives and wants to stop searching.

Selection among the Discrete Set of Alternatives

- Present the *LCLP* to the DM.

- If the DM is satisfied with the presented alternatives and can select one of them as the most promising solution then stop the algorithm.

- If she/he has a difficulty in selecting a final solution from the presented list then outrank the alternatives with Promethee II.

Outranking of the Alternatives with Promethee II

- Ask the DM to set indifference (q_i), preference (p_i) thresholds and weights for each objective.

- For each alternative pair ***a***, ***b*** in the *LCLP*, calculate the outranking degree as

$$\pi(\boldsymbol{a},\boldsymbol{b}) = \frac{1}{W}\sum_{i=1}^{2} w_i F_i(\boldsymbol{a},\boldsymbol{b}),\ \ \forall(\boldsymbol{a},\boldsymbol{b}) \in LCLP$$

where,

w_i	:	Weight associated with each objective
W	:	Total weight ($W = w_1 + w_2$)
$F_i(\boldsymbol{a},\boldsymbol{b})$	:	Function taking values between 0 and 1.

We assume the form of the function $F_i(\boldsymbol{a},\boldsymbol{b})$ as follows:

$$F_1(\boldsymbol{a},\boldsymbol{b}) = \begin{Bmatrix} 0 \ if \ \ L(\boldsymbol{a}) - L(\boldsymbol{b}) < q_1 \\ 1 \ if \ \ L(\boldsymbol{a}) - L(\boldsymbol{b}) > p_1 \\ Linear \ between \ q_1 \ and \ p_1 \end{Bmatrix}$$

$$F_2(\boldsymbol{a},\boldsymbol{b}) = \begin{Bmatrix} 0 \ if \ \ W(\boldsymbol{b}) - W(\boldsymbol{a}) < q_2 \\ 1 \ if \ \ W(\boldsymbol{b}) - W(\boldsymbol{a}) > p_2 \\ Linear \ between \ q_2 \ and \ p_2 \end{Bmatrix}$$

- ❖ Calculate the leaving and entering flow and rank the alternatives decreasing order of number $\phi(\boldsymbol{a})$.

$$\phi^{+}(\boldsymbol{a}) = \sum_{b \in LCLP} \pi(\boldsymbol{a},\boldsymbol{b})$$

$$\phi^{-}(\boldsymbol{a}) = \sum_{b \in LCLP} \pi(\boldsymbol{b},\boldsymbol{a})$$

$$\phi\ (\boldsymbol{a}) = \phi^{+}(\boldsymbol{a}) - \phi^{-}(\boldsymbol{a})$$

where,

$\phi^{+}(\boldsymbol{a})$: Leaving flow, represents the importance of the alternatives outranked by $\boldsymbol{a}$

$\phi^{-}(\boldsymbol{a})$: Entering flow, represents the importance of the alternatives outranking $\boldsymbol{a}$

- ❖ Rank the alternatives in the decreasing order of $\phi(\boldsymbol{a})$ which is a unique complete preorder.

- ❖ Present the alternatives to the DM as the candidate location points in the order of preference.

4.3 EXAMPLES

Example 4.1. Consider a single semi-desirable facility location problem in a 100 x 100 square. Suppose there are 6 demand points with below coordinates;

Table 4.2 – Demand Points of Example 4.1

i	**1**	**2**	**3**	**4**	**5**	**6**
$\mathbf{b^i}$	(50,20)	(18,80)	(2,16)	(75,68)	(90,100)	(85,10)

We assume that the DM has an underlying quasiconcave utility function which we pretend that we do not know.

$$U = -\sum_{i=1}^{2} 0.5\ (R_i - z_i)^2$$

where,

U : Utility function

R_i : i^{th} coordinate of the ideal objective vector

z_i : i^{th} coordinate of an objective vector

Parameters

Branch and Bound Strategies = Strategy Combination 3

M = 200

$\alpha = 12.5$

$\rho = 10^{-3}$

T = 100

Phase 1

Table 4.3 shows the *LCES* along with the upper bound of the squares on the maximin objective and their optimal minsum objective value at the end of Phase 1.

Table 4.3 – *LCES* at the End of Phase 1

i	$UB(S^i)$	$W^*(S^i)$
12	49.00	412.00
20	42.50	382.00
24	46.44	432.00
30	46.00	382.00
32	40.00	382.00
36	42.00	396.00

Table 4.4 shows the *LIFV*. At each branching 100 points are selected. After doing a dominance check at each step, 11 points are left.

Table 4.4 – *LIFV* at the End of Phase 1

I	x^i	$L(x^i)$	$W(x^i)$
1	[50.00,56.50]	36.50	382.00
2	[48.75,56.25]	37.50	384.50
3	[47.50,56.25]	38.75	387.00
4	[46.25,56.25]	40.00	389.50
5	[45.00,57.50]	40.50	392.00
6	[42.50,56.25]	43.75	397.00
7	[41.25,56.25]	45.00	399.50
8	[40.00,56.50]	46.00	402.00
9	[97.50,43.75]	46.25	467.00
10	[98.75,43.75]	47.50	474.50
11	[100.00,43.75]	48.75	482.00

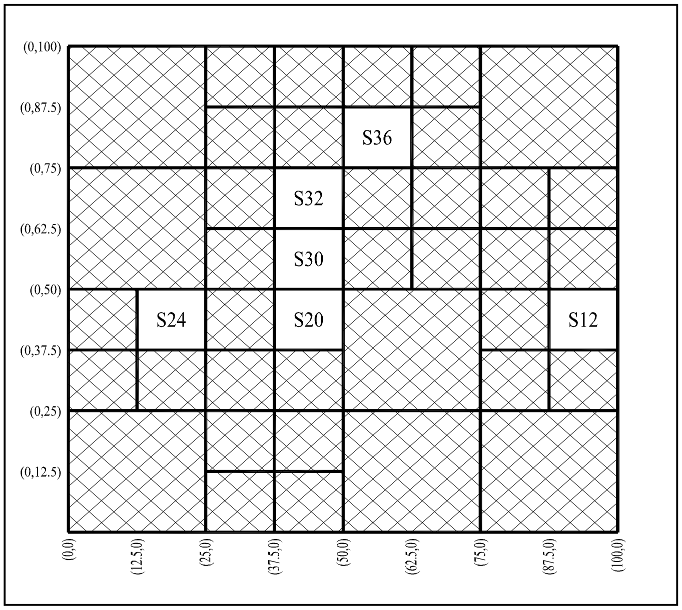

Figure 4.2 - Reduced Feasible Region at the End of Phase 1

Shaded areas in Figure 4.2 shows the regions eliminated at the end of Phase 1.

Phase 2

In this phase, the optimal maximin objective values for each square in the *LCES* (see Table 4.3) are found. As seen from the below table, when the optimal maximin values are found, square 24 is dominated by an element of *LIFV*, $\boldsymbol{x}^{8}$ (see Table 4.4).

Table 4.5 – Elimination in the *LCES* at Phase 2

i	$L^*(S^i)$	$UB(S^i)$	$W^*(S^i)$	Status
12	49.00	49.00	412.00	CANDIDATE
20	42.50	42.50	382.00	CANDIDATE
24	46.00	46.44	432.00	**DOMINATED**
30	46.00	46.00	382.00	CANDIDATE
32	40.00	40.00	382.00	CANDIDATE
36	42.00	42.00	396.00	CANDIDATE

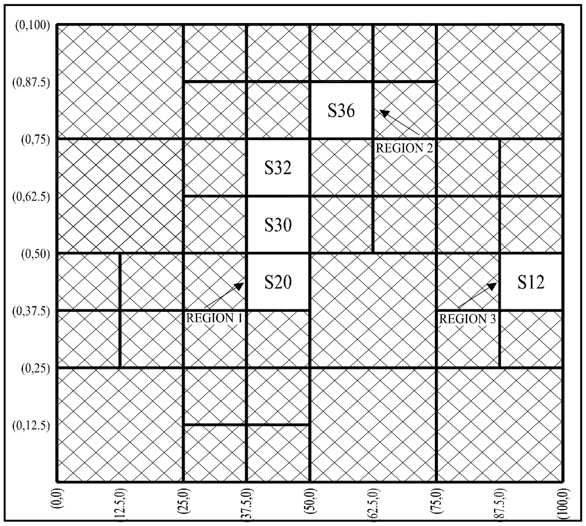

Figure 4.3 - Reduced Feasible Region at the End of Phase 2

Figure 4.3 shows the elimination achieved in Phase 1 and Phase 2 and the below table shows the percent eliminations achieved in Phase 1 and Phase 2. At the end of Phase 2, 92.19 % of the decision region is proved to be inefficient. 7.81 % of the region remains containing both efficient and inefficient points (see Figure 4.3).

Table 4.6 – Percent Elimination Achieved

Phase 1	Phase 2	Overall
90.63	1.56	92.19

We assume that the DM wants to start the search.

Phase 3

Squares 20, 30 and 32 are combined as a single region and 3 regions are presented to the DM (see Figure 4.3). The data related with the regions can be seen in Table 4.7.

It should be noted that had we known the underlying utility function, we could maximize the utility function on the feasible objective region and obtain the best solution as L = 39 and W = 387 with the decision vector $\boldsymbol{x}$ = (47.5, 56.5) and the utility value U = –62.5.

Suppose that the DM wants to use the approximate procedure.

Table 4.7 – Region's Data

Region	**Constraints**	**Optimal Solution to** ***(Maximin-l^1)***	**Optimal Solution to** ***(Minsum-l^1)***	**Ideal Objective Vector**	**Nadir Objective Vector**
I	$37.5<x_1<50$ $37.5<x_2<75$	[37.5,53.5]	[50,69]	[46,382]	[26,407]
II	$50<x_1<62.5$ $75<x_2<87.5$	[52.5,87.5]	[50,75]	[42,396]	[32,436]
III	$87.5<x_1<100$ $37.5<x_2<50$	[100,44]	[87.5,37.5]	[49,412]	[30,482]

Suppose the DM wants to start the search from Region I.

Region I

$\boldsymbol{R}$ = (46, 382)

$\boldsymbol{Q}$ = (26, 407)

Assume that the DM specifies $\boldsymbol{G}^0$ = (32, 390) as the reference point and $\boldsymbol{w}^0$ = (0.8, 0.2) as the weight vector.

By solving the relaxed version of the (ASP), the first continuous solution is found as $\boldsymbol{y}^0$ = (36.5, 382) with the decision vector $\boldsymbol{x}^0$ = (50, 56.5).

Suppose that the DM is not sure about the continuous solution and wants to see some alternative solutions. For finding these, she/he wants to use the direction search in Approach 2. Let the direction vector is $\boldsymbol{\Delta d}$ = (3.5, 2) with p = 2.

p = 1, $\boldsymbol{G}^1$ = (40, 384) $\boldsymbol{y}^1$ = (39.87, 388.73) $\boldsymbol{x}^1$ = (50, 59.87) U($\boldsymbol{y}^1$) = - 64.32
p = 2, $\boldsymbol{G}^2$ = (43.5, 386) $\boldsymbol{y}^2$ = (43.23, 395.47) $\boldsymbol{x}^2$ = (50, 63.23) U($\boldsymbol{y}^2$) = - 107.37

The most preferred continuous vector is $\boldsymbol{K} = \boldsymbol{y}^1$ = (39.87, 388.73). The closest integer nondominated vector is $\boldsymbol{C}^1$ = (39.87, 388.73) with $\boldsymbol{x}$ = (46.63, 56.5). Suppose the DM likes the solution. Since $\boldsymbol{C}^1$ is not dominated by any vector in *LIFV,* it is added to the *LCLP* as a candidate location point.

Suppose that the DM wants to continue the search in Region II.

Region II

$\boldsymbol{R}$ = (42, 396)

$\boldsymbol{Q}$ = (32, 436)

Assume that the DM specifies $\boldsymbol{G^0}$ = (35, 410) as the reference point and $\boldsymbol{w^0}$ = (0.3, 0.7) as the weight vector. The first continuous solution is $\boldsymbol{y^o}$ = (37.02, 401.03) with the decision vector $\boldsymbol{x^0}$ = (50, 75).

Suppose that the DM is not sure about the continuous solution and wants to see some alternative continuous solutions. For finding these, she/he wants to use the direction search in Approach 1.

Percent deviation of the first continuous solution from the first reference points is found as 8 %.

d = (2.02 / 35) + (8.97 / 410) = 0.058 + 0.022 = 0.08

When we perturb $\boldsymbol{G^0}$ in both coordinates with this deviation with P = 2, the resulting reference points and the nondominated continuous objective vectors associated are as below.

i = 1, $\boldsymbol{G^1}$ = (32.2, 410)	$\boldsymbol{y^1}$ = (34.73, 398.73)	$\boldsymbol{x^1}$ = (50, 75)	U($\boldsymbol{y^1}$) = -241.76
i = 2, $\boldsymbol{G^2}$ = (33.6, 410)	$\boldsymbol{y^2}$ = (35.88, 399.88)	$\boldsymbol{x^2}$ = (50, 75)	U($\boldsymbol{y^2}$) = -245.91
i = 3, $\boldsymbol{G^3}$ = (35, 442.8)	$\boldsymbol{y^3}$ = (42.10, 411.20)	$\boldsymbol{x^3}$ = (50, 75)	U($\boldsymbol{y^3}$) = -450.13
i = 4, $\boldsymbol{G^4}$ = (35, 426.4)	$\boldsymbol{y^4}$ = (39.56, 406.12)	$\boldsymbol{x^4}$ = (50, 75)	U($\boldsymbol{y^4}$) = -335.44

The most preferred continuous vector is $\boldsymbol{K} = \boldsymbol{y^1}$ = (34.73, 398.73). The closest integer nondominated vector is $\boldsymbol{C^2}$ = (34.31, 400.61) with $\boldsymbol{x}$ = (50, 77.31). Obviously, $\boldsymbol{C^2}$ is dominated by $\boldsymbol{C^1}$ and deleted from further consideration.

Suppose that the DM wants to stop the search. The final candidate location point is the single element of the *LCLP* which is $\boldsymbol{C^1}$ = (39.87, 388.73). Hence, the semi-desirable facility should be located at $\boldsymbol{x}$ = (46.63, 56.5). In this case the facility is 39.87 distance measure far from the closest demand point, while it has a total distance of 388.73 from all the demand points.

Example 4.2. Consider a single semi-desirable facility location problem in a 100 x 100 square. Suppose that there are 2000 demand points uniformly generated in the interval [0,100].

We assume that the DM has an underlying general monotone utility function:

$$U = \sum_{i=1}^{2} 0.5\ U_i \quad \text{where,}$$

$$U_i = \begin{cases} (z_i - Q_i)^2 & Q_i \le z_i \le a_i \\ 2(a_i - Q_i)^2 - (R_i - z_i)^2 & a_i < z_i \le R_i \end{cases} \quad \text{and} \quad a_i = (R_i + Q_i)/2$$

U	:	Utility function
R_i	:	i^{th} coordinate of the ideal objective vector
Q_i	:	i^{th} coordinate of the nadir objective vector
z_i	:	i^{th} coordinate of an objective vector

Parameters

Branch and Bound Strategies: Combination 3

M = 200

α = 12.5

$\rho = 10^{-3}$

T = 100

In this example, since the demand points are uniformly generated, function values of incumbent points are expected to be close to each other. Besides, the upper and lower bounds on the objective functions are expected to be similar in the regions. Hence, obviously, for any elimination, the feasible region should be divided more compared to the previous example.In addition, as a result of the uniformity in the demand points it will be difficult to eliminate big regions at once. Based on these, the resulting elimination pattern is expected to be more uniform and slower.

<u>Phase 1</u>

Table 4.8 shows the *LCES* along with the upper bound of squares on maximin objective and their optimal minsum objective value at the end of Phase 1.

Table 4.8 - *LCES* at the End of Phase 1

i	$UB(S^i)$	$W^*(S^i)$	i	$UB(S^i)$	$W^*(S^i)$
9	6.87	132240.42	48	5.59	102330.42
11	5.75	129212.78	54	5.56	114068.56
18	6.47	130740.82	56	5.29	129560.04
20	5.03	127713.18	57	6.90	123975.66
25	5.34	129215.68	58	4.97	139486.76
26	6.45	112859.26	59	6.14	112204.98
29	6.13	140986.36	60	7.03	127716.08
30	5.02	124629.94	61	4.86	115047.56
32	7.64	140121.42	62	5.92	99462.10
33	6.56	157262.58	63	4.91	102297.88
34	4.96	140906.16	64	4.27	102304.68
36	5.15	125041.50	65	5.18	105140.46
38	4.66	111258.52	66	4.82	102486.84
40	6.38	126750.00	67	4.81	105322.62
41	5.61	130985.62	68	5.56	99459.20
42	6.36	127534.74	69	5.47	102294.98

Table 4.8 – (cnt'd)

44	4.43	111670.08	70	6.11	127509.00
45	6.56	102938.72	73	5.05	114480.12
46	4.25	99487.84	74	4.80	105963.46
47	5.15	105781.30	76	4.14	102935.82
			77	4.84	99484.94

Table 4.9 shows the *LIFV* at the end of Phase 1. At each branching 100 points are selected. After making a dominance check at each step, 31 points are left.

Table 4.9 – *LIFV* at the End of Phase

i	**x^i**	**$L(x^i)$**	**$W(x^i)$**	**i**	**x^i**	**$L(x^i)$**	**$W(x^i)$**
1	[51.00,49.50]	1.62	99459.35	16	[25.50,51.25]	4.58	112381.02
2	[51.00,49.00]	2.12	99467.99	17	[25.25,51.00]	4.65	112626.17
3	[50.50,49.00]	2.61	99475.92	18	[25.25,51.25]	4.83	112639.00
4	[51.25,46.75]	2.98	99628.81	19	[77.00,80.00]	4.85	131492.13
5	[51.50,46.25]	3.13	99700.84	20	[2.19,38.12]	4.88	149866.23
6	[51.50,46.00]	3.38	99740.29	21	[2.22,37.78]	4.92	149964.83
7	[50.00,58.00]	3.41	100668.79	22	[2.19,37.81]	4.99	150014.76
8	[59.00,59.75]	3.66	102536.23	23	[1.88,37.81]	5.00	150614.51
9	[59.25,60.00]	3.69	102713.02	24	[0.31,49.69]	5.13	150791.76
10	[62.25,62.50]	3.81	105014.31	25	[0.00,49.69]	5.44	151414.99
11	[62.50,62.50]	4.06	105140.16	26	[99.75,36.75]	5.48	152478.23
12	[62.75,62.50]	4.21	105269.38	27	[100.00,37.0]	5.52	152851.22
13	[62.75,62.75]	4.26	105383.51	28	[100.00,36.75]	5.73	152977.36
14	[25.75,51.25]	4.33	112126.25	29	[0.00,37.00]	5.82	154691.89
15	[25.50,51.00]	4.46	112368.19	30	[0.00,36.88]	5.95	154754.79
				31	[0.00,36.67]	6.04	154860.21

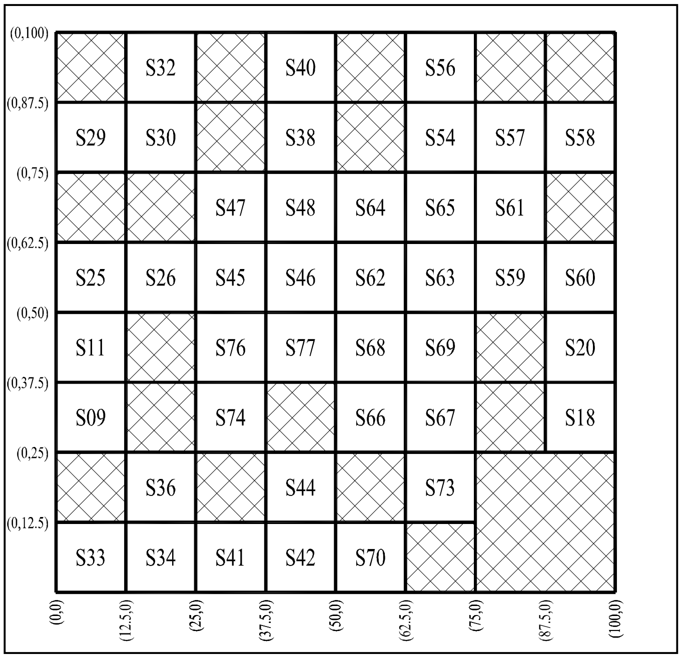

Figure 4.4 - Reduced Feasible Region at the End of Phase 1

Figure 4.4 shows the elimination pattern achieved in Phase 1.

<u>Phase 2</u>

In this phase, the optimal maximin objective values for each square in the *LCES* (see Table 4.8) are found. As seen from the below table, when the optimal maximin values are found, it is observed that 21 squares are dominated by *LIFV*.

Table 4.10 – 1st Elimination in *LCES* at Phase 2

i	$L^*(S^i)$	$UB(S^i)$	$W^*(S^i)$	Status
9	6.09	6.87	132240.42	CANDIDATE
11	5.55	5.75	129212.78	CANDIDATE
18	5.75	6.47	130740.82	CANDIDATE
20	5.03	5.03	127713.18	CANDIDATE
25	5.34	5.34	129215.68	CANDIDATE
26	4.87	6.45	112859.26	CANDIDATE
29	4.39	6.13	140986.36	**DOMINATED**
30	3.29	5.02	124629.94	**DOMINATED**
32	4.97	7.64	140121.42	CANDIDATE
33	5.35	6.56	157262.58	**DOMINATED**
34	4.08	4.96	140906.16	**DOMINATED**
36	3.87	5.15	125041.50	**DOMINATED**
38	3.61	4.66	111258.52	**DOMINATED**
40	4.38	6.38	126750.00	**DOMINATED**
41	4.81	5.61	130985.62	**DOMINATED**
42	3.96	6.36	127534.74	**DOMINATED**
44	3.45	4.43	111670.08	**DOMINATED**
45	4.86	6.56	102938.72	CANDIDATE
46	3.42	4.25	99487.84	CANDIDATE
47	3.10	5.15	105781.30	**DOMINATED**
48	3.41	5.59	102330.42	**DOMINATED**
54	4.89	5.56	114068.56	CANDIDATE
56	4.03	5.29	129560.04	**DOMINATED**
57	4.89	6.90	123975.66	CANDIDATE
58	3.47	4.97	139486.76	**DOMINATED**
59	3.82	6.14	112204.98	**DOMINATED**
60	4.39	7.03	127716.08	**DOMINATED**
61	3.59	4.86	115047.56	**DOMINATED**
62	4.07	5.92	99462.10	CANDIDATE
63	4.27	4.91	102297.88	CANDIDATE

Table 4.10 – (cnt'd)

64	4.17	4.27	102304.68	CANDIDATE
65	4.36	5.18	105140.46	CANDIDATE
66	3.65	4.82	102486.84	CANDIDATE
67	3.89	4.81	105322.62	**DOMINATED**
68	3.53	5.56	99459.20	CANDIDATE
69	3.89	5.47	102294.98	CANDIDATE
70	3.95	6.11	127509.00	**DOMINATED**
73	3.61	5.05	114480.12	**DOMINATED**
74	3.43	4.80	105963.46	**DOMINATED**
76	4.14	4.14	102935.82	CANDIDATE
77	3.47	4.84	99484.94	CANDIDATE

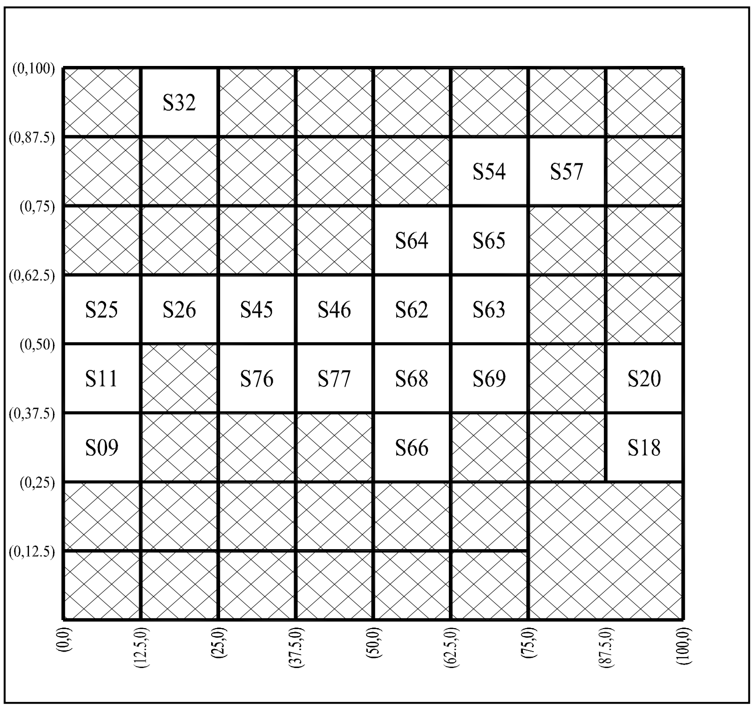

Figure 4.5 - Reduced Feasible Region at the End of Phase 2 ***(First Elimination)***

After the eliminations in Phase 1 and Phase 2, 20 squares remain (see Figure 4.5). At this point, the DM is asked whether she/he wants to divide the regions for any further elimination. Suppose the answer is yes with the second phase stopping side length (β) = 6.25.

With this further division, 80 additional squares are generated. At this step, 62 of them are eliminated with the optimal values and the same incumbent value list. Table 4.11 shows the percent elimination achieved in two phases. Table 4.12 shows the optimal maximin and minsum values of the remaining squares.

Table 4.11 – Percent Elimination Achieved

Phase 1	**Phase 2**		**Overall**
	First Elimination	**Second Elimination**	
35.94	32.81	24.22	92.97

Table 4.12 – *LCES* at the End of Phase 2

i	**$L(S^i)$**	**$W^*(S^i)$**	**I**	**$L(S^i)$**	**$W^*(S^i)$**
80	6.10	142697.00	122	4.86	107074.02
89	5.75	140860.82	127	2.67	99487.84
90	5.32	140400.04	129	3.42	100220.20
92	5.55	139669.36	130	2.79	99462.10
101	3.10	99484.94	132	3.42	100194.46
104	3.53	99459.20	133	4.07	100751.36
111	5.03	138563.86	136	4.27	103030.24
114	5.34	139672.26	139	4.17	102861.58
119	4.87	112859.26	142	4.37	105140.46

Figure 4.6 shows the feasible region after the second elimination in Phase 2.

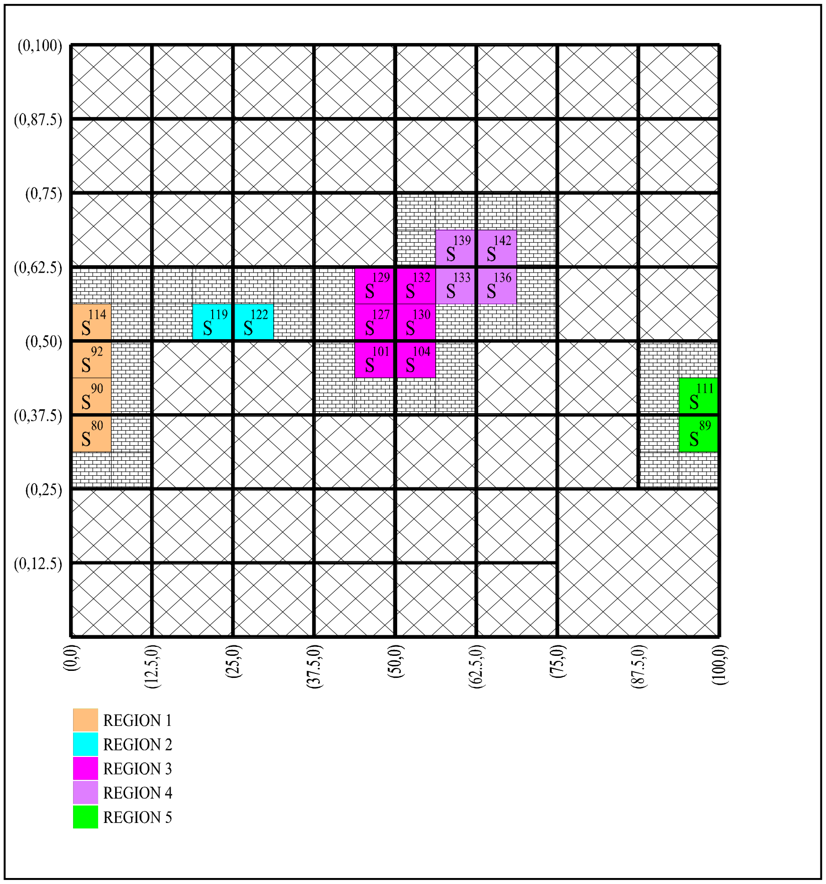

Figure 4.6 - Reduced Feasible Region at the End of Phase 2
(Second Elimination)

<u>Phase 3</u>

As seen from Figure 4.6, the remaining squares can be considered as 5 regions. The data related with the regions are given in Table 4.13. Suppose that the DM wants to use the approximate procedure.

Table 4.13 – Region's Data

Region	Constraints	Optimal Solution to *(Maximin-l^1)*	Optimal Solution to *(Minsum-l^1)*	Ideal Objective Vector	Nadir Objective Vector
I	$0<x_1<6.25$ $31.25<x_2<56.25$	[0,36.73]	[6.25,49.56]	[6.09,139669.36]	[2.03,154828.16]
II	$18.75<x_1<31.25$ $50<x_2<56.25$	[24.9,51.57]	[31.25,50]	[4.87,107074.02]	[1.15,113024.49]
III	$43.75<x_1<56.25$ $43.75<x_2<62.5$	[51.53,45.89]	[51.25,49.56]	[3.53,99459.20]	[1.31,99758.55]
IV	$56.25<x_1<68.75$ $56.25<x_2<68.75$	[62.71,62.61]	[56.25,56.25]	[4.37,100751.36]	[2.46,105298.47]
V	$93.75<x_1<100$ $31.25<x_2<43.75$	[100,36.78]	[93.75,43.75]	[5.75,138563.86]	[0.92, 152962.18]

Region I

$\boldsymbol{R}$ = (6.09, 139669.36)

$\boldsymbol{Q}$ = (2.03, 154828.16)

Assume that the DM specifies $\boldsymbol{G^0}$ = (5, 140000) as the reference point and $\boldsymbol{w^0}$ = (0.5, 0.5) as the weight vector. The first continuous solution is $\boldsymbol{y^o}$ = (5.01, 139676.07) with the decision vector $\boldsymbol{x^0}$ = (6.25, 49.66).

Suppose that the DM is not sure about the continuous solution and wants to see some alternative solutions. For finding these, she/he wants to use the direction search in Approach 2. Let the direction vector is $\boldsymbol{\Delta d}$ = (0.5, 5000) with p = 2.

when;

p = 1, $\boldsymbol{G}^1$ = (5.51, 144676.07) $\boldsymbol{y}^1$ = (5.71, 139681.21)
$\boldsymbol{x}^1$ = (6.25, 49.68) U($\boldsymbol{y}^1$) = 114715040.54

p = 2, $\boldsymbol{G}^2$ = (6.01, 149676.07) $\boldsymbol{y}^2$ = (6.4, 139687.71)
$\boldsymbol{x}^2$ = (6.25, 49.66) U($\boldsymbol{y}^2$) =114616619.74

$\boldsymbol{K} = \boldsymbol{y}^1$ = (5.71, 139681.21) is selected by the DM. The closest integer nondominated vector is $\boldsymbol{C}^1$ = (3.06, 139692.82) with $\boldsymbol{x}$ = (6.25, 48.54). Since $\boldsymbol{C}^1$ is dominated by $\boldsymbol{x}^5$ in *LIFV,* it is deleted.

Region IV

$\boldsymbol{R}$ **=** (4.37, 100751.36)
$\boldsymbol{Q}$ = (2.46, 105298.47)

Assume that the DM specifies $\boldsymbol{G}^0$ = (4.2, 104000) as the reference point and $\boldsymbol{w}^0$ = (0.9, 0.1) as the weight vector.

Suppose that the DM wants to see closest integer nondominated vector to $\boldsymbol{G}^o$ directly. It is found as $\boldsymbol{C}^4$ = (3.49, 102479.02) with $\boldsymbol{x}$ = (58.88, 59.70).

We assume that she/he likes the solution. $\boldsymbol{C}^4$ is added to *LCLP* as a candidate location point.

The final candidate location point is the single element of *LCLP* which is $\boldsymbol{C}^4$ = (3.49, 102479.02). Hence, the semi-desirable facility should be located at $\boldsymbol{x}$ =

(58.88, 59.70). In this case, the facility is 3.49 distance measure far from the closest demand point, while it has a total distance of 102479.02 from all the demand points.

Example 4.3. In this example, we would like to solve the rectilinear version of the real life example given by Skriver and Andersen (2003).

The problem is to solve the uncertainty on where to locate a new international airport in Jutland, Denmark replacing the old one with the fact that the new airport is attractive to a lot of companies and people living nearby the city Arhus.

The region for potential locations is with the x_1-coordinates between 60 and 140; and x_2-coordinates between 100 and 180. In addition, Jutland area is divided into three weighting zones, 100 % zone, 50% zone and 20% zone which reflect the fact that the customers far from the stated region will use the new airport less frequently compared with the customers living closer or inside the region. (see Figure 4.7)

Until now, we have not attributed weights to the demand points while solving *(Maximin-l^1)*. However, in this example, we will use the following weights in coordination with Skriver and Andersen (2003)

w_j^1 = 'population in city j'

To give more importance to larger cities

w_j^2 = 'population in city j multiplied by the weight of the zone'

To give more importance to larger cities and the cities nearby Arhus

The authors have chosen 42 cities to represent the demand points in Jutland the coordinates of which will be presented later. The problem data are presented in the table below.

Table 4.14 – Problem Data

City i	b_1^i	b_2^i	w_1^i	w_2^i	City i	b_1^i	b_2^i	w_1^i
1	7.17	69.31	73422	14684.4	22	88.43	78.39	29376
2	34.42	8.13	8161	1632.2	23	74.09	42.06	21106
3	28.2	52.1	8046	1609.2	24	69.31	20.08	16218
4	7.18	68.83	53012	26506	25	52.58	66.44	8507
5	76	93.21	47839	23919.5	26	99.9	267.2	11365
6	95.12	110.42	48410	48410	27	103.25	289.67	24889
7	99.9	130.49	12067	12067	28	135.27	288.23	24768
8	118.07	142.92	215587	215587	29	83.55	169.25	7201
9	106.12	177.34	56123	56123	30	10.99	83.17	12478
10	67.88	175.9	31872	31872	31	38.72	97.03	9497
11	76.96	146.75	36762	36762	32	11.95	119.98	6949
12	52.1	141.01	14014	7007	33	-4.3	135.27	9166
13	40.63	141.97	29231	14615.5	34	50.67	119.02	6214
14	18.64	166.34	30770	15385	35	0	185.46	7302
15	17.2	181.21	11272	5636	36	33.46	212.71	9319
16	44.45	188.33	20557	10278.5	37	25.33	231.35	12609
17	108.03	162.04	6616	6616	38	19.12	249.52	2574
18	158.7	172.08	14441	14441	39	58.32	244.74	3332
19	91.3	196.94	10704	5352	40	100.86	301.14	6949
20	74.57	216.06	7066	3533	41	136.71	315.96	10674
21	98.47	240.43	119157	59578.5	42	146.75	144.83	4396

Parameters

Branch and Bound Strategies: Combination 3

M = 1000

$\alpha = 10$

$\rho = 10^{-8}$

T = 100

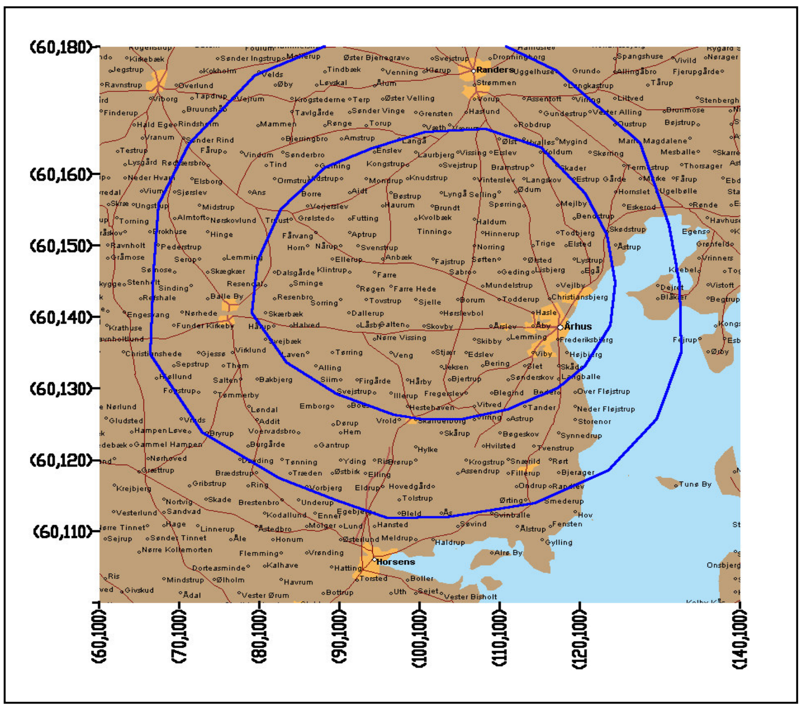

Figure 4.7 – Jutland Area

Phase 1

Table 4.15 shows the *LCES* with the upper bound of the squares on the maximin objective and their optimal minsum value at the end of Phase 1.

Table 4.15 - *LCES* at the End of Phase 1

i	$UB(S^i)$	$W^*(S^i)$	i	$UB(S^i)$	$W^*(S^i)$
8	393652.12	53016039.61	24	178248.78	40714101.10
9	358625.68	54163318.98	27	246046.13	45211201.85
10	329208.78	49076258.52	28	280637.47	42902196.33
11	314665.68	50223537.89	29	234220.47	41642732.26
12	248459.53	45260932.52	32	253703.93	41923834.49
19	362586.90	54225773.00	33	248593.47	40664370.43
20	405575.65	52966308.94	37	277221.34	45161481.11
21	297180.76	50285991.91	39	277221.34	44183119.27
22	314829.37	49026527.85	40	277221.34	48585869.53

Table 4.16 shows the *LIFV* at the end of Phase 1. At each branching 100 points are selected. After making a dominance check at each step, 25 points are left.

Table 4.16 – *LIFV* at the End of Phase 1

i	x^i	$L^*(S^i)$	$W^*(S^i)$
1	[98.00,144.00]	185711.06	40836320
2	[96.00,144.00]	198943.06	41013376
3	[94.00,144.00]	212175.06	41298872
4	[92.00,144.00]	225407.06	41669564
5	[92.00,142.00]	234220.42	41824872
6	[88.00,140.00]	242673.67	43306932

Table 4.16 – (cnt'd)

7	[78.00,140.00]	250594.83	45641140
8	[76.00,140.00]	264996.81	46191384
9	[80.00,134.00]	275342.37	47288532
10	[110.00,116.00]	288289.69	51779496
11	[108.00,114.00]	296727.56	52165900
12	[110.00,114.00]	297081.69	52560548
13	[108.00,112.00]	314665.69	52946952
14	[108.00,110.00]	323457.69	53768668
15	[106.00,108.00]	341041.69	54388252
16	[104.00,106.00]	344995.56	55218856
17	[106.00,106.00]	349833.69	55362944
18	[96.00,104.00]	353392.78	56109048
19	[104.00,104.00]	367417.69	56193548
20	[102.00,102.00]	369129.56	57036596
21	[104.00,102.00]	376209.69	57168240
22	[96.00,102.00]	387442.87	57083740
23	[98.00,100.00]	390850.22	57881380
24	[102.00,100.00]	393263.56	58011288
25	[96.00,100.00]	399870.87	58058432

Phase 2

There is no further elimination in Phase 2.

Figure 4.8 shows the feasible region at the end of Phase 2.

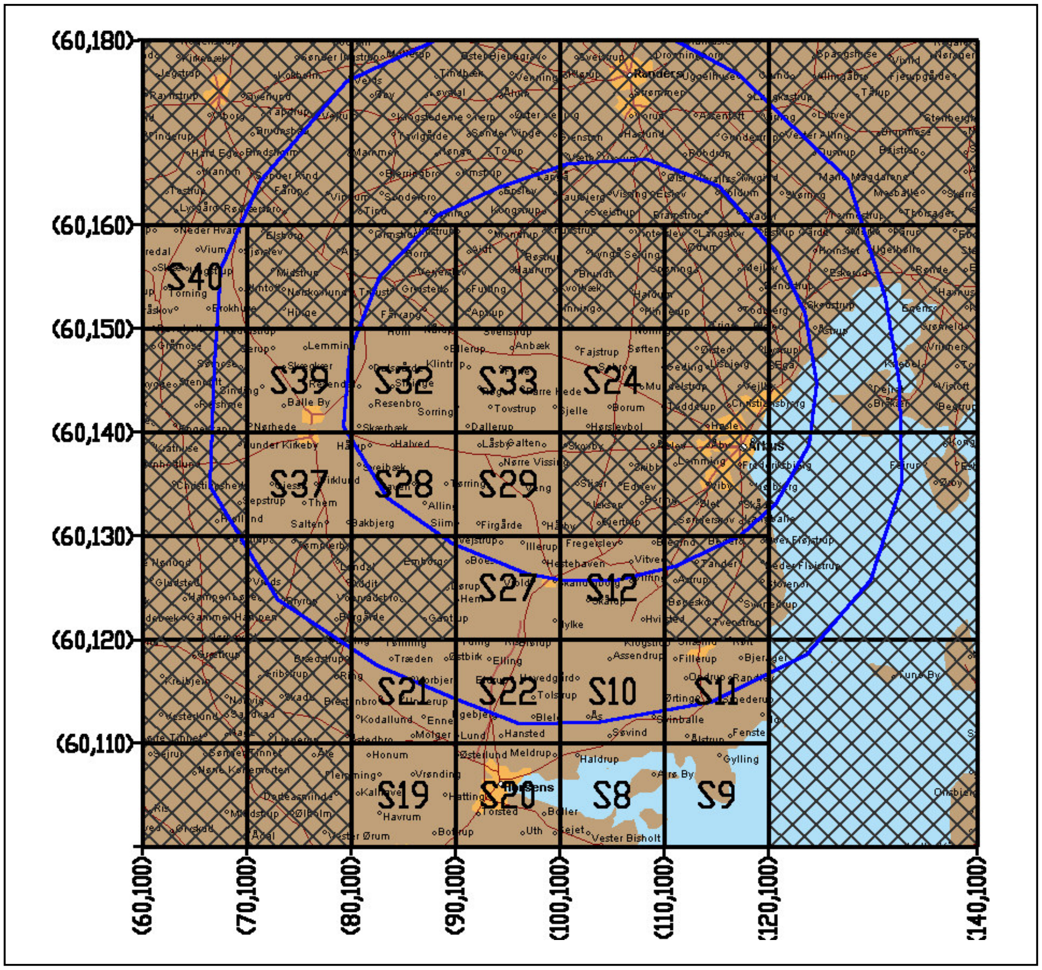

Figure 4.8 - Reduced Feasible Region at the End of Phase 2

The below table shows the percent elimination achieved in Phase 1 and Phase 2.

Table 4.17 – Percent Elimination Achieved

Phase 1	Phase 2	Overall
71.87	0	71.87

Phase 3

Instead of presenting the squares separately to the DM, the squares are combined in five regions.

Region I : Squares19, 20, 8, 9, 21, 22, 10, 11
Region II : Squares 27, 12
Region III : Squares 37, 28, 29, 39, 32, 33
Region IV : Square 24
Region V : Square 40

Suppose that the DM wants to search the regions with the exact approach.

Table 4.18 – Region's Data

Region	**Constraints**	**Optimal Solution to *(Maximin-l^1)***	**Optimal Solution to *(Minsum-l^1)***	**Ideal Objective Vector**	**Nadir Objective Vector**
I	$80<x_1<120$ $100<x_2<120$	[96.83, 100]	[98.47, 120.0]	[405008.13, 49026530]	[143838.72, 57985220]
II	$90<x_1<110$ $120<x_2<130$	[110, 120]	[98.47, 130.0]	[248459.53, 45211200]	[23168.72, 50223536]
III	$70<x_1<100$ $130<x_2<150$	[71.38,142.92]	[98.47,142.92]	[277221.34, 40664370]	[167248.58, 46927152]
IV	$100<x_1<110$ $140<x_2<150$	[104.44, 140]	[100.0, 142.92]	[169558.61, 40714100]	[151199.42, 41977592]
V	$60<x_1<70$ $150<x_2<160$	[64.302, 150]	[70.0, 150.0]	[277221.34, 48585870]	[236192.83, 50992876]

Suppose that the DM wants to start the search from Region 3. Assume that she/he specifies below reference point and weights.

$\boldsymbol{G}^0 = (250000, 45000000)$

$\boldsymbol{w}^0 = (0.9, 0.1)$

The closest integer nondominated vector in region 3 is found as

$\boldsymbol{C}^3 = (248534.05, 42131957)$ with $\boldsymbol{x} = (90.66, 141.85)$

Dominance Check

$\boldsymbol{C}^3$ is not dominated by the *LIFV*, so we project it onto the other regions.

Region 1: $\boldsymbol{C}^1 = (242332.21, 501232470)$ with $\boldsymbol{x} = (109.49, 120)$
$\boldsymbol{C}^1$ does not dominate $\boldsymbol{C}^3$. It is dominated by $\boldsymbol{C}^3$.

Region 2: $\boldsymbol{C}^3$ dominates the ideal point, region 2 is deleted from further consideration.

Region 4: $\boldsymbol{C}^4 = (169558.61, 40806177)$ with $\boldsymbol{x} = (101.52, 142.92)$
$\boldsymbol{C}^4$ does not dominate $\boldsymbol{C}^3$. $\boldsymbol{C}^4$ is not dominated by *LIFV* thus added to the *LCNV*.

Region 5: $\boldsymbol{C}^5 = (244375.24, 490168170)$ with $\boldsymbol{x} = (68.86, 150)$
$\boldsymbol{C}^5$ does not dominate $\boldsymbol{C}^3$. It is dominated by $\boldsymbol{C}^3$.

$\boldsymbol{C}^3$ is proved to be nondominated and added to the *LCLP*.

Suppose that the DM would like to know whether $\boldsymbol{C}^4$ is nondominated or not. It is deleted from the *LCNV*.

$\boldsymbol{G}^1 = \boldsymbol{C}^4 = (169558.61, 40806177)$

$\boldsymbol{w}^1 = (0.5, 0.5)$

Dominance Check

$\boldsymbol{C}^4$ is projected onto the other regions.

Region 1: $\boldsymbol{C}^1 = (143838.64, 49026537)$ with $\boldsymbol{x} = (98.47, 120)$
$\boldsymbol{C}^1$ does not dominate $\boldsymbol{C}^4$. It is dominated by $\boldsymbol{C}^4$.

Region 3: $\boldsymbol{C}^3 = (170265.58, 40686507)$ with $\boldsymbol{x} = (98.22, 142.92)$
$\boldsymbol{C}^3$ dominates $\boldsymbol{C}^4$. $\boldsymbol{C}^3$ is not dominated by *LIFV* thus added to the *LCNV*.

$\boldsymbol{C}^4$ is proved to be dominated.

Suppose that the DM would like to know whether $\boldsymbol{C}^3$ is nondominated or not. It is deleted from the *LCNV*.

$\boldsymbol{G}^2 = \boldsymbol{C}^3 = (170265.58, 40686507)$

$\boldsymbol{w}^2 = (0.5, 0.5)$

Dominance Check

$\boldsymbol{C}^3$ is projected onto the other regions.

Region 1: $\boldsymbol{C}^1$ = (148838.64, 49026537) with $\boldsymbol{x}$ = (98.47, 120)

$\boldsymbol{C}^1$ does not dominate $\boldsymbol{C}^3$. It is dominated by $\boldsymbol{C}^3$.

Region 4: $\boldsymbol{C}^3$ dominates the ideal point, region 4 is deleted from further consideration.

Region 5: $\boldsymbol{C}^5$ = (236192.8, 48585877) with $\boldsymbol{x}$ = (70,150)

$\boldsymbol{C}^5$ does not dominate $\boldsymbol{C}^3$. $\boldsymbol{C}^5$ is dominated by the *LCLP,* thus deleted.

$\boldsymbol{C}^3$ is proved to be nondominated and added to the *LCLP* as $\boldsymbol{C}^{32}$ since there is an element of the *LCLP* from region 3 which we will name $\boldsymbol{C}^{31}$.

Suppose the DM wants to continue searching from region 1 and specifies the below reference point and weights.

$\boldsymbol{G}^3$ = (400000, 51000000)

$\boldsymbol{w}^3$ = (0.5, 0.5)

The closest integer nondominated vector in region 1 is found as

$\boldsymbol{C}^1$ = (365128, 55992647) with $\boldsymbol{x}$ = (104.09, 104.43)

Dominance Check

$\boldsymbol{C}^1$ is not dominated by the *LIFV*, so we project it onto the other regions.

Region 3: $\boldsymbol{C}^3$ = (277221.34, 46791897) with $\boldsymbol{x}$ = (76, 138.302)

$\boldsymbol{C^3}$ does not dominate $\boldsymbol{C^1}$. It is not dominated by the *LIFV* and the *LCLP,* thus added to the *LCNV*.

Region 5: $\boldsymbol{C^5}$ = (277221, 50992717) with $\boldsymbol{x}$ = (76, 138.302)

$\boldsymbol{C^5}$ does not dominate $\boldsymbol{C^1}$. It is dominated by $\boldsymbol{C^3}$, thus deleted.

$\boldsymbol{C^1}$ is proved to be nondominated and added to the *LCLP*.

The elements of the *LCLP* are $\boldsymbol{C^{31}}$ = (248534.05, 42131957), $\boldsymbol{C^{32}}$= (170265.58, 40686507) and $\boldsymbol{C^1}$= (365128, 55992647).

The *LCLP* is presented to the DM. Suppose that DM is satisfied but can not decide between the alternatives.

Suppose that the DM specifies the indifference (q_i) and preference (p_i) threshold levels as follows:

$q_1 = 5000$ $p_1 = 10000$

$q_2 = 500000$ $p_2 = 1000000$

Let the weights be $\boldsymbol{w}$ = (0.3, 0.7)

$F_1(\boldsymbol{C^{31}}, \boldsymbol{C^{32}}) = 1$, $F_1(\boldsymbol{C^{32}}, \boldsymbol{C^{31}}) = 0$, $F_1(\boldsymbol{C^{1}}, \boldsymbol{C^{31}}) = 1$,

$F_1(\boldsymbol{C^{31}}, \boldsymbol{C^{1}}) = 0$, $F_1(\boldsymbol{C^{32}}, \boldsymbol{C^{1}}) = 0$, $F_1(\boldsymbol{C^{1}}, \boldsymbol{C^{32}}) = 1$,

$F_2(\boldsymbol{C^{31}}, \boldsymbol{C^{32}}) = 0$, $F_2(\boldsymbol{C^{32}}, \boldsymbol{C^{31}}) = 1$, $F_2(\boldsymbol{C^{1}}, \boldsymbol{C^{31}}) = 0$,

$F_2(\boldsymbol{C^{31}}, \boldsymbol{C^{1}}) = 1$, $F_2(\boldsymbol{C^{32}}, \boldsymbol{C^{1}}) = 1$, $F_2(\boldsymbol{C^{1}}, \boldsymbol{C^{32}}) = 0$,

$\pi(\boldsymbol{C^{31}},\boldsymbol{C^{32}}) = 0.3$ $\pi(\boldsymbol{C^{32}},\boldsymbol{C^{31}}) = 0.7$ $\pi(\boldsymbol{C^{1}},\boldsymbol{C^{31}}) = 0.3$

$\pi(\boldsymbol{C^{31}},\boldsymbol{C^{1}}) = 0.7$ $\pi(\boldsymbol{C^{32}},\boldsymbol{C^{1}}) = 0.7$ $\pi(\boldsymbol{C^{1}},\boldsymbol{C^{32}}) = 0.3$

$\phi^{+}(\boldsymbol{C^{31}}) = 1$ $\phi^{+}(\boldsymbol{C^{32}}) = 1.4$ $\phi^{+}(\boldsymbol{C^{1}}) = 0.6$

$\phi^{-}(\boldsymbol{C^{31}}) = 1$ $\phi^{-}(\boldsymbol{C^{32}}) = 0.6$ $\phi^{-}(\boldsymbol{C^{1}}) = 1.4$

$\phi\ (\boldsymbol{C^{31}}) = 0$ $\phi\ (\boldsymbol{C^{32}}) = 0.8$ $\phi\ (\boldsymbol{C^{1}}) = -0.8$

According to the net flows, $\boldsymbol{C^{32}}$ outranks $\boldsymbol{C^{31}}$ and $\boldsymbol{C^{1}}$. Therefore $\boldsymbol{C^{32}}$ should be selected as the final objective vector. Hence, the semi-desirable facility should be located in region 3 at $\boldsymbol{x}$ = (98.22, 142.92). In this case the weighted distance of the airport to the closest city is 170265.58 distance measure, and the total weighted distance to the demand points is 40686507 distance measure from all the demand points.

CHAPTER 5

CONCLUSION AND DIRECTIONS FOR FUTURE RESEARCH

In the first part of the thesis, we studied the undesirable facility location problem in a planar region, the findings of which were used in the semi-desirable facility location problem of the second part.

In Chapter 3, a mixed integer mathematical model suggested by Sayin (2000) has been used for the problem of locating an undesirable facility. This model as indicated in Chapter 2, is computationally prohibitive in the existence of large number of demand points and its solution time increases exponentially as the number of demand points increases. Believing that the model is very practical when compared to the approaches in the literature, we have given two solution approaches with which considerable saving in the solution time even for big demand point samples has been achieved. In the first approach, we have investigated different branch and bound strategies and conducted several tests to find out strategies improving the solution time. After selecting the promising strategies, their combined effects have been tested on several problems. Based on the results, it was observed that the solution time of the model has been reduced considerably with some strategy combinations, making the model practical even for big problems.

Further reduction in the solution time of the model has been made possible by use of bounds. We have used the upper bound suggested by White (1996) which, to our knowledge was never tested or used in the literature before, even though it was proved to be better than the upper bound suggested by Drezner and Wesolowsky (1983). Along with the upper bound, a lower bound has been used. Experiments showed that both bounds have a further improving effect on the solution time even though the upper bound was observed to be very loose in all of the test problems.

In the second approach of Chapter 3, a method named as 'Cut and Prune' has been proposed which is based on the idea that some parts of the feasible region can be proved not to contain the optimal point and can be fathomed by the use of upper and lower bounds and incumbent points. Considering that the pruning may not be possible with loose bounds, and White's upper bound failed to be tight in the test problems, the use of a supplementary upper bound was suggested. This approach was illustrated on example problems one of which showed that a problem with big sample of demand points can be solved efficiently with the proposed method.

The second concern of the study, the semi-desirable facility location problem was dealt with in Chapter 4. There has been limited research in the literature on this issue. For the solution of the problem, a new objective function was added to the mixed integer model of Chapter 3 for the purpose of the minimization of the service cost, considering that the facility has desirable properties as well as undesirable in this case. A three-phase interactive geometrical branch and bound algorithm was suggested for the solution of the biobjective model. In the first two phases, we aim to eliminate the parts of the feasible region the inefficiency of which can be proved. The third phase has been suggested for an interactive search in the remaining regions with the involvement of a DM.

In the third phase, the DM is given the opportunity to use either an exact or an approximate procedure to carry out the search. In the exact one, finding an efficient point at the end is guaranteed. This procedure is based on the reference point approach of Wierzbicki (1980) and requires the solution of a mixed integer model for all the regions. On the other hand, in the approximate procedure, an approximately efficient solution can be presented at the end. In this procedure, a hybrid methodology (Karaivanova et al., 1995) is used to increase the efficiency of the reference point approach. With this methodology, we approach the preference regions of the DM in continuous nondominated objective region before starting the search in the integer nondominated objective region. For finding the nondominated continuous solutions, we have used two methods; reference direction approach suggested by Korhonen and Laakso (1986) and perturbation of the reference points suggested by Wierzbicki (1980). The approximate procedure can be used when the DM prefers to see approximately efficient solutions to save on computation time.

The third phase also supports the DM with an outranking method when the search results in multiple efficient points among which the DM has a difficulty in selecting the final location point.

The first two phases of the algorithm was an adaptation of the GBSSS algorithm to the semi-desirable facility location problem. New bounding schemes were used compared to the bounds used in the literature, and whenever optimal values were calculated, the insights of Chapter 3 were used. However, the third phase was completely new considering that there is no interactive approach suggested for the semi-desirable facility location problem in the literature.

The solution approaches in the literature are either approximations that result in regions containing efficient points, or they are aimed at obtaining the complete efficient trajectory. Obviously, these approaches may cause information overload on the DM who may have difficulty in selecting the final location point. Considering these, we believe, our algorithm is the first decision support system for the problem, thus giving the DM an opportunity to have a single -efficient or approximately efficient- location point at the end.

An area for future research should consider forbidden regions which allow modeling real location areas with geographical barriers. In addition, a new distance gauge that properly defines the spread of pollution should be investigated. Moreover, different criteria involving environmental considerations such as, geographical, climatic, should be incorporated to the problem of this area. Another area is the multi-facility version of the problem which will be useful in modeling the real life location problems.

REFERENCES

1. Appa G.M. and Giannikos I. (1994), "Is Linear Programming Necessary for Single Facility Location with Maximin of Rectilinear Distance?", *J. Opl. Res. Soc.*, Vol. 45(1), pp. 97-107.

2. Brans J.P. and Vinckle P. (1985) 'A preference ranking organisation method : The PROMETHEE method for MCDM', *Management Science*, Vol. 31, 6, pp.647-656.

3. Brimberg J. and Juel H. (1998), "On Locating a Semi-desirable Facility on the Continuous Plane", *Int. Trans. Opl. Res*., Vol. 5(1), pp. 59-66.

4. Brimberg J. and Juel H. (1998), "A Bicriteria Model for Locating a Semi-Desirable Facility in the Plane", *European Journal of Operational Research*, Vol. 106, pp. 144-151.

5. Carrizosa E. and Plastria F. (1999), "Location of Semi-obnoxious Facilities", *Studies in Locational Analysis*, Issue 12, pp. 1-27.

6. Cplex (1998), "Using the Cplex Callable Library", Version 6.0, ILOG, Inc., NV, USA.

7. Dasarathy B. and White Lee J. (1980), "A Maxmin Location Problem", *Operations Research,* Vol. 28(6), pp. 1385-1401.

8. Drezner Z. and Wesolowsky G.O. (1980), "A Maximin Location Problem with Maximum Distance Constraints"*, AIIE Transactions,* Vol. 12(3), pp. 249-252.

9. Drezner Z. and Wesolowsky G.O. (1983), "The Location of an Obnoxious Facility with Rectangular Distances", *Journal of Regional Science*, Vol. 23(2), pp. 241-248.

10. Erkut E. and Neuman S. (1989), "Analytical Models for Locating Undesirable Facilities", *European Journal of Operational Research*, Vol. 40, pp. 275-291.

11. Fernandez F., Puerto J. and Rodriguez-Chia AM. (1997), "A Maxmin Location Problem with Nonconvex Feasible Region", *Journal of the Operational Research Society*, Vol. 48, pp. 479-489.

12. Gregory J. (2001), "Rethinking Your Cplex Parameter Settings", ILOG/CPLEX Optimization Newsletter', http://www.ilog.com/products /optimization/times_winter2001/expert.cfm

13. Hansen P., Peeters D. and Thisse J.-F. (1981), "On the Location of an Obnoxious Facility", *Sistemi Urbani*, Vol. 3, pp. 299-317.

14. Karasakal, E. K. and Köksalan, M. "Generating a Representative Subset of the Efficient Frontier in Multiple Criteria Decision Making," *Working Paper No: 01-20,* Faculty of Administration, University of Ottawa, 2001.

15. Karaivanova J., Korhonen P., Narula S., Wallenius J., Vassilev V. (1995), "A reference Direction Approach to Multiple Obective Integer Linear Programming", *European Journal of Operational Research*, Vol. 81, pp. 176-187.

16. Korhonen P. and Laakso J. (1986), "A Visual Interactive Method for Solving the Multiple Criteria Problem", *European Journal of Operational Research*, Vol. 23, pp. 161-179.

17. 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", *J. Opl. Res. Soc.*, Vol. 37(10), pp. 971-977.

18. Melachrinoudis E. and Cullinane T.P. (1985), "Locating an Undesirable Facility within a Geographical Region Using the Maximin Criterion", *Journal of Regional Science*, Vol. 25(1), pp. 115-127.

19. Melachrinoudis E. and Cullinane T.P. (1986), "Locating an Obnoxious Facility within a Polygonal Region", *Annals of Operations Research*, Vol. 6, pp. 137-145.

20. Melachrinoudis E. (1988), "An Efficient Computational Procedure for the Rectilinear Maximin Location Problem", *Transportation Science*, Vol. 22(3), pp. 217-223.

21. Melachrinoudis E. (1999), "Bicriteria Location of a Semi-obnoxious Facility", *Computers & Industrial Engineering*, Vol. 37, pp. 581-593.

22. Melachrinoudis E. and Xanthopulos Z. (2003), "Semi-obnoxious Single Facility Location in Euclidean Space", *Computers & Operations Research*, Vol. 30, pp. 2191-2209.

23. Morales D.R., Carrizosa E. and Conde E. (1997), "Semi-obnoxious Location Models: A Global Optimization Approach", *European Journal of Operational Research*, Vol. 102, pp. 295-301.

24. Plastria F. (1992), "GBSSS: The Genaralized Big Square Small Square Method for Planar Single-facility Location", *European Journal of Operational Research*, Vol. 62, pp. 163-174.

25. Plastria F. (1996), "Optimal Location of Undesirable Facilities: A Selective Overview", *Belgian Journal of Operations Research, Statistics and Computer Science*, Vol. 36(2-3), pp. 109-127.

26. Sayin S. (2000), "Measuring the Quality of Discrete Representations of Efficient Sets in Multiple Objective Mathematical Programming", Mathematical Programming, 87, 543-560.

27. Sayin S. (2000), "A Mixed Integer Programming Formulation for the l-maximin Problem*", Journal of the Operational Research Society*, Vol. 51, pp. 371-375.

28. Sayin S. (2003), "A Procedure to Find Discrete Representations of the Efficient Set with Specified Coverage Errors", Operations Research, 51, 3, 427-436.

29. Shamos M.I. (1975), "Geometric Complexity", *Proceedings of the Seventh ACM Symposium on Theory of Computing*, pp.224-233.

30. Shamos M.I. and and Hoey D. (1975), "Closest Point Problems", *16th Annual IEEE Symposium on Foundations of Computer Science*, pp. 151-162.

31. Skriver A.J.V. and Andersen K.A. (2003), "The Bicriterion Semi-obnoxious Location (BSL) Problem Solved by an $\in$-Approximation", *European Journal of Operational Research*, Vol. 146, pp. 517-528.

32. Steuer R.E. and Harris F.W. (1980), "Intra-set Point Generation and Filtering in Decision and Criterion Space", *Computers and Operations Research*, Vol. (7), pp. 41-53.

33. Steuer R.E. (1986), "Multiple Criteria Optimization: Theory, Computation and Application", John Wiley, New York, p. 245.

34. Wierzbicki A.(1980), "The use of Reference Objective in Multiobjective Optimization", in G. Fandel and T. Gal (Eds.) Multiple Criteria Decision Making, Theory and Application, Springer-Verlag, Berlin, pp. 468-486

35. White D.J. (1996), "Rectilinear Location Revisited", *Journal of the Operational Research Society*, Vol. 47, pp. 181-187.

APPENDIX A

EXPERIMENTS WITH DEFAULT CPLEX STRATEGIES

Table A.1- 1st Experiment with Default Strategies

Number of Demand Points	Optimal Value	Optimal Point	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
25	43.86	[100, 0]	1046	30	0.72
50	30.79	[83.99, 50.87]	2607	63	1.85
100	22.42	[31.17, 100]	7291	143	6.37
500	10.15	[0, 40.79]	75475	627	115.43
1000	8.97	[0, 39.61]	187277	1118	462.48
2000	6.09	[0,36.73]	416041	1360	1686.53
3000	4.97	[0, 37.85]	806176	2731	4232.34

Table A.2- 2nd Experiment with Default Strategies

Number of Demand Points	Optimal Value	Optimal Point	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
25	38.82	[0, 100]	946	31	0.65
50	31.1	[0, 38.17]	3172	95	1.7
100	24.93	[96.21, 0]	6071	111	4.77
500	9.87	[63.26, 100]	106303	959	153.42
1000	6.78	[72.77, 6.66]	198077	1184	514.21
2000	5.35	[56.13, 100]	554268	1730	2109.2
3000	5.35	[56.13, 100]	1118452	2712	5359.02

Table A.3- 3rd Experiment with Default Strategies

Number of Demand Points	Optimal Value	Optimal Point	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
25	45.98	[0, 100]	1346	56	1.17
50	27.85	[100, 100]	3004	83	1.83
100	23.26	[45.39, 0]	4747	65	4.11
500	10.97	[15.93, 100]	91792	483	121.36
1000	7.73	[48.45, 100]	158110	605	478.06
2000	6.07	[25.22, 0]	499928	1823	1905.42
3000	5.11	[0,79.95]	1020872	2562	5171

Table A.4- 4th Experiment with Default Strategies

Number of Demand Points	Optimal Value	Optimal Point	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
25	39.65	[52.99, 100]	1113	36	0.69
50	33.81	[64.77, 100]	2647	81	1.69
100	18.58	[0, 0]	9003	231	6
500	12.57	[11.43, 0]	84007	588	115.53
1000	7.28	[85.93, 100]	291268	1645	619.53
2000	5.85	[60.51, 100]	393027	1992	1667.52
3000	5.21	[59.85, 100]	1089007	2836	5247.45

Table A.5- 5th Experiment with Default Strategies

Number of Demand Points	Optimal Value	Optimal Point	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
25	47.52	[0, 79.88]	1571	57	0.7
50	35.71	[0, 91.69]	3495	87	1.69
100	21.95	[0, 62.23]	7726	147	5.23
500	9.93	[100, 7.95]	84261	617	125.13
1000	7.17	[7.61, 100]	209164	1387	483.47
2000	5.29	[68.39, 0]	467541	2365	1862.75
3000	4.57	[30.89, 32,38]	823076	3554	4509.87

Table A.6- 6th Experiment with Default Strategies

Number of Demand Points	Optimal Value	Optimal Point	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
25	36.74	[0, 17.08]	1188	42	0.81
50	26.96	[100, 47.85]	4703	29	1.92
100	21.02	[0, 30.97]	5235	111	4.27
500	11.78	[57.1, 0]	92882	606	124.59
1000	7.15	[0, 100]	198006	904	511.13
2000	7.11	[100,100]	488145	1194	1733.78
3000	4.69	[100, 9.89]	719409	2382	3736.77

Table A.7- 7th Experiment with Default Strategies

Number of Demand Points	Optimal Value	Optimal Point	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
25	41.31	[100,100]	1057	35	1.06
50	32.36	[0, 0]	1975	46	1.77
100	23.21	[100, 75.13]	6484	109	4.85
500	13.75	[67.63, 0]	101886	575	135.41
1000	8.44	[74.95, 75.6]	132708	676	370.99
2000	5.25	[25.57, 0]	305185	972	1376.5
3000	4.23	[89.11,0]	605421	1529	3654.78

Table A.8- 8th Experiment with Default Strategies

Number of Demand Points	Optimal Value	Optimal Point	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
25	30.03	[83.62, 77.95]	1525	69	0.67
50	28.43	[0, 100]	3253	87	1.75
100	25.88	[2.55, 100]	11012	192	5.36
500	8.97	[51.67, 46.12]	106224	907	146.75
1000	6.95	[0, 100]	180127	1017	422.47
2000	5.79	[72.21, 10]	448723	1620	1761.56
3000	4.47	[0, 97.51]	917776	2019	4645.86

Table A.9- 9th Experiment with Default Strategies

Number of Demand Points	Optimal Value	Optimal Point	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
25	41.05	[0, 40.39]	1477	60	0.66
50	34.83	[100, 98.82]	3056	72	1.73
100	28.35	[2.02, 100]	6327	105	4.34
500	11.15	[33.49, 0]	121599	828	154.75
1000	8.43	[0, 80.43]	216442	1080	570.03
2000	4.96	[0, 47.78]	505357	2243	1819.56
3000	4.23	[90.4, 5.07]	701070	2879	3852.61

Table A.10- 10th Experiment0 with Default Strategies

Number of Demand Points	Optimal Value	Optimal Point	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
25	37.93	[100, 43.22]	911	29	0.59
50	24.87	[100, 47.31]	3797	134	1.83
100	22.74	[80.77, 100]	7525	169	5.05
500	9.07	[76.91, 39.33]	121672	733	137.45
1000	8	[63.7, 0]	226371	1509	513.73
2000	5.57	[0, 16.51]	510926	1982	1873.89
3000	4.92	[100, 44.81]	821763	2028	4114.89

APPENDIX B

COMPARISON OF CPLEX STRATEGIES

Table B.1- 1st Experiment-100 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
100	22.42	[31.17,100]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	7291	143	6.37
VARSEL=-1	9281	145	2
VARSEL=1	5823	93	3
VARSEL=3	4885	65	5
CUTS=NO	8739	180	4
DPRIIND=1	8102	130	4
DPRIIND=2	8204	166	8
MIPEMPHASIS=1	5521	112	2
BRDIR=-1	7787	125	5
BRDIR=1	6922	101	5
NODESEL=0	6961	160	4
NODESEL=2	6394	140	5
NODESEL=3	6450	149	6
Priority on integers	7291	143	6
Combined#1	3970	93	2
Combined#2	4172	139	2
Combined#3	4457	108	2
Combined#4	4053	103	1
Combined#5	4457	108	1
Combined#6	4053	103	1
Combined#7	4666	104	1

Table B.2- 1st Experiment-500 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
500	10.15	[0,40.79]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	75475	627	115.43
VARSEL=-1	156117	1076	137

Table B.2 – (cont'd)

VARSEL=1	122040	1090	108
VARSEL=3	54290	511	156
CUTS=NO	63260	635	66
DPRIIND=1	104411	951	86
DPRIIND=2	92841	658	240
MIPEMPHASIS=1	52337	602	47
BRDIR=-1	133790	658	167
BRDIR=1	155865	1036	174
NODESEL=0	57361	607	100
NODESEL=2	109535	933	146
NODESEL=3	49951	353	92
Priority on integers	75475	627	114
Combined#1	38600	453	29
Combined#2	40815	404	23
Combined#3	42020	456	13
Combined#4	40437	507	21
Combined#5	42020	456	16
Combined#6	40208	501	16
Combined#7	41569	454	21

Table B.3- 1st Experiment -1000 Comparison of Strategies

Number of Demand Points	**Optimal Value**	**Optimal Point**	
1000	8.97	[0,39.61]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	187277	1118	462.48
VARSEL=-1	267320	1135	425
VARSEL=1	244575	1567	377
VARSEL=3	117330	757	517
CUTS=NO	175132	690	277
DPRIIND=1	259919	1293	280
DPRIIND=2	193577	938	1219
MIPEMPHASIS=1	96521	573	149
BRDIR=-1	204682	1182	483
BRDIR=1	175568	905	436
NODESEL=0	85163	612	311
NODESEL=2	153830	1013	408
NODESEL=3	91937	616	324
Priority on integers	187277	1118	458
Combined#1	86668	694	107
Combined#2	83731	612	43
Combined#3	64133	503	30

Table B.3 – (cnt'd)

Combined#4	58935	475	30
Combined#5	64133	503	30
Combined#6	58935	475	28
Combined#7	71242	548	33

Table B.4- 1st Experiment-3000 Comparison of Strategies

Number of Demand Points		**Optimal Value**	**Optimal Point**
3000		4.97	[0,37.85]
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	806176	2731	4232.34
VARSEL=-1	1112735	4840	4360
VARSEL=1	718712	3093	2567
VARSEL=3	747675	2140	6400
CUTS=NO	697444	1074	2981
DPRIIND=1	878356	2392	1987
DPRIIND=2	688723	1920	8980
MIPEMPHASIS=1	429569	1549	1714
BRDIR=-1	822555	2466	4313
BRDIR=1	863045	2008	4555
NODESEL=0	457939	1832	3040
NODESEL=2	762376	2491	4475
NODESEL=3	435744	1626	3311
Priority on integers	823373	3480	3804
Combined#1	382685	1538	1422
Combined#2	401996	1478	364
Combined#3	308765	1377	303
Combined#4	320310	1373	309
Combined#5	308765	1377	301
Combined#6	320310	1373	304
Combined#7	336457	1278	312

Table B.5- 2nd Experiment-100 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
100	24.93	[96.21,0]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	6071	111	4.77
VARSEL=-1	6194	123	2
VARSEL=1	11148	177	4
VARSEL=3	4814	70	5
CUTS=NO	6569	134	3
DPRIIND=1	10242	218	5
DPRIIND=2	6217	130	8
MIPEMPHASIS=1	3825	59	1
BRDIR=-1	6087	95	4
BRDIR=1	4827	63	4
NODESEL=0	7531	208	6
NODESEL=2	5440	95	5
NODESEL=3	5570	114	4
Priority on integers	5832	120	3
Combined#1	3186	95	1
Combined#2	3929	110	1
Combined#3	4175	73	1
Combined#4	4175	73	1
Combined#5	4175	73	1
Combined#6	4175	73	1
Combined#7	4060	68	1

Table B.6- 2nd Experiment -500 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
500	9.87	[63.26,100]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	106303	959	153.42
VARSEL=-1	96939	710	93
VARSEL=1	89110	859	91

Table B.6-(cnt'd)

VARSEL=3	59929	576	175
CUTS=NO	55527	517	61
DPRIIND=1	107527	1036	89
DPRIIND=2	99104	1029	259
MIPEMPHASIS=1	45733	515	41
BRDIR=-1	78465	497	117
BRDIR=1	158789	1227	192
NODESEL=0	56188	529	106
NODESEL=2	124462	1099	168
NODESEL=3	53775	459	103
Priority on integers	115924	1015	137
Combined#1	39140	433	25
Combined#2	42140	579	15
Combined#3	39600	452	12
Combined#4	40264	450	12
Combined#5	39600	452	13
Combined#6	40255	449	12
Combined#7	38520	435	12

Table B.7- 2nd Experiment -1000 Comparison of Strategies

Number of Demand Points		**Optimal Value**	**Optimal Point**
1000		6.78	[72.77,6.66]
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	198077	1184	514.21
VARSEL=-1	404632	2430	716
VARSEL=1	246918	2639	447
VARSEL=3	171486	1689	953
CUTS=NO	146713	791	265
DPRIIND=1	228865	1700	280
DPRIIND=2	142246	1141	955
MIPEMPHASIS=1	132002	1044	214
BRDIR=-1	252290	1546	607
BRDIR=1	212784	1269	541
NODESEL=0	138740	1184	441
NODESEL=2	203827	1335	538
NODESEL=3	145399	1072	450
Priority on integers	263167	1867	552
Combined#1	112334	959	140
Combined#2	106941	890	56

Table B.7 - (cnt'd)

Combined#3	100265	889	47
Combined#4	99396	889	48
Combined#5	100265	889	47
Combined#6	99396	889	47
Combined#7	108570	925	48

Table B.8- 2nd Experiment -3000 Comparison of Strategies

Number of Demand Points	**Optimal Value**	**Optimal Point**	
3000	5.35	[56.13,100]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	1118452	2712	5359.02
VARSEL=-1	1208076	3616	4856
VARSEL=1	805244	3586	3122
VARSEL=3	448941	1696	3374
CUTS=NO	573546	1522	2687
DPRIIND=1	600650	1981	1707
DPRIIND=2	599090	1652	10220
MIPEMPHASIS=1	446621	1932	1836
BRDIR=-1	972364	2396	5100
BRDIR=1	995695	2330	5107
NODESEL=0	462042	1681	3062
NODESEL=2	846755	2494	4525
NODESEL=3	377253	961	2752
Priority on integers	800487	2547	3621
Combined#1	405294	1642	1447
Combined#2	606893	1038	370
Combined#3	281772	1236	249
Combined#4	550147	2407	433
Combined#5	281772	1236	249
Combined#6	293418	1252	248
Combined#7	305388	1252	289

Table B.9- 3rd Experiment-100 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
100	23.26	[45.39,0]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	4747	65	4.11
VARSEL=-1	3370	74	1
VARSEL=1	8652	199	3
VARSEL=3	4607	72	5
CUTS=NO	2809	64	2
DPRIIND=1	6327	105	5
DPRIIND=2	9272	206	9
MIPEMPHASIS=1	5322	140	3
BRDIR=-1	8912	184	6
BRDIR=1	6400	123	5
NODESEL=0	4962	90	3
NODESEL=2	4727	64	4
NODESEL=3	4604	81	4
Priority on integers	4315	59	3
Combined#1	4776	149	1
Combined#2	3858	141	1
Combined#3	3719	95	1
Combined#4	3723	96	1
Combined#5	3719	95	1
Combined#6	3723	96	1
Combined#7	3829	102	1

Table B.10- 3rd Experiment-500 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
500	10.97	[15.93,100]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	91792	483	121.36
VARSEL=-1	171218	998	145
VARSEL=1	110625	1072	101
VARSEL=3	56524	612	177
CUTS=NO	41512	469	55
DPRIIND=1	95083	796	78

Table B.10 – (cnt'd)

DPRIIND=2	54110	559	216
MIPEMPHASIS=1	45958	424	40
BRDIR=-1	161451	726	166
BRDIR=1	211384	1159	221
NODESEL=0	40533	259	80
NODESEL=2	91300	513	121
NODESEL=3	39164	271	80
Priority on integers	76573	533	92
Combined#1	37970	405	23
Combined#2	34286	369	11
Combined#3	35776	408	11
Combined#4	36191	410	11
Combined#5	35776	408	11
Combined#6	36038	401	11
Combined#7	35793	336	10

Table B.11- 3rd Experiment-1000 Comparison of Strategies

Number of Demand Points		**Optimal Value**	**Optimal Point**
1000		7.73	[48.45,100]
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	158110	605	478.06
VARSEL=-1	202737	1016	338
VARSEL=1	260465	1557	399
VARSEL=3	126762	1128	674
CUTS=NO	92293	583	211
DPRIIND=1	183662	1051	277
DPRIIND=2	265104	1734	1269
MIPEMPHASIS=1	101343	705	160
BRDIR=-1	245750	847	548
BRDIR=1	226606	1093	581
NODESEL=0	129142	645	431
NODESEL=2	107568	414	396
NODESEL=3	122289	537	420
Priority on integers	219136	785	452
Combined#1	105295	771	128
Combined#2	100309	673	46
Combined#3	93875	701	43
Combined#4	93282	688	43

Table B.11 – (cont'd)

Combined#5	93875	701	42
Combined#6	93282	688	43
Combined#7	81022	641	40

Table B.12- 3rd Experiment-3000 Comparison of Strategies

Number of Demand Points		**Optimal Value**	**Optimal Point**
3000		5.11	[0,79.95]
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	1020872	2562	5171
VARSEL=-1	479961	1415	2249
VARSEL=1	548214	3201	2333
VARSEL=3	572227	2297	4706
CUTS=NO	778458	1343	3303
DPRIIND=1	804906	1807	1675
DPRIIND=2	371157	473	8506
MIPEMPHASIS=1	470687	1802	1866
BRDIR=-1	1077155	3199	5225
BRDIR=1	793323	2082	4317
NODESEL=0	520572	1968	3399
NODESEL=2	789835	1731	4245
NODESEL=3	401042	1081	2920
Priority on integers	863775	2417	3932
Combined#1	336827	1342	1231
Combined#2	420556	1590	372
Combined#3	315041	1372	262
Combined#4	298969	1378	258
Combined#5	315041	1372	262
Combined#6	298969	1378	255
Combined#7	396435	1541	344

Table B.13- 4th Experiment-100 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
100	18.58	[0,0]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	9003	231	6
VARSEL=-1	6549	160	3
VARSEL=1	10160	276	3
VARSEL=3	7244	144	8
CUTS=NO	8763	174	4
DPRIIND=1	9449	258	5
DPRIIND=2	7784	199	7
MIPEMPHASIS=1	6020	160	3
BRDIR=-1	8018	147	5
BRDIR=1	7763	171	6
NODESEL=0	7211	207	6
NODESEL=2	8380	201	5
NODESEL=3	7345	189	5
Priority on integers	6387	143	4
Combined#1	5421	170	1
Combined#2	4993	182	1
Combined#3	5267	142	1
Combined#4	5267	142	1
Combined#5	5267	142	1
Combined#6	5267	142	1
Combined#7	5665	149	1

Table B.14- 4th Experiment-500 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
500	12.57	[11.43,0]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	84007	588	115.53
VARSEL=-1	66952	508	56
VARSEL=1	85135	636	64
VARSEL=3	45666	348	109
CUTS=NO	71885	526	69
DPRIIND=1	105622	797	79
DPRIIND=2	124596	761	271
MIPEMPHASIS=1	40892	335	35

Table B.14 – (cont'd)

BRDIR=-1	66745	341	89
BRDIR=1	67568	366	98
NODESEL=0	45023	382	85
NODESEL=2	78595	560	109
NODESEL=3	36423	293	80
Priority on integers	79978	494	86
Combined#1	36061	395	22
Combined#2	33840	398	11
Combined#3	34619	374	11
Combined#4	30976	345	9
Combined#5	34619	374	10
Combined#6	30976	345	9
Combined#7	35800	381	11

Table B.15- 4th Experiment-1000 Comparison of Strategies

Number of Demand Points	**Optimal Value**	**Optimal Point**	
1000	7.28	[85.93,100]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	291268	1645	619.53
VARSEL=-1	386362	1999	623
VARSEL=1	228866	1775	369
VARSEL=3	152176	1133	668
CUTS=NO	139760	660	251
DPRIIND=1	258017	1646	308
DPRIIND=2	213449	1456	1403
MIPEMPHASIS=1	144587	978	218
BRDIR=-1	240683	1435	565
BRDIR=1	224582	1549	558
NODESEL=0	137743	1086	419
NODESEL=2	237405	1397	553
NODESEL=3	133360	1030	417
Priority on integers	291751	1781	533
Combined#1	95994	831	120
Combined#2	105245	855	52
Combined#3	92917	657	44
Combined#4	86772	654	42
Combined#5	92917	657	43
Combined#6	87428	652	42
Combined#7	103542	739	47

Table B.16- 4th Experiment-3000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
3000	5.21	[59.85,100]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	1089007	2836	5247.45
VARSEL=-1	550383	2031	2489
VARSEL=1	801274	2801	2867
VARSEL=3	693018	2430	5686
CUTS=NO	587479	1047	2864
DPRIIND=1	786409	2161	1921
DPRIIND=2	838475	2361	8597
MIPEMPHASIS=1	558363	1885	2243
BRDIR=-1	953021	2848	4772
BRDIR=1	1054488	2527	5255
NODESEL=0	525158	2476	3468
NODESEL=2	943567	2523	4699
NODESEL=3	463620	1911	3188
Priority on integers	915214	2665	4116
Combined#1	384665	1402	1360
Combined#2	400681	1142	313
Combined#3	350019	1440	338
Combined#4	365099	1544	321
Combined#5	350019	1440	338
Combined#6	365099	1544	317
Combined#7	344885	1518	314

Table B.17- 5th Experiment-100 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
100	21.95	[0.62.23]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	7726	147	5.23
VARSEL=-1	6853	143	2
VARSEL=1	7245	163	3
VARSEL=3	5957	111	7
CUTS=NO	9791	199	3
DPRIIND=1	9768	154	5
DPRIIND=2	9928	201	8
MIPEMPHASIS=1	5314	153	3

Table B.17 – (cont'd)

BRDIR=-1	8007	134	5
BRDIR=1	7338	173	5
NODESEL=0	6061	122	4
NODESEL=2	7628	164	4
NODESEL=3	6466	139	5
Priority on integers	6833	116	4
Combined#1	3557	108	1
Combined#2	3024	68	1
Combined#3	5509	127	1
Combined#4	4641	121	1
Combined#5	5509	127	1
Combined#6	4965	121	1
Combined#7	4988	116	1

Table B.18- 5th Experiment-500 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point
500	9.93	[100,7.95]

Tested Strategy	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
DEFAULT	84261	617	125.13
VARSEL=-1	110569	571	97
VARSEL=1	106225	934	101
VARSEL=3	61575	689	203
CUTS=NO	50639	475	57
DPRIIND=1	93042	720	71
DPRIIND=2	88378	747	241
MIPEMPHASIS=1	50948	536	45
BRDIR=-1	53706	400	97
BRDIR=1	61502	339	105
NODESEL=0	73458	708	117
NODESEL=2	92701	687	130
NODESEL=3	66484	639	111
Priority on integers	82846	475	96
Combined#1	42877	475	33
Combined#2	44136	531	15
Combined#3	38940	450	12
Combined#4	38489	434	12
Combined#5	38940	450	12
Combined#6	38437	434	11
Combined#7	39858	425	12

Table B.19- 5th Experiment-1000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
1000	7.17	[7.61,100]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	209164	1387	483.47
VARSEL=-1	264691	1294	404
VARSEL=1	192588	1855	312
VARSEL=3	149506	1348	706
CUTS=NO	214314	915	300
DPRIIND=1	146345	826	232
DPRIIND=2	119742	456	969
MIPEMPHASIS=1	126347	944	193
BRDIR=-1	153077	954	398
BRDIR=1	236783	1480	520
NODESEL=0	134898	1020	375
NODESEL=2	234869	1615	526
NODESEL=3	135208	1083	380
Priority on integers	264561	1780	492
Combined#1	98489	747	128
Combined#2	109846	867	56
Combined#3	96878	850	51
Combined#4	96780	854	49
Combined#5	96878	850	50
Combined#6	96780	854	51
Combined#7	98761	784	45

Table B.20- 5th Experiment-3000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
3000	4.57	[30.89,32.38]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	823076	3554	4509.87
VARSEL=-1	937404	3390	3821
VARSEL=1	667741	3792	2578
VARSEL=3	658739	2794	5997
CUTS=NO	1008788	1520	3683
DPRIIND=1	551250	1903	1640
DPRIIND=2	832434	2272	9428
MIPEMPHASIS=1	516714	2262	2041
BRDIR=-1	574202	1971	3438

Table B.20 – (cont'd)

BRDIR=1	722727	2192	3973
NODESEL=0	445829	1931	3002
NODESEL=2	650908	2236	3665
NODESEL=3	440400	1785	2954
Priority on integers	881831	3258	3742
Combined#1	414163	2008	1527
Combined#2	467671	2001	493
Combined#3	442659	1730	373
Combined#4	437048	1711	368
Combined#5	442659	1730	374
Combined#6	437048	1711	368
Combined#7	440223	1713	372

Table B.21- 6th Experiment-100 Comparison of Strategies

Number of Demand Points	**Optimal Value**	**Optimal Point**	
100	21.02	[0,30.97]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	5235	111	4.27
VARSEL=-1	6670	155	3
VARSEL=1	9289	189	3
VARSEL=3	6728	116	6
CUTS=NO	4800	119	3
DPRIIND=1	7148	154	4
DPRIIND=2	9607	267	8
MIPEMPHASIS=1	6466	155	2
BRDIR=-1	9009	214	6
BRDIR=1	9723	191	6
NODESEL=0	6057	140	4
NODESEL=2	5268	111	5
NODESEL=3	5473	102	4
Priority on integers	10305	247	5
Combined#1	4330	99	1
Combined#2	5445	133	1
Combined#3	4062	105	1
Combined#4	4062	105	1
Combined#5	4062	105	1
Combined#6	4062	105	1
Combined#7	4046	105	1

Table B.22- 6th Experiment-500 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point
500	11.78	[57.1,0]

Tested Strategy	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
DEFAULT	92882	606	124.59
VARSEL=-1	80857	591	73
VARSEL=1	74822	822	69
VARSEL=3	47571	464	134
CUTS=NO	57909	402	57
DPRIIND=1	126069	774	86
DPRIIND=2	114828	925	273
MIPEMPHASIS=1	40299	404	33
BRDIR=-1	118864	783	146
BRDIR=1	96715	638	137
NODESEL=0	59541	532	96
NODESEL=2	60019	389	98
NODESEL=3	47394	300	87
Priority on integers	73071	407	96
Combined#1	39956	378	24
Combined#2	37378	444	12
Combined#3	27562	261	9
Combined#4	28548	261	8
Combined#5	27562	261	9
Combined#6	28548	261	8
Combined#7	28813	265	9

Table B.23- 6th Experiment-1000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point
1000	7.15	[0,100]

Tested Strategy	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
DEFAULT	198006	904	511.13
VARSEL=-1	383336	1949	661
VARSEL=1	189213	1395	325
VARSEL=3	199861	1289	888
CUTS=NO	167495	617	268
DPRIIND=1	188841	935	270
DPRIIND=2	245902	1794	1169
MIPEMPHASIS=1	124568	908	187

Table B.23 – (cont'd)

BRDIR=-1	247039	1439	608
BRDIR=1	171864	885	470
NODESEL=0	158978	1214	468
NODESEL=2	189608	847	504
NODESEL=3	136678	820	425
Priority on integers	192034	955	492
Combined#1	97426	855	120
Combined#2	136146	745	53
Combined#3	105503	811	50
Combined#4	105140	707	49
Combined#5	105503	811	49
Combined#6	105140	857	49
Combined#7	109488	724	48

Table B.24- 6th Experiment-3000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point
3000	4.69	[100,9.89]

Tested Strategy	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
DEFAULT	719409	2382	3736.77
VARSEL=-1	1024830	3582	4017
VARSEL=1	673663	3383	2402
VARSEL=3	550818	2403	5058
CUTS=NO	669342	1056	2719
DPRIIND=1	977745	2813	2010
DPRIIND=2	523650	1822	8092
MIPEMPHASIS=1	470048	2080	1919
BRDIR=-1	648252	2354	3574
BRDIR=1	955196	3587	4526
NODESEL=0	493388	1941	3031
NODESEL=2	627850	2144	3438
NODESEL=3	438204	1806	2903
Priority on integers	598510	2037	2805
Combined#1	436391	1864	1559
Combined#2	445329	2148	466
Combined#3	363936	1488	314
Combined#4	362953	1493	310
Combined#5	363936	1488	314
Combined#6	362953	1493	309
Combined#7	407014	1541	332

Table B.25- 7th Experiment-100 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
100	23.21	[100,75.13]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	6484	109	4.85
VARSEL=-1	6378	118	3
VARSEL=1	5439	115	3
VARSEL=3	7096	104	10
CUTS=NO	6515	179	4
DPRIIND=1	7523	153	5
DPRIIND=2	6143	129	7
MIPEMPHASIS=1	5401	99	3
BRDIR=-1	9951	169	6
BRDIR=1	10834	203	7
NODESEL=0	4507	81	5
NODESEL=2	6365	97	6
NODESEL=3	4370	74	4
Priority on integers	8823	175	5
Combined#1	3962	113	1
Combined#2	3835	84	1
Combined#3	4024	87	1
Combined#4	4024	87	1
Combined#5	4024	87	1
Combined#6	4024	87	1
Combined#7	4267	55	1

Table B.26- 7th Experiment-500 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
500	13.75	[67.63,0]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	101886	575	135.41
VARSEL=-1	95533	497	94
VARSEL=1	105252	810	107
VARSEL=3	62693	461	185
CUTS=NO	76463	488	68
DPRIIND=1	105223	812	87
DPRIIND=2	88458	568	217
MIPEMPHASIS=1	44600	381	38

Table B.26 – (cont'd)

BRDIR=-1	100436	531	134
BRDIR=1	106199	639	144
NODESEL=0	39258	301	84
NODESEL=2	93925	545	131
NODESEL=3	35257	267	81
Priority on integers	116950	688	128
Combined#1	35134	350	22
Combined#2	33348	338	10
Combined#3	27016	253	8
Combined#4	27016	253	8
Combined#5	27016	253	8
Combined#6	27016	253	8
Combined#7	55020	468	8

Table B.27- 7th Experiment-1000 Comparison of Strategies

Number of Demand Points		Optimal Value	Optimal Point
1000		8.44	[74.95,75.6]
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	132708	676	370.99
VARSEL=-1	440663	2910	710
VARSEL=1	224139	1815	366
VARSEL=3	150968	1000	656
CUTS=NO	94429	465	210
DPRIIND=1	178165	1002	227
DPRIIND=2	224980	1366	1270
MIPEMPHASIS=1	106028	686	165
BRDIR=-1	225522	1210	518
BRDIR=1	253571	1149	549
NODESEL=0	118334	852	368
NODESEL=2	149470	762	403
NODESEL=3	126469	915	387
Priority on integers	152650	680	320
Combined#1	87663	635	108
Combined#2	96556	623	43
Combined#3	80484	532	36
Combined#4	82185	541	36
Combined#5	80484	532	35
Combined#6	82185	541	36
Combined#7	108232	809	36

Table B.28- 7th Experiment-3000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
3000	4.23	[89.11,0]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	605421	1529	3654.78
VARSEL=-1	618286	2744	2677
VARSEL=1	638108	3093	2567
VARSEL=3	606418	2700	5306
CUTS=NO	554050	1315	2785
DPRIIND=1	567144	1464	1775
DPRIIND=2	622790	1242	8854
MIPEMPHASIS=1	504043	2346	2093
BRDIR=-1	1011844	1660	5057
BRDIR=1	731756	1738	4255
NODESEL=0	486496	1558	3177
NODESEL=2	678487	1663	3852
NODESEL=3	488243	1302	3191
Priority on integers	699847	1628	3335
Combined#1	482602	2334	1815
Combined#2	455751	2168	521
Combined#3	584239	2105	421
Combined#4	586982	2303	438
Combined#5	584239	2105	420
Combined#6	561610	2200	421
Combined#7	472860	1971	411

Table B.29- 8th Experiment-100 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
100	25.88	[2.55,100]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	11012	192	5.36
VARSEL=-1	7056	158	3
VARSEL=1	8929	145	3
VARSEL=3	4905	60	4
CUTS=NO	3588	87	3
DPRIIND=1	6575	78	4
DPRIIND=2	4831	94	7
MIPEMPHASIS=1	4755	83	2

Table B.29 – (cont'd)

BRDIR=-1	6839	107	4
BRDIR=1	4994	82	4
NODESEL=0	4833	72	4
NODESEL=2	8167	156	5
NODESEL=3	5378	81	4
Priority on integers	7486	119	4
Combined#1	4215	96	1
Combined#2	4013	119	1
Combined#3	4780	99	1
Combined#4	4786	99	1
Combined#5	4780	99	1
Combined#6	4786	99	1
Combined#7	4221	68	1

Table B.30- 8th Experiment-500 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point
500	8.97	[51.67,46.12]

Tested Strategy	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
DEFAULT	106224	907	146.75
VARSEL=-1	83469	479	82
VARSEL=1	102299	1041	110
VARSEL=3	69340	731	224
CUTS=NO	48778	541	59
DPRIIND=1	83083	685	78
DPRIIND=2	118400	935	265
MIPEMPHASIS=1	62899	750	57
BRDIR=-1	135053	1003	168
BRDIR=1	83361	535	123
NODESEL=0	62541	486	106
NODESEL=2	118189	893	158
NODESEL=3	53766	425	99
Priority on integers	129928	1221	150
Combined#1	50352	664	33
Combined#2	46415	571	14
Combined#3	47838	504	14
Combined#4	48085	504	14
Combined#5	47838	504	14
Combined#6	48085	504	14
Combined#7	35144	352	15

Table B.31- 8th Experiment-1000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point
1000	6.95	[0,100]

Tested Strategy	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
DEFAULT	180127	1017	422.47
VARSEL=-1	293246	1300	462
VARSEL=1	167265	1460	285
VARSEL=3	134983	1159	640
CUTS=NO	171784	764	276
DPRIIND=1	292798	1923	347
DPRIIND=2	229711	1592	1303
MIPEMPHASIS=1	133904	979	205
BRDIR=-1	21524	1250	489
BRDIR=1	196747	1260	465
NODESEL=0	138327	1226	395
NODESEL=2	175072	1060	433
NODESEL=3	117659	849	359
Priority on integers	257249	1538	462
Combined#1	103505	823	125
Combined#2	102558	807	52
Combined#3	95118	741	44
Combined#4	98198	759	45
Combined#5	95118	741	44
Combined#6	96617	748	44
Combined#7	78611	543	48

Table B.32- 8th Experiment-3000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point
3000	4.47	[[0,97.51]

Tested Strategy	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
DEFAULT	917776	2019	4645.86
VARSEL=-1	694802	2530	2932
VARSEL=1	804768	3654	2700
VARSEL=3	734884	2965	6284
CUTS=NO	690337	1203	2992
DPRIIND=1	566876	1751	1624
DPRIIND=2	525670	1878	7701
MIPEMPHASIS=1	479804	2288	1929

Table B.32 – (cont'd)

BRDIR=-1	1710285	2570	7361
BRDIR=1	1242520	2506	5644
NODESEL=0	427452	1445	2916
NODESEL=2	978318	2241	4759
NODESEL=3	468076	1417	3048
Priority on integers	745948	2196	3453
Combined#1	465160	2072	1597
Combined#2	475099	1962	510
Combined#3	437131	1766	378
Combined#4	464522	1960	406
Combined#5	437131	1766	378
Combined#6	464522	1960	406
Combined#7	545113	2083	425

Table B.33- 9th Experiment-100 Comparison of Strategies

Number of Demand Points		**Optimal Value**	**Optimal Point**
100		28.35	[2.02,100]
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	6327	105	4.34
VARSEL=-1	8049	119	3
VARSEL=1	10841	192	4
VARSEL=3	3811	45	4
CUTS=NO	5354	153	3
DPRIIND=1	5274	68	4
DPRIIND=2	5209	81	9
MIPEMPHASIS=1	3534	54	2
BRDIR=-1	12918	212	7
BRDIR=1	5474	84	5
NODESEL=0	3834	67	4
NODESEL=2	4582	79	4
NODESEL=3	3443	49	4
Priority on integers	5972	96	3
Combined#1	2464	54	1
Combined#2	3186	99	1
Combined#3	4241	71	1
Combined#4	3624	66	1
Combined#5	4241	71	1
Combined#6	3624	66	1
Combined#7	4221	68	1

Table B.34- 9th Experiment-500 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
500	11.15	[33.49,0]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	121599	828	154.75
VARSEL=-1	139324	654	122
VARSEL=1	103357	923	102
VARSEL=3	45432	338	105
CUTS=NO	55780	534	65
DPRIIND=1	85171	618	71
DPRIIND=2	64797	534	206
MIPEMPHASIS=1	48473	457	43
BRDIR=-1	48463	268	89
BRDIR=1	52140	258	96
NODESEL=0	51015	379	95
NODESEL=2	102027	723	139
NODESEL=3	48482	335	95
Priority on integers	95469	636	110
Combined#1	48497	631	31
Combined#2	35143	396	11
Combined#3	34695	384	11
Combined#4	35751	385	11
Combined#5	34695	384	11
Combined#6	35751	385	11
Combined#7	35144	352	10

Table B.35- 9th Experiment-1000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
1000	8.43	[0,80.43]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	216442	1080	570.03
VARSEL=-1	370083	1512	676
VARSEL=1	246001	2103	470
VARSEL=3	145089	1260	818
CUTS=NO	90074	669	219
DPRIIND=1	158696	729	222
DPRIIND=2	234217	1264	1185
MIPEMPHASIS=1	120099	832	187

Table B.35 – (cont'd)

BRDIR=-1	231968	1167	601
BRDIR=1	268889	1734	690
NODESEL=0	133082	930	451
NODESEL=2	170218	775	492
NODESEL=3	130433	951	456
Priority on integers	234552	1306	513
Combined#1	81127	524	92
Combined#2	100045	707	49
Combined#3	74941	508	33
Combined#4	79088	564	36
Combined#5	74941	508	33
Combined#6	79087	563	36
Combined#7	78611	543	34

Table B.36- 9th Experiment-3000 Comparison of Strategies

Number of Demand Points	**Optimal Value**	**Optimal Point**
3000	4.23	[90.4,5.07]

Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	701070	2879	3852.61
VARSEL=-1	996041	4903	4015
VARSEL=1	822464	3730	2732
VARSEL=3	794549	2592	5834
CUTS=NO	758531	1422	3335
DPRIIND=1	818770	3283	1875
DPRIIND=2	589924	2738	8320
MIPEMPHASIS=1	509982	2454	2052
BRDIR=-1	733181	2921	3996
BRDIR=1	649511	2714	3604
NODESEL=0	494454	2201	2969
NODESEL=2	701712	2894	3838
NODESEL=3	495425	2230	3040
Priority on integers	693560	2554	3207
Combined#1	497563	2299	1723
Combined#2	494267	2429	520
Combined#3	466490	1855	382
Combined#4	477556	1913	389
Combined#5	466490	1855	383
Combined#6	477556	1913	390
Combined#7	545113	2083	430

Table B.37- 10th Experiment-100 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point
100	22.74	[80.77,100]

Tested Strategy	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
DEFAULT	7525	169	5.05
VARSEL=-1	8719	214	3
VARSEL=1	7051	155	3
VARSEL=3	5267	75	6
CUTS=NO	5240	150	3
DPRIIND=1	4863	77	4
DPRIIND=2	6090	180	8
MIPEMPHASIS=1	6528	155	3
BRDIR=-1	8365	158	5
BRDIR=1	7250	125	5
NODESEL=0	5453	134	5
NODESEL=2	5972	128	4
NODESEL=3	5838	111	5
Priority on integers	7173	133	4
Combined#1	3989	100	1
Combined#2	4999	145	1
Combined#3	4627	112	1
Combined#4	4627	112	1
Combined#5	4627	112	1
Combined#6	4627	112	1
Combined#7	6509	143	1

Table B.38- 10th Experiment-500 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point
500	9.07	[76.91,39.33]

Tested Strategy	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
DEFAULT	121672	733	137.45
VARSEL=-1	113729	779	103
VARSEL=1	104679	1186	102
VARSEL=3	58398	690	175
CUTS=NO	86371	646	76
DPRIIND=1	101045	1014	93
DPRIIND=2	117158	972	281
MIPEMPHASIS=1	54867	578	49

Table B.38 – (cont'd)

BRDIR=-1	88480	359	114
BRDIR=1	128150	630	148
NODESEL=0	45330	311	82
NODESEL=2	134160	871	153
NODESEL=3	48048	328	84
Priority on integers	67419	442	79
Combined#1	45372	581	30
Combined#2	47844	599	15
Combined#3	44663	513	14
Combined#4	43709	506	14
Combined#5	44663	513	14
Combined#6	43709	506	14
Combined#7	45715	493	14

Table B.39- 10th Experiment-1000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point	
1000	8	[63.7,0]	
Tested Strategy	**Number of Iterations**	**Number of Branch and Bound Nodes**	**CPU Time (sec)**
DEFAULT	226371	1509	513.73
VARSEL=-1	350918	1597	561
VARSEL=1	125571	1039	215
VARSEL=3	142876	1140	684
CUTS=NO	189996	717	278
DPRIIND=1	164757	1111	260
DPRIIND=2	208834	1439	1057
MIPEMPHASIS=1	111025	641	165
BRDIR=-1	200092	1248	482
BRDIR=1	122187	815	368
NODESEL=0	124897	1086	382
NODESEL=2	205330	1382	498
NODESEL=3	115673	801	360
Priority on integers	190851	1290	389
Combined#1	98248	821	121
Combined#2	88801	574	41
Combined#3	75628	519	35
Combined#4	74030	523	34
Combined#5	75628	519	35
Combined#6	74030	523	34
Combined#7	79931	578	38

Table B.40- 10th Experiment-3000 Comparison of Strategies

Number of Demand Points	Optimal Value	Optimal Point
3000	4.92	[100,44.81]

Tested Strategy	Number of Iterations	Number of Branch and Bound Nodes	CPU Time (sec)
DEFAULT	821763	2028	4114.89
VARSEL=-1	807996	2575	3162
VARSEL=1	907164	4438	3220
VARSEL=3	502216	2106	4289
CUTS=NO	735069	1574	3294
DPRIIND=1	793511	2810	1917
DPRIIND=2	586691	1220	9339
MIPEMPHASIS=1	419439	1914	1675
BRDIR=-1	676864	1730	3518
BRDIR=1	570755	1904	3277
NODESEL=0	446224	1826	2992
NODESEL=2	642487	1531	3483
NODESEL=3	470131	1424	2954
Priority on integers	947566	2299	3934
Combined#1	416346	1785	1501
Combined#2	359333	139	366
Combined#3	525904	2065	404
Combined#4	389380	1586	319
Combined#5	525904	2065	403
Combined#6	389380	1586	318
Combined#7	572603	2299	432

APPENDIX C

EFFECT OF BOUNDING

Table C.1- 1st Experiment - Effect of Bounds

	Number of Branch and Bound Iterations			Number of Branch and Bound Nodes			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	701	509	415	27	17	16	0.56	0.42	0.36
50	2118	1499	1299	81	59	27	0.64	0.61	0.39
100	4457	3852	2671	108	89	79	2.12	0.91	0.77
500	42020	35853	31925	456	440	427	13.52	11.89	10.27
1000	64133	55271	51195	503	455	451	30.38	24.66	23.69
2000	181165	177270	166841	1033	1072	1245	135.45	123.75	118.27
3000	308765	288805	207805	1377	1368	1169	301.75	272.95	194.09
4000	600499	544307	307800	1854	1683	1370	585.08	514.92	323.88
5000	742371	699032	546353	1907	1893	2165	761.67	664.33	633.67

Table C.2- 2nd Experiment - Effect of Bounds

	Number of Branch and Bound Iterations			Number of Branch and Bound Nodes			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	795	680	633	31	24	28	0.48	0.42	0.41
50	1623	1471	1077	45	41	34	0.61	0.6	0.55
100	4175	3487	2919	73	67	57	1.00	0.88	0.73
500	39600	37691	39842	452	432	458	15.75	11.64	11.05
1000	100265	90709	90660	889	846	941	43.00	43.00	40.00
2000	232645	226185	184100	1149	1120	1239	160.09	146.44	143.22
3000	281772	304493	206652	1236	1419	1172	245.01	245.27	182.49
4000	546301	523110	528800	1991	1957	2718	517.53	479.25	602.95
5000	828790	807333	589136	2211	2197	2263	962.12	880.55	720.56

Table C.3- 3rd Experiment - Effect of Bounds

	Number of Branch and Bound Iterations			Number of Branch and Bound Nodes			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	568	448	416	22	18	19	0.25	0.23	0.22
50	2193	1772	939	61	58	29	0.39	0.38	0.3
100	3719	3141	3475	95	84	45	0.94	0.86	0.75
500	35776	30656	25158	408	379	348	11.27	10.23	8.42
1000	93875	79835	59135	701	651	540	41.94	36.69	29.62
2000	199889	188468	155102	1026	1037	1135	147.5	133.05	114.44
3000	315041	283190	228168	1372	1355	1172	257.61	233.88	202.5
4000	434621	395946	286060	1418	1401	1351	414.27	369.03	296.97
5000	467239	428475	342208	1205	1189	1183	577.19	510.61	416.34

Table C.4- 4th Experiment - Effect of Bounds

	Number of Branch and Bound Iterations			Number of Branch and Bound Nodes			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	606	574	440	22	22	13	0.89	0.66	0.64
50	1898	1540	1011	57	48	27	0.81	0.75	0.69
100	5267	4429	5138	142	132	95	1.36	1.28	1.17
500	34619	26673	24948	374	323	227	10.8	8.75	7.34
1000	92912	82451	68158	657	644	723	42.95	39.03	34.23
2000	220889	164093	165673	1243	1002	1246	153.00	116.00	125.00
3000	350019	313307	248570	1440	1386	1241	335.8	286.84	239.61
4000	564696	554619	507112	2039	2033	2430	608.42	585.39	526.27
5000	671027	649963	594302	2200	2194	2196	920.02	836.51	693.2

Table C.5- 5th Experiment - Effect of Bounds

	Number of Branch and Bound Iterations			Number of Branch and Bound Nodes			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	708	556	397	29	22	20	0.66	0.59	0.47
50	1494	806	811	60	42	16	0.75	0.59	0.52
100	5509	3800	4184	127	97	85	2.12	1.00	1.89
500	38940	36866	27554	450	420	355	11.89	11.02	8.75
1000	96878	92920	79429	850	851	788	48.86	46.97	42.73
2000	266227	248661	216227	1391	1374	1603	168.47	160.55	159.17
3000	442659	425637	354808	1730	1717	1854	366	351.08	296.7
4000	723870	696064	550021	2270	2258	2495	656.39	618.56	538.58
5000	559871	559087	436930	1737	1733	1665	748.67	698.38	506.47

Table C.6- 6th Experiment - Effect of Bounds

	Number of Branch and Bound Iterations			Number of Branch and Bound Nodes			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	906	660	733	48	36	23	0.45	0.44	0.41
50	2122	1695	1464	67	51	38	0.56	0.47	0.38
100	4062	3519	3263	105	97	73	0.83	0.83	0.83
500	27562	25561	29051	261	255	299	10.05	8.31	7.83
1000	105503	99699	88218	811	797	884	49.16	43.00	42.00
2000	132102	111311	110879	661	641	697	89.27	74.00	80.00
3000	363936	345063	291542	1488	1486	1565	336.72	289.95	253.27
4000	471132	438627	310078	1426	1415	1377	500	435.48	337.56
5000	899503	881742	695743	2423	2413	2552	908.33	848	914.81

Table C.7- 7th Experiment - Effect of Bounds

	Number of Branch and Bound Iterations			Number of Branch and Bound Nodes			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	851	636	456	42	28	14	0.25	0.23	0.23
50	1687	1228	1086	40	30	26	0.33	0.31	0.3
100	4024	3515	3279	87	81	60	0.88	0.88	0.73
500	27016	23139	16487	253	239	219	8.55	7.01	6.55
1000	80484	70916	65956	532	515	617	34.77	31.19	31.18
2000	289741	266585	198794	1384	1351	1418	177.27	160.17	153.17
3000	584239	562626	409514	2105	2083	2228	420.72	393.97	347.06
4000	544223	509273	413621	1971	1946	1867	536.47	494.89	474.8
5000	811979	764144	478705	1711	1698	1808	720.02	662.25	548.36

Table C.8- 8th Experiment - Effect of Bounds

	Number of Branch and Bound Iterations			Number of Branch and Bound Nodes			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	1106	952	901	73	58	62	0.38	0.27	0.25
50	2043	1543	1241	82	66	31	0.59	0.48	0.33
100	4780	3091	2456	99	77	48	0.84	0.7	0.66
500	47838	42192	38686	504	486	548	13.67	12.48	12.19
1000	95118	89996	68842	741	727	707	42.78	41.86	35.59
2000	180478	163003	128342	998	969	961	130.11	114.95	98.47
3000	437131	423276	358008	1766	1744	1900	378.7	348.16	314.83
4000	607375	590206	491804	2165	2119	2232	585.06	548.2	513.22
5000	839269	802166	606279	2450	2421	2250	947.89	855.67	662.99

Table C.9- 9th Experiment - Effect of Bounds

	Number of Branch and Bound Iterations			Number of Branch and Bound Nodes			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	900	544	495	36	24	22	0.28	0.28	0.27
50	1382	1410	758	36	33	19	0.33	0.33	0.28
100	4241	2987	1381	71	49	26	0.73	0.58	0.41
500	34695	29752	27216	384	363	322	10.38	9.41	8.3
1000	74941	65724	61637	508	492	616	33.23	28.56	31.17
2000	262859	242072	226964	1473	1451	1676	186.62	167.09	180.89
3000	466490	447385	341764	1855	1839	1917	388.3	356.8	306.06
4000	556511	509220	405963	1854	1833	1689	495.22	450.36	417.27
5000	903904	895995	921236	2920	2906	3605	1018.9	983.91	957.03

Table C.10- 10th Experiment - Effect of Bounds

	Number of Branch and Bound Iterations			Number of Branch and Bound Nodes			CPU Time (sec)		
Number of Points	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound	Combination 3	With Lower Bound	With Lower and Upper Bound
25	806	650	624	50	40	22	0.52	0.44	0.41
50	1881	1619	1897	83	73	56	0.61	0.56	0.52
100	4627	3941	3117	112	92	50	1.02	0.89	0.72
500	44663	40873	34542	513	495	492	13.95	13.17	11.44
1000	75628	71965	61092	519	523	585	33.67	31.14	28.59
2000	205713	197814	178487	1063	1045	1099	155.41	148.17	121.84
3000	525904	352670	313357	2065	1497	1465	395.06	275.59	267.14
4000	471569	433475	591475	1683	1673	2466	574.75	525.55	559.26
5000	1057582	1023614	790459	2897	2887	2986	1076.3	1016.4	833.28

APPENDIX-D

INTERACTIVE BIG SQUARE SMALL SQUARE (BSSS) ALGORITHM

LCES	: List of Candidate Efficient Squares
LIFV	: List of Incumbent Function Values
LFDP	: List of Filtered Demand Points
LCLP	: List of Candidate Location Points
LCNV	: List of Candidate Nondominated Vectors
$\mathbf{S^0}$	: Initial Square
$\mathbf{a^i}$	: Side Length of Square S^i
$\mathbf{L}(\boldsymbol{x}^i)$	: Maximin Function Value of a Point $\boldsymbol{x}^i$
$\mathbf{W}(\boldsymbol{x}^i)$	: Minsum Function Value of a Point $\boldsymbol{x}^i$
$\mathbf{UB(S^i)}$	: Upper Bound on Optimal Maximin Function Value in Square S^i
$\mathbf{L^*(S^i)}$	: Optimal Maximin Function Value in S^i
$\mathbf{W^*(S^i)}$	: Optimal Minsum Function Value in S^i
$\boldsymbol{R}(\mathbf{S^i})$	: Ideal Objective Vector of Square S^i
$\boldsymbol{Q}(\mathbf{S^i})$	: Nadir Objective Vector of Square S^i
$\boldsymbol{\alpha}$	: Stopping Side Length in Phase 1
$\boldsymbol{\beta}$	: Stopping Side Length in Phase 2
r	: Highest Square Index in *LCES*
N	: Set of Demand Points
$\boldsymbol{b}^j$	: j^{th} Demand Point
$\mathbf{d}(\boldsymbol{x},\boldsymbol{y})$	: Rectilinear Distance Between $\boldsymbol{x}$ and $\boldsymbol{y}$
$\mathbf{d}(\boldsymbol{b}^j,\mathbf{S^i})$	: Smallest Rectilinear Distance Between $\boldsymbol{b}^j$ and Square S^i

1. FINDING CANDIDATE EFFICIENT SQUARES

Phase 1: Pruning with UB(S^i) and W*(S^i)

//Step 1.0 Initialize

- ❖ Find a square approximation of the feasible region S^0
- ❖ Put S^0 into the *LCES*
- ❖ Ask the DM to specify the stopping side length, α
- ❖ Let i = 0

//Step1.1 Branching and Pruning

- ❖ Delete S^i from the *LCES*
- ❖ Pick T feasible points from S^i. Let these points be $\{\boldsymbol{x^1}, \boldsymbol{x^2},..,\boldsymbol{x^T}\}$
- ❖ Evaluate L($\boldsymbol{x^i}$) and W($\boldsymbol{x^i}$) for each of these T points

$$L(\boldsymbol{x^i}) = \left\{ \min_{j=1,\dots,N} (\left|x_1{}^i - b_1{}^j\right| + \left|x_2{}^i - b_2{}^j\right|\right\}$$

$$W(\boldsymbol{x^i}) = \left\{ \sum_{j=1}^{N} (\left|x_1{}^i - b_1{}^j\right| + \left|x_2{}^i - b_2{}^j\right|) \right\} \text{ for i=1,...,T}$$

- ❖ Add these points to the *LIFV* after a dominance check.

Add (L($\boldsymbol{x^i}$), W($\boldsymbol{x^i}$)) to the *LIFV* if and only if there does not exist another objective vector (L($\boldsymbol{x^j}$), W($\boldsymbol{x^j}$)) ∈ *(LIFV)* ∋ L($\boldsymbol{x^j}$) ≥ L($\boldsymbol{x^i}$), W($\boldsymbol{x^j}$) ≤W($\boldsymbol{x^i}$) and (L($\boldsymbol{x^i}$), W($\boldsymbol{x^i}$)) ≠ (L($\boldsymbol{x^j}$), W($\boldsymbol{x^j}$))

If any element of the *LIFV* (L($\boldsymbol{x}^j$), W($\boldsymbol{x}^j$)) is dominated by the newly added objective vector (L($\boldsymbol{x}^i$), W($\boldsymbol{x}^i$)) ∍ L($\boldsymbol{x}^i$) ≥ L($\boldsymbol{x}^j$), W($\boldsymbol{x}^i$) ≤W($\boldsymbol{x}^j$) and (L($\boldsymbol{x}^i$), W($\boldsymbol{x}^i$)) ≠ (L($\boldsymbol{x}^j$), W($\boldsymbol{x}^j$)) delete (L($\boldsymbol{x}^j$), W($\boldsymbol{x}^j$)) from *LIFV*.

- ❖ Compare the *LCES* with the recently generated incumbent value vectors

FOR each square in the LCES
{
FOR each recently generated incumbent value vector $\boldsymbol{x}^j$
IF $UB(S^i) \le L(\boldsymbol{x}^j)$ $\quad \boldsymbol{x}^j \in LIFV$
AND $W^*(S^i) \ge W(\boldsymbol{x}^j)$ $\quad \boldsymbol{x}^j \in LIFV$
THEN delete S^i *from LCES*
}

- ❖ Divide S^i into 4 equal subsquares
- ❖ Number the subsquares from S^{r+1} to S^{r+4}
- ❖ For each recently generated square, solve *UB(Maximin-*l^1*)* with all existing demand points and *(Maximin-*l^1*)* with the demand points located inside the square. Choose the smallest of the solutions as the UB(S^i)
- ❖ In order to find the optimal minsum objective value for each square, solve *(Minsum-*l^1*)* with all existing demand points

- ❖ Compare the recently generated squares with the *LIFV*

For each recently generated square S^i

{

IF $UB(S^i) > L(\boldsymbol{x}^j)$, $\forall\, \boldsymbol{x}^j \in LIFV$

OR $W^*(S^i) < W(\boldsymbol{x}^j)$, $\forall\, \boldsymbol{x}^j \in LIFV$

THEN add S^i *into LCES*

}

- Check the maximum side length

  If $\max_{i=1,\ldots,r}\{a(S^i)\} < \alpha$, $\forall\; S^i \in LCES$

  Then STOP Phase 1. Go to Phase 2.

  Otherwise go to Step 1.2

//Step1.2 Selecting Where to Branch Next

- Select the square with

  $\max_{i=1,\ldots,r}\{UB(S^i)\}$ OR $\min_{i=1,\ldots,r}\{W^*(S^i)\}, \forall\, S^i \in LCES$

- Let i be the index of the selected square, return to Step 1.1.

Phase 2: Pruning with L*(S^i) and W*(S^i)

Finding Optimal Maximin Objective with Filtered Demand Points

//Step 2.0 Initialize

- For all squares in *LCES*

  Extend the sides of S^i into straight lines to cut the plane into 9 regions: N(orth), S(outh), W(est), E(ast), NW, NE, SW, SE, S^i

- Check the demand points $\boldsymbol{b^j}$ in all 9 regions

  $IF\ d(b^j, S^i) \leq UB(S^i)\ for\ \boldsymbol{b^j} \notin S\ OR\ \boldsymbol{b^j} \in S^i\ THEN\ Put\ \boldsymbol{b^j}\ to\ LFDP$

//Step 2.1 Pruning with Optimal Values

- For each square S^i solve *(Maximin-l^1)* with the demand points in the *LFDP*.
- Compare the ideal objective vector of squares in the *LCES* with the *LIFV*

  $IF\ L^*(S^i) \leq L(\boldsymbol{x^j})$

  $AND\ IF\ W^*(S^i) \geq W(\boldsymbol{x^j}),\ \forall\ \boldsymbol{x^j} \in LIFV$

  $THEN\ delete\ S^i\ from\ LCES$

- Ask the DM if she/he wants to divide the region further

  If so, go to Step 2.2.

  Otherwise go to Step 2.3.

//Step 2.2 Further Branching

- Ask the DM to specify the stopping side length, β

For each square S^i in the *LCES*

WHILE ($a(S^i) > \beta,\ \forall\ S^i \in LCES$) DO

- Delete S^i from the *LCES*
- Divide S^i into 4 equal subsquares
- Number the subsquares from S^{r+1} to S^{r+4}
- For each recently generated square, solve *(Maximin-l^1)* and *(Minsum-l^1)* with all existing demand points

- ❖ Compare S^i with *LIFV*

  $IF\ L^*(S^i) > L(\boldsymbol{x}^j)$

  $OR\ W^*(S^i) < W(\boldsymbol{x}^j),\ \forall\ \boldsymbol{x}^j \in LIFV$

  $ADD\ S^i\ to\ LECS$

//Step 2.3 Stopping

- ❖ Combine the remaining squares in regions.

2. SEARCH IN THE CANDIDATE EFFICIENT REGIONS

Phase 3: Interactive Search

- ❖ Ask the DM which procedure she/he wants to use: Exact or approximate

A. Exact Procedure

- ❖ Present each region with its ideal and nadir objective vectors: ***R*(S)**, ***Q*(S)**
- ❖ Ask the DM to choose a region for starting the search.
- ❖ Ask the DM to specify her/his reference point $\boldsymbol{G^0} = (G_1^{\ 0}, G_2^{\ 0})$
- ❖ Ask the DM which objective is important, and how much. Set the initial vector of weights accordingly $\boldsymbol{w^0} = (w_1^{\ 0}, w_2^{\ 0})$
- ❖ Solve the *(ASP)* for the region. Let the solution be ***C***. If the solution is feasible check whether it is dominated by the *LIFV* and *LCLP*.
- ❖ If ***C*** is not dominated by the *LIFV* and *LCLP* then project ***C*** to the other regions by solving *(ASP)* with $\boldsymbol{w^0}$.
- ❖ Check the resulting solutions. If none of them dominates ***C*** then it is proved to be nondominated and add it to the *LCLP*.

- Compare the solutions with *LIFV* and *LCLP*. Put the approximately efficient ones to the *LCNV*.
- If ***C*** is dominated by one of the solutions then add this solution to *LCNV*.
- If ***C*** is dominated either with the *LIFV*, *LCLP* or the resulting solutions then delete it from further consideration.
- Ask the if she/he wants to continue searching the region further. If so, she/he can continue the search from one of the solutions in *LCNV*. Else, he can either select other regions or stop.
- Present the *LCLP* to the DM. If the DM is not satisfied, ask him to determine a new reference point. Otherwise, ask the DM if she/he can select the most preferred solution among the solutions, if so, stop Phase 3, if not go to Step 3.6.

B. Approximate Procedure

//Step 3.0 Starting The Search.

- Present each region with its ideal and nadir objective vectors: ***R(*S*)***, ***Q(*S*)***
- Ask the DM to choose a region

//Step 3.1 Finding a Starting Nondominated Continuous Solution

- Ask the DM to specify her/his reference point $\boldsymbol{G^0} = (G_1^0, G_2^0)$
- Ask the DM which objective is important, and how much. Set the initial vector of weights accordingly $\boldsymbol{w^0} = (w_1^0, w_2^0)$

- Solve the LP relaxation of the *(ASP)* to find a starting continuous solution closest to the reference point. Let the solution be $\boldsymbol{y^0}$=(y_1^0, y_2^0)
- If the DM likes $\boldsymbol{y^0}$=(y_1^0, y_2^0), then set $\boldsymbol{K}$=(K_1, K_2) and go to Step 3.3. Otherwise, go to Step 3.2.

//Step 3.2 Generating Alternative Nondominated Continuous Solutions

Approach 1: Better Perception of the Efficient Frontier with Perturbed Reference Points

- Find the total percent deviation of the reference point from the starting continuous solution (d).

$$d = d_1 + d_2$$
$$d_1 = \left|G_1^0 - y_1^0\right| / G_1^0$$
$$d_2 = \left|G_2^0 - y_2^0\right| / G_2^0$$

- Ask the DM to specify a number of perturbed points in each objective. Let this number be P.
- Find *2P* perturbed reference points as shown below

$$\boldsymbol{G_1^i} = G_1^0 - \left[(d / i)\ G_1^0\right] \ and \ \boldsymbol{G_2^i} = G_2^0 \ for \ i = 1,...,P$$
$$\boldsymbol{G_1^i} = G_1^0 \ and \ \boldsymbol{G_2^i} = G_2^0 + \left[(d /(i-P)\ G_2^0\right] for \ i = (P+1),...,(2P)$$

- Solve the LP relaxation of the *(ASP)* with perturbed reference points $\boldsymbol{G^1..G^{2P}}$ to find additional continuous nondominated solutions. Let the solutions be $\boldsymbol{y^1..y^{2P}}$
- Present the solutions to the DM. Ask her/him if she/he wants to change her/his reference point. If so, return to Step 3.1.Otherwise, ask

him to select one of the continuous solutions $\boldsymbol{y^1..y^{2P}}$.Let the most promising solution be $\boldsymbol{K}$=(K_1, K_2). Go to Step 3.3

Approach 2: Reference Direction Approach

- Ask the DM to specify a reference direction, $\boldsymbol{\Delta d}$ = (Δd_1, Δd_2)
- Ask the DM the number of solutions that she/he wants to see. Let this number be P
- Solve the *(ASPLP) for* p = 1,...,P. Let the solutions be $\boldsymbol{y^1..y^P}$
- Present the solutions to the DM. Ask her/him if she/he wants to change her/his reference point. If so, return to Step 3.1. Otherwise, ask him to select one of the continuous solutions $\boldsymbol{y^1..y^P}$. Let the most promising solution be $\boldsymbol{K}$=(K_1, K_2). Go to Step 3.3

//Step 3.3 Finding Integer Nondominated Solution

- Solve the *(ASP)* to find the closest integer solution to the most preferred continuous solution selected in the previous step $\boldsymbol{K}$=(K_1, K_2). Let the solution be $\boldsymbol{C}$=(C_1, C_2)
- Check $\boldsymbol{C}$. If it is infeasible or dominated by one of the elements of the *LIFV* or the *LCLP*, then delete $\boldsymbol{C}$ from further consideration. Ask the DM if she/he wants to search the region in concern further. If so, set $\boldsymbol{G^0} = (C_1, C_2)$ return to Step 3.1. Otherwise return to Step 3.0. If $\boldsymbol{C}$ is not infeasible or dominated, add it to the *LCLP*. Go to Step 3.4.

//Step 3.4 Stopping the Search

- Present the *LCLP* to the DM. Ask her/him if she/he wants to stop searching. If so, go to Step 3.5. Otherwise go to Step 3.0.

//Step 3.5 Selection among the Discrete Set of Alternatives

- ❖ Present the *LCLP* to the DM. If the DM can select one of the alternatives as the most promising then STOP the algorithm. Otherwise Go to Step 3.6.

//Step3.6 Outranking of the Alternatives with Promethee II

- ❖ Ask the DM to set indifference (q_i), preference (p_i) thresholds for each objective.
- ❖ Rank the alternatives by using Promethee II.