AN INTERACTIVE EVOLUTIONARY ALGORITHM FOR THE MULTIOBJECTIVE RELOCATION PROBLEM WITH PARTIAL COVERAGE

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

BY

BERK ORBAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
OPERATIONAL RESEARCH

APRIL 2011

Approval of the thesis:

AN INTERACTIVE EVOLUTIONARY ALGORITHM FOR THE MULTIOBJECTIVE RELOCATION PROBLEM WITH PARTIAL COVERAGE

submitted by **BERK ORBAY** in partial fulfillment of the requirements for the degree of **Master of Science in Operational Research Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Çağlar Güven
Head of Department, **Operational Research** _____

Assoc. Prof. Dr. Esra Karasakal
Supervisor, **Industrial Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Nur Evin Özdemirel
Industrial Engineering Dept., METU _____

Assoc. Prof. Dr. Esra Karasakal
Industrial Engineering Dept., METU _____

Prof. Dr. Meral Azizoğlu
Industrial Engineering Dept., METU _____

Assoc. Prof. Dr. Canan Sepil
Industrial Engineering Dept., METU _____

Asst. Prof. Dr. Banu Yüksel Özkaya
Industrial Engineering Dept., Hacettepe University _____

Date: 22.04.2011

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: Berk Orbay

Signature:

ABSTRACT

AN INTERACTIVE EVOLUTIONARY ALGORITHM FOR THE MULTIOBJECTIVE RELOCATION PROBLEM WITH PARTIAL COVERAGE

Orbay, Berk

M.S., Department of Industrial Engineering

Supervisor: Assoc. Prof. Esra Karasakal

April 2011, 129 pages

In this study, a bi-objective capacitated facility location problem is presented which includes partial coverage concept and relocation of facility nodes. In partial coverage, a predefined distance between a demand node and a facility node is assumed to be fully covered. After the predefined distance, the service level commences to decay linearly. The problem is designed to consider the existence of already functioning facility nodes. It is allowed to close these existing facilities and open new facilities in potential sites. However, existing facility nodes are strongly favored against new facility nodes. The objectives are the maximization of the weighted total coverage and the minimization of number of facility nodes. A novel interactive multi-objective evolutionary algorithm is proposed to solve this problem, I-TREA. I-TREA is originated from NSGA-II and designed for interactive methods benefiting from quality infeasible solutions. The performance of I-TREA is benchmarked with a modified version of NSGA-II on randomly generated problems with various sizes and utility functions.

Keywords: Interactive Multi-Objective Evolutionary Algorithm, Maximal Covering Location Problem, Partial Coverage, Relocation

ÖZ

KISMİ KAPSAMANIN OLDUĞU ÇOK AMAÇLI YENİDEN YERLEŞTİRME PROBLEMİ İÇİN İNTERAKTİF BİR EVRİMSEL ALGORİTMA

Orbay, Berk

Yüksek Lisans, Endüstri Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Esra Karasakal

Nisan 2011, 129 sayfa

Bu çalışmada, kısmi kapsama ve tesis değiştirme konseptlerinin bulunduğu iki amaçlı kapasite kısıtlı bir yerleşim problemi sunulmuştur. Kısmi kapsamada, bir servis noktası ile talep noktası arasında belli bir uzaklık değeri içinde o talep noktasının tam olarak kapsandığı ve bu değerden sonra kapsamanın azalmaya başladığı varsayılmıştır. Problemde halihazırda servis noktaları olduğu varsayılmış ve bu noktaların kapatılmasına izin verilip potansiyel yerlerde yeni servis noktaları açılmasına izin verilse de halihazırdaki servis noktalarının korunması yeni servis noktalarının açılmasına tercih edilmektedir. Birinci amaç toplam ağırlıklı kapsamayı maksimize etmektir. İkinci amaç ise açılmış servis noktası sayısını minimize etmektir. Bu problem için yeni bir interaktif evrimsel algoritma önerilmiştir, I-TREA. I-TREA, hazırlanışında NSGA-II temel alınmış, olurlu olmayan ama kaliteli sonuçlardan yararlanan, interaktif yöntemlerde kullanmak için tasarlanmış bir çok amaçlı evrimsel algoritmadır. I-TREA'nın performansı NSGA-II'nun modifiye edilmiş bir versiyonu ile karşılaştırılarak rastgele oluşturulmuş, değişik fayda fonksiyonlarına ve büyüklüklere sahip problemler üzerinde test edilmiştir.

Anahtar Kelimeler: Çok Amaçlı Evrimsel Algoritma, Maksimum Kapsama Yerleşim Problemi, Kısmi Kapsama, Yer Değiştirme

To My Family,

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor, Assoc. Prof. Dr. Esra Karasakal for her efforts, guidance, support and constant patience throughout the entire thesis process. It has always been a pleasure to work with her. I would also like to thank her soon-to-be-born son for speeding things up and wish both of them well.

I don't think I can ever pay my debt to my family. Their enormous support during the entire process and at my darkest hours while writing this thesis made this study possible. I would love to write all of them here but I especially want to thank my father Prof. Dr. Atilla Orbay and my mother Prof. Dr. Yeşim Kamile Aktuğlu.

This study again and again proved that dear friends are absolutely necessary to undertake a task like a thesis. I am lucky that I have plenty. I would like to thank İpek Sayın for making the last two years more entertaining and not always about studying. Some friends are too kind to remind you that time is happening through the means of wedding and engagement. I would like to thank Serkan Özpamukçu and Funda Akar, Özlem (Karabulut) and Sinan Karsu, Sinem (Süzen) Günsel, Ayşegül (Demir) and Kerem Demirtaş for being great friends and not leaving me alone in Ankara for the last three years. I would like to thank Doruk Tunaoğlu for being a friend and helping me out with the initiation of my coding. There are so many more to thank: Beril, Jehanzeb, Kemal, Hasan, Yiğit, Erdem, Cansu, Eda, Queen, Deep Purple…

I would also like to thank all of the members of the department for I have learned a lot from them for the last seven years and they are not all about just teaching.

Last, but not the least, I would like to thank Prof. Dr. Osman Coşkunoğlu and Şerife Akuygur Barutçuoğlu. Their support to and interest in my thesis helped more than they think of. Also, I have learned many valuable things from them during my time with them.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Maximal Covering Location Problem (MCLP) is first introduced by Church and ReVelle (1974). The problem's objective is to cover as many demand points as possible by facility nodes within an acceptable distance.

MCLP is a widely studied problem in the literature and it has many real life implementations. To name some of them, Pirkul and Schilling (1988) propose a backup coverage problem for emergency sites which is a variation of MCLP. Farhan and Murray (2008) propose a multi-objective facility location problem for park-and-ride facilities and they use demand coverage as an objective. Harewood (2002) proposes a Bi-Objective Maximum Availability Location Problem for the deployment of ambulances in Barbados where coverage maximization is one of the objectives.

Two of the main assumptions of the classical MCLP are; there are a fixed number of facilities and there is a fixed critical distance where one side of that distance is "fully covered" and the other side is "not covered".

The sudden drop in the coverage value may be valid for some cases, yet it is not realistic for the most. Therefore partial coverage concept is introduced for a smooth decay in the coverage function. With partial coverage, the coverage value begins to decrease according to a function after the critical coverage distance until it reaches another threshold where demand is "not covered" after that distance.

The other assumption is transformed into an objective. In addition, another implicit assumption of "clean slate setting" (i.e. there are no previous installations within the region) is reconsidered by including a distinction of existing facilities from 'new' facilities. Existing facilities are favored in this objective, albeit relocation, opening and closure of facilities are allowed.

The bi-objective Maximal Coverage Location Problem is a capacitated facility location problem. The first objective considers the maximization of the total weighted coverage of demand points, while the second objective is the minimization of number of facilities.

By being a combinatorial problem, obtaining exact solutions as the problem sizes grow gets harder in a reasonable amount of time. A meta-heuristic method will be proposed for fast and reliable results.

The heuristic proposed is an interactive multi-objective evolutionary algorithm (I-TREA) that uses specific functions to enhance its performance and interactive process to focus on parts of the solution space that are of interest to the decision maker and to eliminate undesired parts of the solution space.

I-TREA's performance is tested by benchmarking with exact solutions and a modified version of a distinguished evolutionary algorithm, NSGA-II, for various problems and problem sizes.

The organization of the thesis is as follows. A literature survey is presented to discuss previous studies in the related research areas in Chapter 2. The concepts used in the structure of the thesis are introduced in Chapter 3. The problem definition is discussed in Chapter 4. Then, the proposed algorithm (I-TREA) is explained in detail in Chapter 5. The algorithm is tested and benchmarked with exact solutions and NSGA-II in Chapter 6. Concluding remarks and discussions for further study are given in Chapter 7.

CHAPTER 2

LITERATURE REVIEW

2.1 Location Problems

There are excellent surveys in the literature about facility location problems. To spell some of them; ReVelle et al. (1970) have a survey on location models classified as public and private sector models. Tansel et al. (1983) examine p-center and p-median problems. Brandeau and Chiu (1989) present a comprehensive survey of 50 major location problems up to date. Berman et al. (2010) provide a comprehensive survey on generalized coverage models. Owen and Daskin's (1998) review on strategic facility location gives substantial information on facility relocations.

Alfred Weber's (1909) work is recognized as the beginning of Location Theory with the decision problem of locating a single warehouse with the objective of minimizing the total traveling distance to the demand points. For over a century there are many contributions to this area.

Set Covering Problem (SCP) is introduced by Toregas et al. (1971). Their aim was to cover all demand nodes in the region by minimum number of facilities given a maximum covering distance.

As an alternative to the Set Covering Problem, Church and ReVelle (1974) developed a new model, namely, Maximal Covering Location Problem (MCLP). The model differed from SCP by fixing the number of facilities and removing the

condition to cover all nodes. Also, the objective is changed to maximization of total weighted demand coverage instead of finding the minimum amount of facilities.

2.1.1 Maximum Coverage Location Problem

Maximal Coverage Location Problem is originated from Hakimi's (1964) p-median problem, where a set of p facilities should be picked from a discrete set that leads to the minimum total weighted distance to the demand points. Hakimi's study also covered minimization of the maximum distance from facilities to demand points (p-center).

2.1.2 Capacitated Problems

As Current and Storbeck (1988) state, most of the location problems had an underlying assumption of facilities being uncapacitated. It has been shown that the assumption is not valid for all the cases.

Capacity constraint in Maximal Covering Location Problem is first studied by Chung et al. (1983). Each facility is associated with a capacity on the amount of demand they can serve. With this restriction, assignment of demand points to facilities within coverage distance becomes an imperative.

Pirkul and Schilling (1991) also consider the case where capacities of the facilities are utilized by demand points even if they are not within the covered region by the facility. They add a p-median objective with insignificant effect on the maximal coverage objective to minimize the total distance from uncovered demand points to their assigned facilities.

2.1.3 Backup and Multiple Coverage

Pirkul and Schilling (1988) integrate capacity restrictions and backup coverage in a decision model of siting emergency facilities. The need arises from the settings where high demand occurs and demand cannot be queued. Therefore, backup

coverage is considered for demand points in case the assigned facility is fully utilized and unable to respond.

Daskin and Stern (1981) introduced a multiple coverage approach to Set Covering Problem. While the primary objective is the same as SCP, the secondary objective calculates the number of times the demand nodes are covered beside the first coverage.

Hogan and ReVelle (1986) show that backup coverage can be introduced in coverage models, without serious effect on the performance of primary coverage. Daskin et al. (1988) combine multiple, excess, backup and expected coverage in both SCP and MCLP for emergency services.

Church and Gerard (2003) propose a set covering location model where multi-level coverage is compulsory, which means each demand point should be covered by more than at least one facility.

2.1.4 Partial Coverage

Classical MCLP takes coverage as binary, any demand point within the critical distance is "fully covered" and any demand point beyond that radius is "not covered". Berman and Krass (2002) propose a Generalized Maximal Coverage Location Problem model where coverage is considered as a decreasing step function. Berman et al. (2003) propose a Gradual Cover Decay Model with more general forms of the coverage level decay function. Drezner et al. (2004) examine the planar situation with linear coverage function.

Karasakal and Karasakal (2004) formulate a model with partial coverage called MCLP-P, where partial coverage occurs between two critical distance values.

Eiselt and Marianov (2009) examined partial coverage situation in Set Covering Problems.

Drezner et al. (2010) consider stochastic case of gradual coverage where minimum and maximum critical coverage distances are random variables.

2.1.5 Relocation and Time Horizons

Facility location decisions are occasionally long term due to high costs of realizing a facility which makes the decision of facility location of high importance (Owen and Daskin 1998). Nevertheless, there are situations where some facilities already exist and individuals face the decision of opening new facilities and/or removing former ones. In the literature, it is widely referred as opening and closing of facilities or relocation of facilities. Relocation concept is closely associated with dynamic location problem since time horizons are often involved.

Wesolowsky (1973) proposes a model for single dynamic facility location problem that allows relocation and takes relocation costs into account.

Wesolowsky and Truscott (1975) examine the multiperiod location-allocation problem with relocation of facilities. They propose a model to locate a fixed number of facilities and allow relocation.

Van Roy and Erlenkotter (1982) study an uncapacitated dynamic facility problem where the objective is to minimize the discounted cost of transporting goods from facilities to demand points. The model allows opening new facilities and closing existing ones over the time periods.

Drezner and Wesolowsky (1991) examine the problem of locating a facility in a changing environment, whereas demand shifts occur over time with the objective of minimizing total expected cost within the predictable time horizon.

Dell's (1998) study shows a military application of optimizing the closure and relocation of army bases in the U.S. due to the shrinkage of army personnel which uses an mixed integer programming (MIP) method, BRACAS (Base Realignment and Closure Action Scheduler).

Wang et al. (2003) consider a budget constrained single period uncapacitated facility location problem where opening and closing of facilities are allowed. They explain the need of relocation as the shift in demand over time and that the position of existing facilities might not be adequate. The objective is to minimize the total weighted distance from the demand points, analogous to p-median problem. They restrict the maximum of number of open facilities in addition to the budget constraint.

In Farahani et al. (2009), a single facility case is considered with multiple relocation opportunities for finite or infinite time horizons. Their objective is to minimize location and relocation cost.

Batta and Huang (1989) and ReVelle and Serra (1991) take the relocation issue in competitive facility location problems. Batta and Huang (1989) discuss a relocation promotion model that tries to maximize the profit due to relocation and/or promotion subject to exponential decay of demand with distance and diminishing monetary returns in promotion of demand functions. ReVelle and Serra (1991) present a model to locate new facilities and relocate existing facilities to maximize the captured market demand in a competitive environment according to Cournot and Stackelberg strategies.

2.2 Interactive Multi-Objective Decision Making

Step Method, or STEM, is proposed by Benayoun et al. (1971). The method first finds the optimal points for each objective separately and then finds non-dominated points benefiting from the weighted Tchebycheff distance metric. Decision maker is asked to evaluate the objectives to sacrifice from or favor them.

Geoffrion et al. (1972) propose a method based on the Frank-Wolfe gradient ascent algorithm (1956), using decision maker’s preference information by pair-wise comparisons.

Zionts and Wallenius (1976, 1983) propose a reduced feasible weight space method. In this method decision maker is asked for preference information about trade-off vectors and adjacent extreme points. On the presence of conflicting information, preference information is deleted starting from the oldest until inconsistency is resolved.

Steuer's (1977) study is a criterion cone reduction method that forms convex combinations of non-dominated criterion vectors and benefits from decision maker information in an a posteriori way to find the convex combinations that gradient closest to decision maker's preferences.

Korhonen and Laakso (1986) develop a visual interactive method that uses achievement scalarizing programming to direct the search on the non-dominated front.

Köksalan and Sagala (1995) make use of fictional superior solutions (partition ideals) in the discrete solution space, that dominate several solutions to ask the decision maker between a preferable feasible solution and the partition ideal to reduce the feasible solution space quickly and with the least amount of questions possible.

Köksalan and Karasakal (2006) use Köksalan and Sagala's (1995) partition ideal idea for the continuous space problem to reduce the feasible solution space and to approximate decision maker's utility function by reducing the feasible weight space at the same time.

Balıbek and Köksalan (2010) apply Korhonen and Laakso's (1986) approach in public debt management strategy problem.

Miettinen et al. (2010) propose their study on the assumption of the human perception of gain and loss. They propose a model called NAUTILUS. It takes a solution from the nadir point and iteratively improve the solution by steering according to preference information from the decision maker. The interactive part is

based on behavioral assumptions like the asymmetry of reaction against gain and loss and past experience to affect future decisions.

2.3 Multi-Objective Facility Location Problems

Until now, only single objective models are covered in this survey. Though, most real life facility location problems have several conflicting objectives. Different solving methods are considered for multi-objective problems. Because, there might exist a multitude of solutions .

ReVelle et al. (1977) examine the problem of locating fire stations according to several criteria; coverage of fires, area, population, property value, property value hazard and population hazard.

Ross and Soland (1977) show the similarities between Generalized Assignment Problem (GAP) and facility location problems. Ross and Soland (1980) benefit from these similarities in multi-objective facility location problems. The study presents an interactive approach for the solution of the problem, using satisfaction levels.

Min (1988) develops a multi-objective MIP goal programming model for the relocation and expansion decisions of public facilities that resemble capacitated dynamic facility location problem in a fuzzy decision environment. The objectives are “coverage of populations and proximity to center of gravity of each community”, “proximity to the old facility to be closed” and “accessibility to major transportation arteries or public transportation systems or availability of spacious parking lots”.

ReVelle and Laporte (1996) introduce new models and concepts to facility location problems. Their study compartmentalizes facility location problems as problems with multiple objectives, multiple products and machines and spatial interactions.

Brimberg and ReVelle (1998) propose a bi-objective model that minimizes total cost and maximizes demand coverage for an uncapacitated facility location problem. They solve the bi-objective model using weighting methods.

Jayaraman (1998) proposes a multi-objective model for a capacitated multi-product public facility location problem. There are two cost objectives (i.e. fixed cost of operations, variable operating costs) and average response time to be minimized. The number of facilities to be opened is fixed to a pre-determined value.

Melachrinoudis and Min (2000) present a multi-objective, multi-echelon capacitated dynamic location allocation model that takes relocation and phase-out of facilities into account over the time horizon. The objectives are the minimization of cost and the maximization of traffic access and local incentives.

Melachrinoudis et al. (2000) also present a multi-objective, dynamic location problem which determines an optimal relocation site and phase-out schedule of a single manufacturing and warehouse facility. The criteria are "total costs during the planning horizon", "average access times from the new site to each customer, supplier and transportation infrastructure respectively" and local incentives (tax credits, labor quality etc.). Problem is modeled by using Physical Programming (a variation of Goal Programming), six objective ranges are set as ideal, desirable, tolerable, undesirable, highly undesirable and unacceptable with different weights of penalty. The ranges are set by the decision maker. Relocation is also discussed.

Harewood (2002) propose a model for the bi-objective version of Maximum Availability Location Problem for the deployment of ambulances in Barbados. The first objective is to maximize coverage and the second is to minimize the costs incurred. Real data obtained from Barbados Emergency Ambulance Service is analyzed. The model is optimally solved and tested by simulation with different parameters.

Liu et al. (2005) introduce a multi-objective emergency facility location model with the presence of existing facilities. In the model three objectives are considered; maximization of coverage that are not previously covered by existing facilities within a critical distance, achieving a reasonable distance between fire stations (i.e. minimization of deviation from a fixed distance) and maximization of coverage

within another critical distance. They propose a multi-objective ant algorithm for the pareto ranking of the problem.

Farahani and Asgari (2007) examine a real-world military logistics case to locate distribution centers. A bi-criteria set coverage model is proposed that minimizes the number of distribution centers to be located and maximizes the quality of locations chosen. Quality objective consists of a number of other criteria (i.e. temperature, economical, infrastructure etc.) and the decision maker is asked to determine these criteria's importance (i.e. weights). Then the quality objective is composed as a weighted sum where weights are determined by the decision maker. There are also some scenarios tried on the model concerning the facilities. In the first scenario no facility exists, in the second scenario there are some existing facilities and these facilities should remain open and in the third scenario it is allowed to close existing facilities albeit they are favorable over prospective facilities.

Farhan and Murray (2008) develop a multi-objective spatial optimization model for the location of park-and-ride facilities. They consider three objectives; maximization of demand coverage, minimization of the total distance between park-and-ride facilities and major roadways and maximization of the preservation of existing facilities. Model is solved optimally and a non-dominated front is obtained for the decision making process.

2.4 Multi-Objective Evolutionary Algorithms (MOEA)

MOEA or EMO (Evolutionary Multiobjective Optimization) is a relatively new but rapidly growing field. There are already numerous contributions by distinguished researchers. One of the most promising topics in MOEA is the elitist approach, where good solutions are favored in reproduction and protected against selection.

Non Dominated Sorting Algorithm (NSGA-II), proposed by Deb et al. (2002), is an elitist algorithm. NSGA-II's working principle is based on non-dominated fronts (i.e. ranks) and crowding distance metric. Crowded tournament selection concept, an elitist method to fill the mating pool, is also first coined in NSGA-II.

Zitzler et al. (2002) propose an elitist evolutionary algorithm called Strength Pareto Evolutionary Algorithm (SPEA-II) using an external non-dominated population and clustering. Fitness and strength functions are based on dominance. Diversity is maintained by density estimation. This algorithm is an improvement of its predecessor SPEA, proposed by Zitztler and Thiele (1998).

Pareto Archived Evolutionary Strategy (PAES) is developed by Knowles and Corne (2000). The motivation for this method originates from the need of solving telecommunications network problems. PAES mechanism is based on local search strategy. A single parent, subject to mutation, creates a single off-spring. A finite sized archive is kept to record non-dominated solutions. The dominance relation between the parent and the off-spring determines the next generation's parent and the state of the archive.

Constraint handling is an important part of the evolutionary algorithms. The fate of the infeasible solutions is not determined by default. There are times when they can be useful indeed. Ray et al. (2009) propose a method that includes infeasible solution as the driving force of the algorithm called Infeasibility Driven Evolutionary Algorithm (IDEA). The study also reports that IDEA performs better than popular MOEA methods like NSGA-II in several ways. They also stress that marginally infeasible solutions make good trade-off alternatives.

2.4.1 Interactive Methods in MOEA

Phelps and Köksalan (2003) propose an evolutionary meta-heuristic that uses "clubs" that include best performing solutions for each objective and estimated utility function to approximate decision maker's utility by extracting preference information using pair-wise comparisons.

Deb and Kumar's (2007) study is to enhance NSGA-II with the reference direction approach. They replace the ranking method where the distance from the reference direction vector is the primary measure to form ranks (i.e. Rank 1 is the closest, Rank 2 is the next closest, etc.).

Deb and Kumar (2007a) also propose to incorporate Jaszkiewicz and Slowinski's (1994) light beam search (LBS) method in NSGA-II to find a preferred set of solutions instead of the representation of the whole non-dominated front.

Pfeiffer et al. (2008) study reference point based evolutionary algorithms from the perspective of group decision making. They implement ranking based and distance based approaches to find solutions where consensus can be achieved.

Thiele et al. (2009) propose a preference based evolutionary algorithm that combines fitness with achievement scalarizing functions. The procedure quickly generates an approximation of the non-dominated front and then concentrates the population on the most preferred region in the objective space, according to the preference information from the decision maker.

Fowler et al. (2010) propose a customized evolutionary algorithm similar to Phelps and Köksalan (2003), to sort solutions that are not evaluated by the decision maker by forming convex preference cones based on evaluations on a set of solutions by the decision maker in the interactive part.

Deb et al. (2010) propose a progressive interactive evolutionary algorithm that uses value functions as the evaluation engine of the decision maker's preferences by asking several solutions to the decision maker to rank them from best to worst.

2.4.2 MOEA in Location Problems

There are surprisingly few evolutionary algorithms for multi-objective facility location problems.

Villegas et al. (2006) propose a cost-coverage uncapacitated facility location-allocation problem. They redesign the supply network of a Colombian coffee association. The bi-objective nature of the problem has required them to generate the non-dominated front of the solutions, at least an approximation. They use algorithms based on Non-Dominated Sorting Genetic Algorithm II (NSGA-II) and Pareto

Archive Evolution Strategy (PAES) and an algorithm based on mathematical programming. They also present a benchmark of the algorithms' performance.

Doerner et al. (2009) take the location problem from a disaster perspective. They propose a multi-objective location model for the post-disaster (tsunami to be specific) decision of locating public facilities (primarily schools). There are three objectives; first objective is a combination of minimization of total weighted distance (p-median) and maximal coverage, second objective is the minimization of the risks and the last objective is the minimization of total costs. A variation of NSGA-II metaheuristic is used to solve the problem and its performance is discussed.

Konstantinidis et al. (2009) propose a bi-objective problem for the location of wireless sensor networks. The model they propose is named multi-objective Deployment and Power Assignment Problem (DPAP). In addition to maximal coverage, maximization of network lifetime is chosen as an objective. They propose solving the problem using Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) and compare its performance with NSGA-II's.

Karasakal and Silav (2010) propose an uncapacitated bi-objective problem that combines well-known concept maximal coverage and p-center. Objectives are the maximization of weighted coverage and the minimization of the maximum distance between a demand point and its closest facility. To solve this problem, they propose a novel algorithm "Modified SPEA-II" (mSPEA-II). Modified SPEA-II uses the fitness function of SPEA-II and crowding distance concept of NSGA-II.

CHAPTER 3

THEORETICAL BACKGROUND

3.1 Multi-objective Decision Problem

The general mathematical representation of a multi-objective decision problem is as follows;

Minimize/Maximize" $\{Z_1(x), Z_2(x), \dots, Z_P(x)\}$

$s.t.\ \ x \in X$

x: Decision vector

X: Feasible space

3.1.1 Dominated Solutions

A solution $x \in X$ is denoted as inefficient or dominated if there exists at least one other solution that is not worse in any objective and is better in at least one objective than x. *(assume all objectives are maximization)*

$Z_i(x') \geq Z_i(x) \quad i = 1,2, \dots, P$

$Z_k(x') > Z_k(x) \quad k \leq P$

3.1.2 Non-dominated Solutions

A solution $x \in X$ is denoted as efficient or non-dominated if it's not dominated by any other solution.

3.1.3 Non-dominated Front

The set of non-dominated solutions in the objective space is called non-dominated front. Pareto front is also used.

3.2 Epsilon Constraint Method

Epsilon constraint method is proposed by Haimes (1971) and used to find non-dominated solutions in multi-objective optimization problems by transforming the multi-objective problem into a special single objective weighted sum optimization problem.

The process is initialized by constructing a payoff table by solving each objective independently to determine the ranges of the objectives. Then, the optimization problem is transformed into the ε -constrained model.

Its representation for a maximization problem is as follows;

$$Max\ Z_t(x) + \emptyset \sum_{i \neq t} Z_i(x)$$

$$s.t.$$

$$Z_i(x) \geq \varepsilon_i \quad \forall i \neq t$$

$$x \in X$$

$\emptyset$: a very small positive number

The problem is first solved with all ε values set to zero to find the first solution where $Z_t(x)$ is the highest, then ε_i values are incremented to $Z_i(x)$ + $\emptyset'$ ($\emptyset'$: a very

small positive number) for each objective to find other efficient solutions and solved again.

For example, let's consider a bi-objective problem.

$Max\ Z_1(x) + \emptyset\ Z_2(x)$

$s.t.$

$Z_2(x) \geq \varepsilon$

$x \in X$

$\emptyset$: a very small positive number

The problem is solved repeatedly by incrementing the ε value (i.e. $\varepsilon^1 = Z_2^0(x) + \emptyset'$, $\varepsilon^2 = Z_2^1(x) + \emptyset^{'}$, $etc.$). The graphical representation of the results is given in Figure 1.

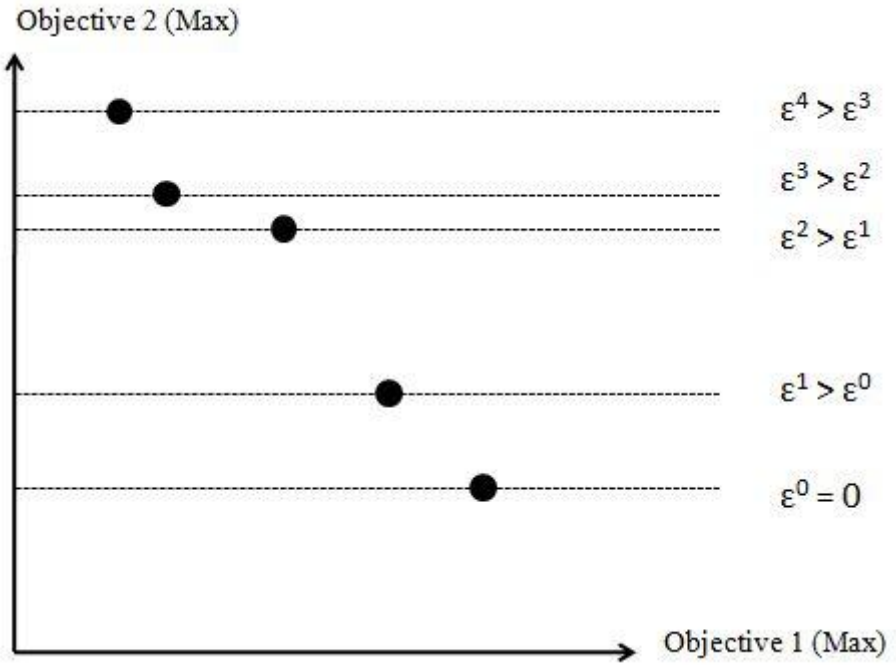

Figure 1. Epsilon constraint method

Different efficient points are found as the epsilon ε is incremented. The resulting set comprises the efficient frontier.

3.3 Estimated Utility Function

For some of the interactive methods in multi-objective decision making problem where decision maker's utility function is assumed to be implicit, there is a need for a mechanism to process the preference information from the decision maker to approximate his/her utility function. One of those methods is to derive a linear estimated utility function.

3.3.1 Linear Estimated Utility Function

$$u(Z) = \sum_{i=1}^{P} w_i^* Z_i$$

Where w_i^* are the weights and Z_i are the objective values of each objective i.

To determine the initial values of w_i^*s, all weights could be taken as equal (i.e. 1 / P for i = 1,…,P) or some preference information could be asked from the decision maker in the form of pair-wise comparisons.

3.3.2 Preference Information

For the sake of consistency, the estimated utility values of the alternatives should be in line with the decision maker's preferences (i.e. preferred solution's estimated utility value should not be lower than the not-preferred solution's utility value). Therefore preference constraints should be added.

$$\sum_{i=1}^{P} w_i Z_i^r - \sum_{i=1}^{P} w_i Z_i^n \geq \varepsilon$$

Z_i^r is the i[th] objective value of the preferred solution and Z_i^n is the i[th] objective value of the not-preferred solution. ε is a very small positive constant to denote there is at least a marginal difference between the preferred and the not-preferred solution in terms of utility value.

3.3.3 Mid-Point Approach

Weights are now to be determined from the reduced feasible weight space. A mid-point approach can be adopted. Mid-point approach is used to determine a set of weights that is as far as possible from the closest preference constraint.

Köksalan et al. (1984) proposed a linear programming problem for the mid-point approach.

$$maximize\ v$$

$$s.t.$$

$$\sum_{i=1}^{P} w_i Z_i^r - \sum_{i=1}^{P} w_i Z_i^n - v \geq \varepsilon \ , \ \forall (r,n) \in Q$$

$w_i - v \geq 0\,,\ \forall i$

$$\sum_{i=1}^{P} w_i = 1$$

$v \geq 0$

v is the variable that is used to set the weights (w_i) approximately at the mid-point of the solution space of the problem.

This model is updated and solved as new preference information is obtained therefore providing new weights each time. One weakness of this method is the inconsistency between the form of the decision maker's utility function and the linear estimated utility function. This type of inconsistency usually causes infeasibility in the mathematical model. One way to overcome this obstacle is to remove preference information (i.e. preference constraints) starting from the oldest until inconsistency is resolved and the model can perform again.

3.4 Partition Ideal

The concept of partition ideals are proposed by Köksalan and Sagala (1995). The method is developed for discrete alternative multi-objective problems under the assumption of decision maker having an implicit general monotone utility function.

The idea is to form "super-solutions" that dominate a portion of the efficient frontier, but weak enough in terms of decision maker's preferences that there exists a solution preferred to the partition ideal.

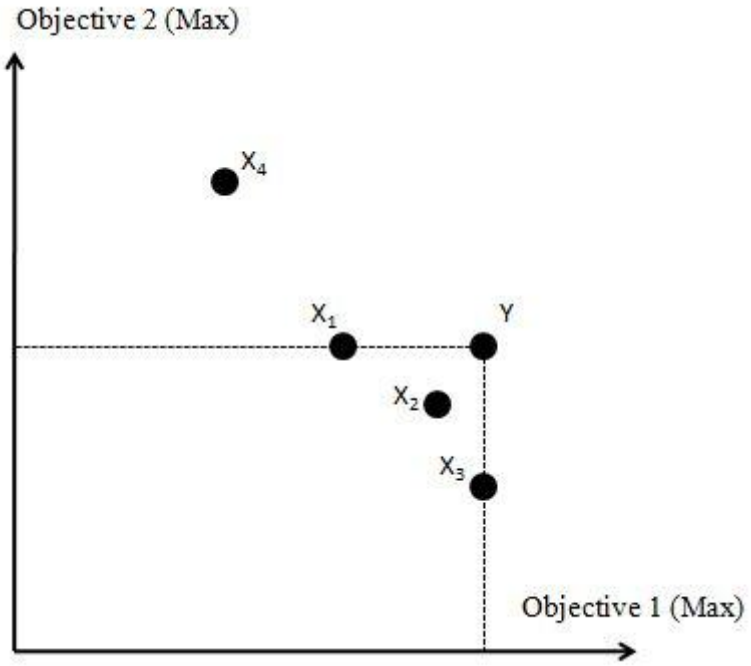

Figure 2. Partition ideal representation

In the example shown in Figure 2, solution Y is a partition ideal that dominates solutions X_1, X_2 and X_3. X_4, on the other hand, is expected to be preferred to Y in order to save the decision maker from the trouble of asking X_1, X_2 and X_3 versus X_4 seperately.

The name "partition ideal" originates from the method of determining partition ideals. The solution space partitioned into equal sections, and then solutions within each section are aggregated to form partition ideals.

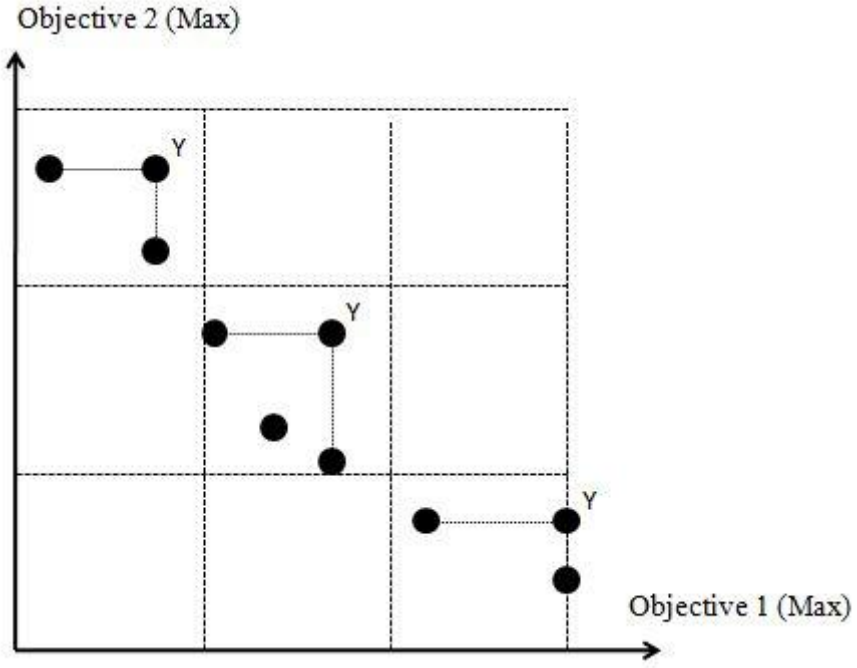

Figure 3. Formation of partition ideals

Partition ideals can be formed in different ways. Another method is proposed in Köksalan and Karasakal (2006) for the continuous space. And in this algorithm, some infeasible solutions will be used for partition ideal role.

3.5 NSGA-II

Non-Dominated Sorting Genetic Algorithm II is proposed by Deb et al. (2000). This elitist multi-objective genetic algorithm is based on the framework of non-dominated ranks and crowding distance measures.

3.5.1 Ranks

Each non-dominated front constitutes a "rank". Ranks are measured according to their domination relation. If a solution is not dominated by any other solution, then it is assigned to Rank 1. And if a solution is only dominated by Rank 1 solutions, then it is assigned to Rank 2 and so on.

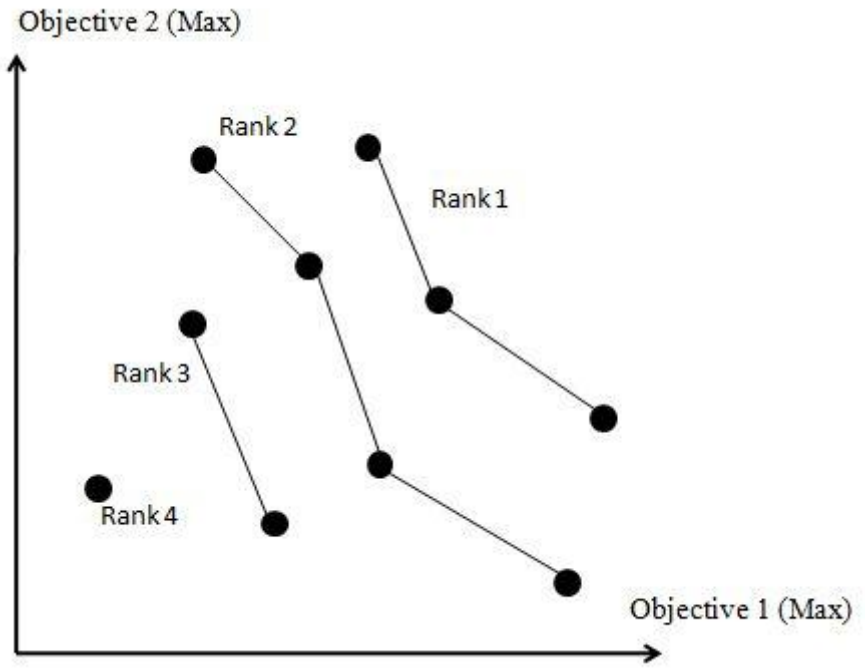

Figure 4. Formation of ranks

3.5.2 Crossover

The crossover phase of NSGA-II is undertaken by a crowded tournament selection operator. In crowded tournament selection operator, two solutions in the population are compared according to their ranks and crowded distance values. The prevalent solution is added to the mating pool for crossover. The conditions for a solution to win a "tournament";

i. It should have a better rank.
ii. If both solutions have the same rank, then it should have a better crowded distance value.

3.5.3 Crowding Distance

Crowded distance is a property of NSGA-II. It is used to find the density of solutions occupying the same rank. A solution's crowded distance is simply calculated by the aggregation of its adjacent solutions' distance for each objective.

The solutions are sorted according to their objective values for each objective and the absolute difference for that objective between the previous and the next solution gives a crowding distance value. The sum of crowding distances for all objectives for that solution gives the crowded distance value of that solution. Crowding distance values for the boundary solutions (i.e. solutions that obtain the maximum or minimum value in any objective) are set to either too large or infinite values.

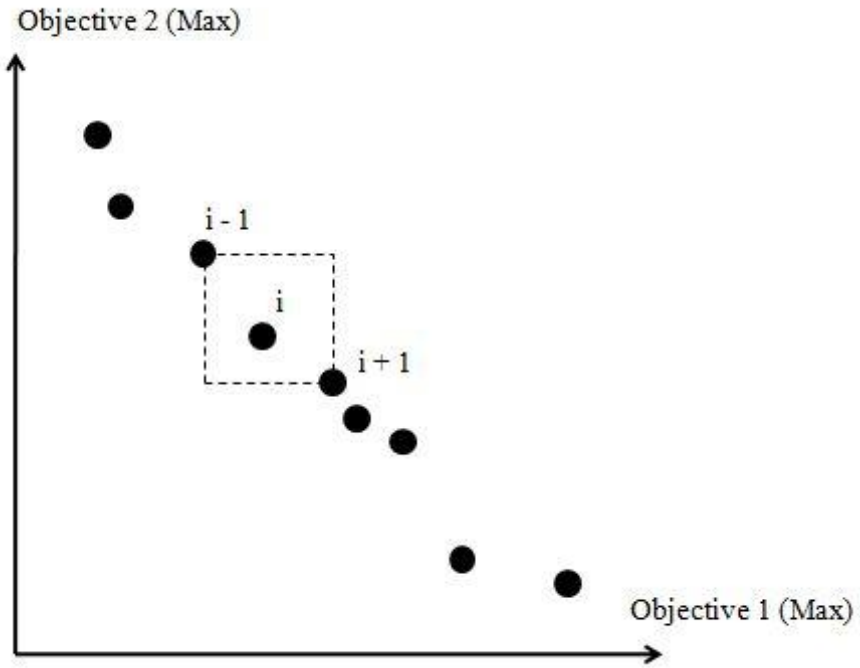

Figure 5. Cuboid distance representation

The equation to calculate crowding distance is as follows:

$$d_{I_j^m} = \frac{Z_m^{(I_{j+1}^m)} - Z_m^{(I_{j-1}^m)}}{Z_m^{max} - Z_m^{min}}$$

where

I_j^m : Index of the j[th] member of the sorted list for the m[th] objective.

$Z_m^{(I_{j+1}^m)}, Z_m^{(I_{j-1}^m)}$: Objective values of the neighboring solutions of the jth member of the sorted list for the mth objective.

Z_m^{max}, Z_m^{min} : Maximum and minimum objective values for the mth objective.

3.5.4 Elimination

Elimination starts from the members of lowest rank and continues to the higher ranks. It is quite common that remaining number of solutions required to be eliminated is lower than the number of members in that rank. If that is the case, then elimination proceeds by the crowded distance criteria. The solutions with the lowest crowded distance values are removed from the population until the population level is back to normal.

3.5.5 Constrained NSGA-II (C-NSGA-II)

On the presence of constraints, NSGA-II would require some changes in determination of parent selection and ranking by redefining the domination concept of NSGA-II. The C-NSGA-II method proposed by Deb et al. (2002) is as follows.

"A solution *i* is said to *constrained-dominate* a solution *j*, if any of the following conditions is true.

- Solution *i* is feasible and solution j is not.
- Solutions *i* and *j* are both infeasible, but solution has *i* a smaller overall constraint violation.
- Solutions *i* and *j* are both feasible and solution *i* dominates solution *j*."

3.6 Maximum Coverage Location Problem

Proposed by Church and ReVelle (1974), MCLP is a problem where a fixed number of facilities are to be set to serve a maximum number of demand nodes with a fixed coverage distance.

The formulation of the problem is as follows;

$$maximize \sum_{i=1}^{I} X_i$$

$$s.t.$$

$$\sum_{i \in I^j}^{I} X_i \leq \beta Y_j \qquad \forall j$$

$$\sum_{j=1}^{J} Y_j = P \qquad \forall i$$

$$X_i , Y_j = \{0,1\} \qquad \forall i, j$$

Where;

i: The index for demand points ($i = 1, \dots, I$)

j: The index for facilities ($j = 1, \dots, J$)

I^j : Set of demand points covered by facility j.

X_i : Binary decision variable to determine whether ith demand point is covered or not.

Y_j : Binary decision variable to determine whether jth facility is opened or not.

P : A limit on the number of facilities to be opened.

β : A sufficiently large number (greater than I).

3.6.1 Partial Coverage

One of the strict assumptions in the maximal coverage location problem is the property of the critical distance. Under this assumption, demand points whose distances are lower than the critical distance value are denoted as “covered” and

those of which that are not within the critical distance value as “not covered”. This binary behavior might not be that realistic. For example, let’s call D_c as the critical distance value and ε as a marginal distance value. A demand point whose distance from a facility is D_c - ε is “covered”, but another demand point whose distance from that facility is D_c + ε is “not covered”. It is a severe classification, considering the very small difference between two demand points’ distances from the facility node.

Partial coverage concept is studied by Berman et al. (2003) and Karasakal and Karasakal (2004), where the coverage function is transformed into a decaying form after a threshold point.

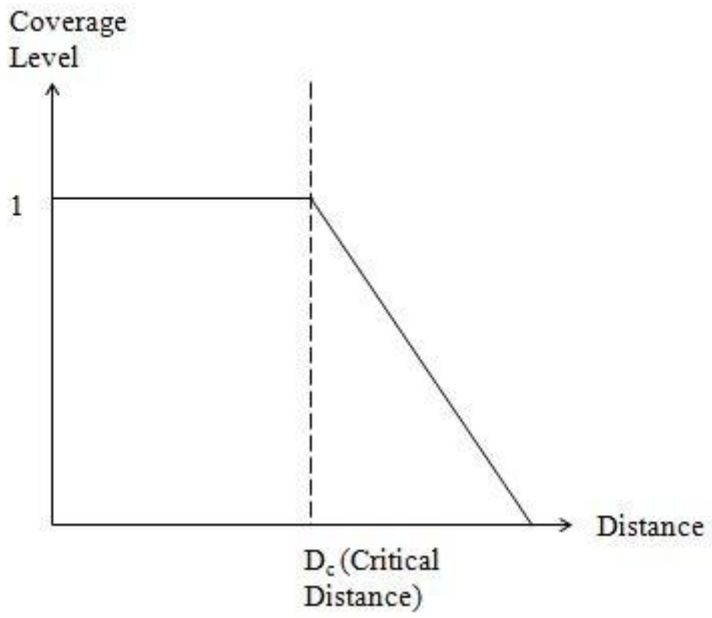

Figure 6. Graphical representation of partial coverage concept

Figure 6 shows a linear decaying function, where the service (coverage) level begins to drop after the critical distance D_c. This type of flexibility may cause change solutions in terms of facility objective. The problem in the Figure 7 is an example.

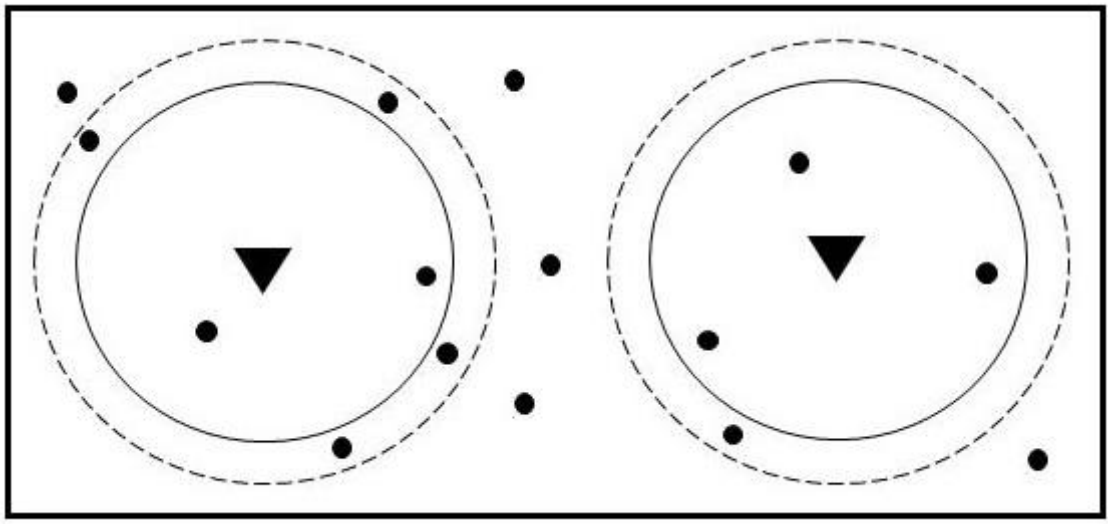

Figure 7. Partial coverage example

In the setting illustrated by Figure 7, there are two facilities represented as reverse triangles and demand points as dots. The closer perimeter (the solid circle) represents the critical coverage distance and the dashed circle determines the partial coverage distance. And let us assume all demand points have the same weight and facilities' capacities are infinite.

Under the critical distance assumption, the facility on the left covers two demand points and the facility on the right covers three demand points. It's clear that the facility on the right covers more and would be preferable to its counterpart.

Though when partial coverage is included the facility on the left covers four more demand points and the facility on the right covers only one more demand point. Now, the facility on the left covers six demand points four of which are partial and the facility on the left covers four demand points only one of which is partial. The distinction between two facilities becomes ambiguous.

CHAPTER 4

PROBLEM DEFINITION

4.1 Motivation

Location problems are widely studied and the research area has significant real life applications. Some noteworthy examples are Pirkul and Schilling's (1988) study of locating emergency facilities considering backup coverage and capacity restrictions and Dell's (1998) study of relocating U.S. Army bases due to shrinkage of army personnel.

Therefore, most of the concepts come from corporeal needs. Maximal Coverage Location Problem (MCLP) is also a popular problem in location theory. Many of its variations exist in the literature. To name some of them, Daskin et al. (1988) combine multiple, excess, backup and expected coverage both in SCP and MCLP; Karasakal and Karasakal (2003) integrate partial coverage concept to MCLP and Brimberg and ReVelle (1998) consider a case where cost is also an objective.

However, multi-objective approaches to location problems are not as popular as the single objective ones. Additionally, cost-coverage bi-objective problems are even a small set of multi-objective location problems.

There is an unrealistic approach to coverage problems that a demand node is deemed uncovered if it's further than a designated critical distance, no matter the amount of the excess distance. For example, let us assume the critical distance is 5 units (km, mile, minutes etc.). If there is a demand point which has a distance of 4.999 units

from a facility it is considered as covered. Suppose there is another demand point that is 5.0001 units away from the same facility and it's considered as not covered. Most of the studies internalize this assumption without question.

Another assumption that might not be so realistic is that the problem setting is always a clean slate (i.e. there are no previous installations). In fact, there might be a structure consisting of several facilities that are already built concerning the previous state of demand. If that is the case, the new model should be an update not a start from scratch.

In this study, a bi-objective model is proposed to tackle all the issues considered above.

4.2 Proposed Model

One of the objectives of this model is the *maximization of total weighted coverage*. This objective originally comes from Church and ReVelle's (1974) Maximal Covering Location Problem (MCLP) which is a special variation of Hakimi's (1964) p-median problem. For further information, Church and ReVelle (1976) discuss the relationship between p-median and covering problems in detail. The weight concept comes from each demand point having different significance (i.e. population). For example, a demand point might have a significance value of 10, but another demand point might have a significance value of 15. Significance can be measured differently, though the most basic way is to think them as population values. Population of households in a specific area might be higher or lower than te population of other areas. Additionally, coverage (service) level affects the value of this objective. Coverage value is usually determined by the distance between a demand node and a facility node. This is generally a binary value, if the demand point's distance is closer than a designated critical distance value from the facility node the demand node is considered as covered; else it is not. For this specific problem, this assumption is replaced with partial coverage concept.

Partial coverage is represented in the model as the coverage (service) level. The coverage is assumed to decay (linearly in this case) from the full coverage after a critical distance value.

The other objective is the *minimization of the total number of open facilities*. There are two types of facilities; existing and new. Existing facilities can be relocated or closed. On the other hand, it's always possible to open new facilities.

Relocation concept is a significant part of the problem. It assumes that there is a current setting of facilities. These *existing (also referred as old) facilities* may become obsolete as demand shifts from one part of the area to another. They may have to be moved to other potential nodes or closed entirely. It's also possible to place *new facilities* on these potential nodes, therefore increase the total number of facilities. Relocation, which is basically moving a facility to another position, consists of two actions; closing an existing facility and opening a new facility on a potential node.

The relocation process is handled as putting a distinction between existing facilities and new facilities. Although one of the objectives is the minimization of total number of facilities, existing facilities have much lower significance in the indicated objective. In other words, it will always be beneficial to hold an existing facility, instead of closing it and opening a new facility. The significance of this trade-off can be explained as there is no trade-off but priorities between these two types of facilities. In other terms, keeping the entire existing facilities set open is more beneficial than closing them all and opening a facility in a potential node.

Capacity limits are set for each facility. Demand cannot be split, therefore cannot be assigned to more than one facility.

4.2.1 Mathematical Model

4.2.1.1 Indices and Sets

i: Demand points (nodes). $(i = 1, \dots, I)$

j: Facilities. $(j = 1, \ldots, J)$

New: Set of potential sites to place new facilities.

Old: Set of already existing facilities.

4.2.1.2 Decision Variables

$$X_{ij} = \begin{cases} 1 & \text{if the demand point } i \text{ is assigned to facility } j. \\ 0 & \text{otherwise} \end{cases}$$

$$Y_j = \begin{cases} 1 & \text{facility } j \text{ is opened.} \\ 0 & \text{otherwise} \end{cases}$$

4.2.1.3 Parameters

a_i: Demand (weight) of demand point i.

s_{ij}: Coverage (service) level of facility j to demand point i.

$\emptyset$: A sufficiently small constant. (i.e. < 1/(number of new facilities))

c_j: Capacity value of facility j.

4.2.1.4 Representation

$$maximize \sum_{i=1}^{I} \sum_{j=1}^{J} a_i s_{ij} X_{ij} \quad (1)$$

$$minimize \sum_{j \in New}^{J} Y_j + \emptyset \sum_{j \in Old}^{J} Y_j \quad (2)$$

$s.t.$

$$\sum_{j=1}^{J} X_{ij} = 1 \qquad \forall i \quad (3)$$

$$\sum_{i=1}^{I} a_i X_{ij} \leq c_j Y_j \quad \forall j \tag{4}$$

$$X_{ij}\ , Y_j = \{0,1\} \quad \forall i, j \tag{5}$$

The first objective (1) is the maximization of total weighted coverage, where each demand point is evaluated by their demand weights and service levels due to their assignments to open facilities. The second objective (2) is the minimization of the number of facilities. Here, there are two types of facility nodes; already existing facilities (old) and potential sites for placing facilities (new). Keeping the already existing facilities is the priority; therefore a sufficiently small objective coefficient to ensure there is a significant trade-off between an already existing facility and a new facility.

Constraints ensure assignments are in check, capacity restrictions are not violated and denote the decision variables as binary variables. The first constraint set (3) makes sure that no demand point is assigned to more than one facility. The second constraint set (4) prevents assignments to closed facilities or to exceed capacities of the open facilities. The last constraint (5) forces the assignment and facility existence capacities to take binary values (i.e. 0 or 1).

4.3 Computational Complexity

Decision version of the single objective maximum covering location problem on a general network is shown to be NP-Complete by Megiddo et al. (1983). Hence, as the problem size grows (i.e. number of demand points and number of facilities), computation time required to generate the exact non-dominated front grows exponentially.

The problem presented is a bi-objective capacitated facility location problem, therefore it is also expected to be computationally complex.

Table 1. Problem sizes

	Problem Type 1	**Problem Type 2**	**Problem Type 3**	**Problem Type 4**
Demand Points	50	100	150	200
Facilities	10	20	30	40
Existing Facilities	5	10	15	20
Computation Time	< 1 min	~5-6 mins	> 8 hrs	> 8 hrs

To elaborate on the issue, there are four problem types shown in Table 1. Problem Type 1 is with the smallest number of facility and demand point nodes and can be solved quickly (just under a minute). In Problem Type 2, it took a little longer than Problem Type 1. In a few minutes, the non-dominated front is formed. Though, if problem size is increased a notch further, it takes hours to solve a problem.

Therefore, a need arises to solve the problems quickly and efficiently. In the next chapter (Chapter 5), an interactive evolutionary algorithm is proposed and in Chapter 6, this algorithm is evaluated by benchmarking with exact solution and another similar metaheuristic method.

CHAPTER 5

PROPOSED ALGORITHM (I-TREA)

This chapter covers an overview of the proposed algorithm, the fundamentals used in various parts of the algorithm and the step by step algorithm, intended to elaborate on the process and to provide rationale for using those tools. The fundamentals include chromosome representation, ranks, infeasible solutions and the triggers used to determine the automated actions taken by the algorithm by explaining concepts and processes used in the algorithm. The pseudo code of the algorithm can be seen in Appendix A and the parameter list of the algorithm with values for the specific problem can be seen in Appendix B.

5.1 OVERVIEW

For the specified bi-objective facility location problem, an evolutionary algorithm is proposed. This evolutionary algorithm combines several aspects of other algorithms in the literature and also provides unique futures, some of which are specific and essential to the specified problem. The algorithm is responsive to decision maker's preferences in a progressive process, benefiting from preference information both for sorting out the solution space and concentrating the algorithm's attention to the desired parts of the solution space; uses infeasible but good (i.e. not dominated by feasible or inferior solutions) solutions as flagships of the population progress; and provides enhancements and repairs to the chromosome by mutation and its consolidating functions.

The algorithm is founded on principles combined from different methods in the literature. Some of NSGA-II's properties provide a canvas for the algorithm, its ranking system formed of non-dominated fronts is twisted into three ranks based on feasibility and domination. The crowding distance measure is applicable to only a small portion of the algorithm, though some variations of the measure can be found in the algorithm. It is infeasibility driven, which means a portion of the population is reserved for infeasible but good solutions (i.e. not dominated by the non-dominated front of feasible solutions).

Interaction with decision maker is established throughout the process, preference information is gathered from the decision maker by pair-wise comparisons to estimate decision maker's desired outcome and sometimes denoting a portion of the solution space undesirable for further inquiry.

Decision maker is assumed to be consistent in his preference information. The decision maker is not asked to pick a solution for a second time when it is not preferred for the first time. Also any solution dominated by the non-preferred solutions is assumed to be inefficient, therefore not asked to the decision maker.

"Best" solution is defined by the decision maker's estimated utility and preference information.

The algorithm is simply named as Interactive Three Ranks Evolutionary Algorithm (I-TREA). Like most of the evolutionary algorithms, this algorithm should also be tailored according to the problem. Mutation section is the part where the effect of customization is perceived most.

Briefly, it is a meta-heuristic algorithm consisting of a caste system, protecting a portion of good but infeasible solutions, using decision maker preference information as guidance, benefiting from the unique methods in the mutation function and aimed to provide the most suitable solution according to decision maker's utility. It is an interactive multi-objective genetic algorithm heralded by elitism.

5.2 CHROMOSOME REPRESENTATION

Chromosome representation of the algorithm is easy to obtain information from, simple in design to be suitable to use in the functions of the algorithm without much effort and robust in a way that minimizes the need of repairing the chromosomes (capacity violation is not a defect, albeit undesirable).

Four pieces of information should be derived from the chromosome;

i. Number of open 'new' facilities
ii. Number of open 'existing' facilities
iii. Total weighted coverage
iv. The capacity violation

Coverage matrix, demand point weights and facility capacities are given. Also, the existing and new facilities are known.

5.2.1 Proposed Representation

An example of the proposed chromosome representation is given in Figure 8.

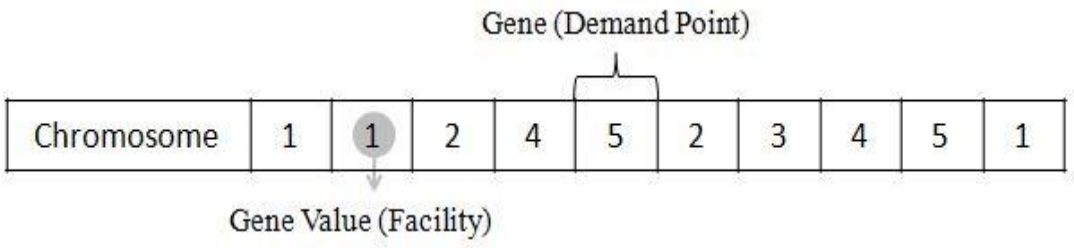

Figure 8. Chromosome representation

Each demand point is represented as a single gene. The values of the genes denote the facilities the demand points are assigned to.

The open facilities can be found by looking at the gene values in the chromosome. Any facility showing up in there means a demand point is assigned to that facility therefore it should be open.

5.2.2 Distinction of New and Existing Facilities

The numbering of new and existing facilities is not entirely random. The first section of numbers is reserved to existing facilities and the second part belongs to new facilities. For example if there are J facilities in total and E of them are existing, then facilities 1 to E are denoted as existing facilities and E+1 to J are new facilities. This way the algorithm can make a distinction between new and existing facilities and evaluate the facility objective value of the solution. As can be seen in Table 2, the first three facilities are existing facilities and the last three facilities are new facilities.

Table 2. Coverage matrix

	Fac\Dem	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Existing Facilities	**1**	1	0	0.8	0	0	1	0
	2	0.5	0	1	0	0	0.4	0.7
	3	0	0	0.2	1	1	0.3	0
New Facilities	**4**	0	0	1	1	0	0	0
	5	1	0.1	0	0	1	0	0
	6	0	0	0	0.6	0.9	0	1

5.2.3 Calculation of Total Weighted Coverage

Total weighted coverage can be derived using the coverage matrix, demand point weight list and the assignment information from the chromosome.

Table 3. Demand weight (population) list

Demand Point	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Weight (Population)	10	15	20	10	10	40	25

For example, let's calculate the weighted coverage of the sixth demand point of the solution in Figure 8. From the chromosome representation it is assigned to the second facility. Using the coverage matrix in Table 2 its coverage value is 0.4 and using the demand point weight list in Table 3 its weight is 40. Then the weighted coverage value of the sixth demand point is 0.4 * 40 = 16. Total weighted coverage can be derived by repeating the same action for each demand point.

5.2.4 Calculation of Capacity Violation

Capacity violation can be derived using the assignment information from the chromosome, demand point weight list and the capacity list.

Table 4. Facility capacities list

Facility	**1**	**2**	**3**	**4**	**5**	**6**
Capacity	250	250	250	250	250	250

Though, the design has its weaknesses. The design unintentionally hinders the improvement of the facility objective, mostly because of the crossover mechanism. The crossover of two chromosomes with different number and types of open facilities is usually a cause of offspring to be weak in the facility objective (see Figure 9).

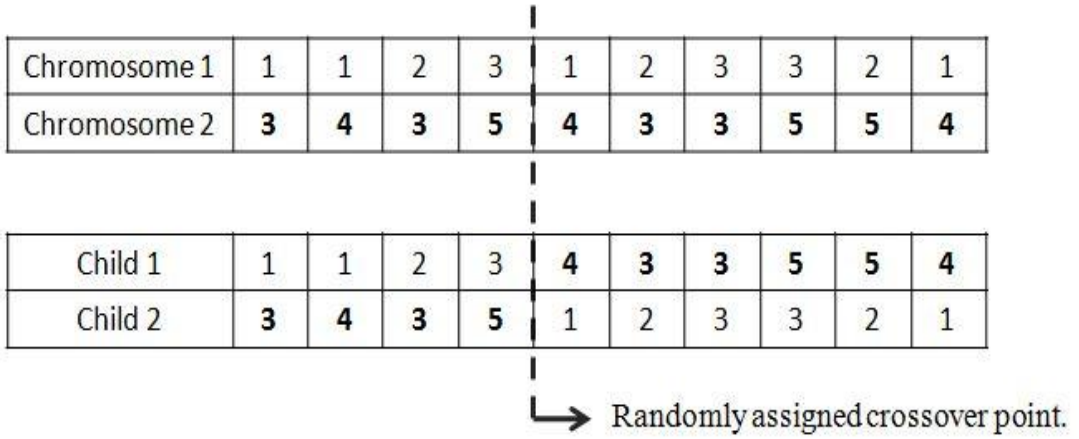

Figure 9. Crossover weakness example

The example in Figure 9 shows that both chromosomes have three facilities open (1-2-3 and 3-4-5 respectively). At the end of the crossover both chromosomes have five facilities open. This action causes a severe setback in the facility objective.

The length of the chromosome is determined by the number of demand points, which is usually a large number. It is expected to affect the runtime and memory requirement of the algorithm seriously.

The proposed chromosome representation, despite its weaknesses is simple in design and effective in implementation.

5.3 RANKS

The structure resembles the ranking system of NSGA-II, though with pre-determined number of ranks. There are three ranks considered for the algorithm and they are simply named Rank 0, Rank 1 and Rank 2. Though they can also be called as ‘infeasible good’, ‘feasible good’ and ‘rejects’ respectively. Each rank differs from the other with two binary properties; being dominated by a feasible or inferior solution and being feasible.

There are two types of solutions in Rank 2 (Rejects). *Inefficient* solutions are solutions that are either dominated by a feasible solution or an *inferior* solution.

Inferior solutions are determined by preference information, made up by not-preferred solutions. They are usually not dominated by feasible solutions. They practically mark the reduced objective space.

The ranks are;

Rank 0 (Infeasible Good): Infeasible solutions that are not dominated by any feasible or inferior solution belong here.

Rank 1 (Feasible Good): Also known as the non-dominated front, Rank 1 consists of solutions that are feasible and not dominated by any feasible or inferior solution.

Rank 2 (Rejects): Rest of the solutions belong here, no matter the feasibility.

The ranks' formation is illustrated in Figure 10.

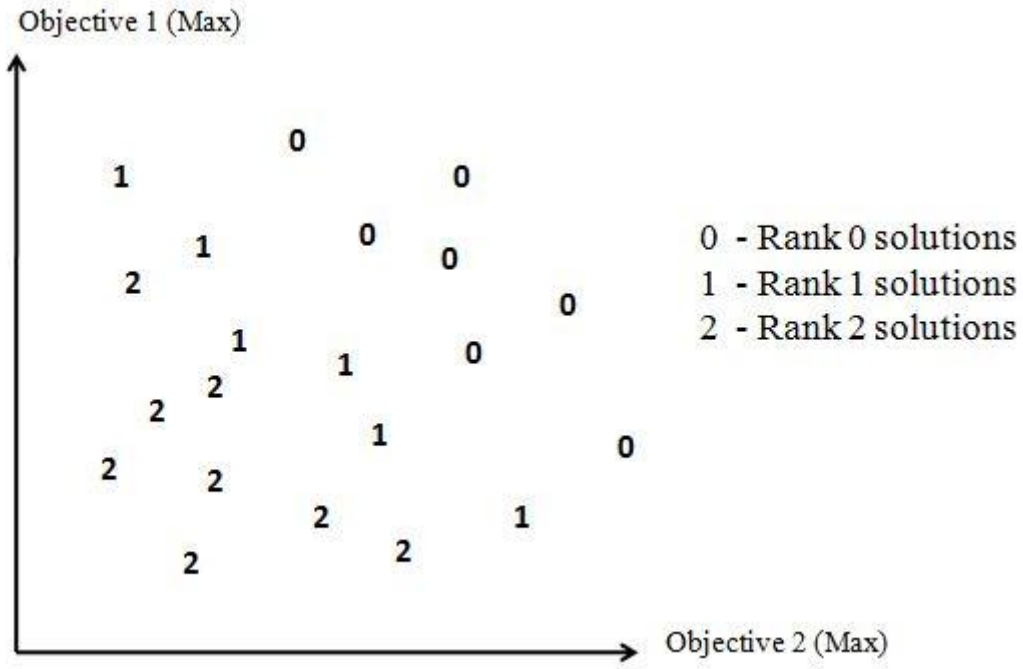

Figure 10. Ranks

5.4 INFEASIBLE SOLUTIONS

One of the defining properties of the algorithm is the presence and protection of potentially beneficial infeasible solutions. Those solutions are named as Rank 0 (or infeasible good) solutions. They are neither dominated by Rank 1 solutions nor in the Rank 2 section dominated by inferior solutions (i.e. previously not preferred alternatives).

There are two reasons to keep Rank 0 solutions;

i. They are useful in the sense of advancement in the population. The better but infeasible solutions may produce better and less infeasible (i.e. less total capacity violation) off-spring and feasibility oriented mutations may process them into efficient feasible solutions.
ii. They can be used as alternatives against the incumbent in the interactive process.

A predefined portion of the Rank 0 solutions are preserved from elimination.

5.5 TRIGGERS

Some functions do not repeat every generation as the rest of the functions in the algorithm. Asking questions to the decision maker (interactive process) is a good example. If there were a question for every generation during the run of the algorithm, hundreds of questions should have been asked to the decision maker. Therefore conditions are set for some functions. When requirements for a function are fulfilled then that function is triggered for once.

The functions triggered by conditions in the algorithm are interactive process, Forced Closure and Second Chance. Termination conditions are also considered within this concept.

5.5.1 Trigger Conditions

For the algorithm, most of the trigger conditions are related to the progress of the population. For example if the incumbent stays the same or the average of objectives of Rank 1 solutions does not change for a number of generations a function might trigger. Also there might be a condition that two questions should not be asked subsequently, and there should be some generations between two questions no matter what. This also applies as a trigger condition.

The trigger conditions are designed in a way that some functions would work only as needed, therefore sparing the algorithm of an abundant computation time and multitudes of questions. They also determine when to terminate the algorithm, providing ample time to the population to progress.

More information on trigger conditions can be found within the explanations of the individual functions.

5.6 STEP BY STEP ALGORITHM

The algorithm consists of seven parts, five of which repeats in a cycle in each generation. Those are initialization, crossover, mutation, classification, elimination, interactive process and termination. Initialization and termination are used only once at the beginning and the end of algorithm respectively.

Initialization: Introduction of the initial members of the population. (Executed only once at the start of the algorithm. Includes *Classification*.)

Crossover: One-point, random-point crossover with crowded tournament selection.

Mutation: Changing a gene either by chance or a special function.

Forced Closure: Closing of a facility. If the closed facility is a ‘new’ facility, then re-opening of an existing facility.

Second Chance: Nullifying of the assignments of demand points which do not return coverage or can return better coverage.

Classification: Putting each solution to their proper ranks. (Rank 0, 1 or 2)

Elimination: Discriminating 'worse' solutions from 'better' solutions and then proceed with the better solutions.

Interactive Process: Extracting preference information from the decision maker.

Measurement and Termination Check: Gathering the population statistics, calculation and decision of the fulfillment of termination conditions.

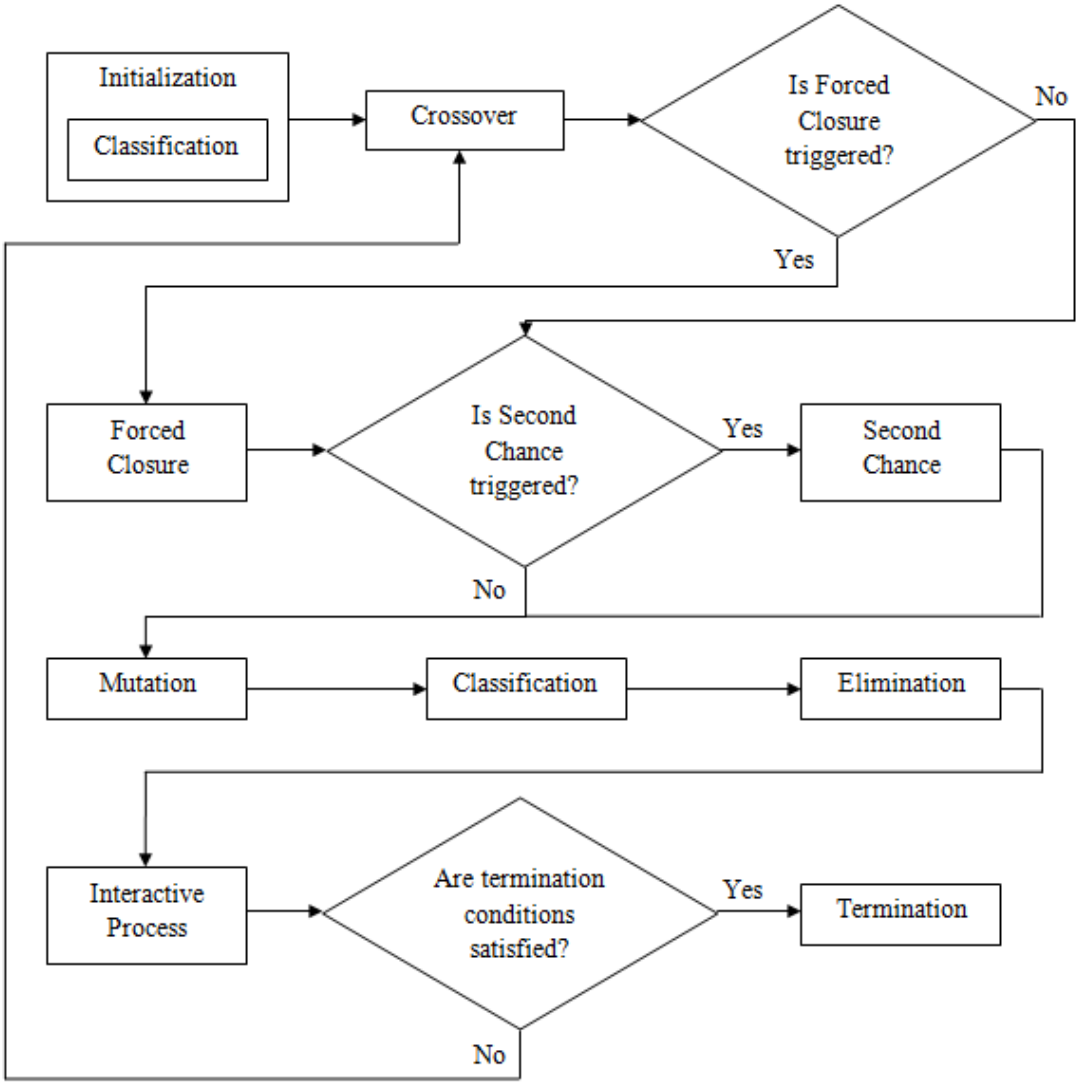

Figure 11. I-TREA Flowchart

5.6.1 Initialization

Initialization is the preparation of the commencement of the algorithm (i.e. generation zero). Initial solutions are introduced to the population and then classified into ranks. The non-dominated feasible solution with the highest estimated utility function value becomes the incumbent.

Incumbent is a common term used in interactive algorithms. Incumbent is assumed as the 'best' solution for the decision maker that can be found so far. Incumbent is prone to changes by either domination or preference information by the decision maker.

5.6.2 Crossover

Crossover is the most ordinary part of the algorithm. The methods used are the simplest and the most common amongst the genetic algorithms. There is also an elitist twist called crowded tournament selection method, the one that is also used in NSGA-II (for detailed information see Theoretical Background chapter). The algorithm's mate selection system benefits from a slightly modified method.

The decision of solutions to be paired for crossover is resolved by implementing crowded tournament selection method. Every solution in the population has a chance to enter the mating pool.

NSGA-II's (Deb, 2001) crowded tournament selection method is as follows:

"A solution i wins a tournament with another solution j if any of the following conditions are true:

- If solution i has a better rank, that is $r_i < r_j$
- If they have the same rank but solution i has a better crowding distance than solution j, that is, $r_i = r_j$ and $d_i > d_j$.

The first condition makes sure that chosen solution lies on a better non-dominated front. The second condition resolves the tie of both solutions being on the same non-dominated front by deciding on their crowding distance."

The algorithm's modified version is as follows:

A solution i wins a tournament with another solution j if any of the following conditions are true:

- If solution i has a better rank, that is $r_i < r_j$ (Rank 1 < Rank 0 < Rank 2)

- If they have the same rank but solution i has a better estimated utility value than solution j, that is, $r_i = r_j$ and $d_i > d_j$.

The first condition makes sure that chosen solution resides in a better rank. The second condition resolves the tie of both solutions being on the same non-dominated front by deciding on their estimated utility.

There are several reasons for the algorithm's crowded tournament selection differs from the Deb's NSGA-II crowded tournament selection. First, the algorithm does not necessarily consist of non-dominated fronts as NSGA-II, therefore it is not so convenient to assess a solution's virtue by its crowding distance. Second, the algorithm is intended to be a little more elitist than other elitist genetic algorithms and therefore, at least in the crossover section, sacrificing diversification for the sake of intensification is preferred. Last, since the algorithm reacts on decision maker's preferences, it is only proper to use an indicator that is the algorithm's perception of DM's utility.

After the mating pool's inhabitants are formed, it is time to crossover. Crossover is fairly simple and random. Two solutions (chromosomes) are paired from a single, random point to give birth to two other solutions. The process is shown in Figure 12.

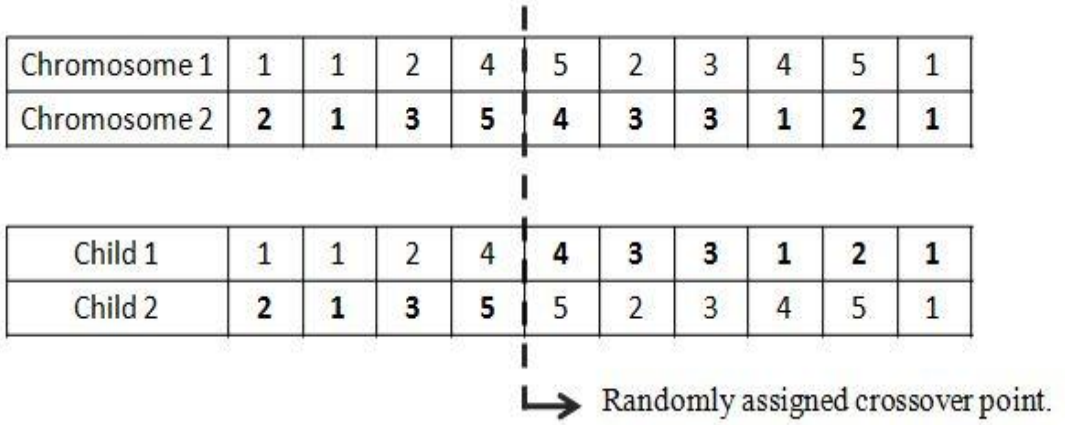

Figure 12. One-point, random point crossover

After the crossover process, the population is expected to double. Next, the new arrivals are processed for mutation.

5.6.3 Mutation

Mutation is the part where customization is paramount in the algorithm. Due to flaws, as indicated before, in the structure of the chromosome representation and the specific nature of the multi-objective problem, mutation should be handled differently and consolidated with other functions' periodic intervention. Those consolidating functions Forced Closure and Second Chance are also explained in this section.

The mutation function used in the algorithm is rather a smart function than a random process; also showing opportunistic behavior, albeit restricted by feasibility concerns.

Mutation process begins with its auxiliary functions; Forced Closure and Second Chance, respectively. They leave some genes in the chromosome unassigned by revoking the respective demand points' assignments to their former facilities. In addition, some other demand points' assignments can be also revoked par chance (i.e. usually with a small probability determined by the mutation probability

parameter). Then, all these demand points will be reassigned under the mutation function.

5.6.3.1 Forced Closure

Forced Closure is a special function that is triggered only if special conditions apply and when triggered it applies to the solutions only occasionally.

Forced Closure is an effective function that is aimed to improve the facility objective. The procedure is as follows:

- Close a facility.
- If the closed facility is a ‘new’ facility, open an ‘existing’ facility.

The necessity for the Forced Closure function arises from the lack of performance of crossover and mutation functions to reduce the gene value variety. Therefore facilities should be shut down manually, instead of waiting for the natural progress to take place. It is fairly easy to go back in the facility objective therefore a ‘forced opening’ function is unnecessary.

The decision of which facility to close, and which ‘existing’ facility to open on the event of the closure of a ‘new’ facility, can be handled in any way that is found subtle. The methods can range from entire randomness to complicated measures to appreciate the potential of the facilities.

The forced closure process used in the algorithm for this specific problem takes facilities’ aggregated weighted coverage values into consideration in a stochastic manner to determine the chances of election. In other words, a facility which gets to cover more weighted demand is less likely to be closed and an ‘existing facility’ which gets to cover more weighted demand is more likely to be opened.

Additionally, as a rule, if a facility’s coverage of a demand point is the highest among other open facilities the weighted coverage score it gathers from that demand point is boosted by %50. The rationale for this bonus is for distinction of facilities

that can cover more than other facilities for individual demand points rather than facilities that cover an abundance of demand points with average coverage values. The coefficient of additional score (%50 for this problem) is determined by preliminary runs.

For example a facility that can provide full coverage (i.e. coverage value of 1) to two demand points is preferable to a facility that can provide half coverage (i.e. coverage value of 0.5) to four demand points. Assume all six demand points have equal weights.

Table 5 provides problem information for an illustrative example of Forced Closure function.

Table 5. Forced closure example

State	Fac\Dem	1(10)	2(15)	3(20)	4(10)	5(10)	Potential Gain
Open	**1**	1*	0	0.8	0	0	**31** = 15* + 16
Open	**2**	0.5	0	1*	0	0	**35** = 5 + 30*
Open	**3**	0	0	0.2	1*	1*	**34** = 4 + 15* + 15*
Open	**4**	0	0	1*	1*	0	**45** = 30* + 15*

The values with asterisk (*) denote the highest coverage values for a demand point (i.e. facilities will get extra scores from that coverage values). The values in parentheses in the first row are weights (population) of respective demand points. Such as the weight of the first demand point is 10, the second 15 and so on.

The aggregate coverage score of facility 1 is calculated as follows. Its coverage of the first demand point is the highest value among the open facilities. Therefore, the coverage score it gets from that demand point is boosted by %50; it gets 10 + 0.5*10 = 15 from the first demand point. The coverage score it gets from the third demand point is 20 * 0.8 = 16. The aggregate coverage score of facility 1 is 15 + 16 = 31.

The aggregate coverage scores for other facilities are given on the last column of Table 5.

The stochastic process of closing a facility is calculated as follows. As mentioned above, if a facility's aggregated coverage is higher it is less likely to be closed. So, reciprocals of aggregate coverage scores of facilities are taken and the probability of a facility to be closed is determined by dividing its reciprocal aggregate coverage score by the sum of reciprocal aggregate coverage of all open facilities.

For example, the probability of closing facility 3 is (1/34) / (1/31 + 1/35 + 1/34 + 1/45) ~= %26.

The process of opening an 'existing facility' on the event of closing a 'new facility' is similar. The same aggregate facility score procedure applies for closed 'existing facilities'. The only difference is the scores are used in probability calculations without change, instead of their reciprocal values. For example, suppose the facilities and their aggregate coverage scores shown in Table 5 are used for opening an 'existing facility'. Then the probability of opening facility 3 would be 34 / (31 + 35 + 34 + 45) ~= %23.

The Forced Closure function quickens the improvement of solutions and is proved to be effective.

5.6.3.2 Second Chance

Second Chance is a special function that is triggered only if special conditions apply and when triggered it applies to the solutions only occasionally.

Second Chance is a simple but effective function that is aimed to improve the coverage objective by revoking any demand point assignment that could not deliver its maximum coverage potential. Any demand point assignment with zero coverage is revoked by default to give up space to any potentially profitable assignments.

Table 6. Second chance example

Fac\Dem	1	2	3	4	5
1	**1**	0	0.8	0	0
2	0.5	**0**	1	0	0
3	0	0	**0.2**	**1**	1
4	0	0	1	1	**0**

An illustrative example is given in Table 6. There are 5 demand points and 4 facilities; hence coverage values that are marked with bold indicate the assignments (e.g. 3rd demand point is assigned to the 3rd facility with coverage value of 0.2).

Assignments of demand points 2, 3 and 5 are revoked. 2 and 5 return zero coverage from their assignments and there are better alternatives for the 3rd demand point.

This function assists the mutation process in two ways. First, since mutation is programmed to be extremely careful about feasibility, it opens up space by removing demand point assignments that do not contribute to the coverage objective or where there is potential for improvement for that demand point's assignment. Second, the Second Chance function pinpoints the demand points whose contribution to the coverage objective can be improved and immediately includes them in the mutation, without leaving it to chance (i.e. mutation probability).

Second Chance function enables, even if not guarantees, the mutation to achieve better performance in the coverage objective.

5.6.3.3 Step by Step Process

The core process of mutation is more intricate than usual mutation functions. The order of mutation is rather iterative than linear. It does not follow the gene order in the chromosomes. Instead, a supply and demand oriented approach is adopted to alleviate the restrictions from the facility capacities and avoid infeasibility at the same time.

There are three ways of getting a gene's (demand point) value (facility) revoked. Two of them are explained before; Forced Closure and Second Chance. The third way is the ordinary approach, mutation probability. Before the start, some genes assignments are revoked par chance subject to the mutation probability.

The mutation process, designed for this specific problem, is choosing a facility and choosing a demand point to assign them to each other. The process is repeated until there is no demand point left unassigned.

The decision of which facility to choose and to assign which demand points to that facility is subject to design. There are many ways of handling facility and demand point potentials. The main steps of the mutation (separately from Forced Closure and Second Chance) are given in a flowchart in Figure 13.

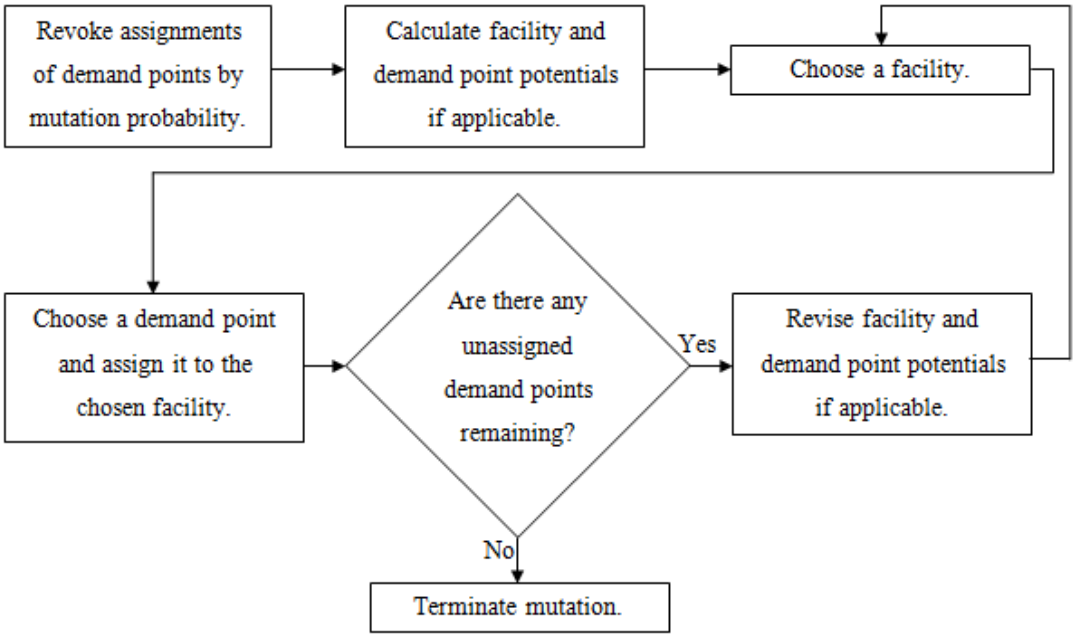

Figure 13. Mutation flowchart.

The extra process (i.e. calculating and revising potentials) in the flowchart might be required if the design of the mutation requires some appreciation of facilities and

demand points according to coverage, weight and capacity values. The process is repeated until all the demand points are assigned to a facility.

The process designed for the specific problem is as follows. There are three measures used to assess the potentials of facilities and demand points. *Aggregate coverage score* is the sum of weighted coverage values of a facility. If a facility covers two demand points with coverage values 0.5 and 1 and weights 10 and 15 respectively, the aggregate coverage score of that facility is 20. It is used to assess a facility's weighted coverage potential.

Maximum coverage keeps the record of a demand point's maximum potential. For example if there are three facilities that cover a demand point with coverage values 0.3, 0.6 and 0.9, the demand point's maximum coverage value is 0.9.

Total coverage is the sum of coverage values that cover a demand point. For example if there are three facilities that cover a demand point with coverage values 0.3, 0.6 and 0.9, the demand point's total coverage value is 1.8. It is used to assess the existence of alternatives for a demand point.

There is a rule, though. If a demand point has a weight (population) that is higher than the remaining capacity of a facility, the coverage relationship is not included in the calculation of these measures. For example, suppose there is a facility with 15 units of capacity left and there are two demand points with weights 10 and 20. The coverage value of second demand point is not included in any of the three measure calculations. Though, if there are two demand points with weights 10 and 15 they will both be included.

The assignment commences from the facility with the highest aggregate coverage score. A demand point is assigned to that facility stochastically. Demand point probabilities are evaluated by their coverage values by the chosen facility and their maximum and total coverage values.

The assignment process for the specific problem is as follows. If;

- A demand point's *total coverage* value is more than twice of its *maximum coverage* value, or
- A demand point's *total coverage* value is more than 1 and its coverage value is less than its *maximum coverage* value,

that demand point's score is calculated as:

$$Score = (Coverage\ Value / Total\ Coverage)^2$$

Else:

$$Score = (Coverage\ Value)^2$$

This procedure is repeated for each available demand point. Then the probability to assign a demand point to that facility is that demand point's individual score divided by the sum of all demand points' scores.

The rationale for making the assignment scoring process so convoluted is to further increase the chances of a demand point with a high coverage value available for that assignment. For example, suppose six demand points with the same weights are covered by a facility and their coverage values are 0.2 for five demand points and 1 for the remaining demand point. The chance of assigning the demand point with more coverage is 1 / (0.2 + 0.2 + 0.2 + 0.2 + 0.2 + 1) = %50. But, if the proposed procedure is used (assume all presumed demand points are covered by only that facility, therefore total coverage values are equal to their coverage values) the chance of choosing the demand point with the highest coverage value would be increased to 1^2 / [$(0.2)^2$ + $(0.2)^2$ + $(0.2)^2$ + $(0.2)^2$ + $(0.2)^2$ + 1^2] ~= %83.3.

The measures *aggregate coverage score*, *maximum coverage* and *total coverage* values are updated after the assignment and the whole assignment process is repeated. If there are demand points left with zero *maximum coverage* value, they are assigned randomly but without violating capacity constraints if possible.

An illustrative example is provided. Table 7 shows a coverage matrix of 5 demand points and 4 facilities. The numbers in parentheses are weights (population) for each demand point and remaining capacities for each facility. Suppose all demand points should be reassigned.

Table 7. Mutation example

Fac\Dem	**1**	**2**	**3**	**4**	**5**
1	1	0	0.8	0	0
2	0.5	0	1	0	0
3	0	0	0.2	1	1
4	0	0	1	1	0

First, a*ggregate coverage scores* of the facilities are calculated. Weighted potential of the 1st facility is 26 = 1 * 10 + 0.8 * 20. Though, weighted potential of the 2nd facility is only 5 = 0.5 * 10. It is because the weight of the 3rd demand point is more than the 2nd facility's capacity. There are no such issues for the 3rd and the 4th facilities; their weighted potentials are 24 and 30 respectively.

Next, *maximum coverage* and *total coverage* values of demand points are calculated. 1st demand point has a maximum coverage of 1 and total coverage of 1.5. 2nd demand point has neither a maximum coverage nor a total coverage value, since it is not covered by any facility. 3rd demand point has a maximum coverage of 1 only because it is covered by the 4th facility with full coverage. Hence it cannot be assigned to the 2nd facility because of capacity issues. And its total coverage value is 2. The 4th demand point's maximum coverage value is 1 and total coverage value is 2. Lastly, the 5th demand point's both maximum coverage and total coverage values are 1.

The facility with the highest weighted potential is the 4th facility (30). So, it is chosen as the first facility to assign demand points to. The scores of candidate demand points for assignment are as follows. 1st, 2nd and 5th demand points are not

considered since they are not covered by the 4th facility (i.e. zero coverage value). 3rd demand point's chance is calculated as follows. Its total coverage (2, hence it cannot be covered by 2nd facility) is twice of its maximum coverage (1) then its chance score is $0.25 = (1 / 2)^2$. The 4th demand point has also the same chance score 0.25. The actual chance of assigning the 3rd demand point is %50 = 0.25 / (0.25 + 0.25).

After 3rd or 4th demand point's assignment to the 4th facility, *aggregate coverage score*, *maximum coverage* and *total coverage* values will be updated and the process will be repeated.

The rationale for making a mutation process so complicated can be explained as the diminishing effectiveness of intensification for large problem sizes. As the problem size grows, solutions' neighborhood also expands. Therefore a simple mutation function might render useless, against increasing sizes of solution neighborhood.

Nevertheless, a variation of I-TREA with a simple mutation procedure will be introduced to show the relative ineffectiveness of the simpler approaches. Forced Closure function will be maintained, but Second Chance function will not be included. The demand points subject to mutation will be assigned to facilities randomly without violating capacity restrictions.

5.6.4 Classification

After the crossover and mutation processes, the off-springs will be ready to be introduced to the population and classified. The classification system is in accord with the ranking system. A flowchart is given in Figure 14. The process is repeated for each off-spring until all the off-springs are properly placed.

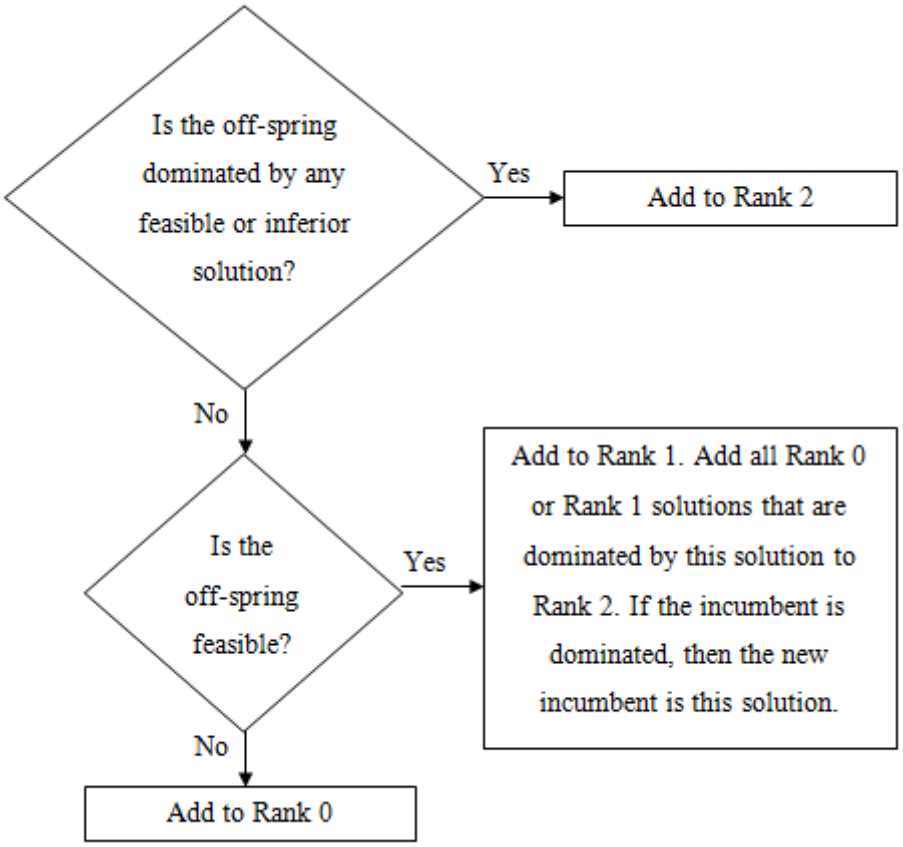

Figure 14. Classification flowchart

5.6.5 Elimination

Since crossover process doubles the number of solutions an elimination function is required to keep the individual level in the population constant. The principles of elitism and diversity should be represented in balance in the elimination process for the sake of the progress of the algorithm.

Elimination is a relatively straightforward process, though with different special rules within the ranks. Each rank goes under elimination process twice. Generally speaking, first elimination process acts as a sweep of ‘twin’ (i.e. having the same objective values) and ‘weaker’ (i.e. dominated within the rank) solutions and the second elimination process uses measurements like estimated utility value and distance as criteria.

There is a special quota for Rank 0 solutions. Rank 0 solutions are exempt from further elimination if their number of members falls below a designated proportion (e.g. %10) of the population.

Elimination continues until the population level is reduced to its originally designated value. The population level is checked after each eliminated solution. If the population is lowered to its designated level, elimination process ceases its operation.

Elimination order within each subsection starts from the 'older' solutions (i.e. solutions that are classified in previous generations). A high solution turnover is aimed within the population.

5.6.5.1 Duplicate Elimination

There are two types of chromosomes that return the exact objective values; duplicates and twins. *Duplicate* solutions have the exact chromosome content. On the other hand *twin* solutions only return the same objective values, yet some of their gene values differ.

Duplicate solutions are highly undesired. Therefore, they are eliminated first.

5.6.5.2 First Rank 2 Elimination

The solutions are evaluated for elimination according to their objective values and capacity violation. The solutions that are dominated by other Rank 2 solutions are eliminated. If there are two solutions that have the same objective values but different capacity violation values, the solution with the higher capacity violation is eliminated. If there are two solutions that have the same objective values and capacity violation values, the 'older' solution is eliminated.

The expected outcome is a 'non-dominated front' of Rank 2 solutions.

5.6.5.3 First Rank 0 Elimination

The purpose of the first Rank 0 elimination is to exclude any Rank 0 solution that is 'totally dominated' by any other Rank 0 solution both in terms of objective values and capacity violation.

The solutions that are dominated by and have higher capacity violation values than any other Rank 0 solutions are eliminated. For example if a solution has objective values of 0.5 and 0.6 with capacity violation value of 30 and another solution has objective values of 0.7 and 0.8 with capacity violation value of 20, the former solution is eliminated.

5.6.5.4 Second Rank 0 Elimination

After the first elimination of Rank 0 solutions, second elimination of Rank 0 is immediate. Rank 0 solutions are eliminated by their minimal radial (euclidian) distance. This diversity approach aims to spread the Rank 0 solution as evenly as possible.

Inevitably, every time, there will be at least two solutions with the same minimum radial distance. In this case, the Rank 0 solution with the lower estimated utility is eliminated.

5.6.5.5 First Rank 1 Elimination

Eliminate Rank 1 solutions if there are any other Rank 1 solutions with the same objective values (i.e. twins).

5.6.5.6 Second Rank 2 Elimination

Eliminate remaining Rank 2 solutions according to the crowding distance principles that are used in NSGA-II.

5.6.5.7 Second Rank 1 Elimination

The second phase of Rank 1 elimination is handled by the estimated utility criteria. Elimination begins from the Rank 1 solution with the lowest estimated utility value.

The elimination continues upwards until the population is lowered back to its normal level.

After the elimination process, the population is expected to return to its normal level. The objective of elimination is sorting the most useful solutions due to population constraint and removing the rest from the population. Although elitism is championed, a necessary dose of diversity is applied. Elimination process is mainly straightforward with well defined boundaries between ranks with an array of elimination methods within the ranks.

5.6.6 Interactive Process

In this part of the algorithm, the decision maker is asked for preference information by presenting pair-wise comparisons. The main objectives of the interactive process is to reduce the objective space so the search can be concentrated on regions that are more desired by the decision maker; and, more importantly, to find the most preferable solution by the decision maker.

The main question types are asking a Rank 0 solution against the incumbent and asking a Rank 1 solution against the incumbent.

The interactive process is activated only when triggered and there is at least one other Rank 1 solution than the incumbent. The interactive process flowchart can be seen in Figure 15. The trigger conditions and methodology for choosing Rank 0 and Rank 1 solutions differ.

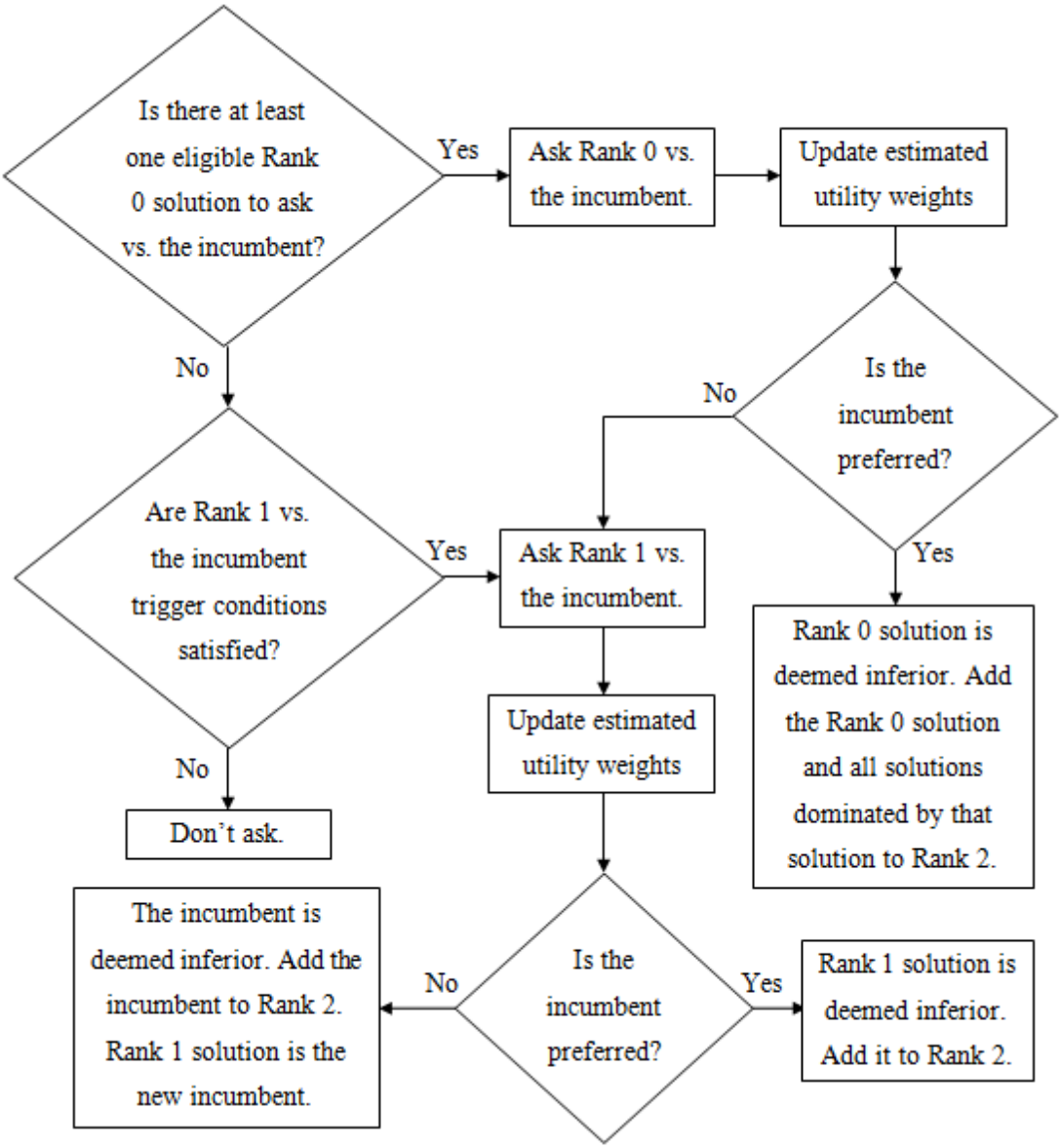

Figure 15. Interactive process flowchart

5.6.6.1 Rank 0 vs. the Incumbent

When the decision maker is asked for preference information by pair-wise comparison, it is possible to avoid asking so many questions for each Rank 1 solution by asking a few Rank 0 solutions against the incumbent.

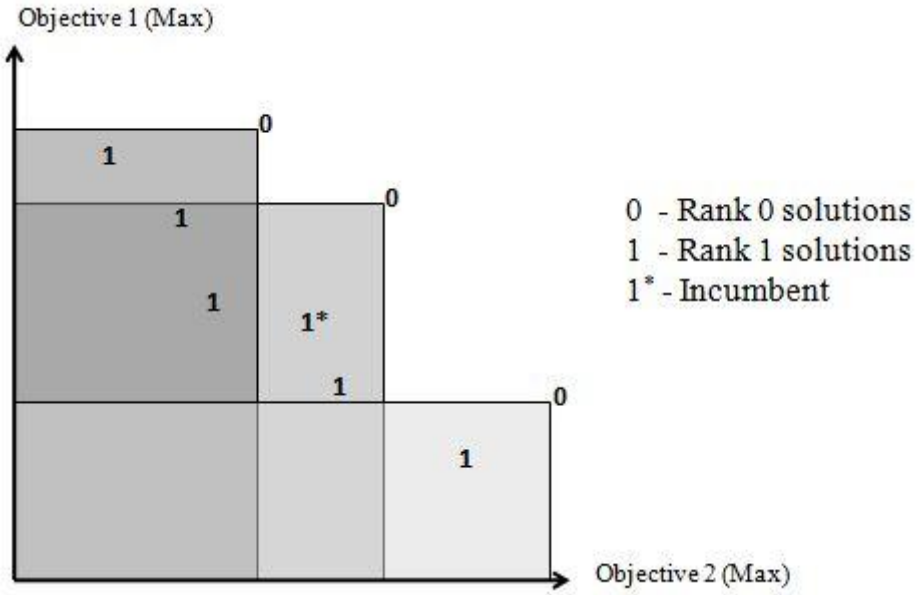

Figure 16. Rank 0 solutions

Eligibility of Rank 0 solutions is mostly subject to design. Though, there are a couple of points when it comes to choosing Rank 0 solutions.

In Figure 16, there are three Rank 0 solutions. The Rank 0 solution in the middle is not eligible by default because it dominates the incumbent.

The Rank 0 solution on the left dominates three Rank 1 solutions and none of which is the incumbent. The solution is eligible and a good candidate. Because if it is not preferred to the incumbent, three Rank 1 alternatives for the incumbent will be removed with only one question.

The Rank 0 solution on the right can also be asked but it only dominates one Rank 1 solution. The only difference between asking that Rank 1 solution is the region that is dominated by the Rank 0 solution but not dominated by the Rank 1 solution, if the decision maker prefers the incumbent to the Rank 0 solution. Especially if there is a maximum on number of questions allowed to ask to the decision maker, the Rank 0 solution on right will be a poor choice.

The methodology of choosing a suitable Rank 0 solution for the specific problem is designed as follows. A Rank 0 solution is a suitable candidate if:

- It does not dominate the incumbent, and
- It dominates at least two Rank 1 solutions.

There might be two suitable Rank 0 solutions. Then the comparison between two Rank 0 solution is as follows. A Rank 0 solution is more suitable than another Rank 0 solution if:

- It dominates more Rank 1 than solutions than the other Rank 0 solutions, or
- It dominates the same number of Rank 1 solutions and have lower estimated utility than the other Rank 0 solution.

Let us assume the solution on the left is asked versus the incumbent to the decision maker. If the decision maker prefers the incumbent to the Rank 0 solution, the Rank 0 solution will be deemed inferior and moved to Rank 2. Also any solution dominated by that solution will be moved to Rank 2 (see Figure 17). Else, a Rank 1 solution will be asked immediately after.

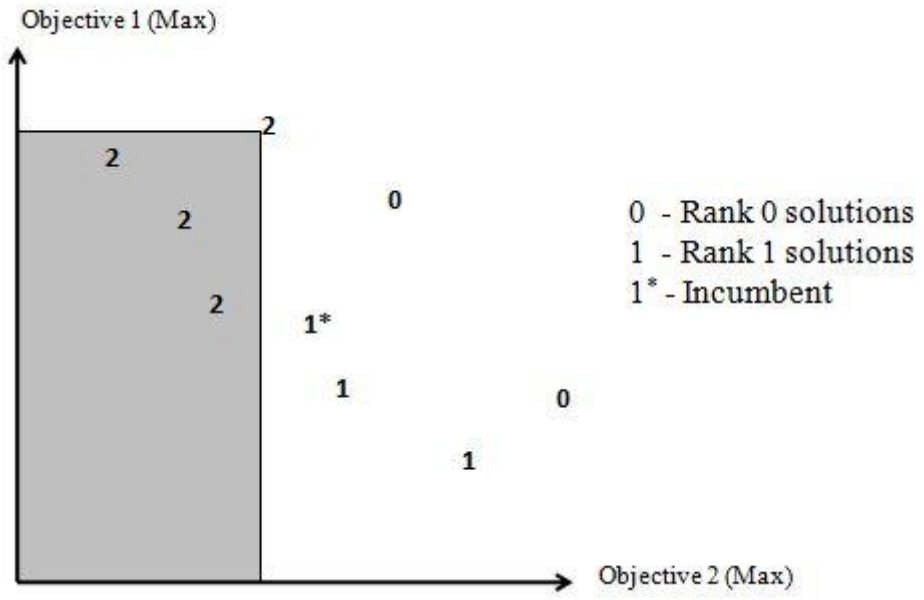

Figure 17. The decision maker prefers the incumbent

5.6.6.2 Rank 1 vs. the Incumbent

There are three ways of asking a Rank 1 solution against the incumbent. If the decision maker is asked for preference information between a Rank 0 solution and the incumbent; and the decision maker prefers the Rank 0 solution, a Rank 1 solution will be picked to ask against the incumbent immediately.

If there are no suitable Rank 0 solutions and trigger conditions for asking a Rank 1 solution is satisfied, a Rank 1 solution will be picked to ask against the incumbent.

If the population convergence is imminent and there are still some Rank 1 solutions left, a Rank solution will be picked to ask against the incumbent. This last action is named *terminal interactive process*.

The last two actions are designed as redundancy measures. Mostly due to triggers that prevent a multitude of questions and lack of suitable Rank 0 solutions, Rank 1 solutions can be required to be asked independently.

The methodology of choosing a suitable Rank 1 for the specific problem is designed as follows. A Rank 1 solution is suitable if: its estimated utility is not higher than %5 above the incumbent's estimated utility value and is higher than other Rank 1 solutions whose estimated utility values are not higher than %5 above the incumbent's estimated utility value.

The rationale for determining a upper bound for a prospective Rank 1 solution is as follows. The incumbent is assumed to be an indicator of the decision maker's preferences. A potential Rank 1 solution's estimated utility value is expected to be higher than the current incumbent's with the updated weights, because estimated utility function is intended to be an approximate to the decision maker's utility function. But, at the same time, the current incumbent is chosen using the same utility function (albeit probably with different weights). Therefore, seeking a solution close to the incumbent would increase the accuracy of the interactive process.

The process of asking the decision maker for preference information between a Rank 1 solution and the incumbent is fairly easy. If the incumbent is preferred the Rank 1 solution will be deemed inferior and moved to Rank 2. Else, the incumbent will be deemed inferior and moved to Rank 2; and the Rank 1 solution will be the new incumbent. The objective of asking Rank 1 solutions against the incumbent is to find strong alternatives that are more preferable than the current incumbent.

5.6.7 Termination

The algorithm proceeds until the conditions for termination trigger are satisfied. The records required to keep track of these criteria are calculated at the end of every generation. The incumbent reported after the termination is the final.

CHAPTER 6

COMPUTATIONAL RESULTS

In this chapter, the success of the algorithm is tested against exact solutions in smaller types of problems and another meta-heuristic method for all types of problems. The aim of this chapter is to show that the proposed algorithm (I-TREA) can produce the exact (or very close) "most preferred" solution in the small size problems and does surpass the performance of a modified version of a meta-heuristic method widely used in the literature (NSGA-II) by using four types of problems (two small, two large) for objective assessment.

Then, difficulties in the implementation and some potential problems are discussed for further development of the proposed evolutionary algorithm (I-TREA).

To test I-TREA's effectiveness 20 different problems are randomly generated from 4 types of problem settings with different sizes. There are 5 different problems for each problem size.

2 types of decision maker utility function are used; chebyshev and linear. There are 5 randomly generated utility functions for each utility function type to be used on the 5 problems for each problem size. In other words, a problem is solved by both utility function types and each of the 5 problems gets different utility weights. Utility weights are generated randomly between 0 and 1 where each function's sum of weights is 1 (i.e. $w_1 + w_2 = 1$).

For each utility function associated with a problem, 5 runs are performed and their averages and standard deviations are reported. There are a total of 200 runs performed using different utility functions, problems and problem sizes to evaluate the performance of I-TREA.

For benchmarking reasons, the same process is performed using a modified version of NSGA-II and their performances are compared. Also, for a single problem size, a simplified version of I-TREA is evaluated to compare with the performance of the original I-TREA.

6.1 Problem Settings

There are four different problem sizes to evaluate the effectiveness and consistency of I-TREA. The parameters of the problems sizes are given in Table 8.

Table 8. Problem Sizes

	1 (50x10)	**2 (100x20)**	**3 (150x30)**	**4 (200x40)**
Demand Points	50	100	150	200
Weights (Interval)	5-50	5-50	5-50	5-50
Facilities	10	20	30	40
Existing Facilities	5	10	15	20
Facility Capacities	250	250	250	275
Critical Cov. Distance	40	40	40	40
Partial Cov. Distance	50	50	50	50
Side Length	200	200	200	200

The first two problem sizes are small enough to be optimally solved in a reasonable amount of time. These problems are solved using CPLEX solver under GAMS by using the epsilon methodology. Problems of the third problem size cannot be optimally solved within 8 hours using CPLEX. Therefore, the third and the fourth problem sizes are denoted as large problems.

6.2 I-TREA Settings

6.2.1 Initial Population

Initial population for the I-TREA and modified NSGA-II for each problem type is generated randomly from a uniform distribution within problem parameters given in Table 8.

6.2.2 Normalization

Both objectives are normalized to simplify the trade-off assessment between objectives and the evaluation of results. The upper and lower bounds for each objective are calculated as follows.

For the number of facilities objective, the lower bound is the least number of facilities with lowest weight to maintain a feasible solution (i.e. all demand points are assigned to facilities and no capacity violation happens). For example, in a problem setting where capacities of facilities are 100 and total demand weight is 250, the minimum required number of facilities should be 3. The upper bound for the facility objective is the maximum value of the objective (i.e. all facilities are open).

The lower bound for the weighted coverage objective is the weighted coverage objective value of the solution with the lowest weighted coverage objective value in the initial population. The upper bound is calculated by solving the LP relaxed version of the problem with single objective (maximize total weighted coverage).

6.2.3 Decision Maker

Originally, a decision maker whose utility function is assumed to be implicit is asked for preference information between two alternative solutions.

For testing purposes, the decision maker is represented by utility functions with randomly assigned weights. There are 2 types of utility functions with 5 different

weight sets for each type. In other words, 10 randomly generated decision maker "personas" are used to test I-TREA for different types of decision preferences.

The first type of utility function is a chebyshev function where Z_1 and Z_2 are the normalized objective values and the maximum weighted deviation from the ideal value of the two objectives is to be minimized. The sum of weights is equal to 1. ($w_1 + w_2 = 1$)

$$U(Z) = Maximum\ of\ \{w_1(1 - Z_1);\ w_2(1 - Z_2)\}$$

The second type of utility function is a linear function, where Z_1 and Z_2 are the normalized objective values and their weighted sum is to be maximized. The sum of weights is equal to 1. ($w_1 + w_2 = 1$)

$$U(Z) = w_1 Z_1 + w_2 Z_2$$

6.2.4 Algorithm Parameters

Although I-TREA's process is simple, there are numerous parameters to be adjusted. In addition, there are triggers. Trigger is a special type of algorithm parameter which determines the conditions for a function to be activated (e.g. asking questions, termination).

There are an abundant number of parameters in I-TREA, some of which are in specially designed mutation functions forced closure and second chance. There are also ordinary parameters like Rank 0 preservation ratio or termination conditions. The determination of parameter values were subject to a vast number of preliminary runs.

The complete list of parameters and their values are given in Appendix B.

6.3 Modified NSGA-II Settings

Basically, it is the constrained NSGA-II (CNSGA-II) described by Deb et al. (2002) in the literature. The modifications are solely based on implementing the interactive process of I-TREA in NSGA-II.

The only difference in the population mechanism of CNSGA-II is to preserve a small proportion of infeasible solutions that are not dominated by any feasible solution (Rank 0 solutions of I-TREA).

The interactive process is slightly different from I-TREA's to not to distort the ranking system of NSGA-II. After preference information, there are no subtractions from the objective space by moving the not-preferred solution and solutions dominated by that solution into inferior ranks. Instead, the portion that is dominated by the not preferred solution is not further included in the interactive process. For example, a not-preferred Rank 1 solution is still a Rank 1 solution in the crossover and the elimination processes. But it is not counted as an eligible solution in the interactive process.

6.4 Simplified I-TREA Settings

To allay the concerns against the convoluted mutation process of I-TREA, a simplified version is also tested. In the simplified version, Second Chance function is not included and the main mutation function is entirely randomized.

In the main mutation function a demand point is randomly assigned to an open and available facility (i.e. demand weight plus the current capacity utilization of that facility should not exceed the facility's capacity). If there are no suitable facilities, the facility with the minimum capacity violation is chosen.

Forced Closure function is retained.

6.5 Evaluation and Computation Settings

6.5.1 Technical Specifications

All the computer runs are performed on the same computer with Intel® Core™ 2 Duo CPU P8400 2.26 GHz, 4 GB of RAM (3 GB RAM usable because of the OS being x86 or 32 bit).

To find exact solutions CPLEX solver is used on GAMS 23.0 because of its popularity as the best solver for MIP problems. Epsilon method is used to generate the non-dominated front.

6.5.2 Performance Metrics

6.5.2.1 Solution Quality

In the literature, there is quite an array of performance metrics to assess the solution quality for the multi-objective evolutionary algorithms. To name some of them, Deb (2001) classifies performance metrics as metrics for convergence, metrics for diversity and metrics for both convergence and diversity; Zitzler et al. (2000) also suggest three goals, minimization of distance of solutions to the true Pareto set, an evenly distribution of solutions and solutions belonging to the non-dominated front should include most of the values for each objective.

Some metrics are devised according to these principles. For example Hyper-volume Metric proposed by Zitzler and Thiele (1998) is used to calculate how much of the solution space is dominated by the efficient solutions of the algorithm. Another method Inverted Generational Distance (IGD) proposed by Bosman and Thiernes (2003) calculates the deviation from the closest efficient solution. There is also another metric that measures the ratio of efficient solutions covered by the non-dominated front of the evolutionary algorithm.

Unfortunately, none of the above methods are applicable in exact form for the specific problem. First of all, the solution of the problem is undertaken by an

interactive method (i.e. directly aimed to find a set of solutions that is of interest to the decision maker). Therefore concentrating on specific parts of the solution space is a priority. This objective diminishes the merits of measuring the spread of solutions.

The methodology used in the algorithm is a subtractive one. It takes portions of the solution space and removes them from interest using infeasible solutions. Therefore, it is not so sensible to calculate how much of the efficient frontier is covered by the population of the evolutionary algorithm.

Interactive methods are only interested in one solution (in some cases, a set of solutions). If a solution is not of interest, it is not so proper to assess a solution's quality by calculating the distance from that point.

There is another method to assess the solutions for the specific problem; the decision maker's utility function. The deviation from the optimal utility value might provide a good assessment of the solution quality.

The main metric used is the absolute deviation from the most preferred solution in terms of decision maker's utility values. It is calculated as follows.

$$U_{dev} = \left| U_{most\ preferred} - U_{best\ found} \right|$$

For the first two problem types, exact non-dominated front is available and it is easy to find the solution that is preferred to all other feasible solutions within the non-dominated front. For the last two problem types, CPLEX is provided with the decision maker's utility function respectively for each individual problem and the bi-objective problem is solved as a single objective problem to find the most preferred solutions. For the problems which CPLEX failed to solve optimally within reasonable time, the approximate results are used as most preferred solutions.

6.5.2.2 Question Limit

There is no strict limit put on the number of questions to be asked to the decision maker. However, to not to consult to the decision maker repeatedly, it is aimed to not to ask more than 15 questions and to keep the average number of questions around 10 questions for each problem type. This is especially hard for the large problem sizes.

6.5.2.3 Runtime

It is aimed to keep the algorithm as fast as possible. Though, the runtime of the CPLEX solutions (which is more than 8 hours per run for problem types 3 and 4) allows I-TREA to not to consider runtime as a strictly binding constraint. The process of I-TREA is extremely rapid.

6.6 Results

6.6.1 Solution Quality

Average absolute utility deviations from the most preferred solutions of I-TREA and modified NSGA-II for all problem sizes are reported in Table 9 for chebyshev utility functions and Table 10 for linear utility functions. For the larger two problem types, the best solutions from the runs are taken as most preferred solution and deviations are calculated accordingly. Because, the larger problem types are not solvable in a reasonable amount of time and the complete non-dominated front is unknown.

Table 9. Absolute utility deviation results for chebyshev utility functions

		50x10	**100x20**	**150x30**	**200x40**
	Std. Dev.	0.0042	0.0008	0.0021	0.0035
I-TREA	**Average**	0.0008	0.0003	0.0021	0.0023
	Max.	0.0209	0.0035	0.0065	0.0132
	Std. Dev.	0.1739	0.1485	0.1040	0.0613
NSGA-II	**Average**	0.1741	0.2192	0.1749	0.1620
	Max.	0.5138	0.5247	0.3440	0.2914

Table 10. Absolute utility deviation results for linear utility functions

		50x10	100x20	150x30	200x40
I-TREA	**Std. Dev.**	0.0078	0.0043	0.0080	0.0372
	Average	0.0016	0.0031	0.0111	0.0296
	Max.	0.0390	0.0158	0.0246	0.1071
NSGA-II	**Std. Dev.**	0.1453	0.1365	0.1376	0.1315
	Average	0.1275	0.1291	0.1370	0.1572
	Max.	0.3671	0.3422	0.3957	0.3401

The performance of I-TREA clearly surpasses the performance of modified NSGA-II in every problem size and utility function type. The average absolute deviation from the most preferred solutions are quite small for I-TREA, though the 'best' solutions found in larger problems (150x30 and 200x40) are yet to be proved to be in the exact non-dominated front of their corresponding problems. Their deviations may be larger than reported in the utility deviation tables. Nevertheless, the results show that the solutions are close to each other in I-TREA even for larger problems. The consistency shows that even though there is room for improvement the algorithm is robust and resilient against increasing problem size.

6.6.1.1 Benchmarking With Simplified I-TREA (I-TREA-S)

A simplified version of I-TREA (I-TREA-S) is tested for a large problem size (150x30). I-TREA-S performs reasonably well when compared to modified NSGA-II's performance, but I-TREA provides more satisfying results. The results given in Table 11 confirm that the complex mutation procedure of I-TREA is a requirement for the specific bi-objective problem.

Table 11. Benchmarking the performance of I-TREA-S

		Chebyshev	Linear
I-TREA	**Std. Dev.**	0.0021	0.0080
	Average	0.0021	0.0111
	Max.	0.0065	0.0246
I-TREA-S	**Std. Dev.**	0.0130	0.0243
	Average	0.0179	0.0457
	Max.	0.0473	0.0903
NSGA-II	**Std. Dev.**	0.1040	0.1376
	Average	0.1749	0.1370
	Max.	0.3440	0.3957

6.6.2 Resource Consumption

The price of high performance is high resource consumption. Although the resource requirements of I-TREA are modest, Table 12 and Table 13 show the need for more resources as the problem size increases.

Table 12. Resource requirements of I-TREA for chebyshev utility functions

		50x10	100x20	150x30	200x40
Runtime (sec.)	**Std. Dev.**	5.34	18.32	59.63	124.92
	Average	14.84	54.48	140.76	298.28
	Max.	28	85	278	610
Generations	**Std. Dev.**	1251.77	1920.64	4380.59	5778.70
	Average	3580.12	6997.48	11128.00	15001.44
	Max.	6747	9516	20900	28362
Questions	**Std. Dev.**	1.21	2.60	2.86	5.21
	Average	5.04	8.12	9.48	11.00
	Max.	7	13	15	28

Table 13. Resource requirements of I-TREA for linear utility functions

		50x10	100x20	150x30	200x40
Runtime (sec.)	**Std. Dev.**	4.08	16.91	58.68	102.91
	Average	13.72	39.72	95.20	220.12
	Max.	21	75	218	365
Generations	**Std. Dev.**	863.51	1959.79	4189.16	5135.86
	Average	3201.60	5272.04	7696.60	11234.92
	Max.	5334	9878	16608	18906
Questions	**Std. Dev.**	1.55	2.33	2.48	2.42
	Average	5.40	6.76	7.68	8.20
	Max.	10	11	14	13

I-TREA requires increasingly more generations and runtime as problem size increases. This is due to the expansion of chromosome size and solution space. The number of questions asked also increases. The largest problem size (200x40) for the chebyshev utility function exceeds the hypothetical limit of 10 questions average. Question averages are lower for linear functions because the estimated utility function and decision maker utility function types correspond (i.e. they are both linear utility functions).

Modified NSGA-II seems to consume fewer resources. This is mostly due to its failure to reach all parts of the objective space and find viable alternatives for the incumbent it finds. Thus, the runs converge prematurely and the performance falters. Table 14 and Table 15 show the resource consumption of the modified NSGA-II.

Table 14. Resource requirements of modified NSGA-II for chebyshev utility functions

		50x10	100x20	150x30	200x40
Runtime (sec.)	**Std. Dev.**	1.50	2.37	3.41	2.26
	Average	10.60	14.84	19.72	23.52
	Max.	13	19	29	29
Generations	**Std. Dev.**	251.97	347.60	328.98	112.84
	Average	2395.68	2400.12	2359.80	2164.24
	Max.	3036	3594	3484	2580
Questions	**Std. Dev.**	1.16	0.87	0.58	0.20
	Average	1.44	1.00	0.40	0.04
	Max.	5	3	2	1

Table 15. Resource requirements of modified NSGA-II for linear utility functions

		50x10	100x20	150x30	200x40
Runtime (sec.)	**Std. Dev.**	1.41	1.70	3.32	4.08
	Average	10.60	14.84	18.92	24.36
	Max.	13	17	28	37
Generations	**Std. Dev.**	321.62	280.33	333.41	376.50
	Average	2367.32	2364.72	2247.12	2279.80
	Max.	3036	2782	3560	3890
Questions	**Std. Dev.**	1.46	1.03	0.54	0.28
	Average	1.96	1.32	0.28	0.08
	Max.	5	4	2	1

The decline in number of questions is highly unorthodox. This is an indicator of modified NSGA-II's failure to find alternative solutions in the objective space. Otherwise there would be alternatives to ask to the decision maker. The main difficulty is to close facilities without a forced closure function. The lack of finding

solutions with higher facility objective values results in premature convergence. Therefore all runs end within about 2000 generations. The runtime increase is due to expanded chromosome size. The results clearly indicate NSGA-II is not equipped to handle the specific problem.

6.6.3 Overall Assessment

In each of the four problem types, I-TREA shows significant success. For the small sized problems, its results match with exact solutions to a good extent. I-TREA clearly claims superiority against modified NSGA-II with the help of its special mutation functions, thus presenting itself as a good alternative to exact solutions when it comes to problems with larger sizes where exact solutions cannot be carried out in a reasonable amount of time (8 hours in this case).

For the larger problem sizes, where solvers fail to deliver in a reasonable amount of time, I-TREA provides quick and robust results (under a few minutes). Although it is not possible to compare with exact non-dominated solutions, the closeness of results to the ‘best’ solutions is an indicator of the consistency of the solution quality.

However, it is expected to deviate from the “most preferred” solution increasingly as the problem size grows. The important thing is the increase is limited and can be tackled by change of parameters (especially triggers). The improvements will come at the cost of runtime most probably, yet the algorithm is fast enough to tolerate this kind of trade-off.

Average number of questions increases at a decreasing rate, mostly due to interactive process gets activated by triggers that are related with population progress (i.e. if the population stops progressing, it may trigger an interactive process event).

Overall, the algorithm's progress is good, yet it will eventually deviate even more from the most preferred solutions at greater problem sizes. Then, there will be a need to change the parameters.

6.7 Weaknesses

Although I-TREA thrives, it has certain weak points that should be discussed. Question timing and increasing average number of questions are the first among them.

It is previously discussed that placing questions at the right time is extremely important. If a question is placed earlier it might curb a potential efficient point's progress before it reached its full potential. It either diminishes the chance of its improvement or leads to ask against a slightly improved version of the defeated solution, which will probably be a waste of decision maker attention. And if a question is asked too late, it would be a waste of generations that passed in the best case. It would probably lead to premature convergence and therefore early termination. A balance should be maintained, albeit it is a delicate balance.

Another issue is the increase in the average number of questions as the problem size grows. It can be expected that the size of the efficient frontier to increase as the problem size increases. Thus, it would eventually lead to an increase in the questions asked. The current setting, although low, is more prone to violation of a virtual maximum question limit of 15. Asking an abundance of questions is undesirable, because one of the objectives of the interactive system is to steer the algorithm to the decision maker's desired regions in the solution space without disturbing the decision maker too much. The situation is mostly circumvented by accepting the incumbent as preference information (i.e. picking incumbent alternatives closet to the incumbent's estimated utility).

To tackle the first and the second issues at the same time, question asking is entitled to several conditions related with the progress of the population and delayed as further as possible. Though, the one-size-fits-all solution might fail at some point at

the increased levels of problem size. The trigger conditions might render useless as they are bounded by the termination conditions. Another trigger system can be used that is more sensitive to problem size changes.

Another method might be changing the structure of the interactive process. Instead of asking pair-wise comparisons, DM might be asked to assess or rank several solutions at a time.

The midpoint approach can be faulty at times. Using a linear utility function might cause the algorithm to overlook convex dominated solutions. That means extra questions in the best case. And resolution of the conflicting decisions from the decision maker is not very subtle. One irrational decision might wipe all the previous preference information.

Although I-TREA's process is pretty simplified and its performance is improved with infeasible solutions, diversity might still be an issue. Its deficiency is not felt by I-TREA, because its aim is to find interesting regions in the solution space and to disregard the rest of the solution space. Anyway, for I-TREA, most of the internal benchmark is done with estimated utility values. Though, without the interactive process, diversity will gain weigh again and should be considered more seriously.

The weakness in the chromosome representation discussed in Section 5.2 is specific to this problem. Therefore the algorithm's mutation needed to be augmented by other functions.

CHAPTER 7

CONCLUSION

In this thesis, a bi-objective capacitated facility location problem is studied with the objectives; the maximization of the total weighted coverage and the minimization of number of facilities to be opened. Presence of a singular critical coverage value is claimed to be unrealistic and the assumption of there are no previous installation within the region (i.e. no existing facilities) is dismissed. Therefore, partial coverage concept is introduced in the problem and a distinction is assumed between the 'existing' and 'new' facilities.

The problem, due to its combinatorial nature, is reported to be unsolvable in polynomial time. Therefore an interactive evolutionary algorithm is proposed (I-TREA). I-TREA benefits from infeasible solutions that are not dominated by the feasible solutions both for the progress of the algorithm and as trade-off alternatives like partition ideals explained in Section 3.4. I-TREA is tailored with specific mutation functions to enhance its performance and to progress quickly.

The algorithm is tested against exact solutions in solvable problem sizes and a modified version of NSGA-II in all problem sizes. It performed reasonably well in the small sized problems as it returned exact or close solutions. I-TREA clearly outperformed the modified NSGA-II in every problem and every problem size. The algorithm proved to be robust and efficient for this specific problem.

7.1 Future Studies

There is still much room for improvement. With so many parameters to fiddle with, the algorithm can be improved further. I-TREA is only tested against bi-objective problems. It can also be tested against problems with three or more objectives for its robustness against increased number of objectives. The algorithm is also overly elitist. Further diversity measures can be devised at some point in the future.

REFERENCES

Balıbek E., Köksalan M., (2010) A multi-objective multi-period stochastic programming model for public debt management. *European Journal of Operational Research*, 205, 1, 205-217.

Batta R., Huang W.V., (1989) On the synthesis of advertising and relocation decisions for facility. *Computers and Industrial Engineering*, 16, 1, 179–187.

Benayoun R, de Montgolfier J, Tergny J, Laritchev O (1971). Linear programming with multiple objective functions: Step method (STEM). *Math Program*, 1, 366-375.

Berman O., Krass D., (2002) The generalized maximal covering location problem. *Computers and Operations Research*, 29, 563–591.

Berman O., Krass D., Drezner Z., (2003) The gradual covering decay location problem on a network, *European Journal of Operational Research*, 151, 474–480.

Berman O., Drezner Z., Krass D. (2010) Generalized Coverage: New Developments in Covering Location Models. *Computers and Operations Research*, 37, 10, 1675-1687.

Brandeau M.L., Chiu S.S., (1989) An Overview of Representative Problems in Location Research. Management Science, 35, 6, 645-674.

Brimberg, J., ReVelle, C. (1998). A bi-objective plant location problem: cost vs. demand served. *Location Sci*ence, 6: 121–135.

Chung C.H., Schilling D.A., Carbone R. (1983) The Capacitated Maximal Covering Problem: A Heuristic Solution. *Modelling and Simulation* 14, 1383-1388.

Church, C., ReVelle, S. (1974) The Maximal Covering Location Problem. *Papers Of The Regional Sci. Assoc.*, 32, 101-118.

Church, C., ReVelle, S. (1976) Theoretical and computational links between the p-median, location set-covering and the maximal covering location problem. *Geographical Analysis*, 8, 406-415.

Current, J., Storbeck J. (1988) Capacitated Covering Models. *Environment And Planning B.*, 15, 153-164.

Daskin M.S., Hogan K., Revelle C. (1988) Integration of Multiple, Excess, Backup, and Expected Covering Models. *Environment And Planning B: Planning And Design* 15, 15–35.

Daskin M.S., Stern E.H., (1981) A Hierarchical Objective Set Covering Model For Emergency Medical Service Vehicle Deployment. *Transportation Science* 15, 2, 137-152.

Deb K. (2001) Multiobjective Optimization Using Evolutionary Algorithms. *John Wiley & Sons*.

Deb K., Kumar A. (2007) Interactive evolutionary multi-objective optimization and decision-making using reference direction method. *GECCO'07 - Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*,. London, England.

Deb K. Kumar A. (2007a) Light beam search based multi-objective optimization using evolutionary algorithms. *Congress on Evolutionary Computation (CEC)*. Singapore.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6, 181–197.

Deb, K.; Sinha, A.; Korhonen, P.J.; Wallenius, J. (2010) An Interactive Evolutionary Multiobjective Optimization Method Based on Progressively Approximated Value Functions. *IEEE Transactions on Evolutionary Computation*, 14, 5, 723 – 739.

Dell R. (1998) Optimizing army base realignment and closure. *Interfaces*, 28, 1–18.

Doerner K.F., Gutjahr W.J., Nolz P.C. (2009) Multi-criteria location planning for public facilities in tsunami-prone coastal areas. *OR Spectrum*, 31, 3, 651–678.

Drezner T., Drezner Z., Goldstein Z. (2010) A stochastic gradual cover location problem. *Naval Research Logistics*, 57, 4, 367-372.

Drezner Z., Wesolowsky G.O., (1991) Facility location when demand is time dependent. *Naval Research Logistics*, 38, 763-777.

Drezner Z., Wesolowsky G.O., Drezner T., (2004) The gradual covering problem. *Naval Research Logistics*, 51, 841–855.

Eiselt H.A., Marianov V., (2009) Gradual location set covering with service quality. *Socio-Economic Planning Sciences*, 43, 121–130.

Farhan B., Murray A.T., (2008) Siting park-and-ride facilities using a multi-objective spatial optimization model. *Computers and Operations Research* 35, 445–456.

Farahani R. Z., Z. Drezner, and N. Asgari (2009) Single Facility Location and Relocation Problem with Time Dependent Weights and Discrete Planning Horizon. *Annals of Operations Research*, 167, 353-368.

Fowler J.W., Gel E.S., Köksalan M., Korhonen P., Marquis J.L., Wallenius J., (2010) Interactive evolutionary multi-objective optimization for quasi-concave preference functions, *European Journal of Operational Research*, 206, 2, 417-425.

Frank M., Wolfe P. (1956) An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3, pp. 95-110.

Haimes Y.Y., Lasdon L.S., and Wismer D.A. (1971) On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics* 1, 3, 296-297.

Hakimi S.L. (1964) Optimal Locations of Switching Centers and the Absolute Centers and Medians of a Graph. *Operational Research*, 12, 450-459.

Harewood S.I. (2002). Emergency ambulance deployment in Barbados: a multi-objective approach. *Journal of the Operational Research Society*, 53, 185–192.

Hogan, K., Revelle, C. (1986) Concepts and Applications Of Backup Coverage. *Management Science*, 32, 1434-1444

Jayaraman, V. (1999) A Multi-Objective Logistics Model for a Capacitated Service Facility Problem. *International Journal of Physical Distribution & Logistics*, 29, 1, 65–81.

Karasakal E., Silav A. (2010) A Multi-Objective Genetic Algorithm for a Bi-Objective Facility Location Problem with Partial Coverage. *Technical Report 10-05*, IE, METU, Ankara.

Karasakal O., Karasakal E. (2004) A maximal covering location model in the presence of partial coverage. *Computers and Operations Research*, 31, 15–26.

Knowles J.D., Corne D.W. (2000) Approximating the non-dominated front using the Pareto archived evolution strategy. *Evolutionary Computation Journal,* 8, 2, 149-172.

Köksalan M, Karasakal E. (2006) An interactive approach for multiobjective decision making. *Journal of the Operational Research Society*, 57, 532-540.

Köksalan M, Sagala P.N.S. (1995) Interactive appreaches for discrete alternative multiple criteria decision making with monotone utility functions. *Management Science,* 41, 1158-1711.

Korhonen P., Laakso J. (1986) A visual interactive method for solving the multiple criteria problem. *European Journal of Operational Research,* 24, 277-287.

Liu N., Huang B., Xiaohong P., (2005) Using the Ant Algorithm to Derive Pareto Fronts for Multiobjective Siting of Emergency Service Facilities. *Journal of the Transportation Research Board*, 1935, 120-129.

Melachrinoudis, E., Min, H., (2000) Dynamic relocation and phase-out of a two-echelon, hybrid plant and warehousing facility: A multiple objective approach. *European Journal of Operational Research,* 123, 1, 1–15.

Melachrinoudis E., Min H., Messac A., (2000) The relocation of a manufacturing/distribution facility from supply chain perspectives: A physical programming approach. *Applications of Management Science: Multi-criteria Applications,* 1st ed., vol. 10, K.D. Lawrence, G.R. Reeves, and R.K. Klimberg Ed. Amsterdam: JAI Press, 15-39.

Miettinen K., Eskelinen P., Ruiz F., Luque M. (2010) NAUTILUS method: An interactive technique in multiobjective optimization based on the nadir point. *European Journal of Operational Research*, 206, 2, 426-434.

Min H. (1988) Dynamic expansion and relocation of capacitated public facilities: A multi-objective approach. *Computers and Operations Research,* 15, 3, 243-252.

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

Pfiffer J., Golle U., Rothlauf F. (2008) Reference Point Based Multi-Objective Evolutionary Algorithms for Group Decisions. *GECCO'08 - Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, Atlanta, Georgia, USA.

Phelps S.P., Köksalan M. (2003) An interactive evolutionary metaheuristic for multiobjective combinatorial optimization. *Management Science*, 49, 12, 1726-1738.

Pirkul H., Schilling D.A., (1988) The Siting of Emergency Service Facilities with Workload Capacities and Backup Service. *Management Science*, 34, 7, 896-908.

Pirkul H., Schilling D.A. (1991) The Maximal Covering Location Problem with Capacities on Total Workload. *Management Science*, 37, 2, 233–48.

Ray T., Singh H.K., Isaacs A., Smith W. (2009) Infeasibility Driven Evolutionary Algorithm for Constrained Optimization. *Studies in Computational Intelligence*, 198, 145-165.

ReVelle C.S., Marks D., Liebman J.C., (1970) An Analysis of Private and Public Sector Location Models. Management Science, 16, 11, 692-707.

ReVelle C.S., Bigman D., Schilling D., Cohon J., Church R., (1977) Facility location: A review of context-free and EMS models. *Health Services Research*, 12, 1129–1146.

ReVelle, C.S., Laporte, G., (1996) The plant location problem: New models and research prospects. *Operations Research*, 44, 6, 864–874.

ReVelle, C.S., Serra. D. (1991) The Maximum Capture Problem Including Relocation. *Information and Operations Research*, 29, 130–38.

Ross G.T. and Soland R.M., (1977) Modeling facility location problems as generalized assignment problems, *Management Science* 24, 3 ,345–357.

Ross, G.T., Soland, R.M., (1980) A multicriteria approach to location of public facilities. *European Journal of Operational Research* 4, 307–321.

Steuer R.E., (1977) An interactive multiple criteria linear programming procedure, TIMS Studies. *Management Science*, 6, 225-239.

Tansel B.C., Francis R.L., Lowe T.J., (1983) Location on Networks: A Survey. Part I: The P-Center and P-Median Problems. *Management Science*, 29, 4, 482-497.

Thiele L., Miettinen K, Korhonen P.J., Molina J. (2009) A Preference-Based Evolutionary Algorithm for Multi-Objective Optimization. *Evolutionary Computation*, 17, 3, 411-436.

Toregas, C., Swain R., ReVelle. C., (1971) The Location of Emergency Service Facilities. *Operational Research*, 19, 1363-1373.

VanRoy T.J., Erlenkotter D. (1982) A dual-based procedure for dynamic facility location, *Management Science*, 28, 10, 1091-1105.

Villegas, J.G., Palacios, F., Medaglia, A.L. (2006) Solution methods for the bi-objective (cost-coverage) unconstrained facility location problem with an illustrative example. *Annals of Operational Research*, 147, 109–141.

Wang Q., Batta R., Bhadury J., Rump C.M. (2003) Budget constrained location problem with opening and closing of facilities. *Computers and Operations Research*, 30, 2047–2069.

Weber, A., (1909) Uber Den Standort Der Industrien; Translated As Alfred Weber's Theory Of The Location Of Industries. *University Of Chicago*, 1929.

Wesolowsky G.O. (1973) Dynamic facility location. *Management Science*, 19, 11, 1241-1248.

Wesolowsky G.O., Truscott W.G. (1976) The multiperiod location-allocation problem with relocation of facilities. *Management Science*, 22, 1, 57-65.

Zionts S., Wallenius J. (1976) An interactive programming method for solving the multiple criteria problem. *Management Science*, 22, 652-663.

Zionts S., Wallenius J. (1983) An interactive multiple objective linear programming for a class of underlying nonlinear utility functions. *Management Science*, 29, 519-529.

Zitzler, E., Deb, K., Thiele, L. (2000) Comparison of multiobjective evolutionaryalgorithms: Empirical results. *Evolutionary Computation,* 8, 2, 173-195.

Zitzler E., Laumanns M., Thiele L. (2001) SPEA 2: Improving the Strength Pareto Evolutionary Algorithm. *TIK Report No: 103. Computer Engineering and Networks Laboratory (TIK),* Swiss Federal Institute of Technology, Zurich.

Zitzler, E., Thiele, L., (1998) Multiobjective Optimization Using EvolutionaryAlgorithms - A Comparative Case Study. *Parallel Problem Solving From Nature,* 1498, 292-301.

Zitzler E., Thiele L. (1998a) An evolutionary algorithm for multiobjective optimization: The strength pareto approach. *Technical Report 43, Computer Engineering and Networks Laboratory (TIK),* Swiss Federal Institute of Technology, Zurich.

APPENDIX A

PSEUDO CODE OF I-TREA

Initialization()

Fill the population with initial solutions.

Classification()

For each new member of the population:

*Add to Rank 0 if it is infeasible and not dominated by any other Rank 1 or defeated solutions.

*Add to Rank 1 if it's feasible and not dominated by any other Rank 1 or defeated solutions. Add any Rank 1 and 0 solution dominated by the newly added Rank 1 solution to Rank 2. If it dominates the incumbent, it is the new incumbent.

*Add to Rank 2 if it does not belong to Rank 0 or Rank 1.

Denote the Rank 1 solution with the highest estimated utility value as "incumbent".

For each new generation until termination:

Crossover()

Fill the mating pool by "crowded tournament selection".

One point, random point crossover.

Mutation()

If triggered, Forced Closure ()

Calculate facility potentials for open facilities.

Close a facility using potentials as probability values.

If closed facility is a ‘new facility’ and there is at least one closed ‘existing’ facility:

Calculate facility potentials for closed ‘existing’ facilities.

Open an existing facility using potentials as probability values.

If triggered, Inefficiency Reset()

Revoke all demand point assignments if they return zero coverage or less coverage than they could be by assignment to another open facility regardless of the capacity violation.

For each gene that has a value;

Revoke its assignment by chance, whose probability is determined by the mutation probability.

For each open facility;

Calculate facility potential scores.

For each demand point;

Calculate demand points total scores and maximum scores.

While facility potential scores > 0;

Choose the facility with the highest potential facility score.

Calculate the probabilities of the demand points to be assigned to that facility with regard to maximum scores and total scores.

Assign a demand point to the chosen facility.

Update capacity information for that facility.

Update facility potential scores.

Update demand point maximum and total scores.

Assign the remaining demand points to facilities without any specific order.

Classification()

Elimination(Continue until population level is back to normal)

Eliminate the duplicates.

Eliminate Rank 2 solutions according to domination principle. If a Rank 2 solution is dominated by another Rank 2 solution, it is eliminated.

Eliminate Rank 0 solutions according to both domination and capacity violation. If a Rank 0 solution is dominated by another Rank 0 solution and have greater capacity violation than its match, it is eliminated. Stop eliminating Rank 0 solutions if number of Rank 0 solutions has reached to Rank 0 preservation limit.

Eliminate Rank 0 solutions according to radial distances. Start eliminating from the closest Rank 0 solution. There will always be two Rank 0 solutions, eliminate the one with the higher capacity violation. Stop eliminating Rank 0 solutions if number of Rank 0 solutions has reached to Rank 0 preservation limit.

Eliminate Rank 1 solutions that return the same objective values (twins).

Eliminate Rank 2 solutions according to crowding distances.

Eliminate Rank 1 solutions according to crowding distances.

Interactive Process()

If trigger conditions for asking are satisfied;

Try to find a suitable Rank 0 solution to ask vs. the incumbent.

If there is one, ask the DM Rank 0 vs. the incumbent;

If the incumbent is preferred, Rank 0 solution is a defeated solution. That solution and all solutions that are dominated by that solution are moved to Rank 2.

Else, find a suitable Rank 1 solution to ask vs. the incumbent.

If the incumbent is preferred, Rank 1 solution is now a defeated solution and moved to Rank 2.

Else, incumbent is now a defeated solution and moved to Rank 2. Rank 1 solution is the new incumbent.

Else, if trigger conditions for asking Rank 1 solutions are satisfied;

If there is a suitable Rank 1 solution to ask vs. the incumbent.

If the incumbent is preferred, Rank 1 solution is now a defeated solution and moved to Rank 2.

Else, incumbent is now a defeated solution and moved to Rank 2. Rank 1 solution is the new incumbent.

Calculate population averages, check for termination conditions.

If termination conditions are satisfied, then terminate the run.

APPENDIX B

LIST OF I-TREA PARAMETERS

- Population size is 100.
- The preservation ratio of Rank 0 solutions is %10.
- Mutation probability is %5.
- Initial estimated utility weights are equal (i.e. 0.5 and 0.5).
- To ask a Rank 0 solution against the incumbent, the conditions are as follows.
  - There should be at least a Rank 0 solution.
  - 100 generations should pass after the last question.
  - And;
    - 150 generations should pass after the last non-dominated front change.
    - 1000 generations should pass after the last incumbent change.

      or
    - Number of questions asked should be less than 2.
    - 500 generations should pass after the last incumbent change.
- A Rank 0 solution is eligible to ask if it dominates at least two Rank 1 solutions none of which is the incumbent.
- To ask a Rank 1 solution against the incumbent, the conditions are as follows.
  - There should be no eligible Rank 0 solution to ask.
  - There should be at least one Rank 1 solution which is not the incumbent.
  - And;
    - 250 generations should pass after the last non-dominated front change.

    - 200 generations should pass after the last question.

      or

    - Number of questions asked should be less than 2.
    - 500 generations should pass after the last incumbent change.
- The trigger conditions for terminal interactive process are as follows.
  - 200 generations should pass after the last non-dominated front change.
  - 1500 generations should pass after the last incumbent change.
- A maximum of 3 Rank 1 solutions can be asked against the incumbent during the terminal interactive process.
- For a Rank 1 solution to be eligible to ask against the incumbent, its estimated utility value should not exceed %5 above the incumbent's estimated utility value.
- Forced closure function is triggered if 20 generations have passed after the last non-dominated front change. Force closure activates once in every 5 generations after being triggered.
- The bonus for highest coverage values in the forced closure function is %50.
- Second chance function is triggered if 10 generations have passed after the last non-dominated front change. Force closure activates once in every 2 generations after being triggered.
- Termination conditions are as follows.
  - 200 generations should pass after the last non-dominated front change.
  - 1500 generations should pass after the last incumbent change.

APPENDIX C

DETAILED RUN RESULTS

Table 16. Problem type 1 I-TREA linear utility function results

All Problems (Linear Utility Function)				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	7	0.0000	1699	3
Maximum	21	0.0390	5334	10
Std.Dev.	4.08	0.0078	863.51	1.55
Average	13.72	0.0016	3201.60	5.40
Problem Set 1				
Minimum	12	0.0000	3064	5
Maximum	21	0.0000	5334	7
Std.Dev.	3.42	0.0000	852.28	0.84
Average	16.20	0.0000	3939.80	6.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	21	0.0000	5334	7
2	17	0.0000	3936	7
3	12	0.0000	3064	5
4	17	0.0000	3859	6
5	14	0.0000	3506	6

Table 16. Problem type 1 I-TREA linear utility function results (cont.)

Problem Set 2				
Minimum	11	0.0000	2980	3
Maximum	16	0.0390	4142	5
Std.Dev.	1.95	0.0175	423.90	0.89
Average	14.40	0.0078	3645.40	3.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	16	0.0000	4142	3
2	15	0.0000	3789	3
3	11	0.0000	2980	5
4	15	0.0390	3593	3
5	15	0.0000	3723	3
Problem Set 3				
Minimum	7	0.0000	1699	6
Maximum	11	0.0000	2796	10
Std.Dev.	1.64	0.0000	431.19	1.64
Average	8.20	0.0000	2043.00	7.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	7	0.0000	1942	6
2	8	0.0000	1870	7
3	7	0.0000	1908	7
4	11	0.0000	2796	10
5	8	0.0000	1699	6
Problem Set 4				
Minimum	10	0.0000	2334	5
Maximum	15	0.0000	3590	6
Std.Dev.	2.30	0.0000	594.60	0.45
Average	12.40	0.0000	3022.80	5.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	14	0.0000	3384	5
2	13	0.0000	3382	5
3	10	0.0000	2334	5
4	15	0.0000	3590	6
5	10	0.0000	2424	5

Table 16. Problem type 1 I-TREA linear utility function results (cont.)

Problem Set 5				
Minimum	12	0.0000	2615	5
Maximum	20	0.0000	4092	5
Std.Dev.	3.29	0.0000	594.00	0.00
Average	17.40	0.0000	3357.00	5.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	20	0.0000	2907	5
2	12	0.0000	2615	5
3	20	0.0000	4092	5
4	18	0.0000	3668	5
5	17	0.0000	3503	5

Table 17. Problem type 1 I-TREA chebyshev utility function results

All Problems (Chebyshev Utility Function)				
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	8	0.0000	2022	2
Maximum	28	0.0209	6747	7
Std.Dev.	5.34	0.0042	1251.77	1.21
Average	14.84	0.0008	3580.12	5.04
Problem Set 1				
Minimum	8	0.0000	2022	5
Maximum	13	0.0000	3456	7
Std.Dev.	2.49	0.0000	704.78	1.00
Average	11.20	0.0000	2846.80	6.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	13	0.0000	3220	6
2	8	0.0000	2142	5
3	13	0.0000	3394	7
4	9	0.0000	2022	5
5	13	0.0000	3456	7

Table 17. Problem type 1 I-TREA chebyshev utility function results (cont.)

Problem Set 2				
Minimum	13	0.0000	3043	2
Maximum	28	0.0000	6747	6
Std.Dev.	5.76	0.0000	1448.35	1.82
Average	19.80	0.0000	4756.80	4.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	16	0.0000	3744	6
2	28	0.0000	6747	3
3	22	0.0000	5438	6
4	13	0.0000	3043	5
5	20	0.0000	4812	2
Problem Set 3				
Minimum	8	0.0000	2283	4
Maximum	16	0.0000	4553	6
Std.Dev.	2.88	0.0000	844.11	0.71
Average	12.40	0.0000	3303.60	5.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	8	0.0000	2283	5
2	12	0.0000	2999	4
3	16	0.0000	4553	6
4	13	0.0000	3617	5
5	13	0.0000	3066	5
Problem Set 4				
Minimum	14	0.0000	3296	3
Maximum	28	0.0209	6085	7
Std.Dev.	5.83	0.0093	1150.44	1.48
Average	19.00	0.0042	4435.20	4.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	21	0.0209	4865	3
2	18	0.0000	4534	5
3	14	0.0000	3396	5
4	28	0.0000	6085	4
5	14	0.0000	3296	7

Table 17. Problem type 1 I-TREA chebyshev utility function results (cont.)

Problem Set 5				
Minimum	10	0.0000	2156	5
Maximum	13	0.0000	2722	5
Std.Dev.	1.10	0.0000	230.88	0.00
Average	11.80	0.0000	2558.20	5.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	12	0.0000	2722	5
2	10	0.0000	2156	5
3	12	0.0000	2631	5
4	13	0.0000	2694	5
5	12	0.0000	2588	5

Table 18. Problem Type 2 I-TREA linear utility function results

Problems Average				
	Time	**DMDev (abs)**	**Generation**	**Questions**
Minimum	13	0.0000	2344	2
Maximum	75	0.0158	9878	11
Std.Dev.	16.91	0.0043	1959.79	2.33
Average	39.72	0.0031	5272.04	6.76
Problem Set 1				
Minimum	24	0.0003	3562	2
Maximum	53	0.0035	7256	9
Std.Dev.	12.62	0.0013	1370.73	2.88
Average	39.80	0.0014	5297.20	4.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	53	0.0003	7256	9
2	32	0.0014	4638	2
3	52	0.0008	5778	4
4	24	0.0035	3562	2
5	38	0.0008	5252	5

Table 18. Problem Type 2 I-TREA linear utility function results (cont.)

Problem Set 2				
Minimum	17	0.0000	2344	4
Maximum	46	0.0117	5384	6
Std.Dev.	11.10	0.0058	1162.04	0.89
Average	30.40	0.0047	3717.40	5.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	17	0.0014	2344	6
2	46	0.0000	5384	6
3	32	0.0001	4133	6
4	34	0.0117	3759	6
5	23	0.0102	2967	4
Problem Set 3				
Minimum	13	0.0000	2492	6
Maximum	37	0.0158	6022	9
Std.Dev.	8.79	0.0067	1311.02	1.30
Average	24.20	0.0039	4004.80	7.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	13	0.0012	2492	6
2	37	0.0012	6022	9
3	27	0.0000	4352	8
4	23	0.0158	3710	7
5	21	0.0012	3448	6
Problem Set 4				
Minimum	29	0.0011	3978	6
Maximum	69	0.0086	8002	11
Std.Dev.	14.53	0.0029	1512.12	2.07
Average	52.20	0.0054	6264.00	8.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	69	0.0041	8002	7
2	29	0.0086	3978	6
3	57	0.0065	7172	11
4	53	0.0011	6012	9
5	53	0.0067	6156	10

Table 18. Problem Type 2 I-TREA linear utility function results (cont.)

Problem Set 5				
Minimum	27	0.0000	3844	6
Maximum	75	0.0003	9878	10
Std.Dev.	18.73	0.0002	2340.83	1.58
Average	52.00	0.0001	7076.80	8.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	75	0.0003	9878	10
2	61	0.0000	8338	9
3	27	0.0003	3844	6
4	57	0.0000	7580	7
5	40	0.0000	5744	8

Table 19. Problem Type 2 I-TREA chebyshev utility function results

Problems Average				
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	16	0.0000	2380	3
Maximum	85	0.0035	9516	13
Std.Dev.	18.32	0.0008	1920.64	2.60
Average	54.48	0.0003	6997.48	8.12
Problem Set 1				
Minimum	16	0.0000	2380	3
Maximum	58	0.0000	7896	8
Std.Dev.	18.15	0.0000	2401.77	2.07
Average	36.40	0.0000	5137.60	5.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	23	0.0000	3412	4
2	16	0.0000	2380	3
3	33	0.0000	4697	6
4	58	0.0000	7896	8
5	52	0.0000	7303	7

Table 19. Problem Type 2 I-TREA chebyshev utility function results (cont.)

Problem Set 2				
Minimum	60	0.0001	6706	5
Maximum	85	0.0020	9516	10
Std.Dev.	9.20	0.0008	1046.10	2.00
Average	73.20	0.0006	8163.20	8.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	74	0.0004	8354	9
2	70	0.0020	7682	5
3	85	0.0002	9516	10
4	60	0.0001	6706	7
5	77	0.0004	8558	9
Problem Set 3				
Minimum	19	0.0000	3084	5
Maximum	58	0.0000	8734	12
Std.Dev.	14.97	0.0000	2136.00	3.11
Average	44.00	0.0000	6701.20	8.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	19	0.0000	3084	5
2	43	0.0000	6868	6
3	58	0.0000	8734	12
4	49	0.0000	7388	11
5	51	0.0000	7432	10
Problem Set 4				
Minimum	46	0.0000	5612	6
Maximum	76	0.0035	8532	13
Std.Dev.	14.15	0.0015	1426.07	2.59
Average	63.80	0.0008	7215.00	9.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	46	0.0035	5612	8
2	72	0.0000	8103	10
3	74	0.0002	8110	9
4	51	0.0000	5718	6
5	76	0.0005	8532	13

Table 19. Problem Type 2 I-TREA chebyshev utility function results (cont.)

Problem Set 5				
Minimum	39	0.0000	5436	6
Maximum	62	0.0000	8662	12
Std.Dev.	9.14	0.0000	1317.60	2.24
Average	55.00	0.0000	7770.40	9.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	62	0.0000	8662	6
2	58	0.0000	8202	10
3	57	0.0000	8242	9
4	39	0.0000	5436	8
5	59	0.0000	8310	12

Table 20. Problem Type 3 I-TREA linear utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	25	0.0000	2458	5
Maximum	218	0.0246	16608	14
Std.Dev.	58.68	0.0080	4189.16	2.48
Average	95.20	0.0111	7696.60	7.68
Problem Set 1				
Minimum	109	0.0163	8544	8
Maximum	218	0.0195	15408	12
Std.Dev.	44.12	0.0015	2721.59	1.58
Average	152.80	0.0178	11287.60	10.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	137	0.0176	10694	11
2	218	0.0163	15408	12
3	176	0.0166	12390	10
4	124	0.0191	9402	8
5	109	0.0195	8544	9

Table 20. Problem Type 3 I-TREA linear utility function results (cont.)

Problem Set 2				
Minimum	37	0.0000	2811	5
Maximum	47	0.0007	4000	6
Std.Dev.	3.71	0.0003	478.70	0.55
Average	42.40	0.0001	3636.40	5.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	44	0.0000	4000	6
2	47	0.0007	3928	5
3	37	0.0000	2811	5
4	41	0.0000	3679	6
5	43	0.0000	3764	5
Problem Set 3				
Minimum	45	0.0150	4366	6
Maximum	99	0.0246	8524	10
Std.Dev.	24.90	0.0038	1965.69	1.67
Average	71.20	0.0196	6429.20	7.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	73	0.0217	6888	6
2	45	0.0246	4366	6
3	99	0.0198	7984	8
4	47	0.0171	4384	7
5	92	0.0150	8524	10
Problem Set 4				
Minimum	25	0.0043	2458	5
Maximum	185	0.0156	16608	14
Std.Dev.	67.28	0.0050	5903.22	3.83
Average	66.00	0.0124	6169.40	8.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	185	0.0043	16608	14
2	25	0.0156	2458	5
3	28	0.0156	2946	5
4	43	0.0108	4168	7
5	49	0.0156	4667	10

Table 20. Problem Type 3 I-TREA linear utility function results (cont.)

Problem Set 5				
Minimum	104	0.0044	8572	6
Maximum	192	0.0077	13436	10
Std.Dev.	34.70	0.0014	1995.49	1.67
Average	143.60	0.0054	10960.40	7.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	104	0.0044	8572	6
2	117	0.0056	9322	6
3	157	0.0049	12058	10
4	148	0.0044	11414	8
5	192	0.0077	13436	7

Table 21. Problem Type 3 I-TREA chebyshev utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	48	0.0000	4326	4
Maximum	278	0.0065	20900	15
Std.Dev.	59.63	0.0021	4380.59	2.86
Average	140.76	0.0021	11128	9.48
Problem Set 1				
Minimum	121	0.0044	8870	6
Maximum	205	0.0044	14940	11
Std.Dev.	33.90	0.0000	2492.84	1.92
Average	163.20	0.0044	12044.8	8.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	151	0.0044	11230	9
2	190	0.0044	14194	10
3	205	0.0044	14940	11
4	149	0.0044	10990	6
5	121	0.0044	8870	8

Table 21. Problem Type 3 I-TREA chebyshev utility function results

Problem Set 2				
Minimum	57	0.0014	4372	4
Maximum	243	0.0054	17058	10
Std.Dev.	72.16	0.0014	5056.38	2.12
Average	163.00	0.0034	11930.4	7.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	179	0.0036	13068	7
2	130	0.0035	9686	7
3	243	0.0031	17058	10
4	206	0.0014	15468	7
5	57	0.0054	4372	4
Problem Set 3				
Minimum	69	0.0000	6478	8
Maximum	278	0.0000	20900	15
Std.Dev.	80.28	0.0000	5599.19	2.77
Average	158.40	0.0000	12622.8	11.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	278	0.0000	20900	15
2	103	0.0000	8498	8
3	69	0.0000	6478	9
4	178	0.0000	13878	12
5	164	0.0000	13360	12
Problem Set 4				
Minimum	75	0.0000	6472	9
Maximum	184	0.0065	17272	14
Std.Dev.	45.42	0.0028	4371.19	2.07
Average	137.80	0.0023	#######	12.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	109	0.0000	9860	13
2	184	0.0036	17272	14
3	148	0.0015	12288	13
4	173	0.0000	15734	14
5	75	0.0065	6472	9

Table 21. Problem Type 3 I-TREA chebyshev utility function results (cont.)

Problem Set 5				
Minimum	48	0.0006	4326	7
Maximum	102	0.0006	8264	9
Std.Dev.	20.51	0.0000	1460.91	1.10
Average	81.40	0.0006	6716.80	7.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	102	0.0006	8264	9
2	80	0.0006	6820	9
3	84	0.0006	6820	7
4	48	0.0006	4326	7
5	93	0.0006	7354	7

Table 22. Problem Type 4 I-TREA linear utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	48	0.0000	2686	5
Maximum	365	0.1071	18906	13
Std.Dev.	102.91	0.0372	5135.86	2.42
Average	220.12	0.0296	11234.92	8.20
Problem Set 1				
Minimum	218	0.0098	11582	7
Maximum	365	0.0155	18906	10
Std.Dev.	59.98	0.0026	2895.44	1.10
Average	311.40	0.0135	16298.40	8.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	365	0.0098	18906	9
2	300	0.0119	16032	9
3	363	0.0150	18380	9
4	218	0.0154	11582	7
5	311	0.0155	16592	10

Table 22. Problem Type 4 I-TREA linear utility function results (cont.)

Problem Set 2				
Minimum	48	0.0000	2686	5
Maximum	128	0.0009	6346	7
Std.Dev.	29.95	0.0004	1376.79	0.84
Average	77.80	0.0003	4044.00	5.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	128	0.0005	6346	7
2	48	0.0000	2686	6
3	66	0.0000	3440	5
4	75	0.0009	3964	6
5	72	0.0000	3784	5
Problem Set 3				
Minimum	83	0.0000	4642	5
Maximum	250	0.0097	11948	10
Std.Dev.	67.45	0.0046	3084.66	1.95
Average	175.60	0.0054	8844.00	7.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	215	0.0007	10774	10
2	131	0.0083	6598	6
3	199	0.0083	10258	8
4	83	0.0097	4642	5
5	250	0.0000	11948	8
Problem Set 4				
Minimum	187	0.0961	10208	5
Maximum	365	0.1071	17230	13
Std.Dev.	75.01	0.0046	3189.44	3.58
Average	260.20	0.0995	13443.60	9.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	249	0.0962	12844	13
2	196	0.1071	10682	5
3	304	0.0961	16254	7
4	187	0.0979	10208	13
5	365	0.1003	17230	10

Table 22. Problem Type 4 I-TREA linear utility function results (cont.)

Problem Set 5				
Minimum	171	0.0226	8660	6
Maximum	334	0.0330	16950	11
Std.Dev.	69.31	0.0040	3420.37	1.95
Average	275.60	0.0291	13544.60	9.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	171	0.0330	8660	6
2	324	0.0226	15754	11
3	239	0.0313	11420	10
4	334	0.0280	16950	10
5	310	0.0304	14939	10

Table 23. Problem Type 4 I-TREA chebyshev utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	95	0.0000	5540	5
Maximum	610	0.0154	28362	28
Std.Dev.	124.92	0.0056	5778.70	5.21
Average	298.28	0.0061	15001.44	11.00
Problem Set 1				
Minimum	176	0.0049	9498	8
Maximum	413	0.0130	22140	24
Std.Dev.	106.36	0.0035	5480.45	6.54
Average	307.60	0.0072	16412.00	13.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	212	0.0049	11710	10
2	413	0.0079	22140	16
3	380	0.0049	19562	8
4	357	0.0053	19150	24
5	176	0.0130	9498	10

Table 23. Problem Type 4 I-TREA chebyshev utility function results (cont.)

Problem Set 2				
Minimum	195	0.0011	9528	8
Maximum	279	0.0069	14670	12
Std.Dev.	33.49	0.0027	2010.26	1.52
Average	239.40	0.0035	12346.00	9.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	261	0.0022	13862	9
2	244	0.0069	11804	12
3	195	0.0060	9528	10
4	218	0.0011	11866	8
5	279	0.0016	14670	9
Problem Set 3				
Minimum	223	0.0000	11564	7
Maximum	508	0.0037	23910	14
Std.Dev.	117.45	0.0018	4921.98	2.51
Average	337.80	0.0018	16611.20	10.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	260	0.0018	13834	10
2	223	0.0037	11564	11
3	407	0.0036	19144	10
4	291	0.0000	14604	7
5	508	0.0000	23910	14
Problem Set 4				
Minimum	187	0.0132	10154	5
Maximum	610	0.0154	28362	28
Std.Dev.	165.63	0.0010	7392.33	9.02
Average	407.00	0.0146	19774.40	12.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	370	0.0137	17568	8
2	341	0.0153	16802	11
3	187	0.0154	10154	5
4	527	0.0154	25986	10
5	610	0.0132	28362	28

Table 23. Problem Type 4 I-TREA chebyshev utility function results (cont.)

Problem Set 5				
Minimum	95	0.0000	5540	5
Maximum	317	0.0132	14652	13
Std.Dev.	81.40	0.0057	3376.02	3.39
Average	199.60	0.0035	9863.60	9.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	175	0.0000	8540	8
2	317	0.0000	14652	13
3	182	0.0044	9332	7
4	229	0.0000	11254	12
5	95	0.0132	5540	5

Table 24. Problem type 1 modified NSGA-II linear utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	8	0.0005	1472	0
Maximum	13	0.3671	3036	5
Std.Dev.	1.41	0.1453	321.62	1.46
Average	10.60	0.1275	2367.32	1.96
Problem Set 1				
Minimum	9	0.1441	2394	1
Maximum	12	0.3460	3036	5
Std.Dev.	1.14	0.1038	267.84	1.41
Average	10.60	0.2564	2666.80	3.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	12	0.1441	3036	5
2	10	0.3460	2537	3
3	11	0.3431	2512	1
4	11	0.1441	2855	3
5	9	0.3049	2394	3

Table 24. Problem type 1 modified NSGA-II linear utility function results (cont.)

Problem Set 2				
Minimum	12	0.0005	1472	0
Maximum	13	0.0194	2505	4
Std.Dev.	0.55	0.0085	435.28	1.64
Average	12.40	0.0043	2232.00	2.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	12	0.0005	2263	4
2	12	0.0005	1472	3
3	13	0.0194	2440	0
4	12	0.0005	2480	4
5	13	0.0005	2505	3
Problem Set 3				
Minimum	9	0.0103	2307	1
Maximum	10	0.0103	2452	3
Std.Dev.	0.55	0.0000	71.34	1.10
Average	9.40	0.0103	2363.40	1.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	9	0.0103	2316	1
2	9	0.0103	2307	1
3	10	0.0103	2430	3
4	9	0.0103	2312	1
5	10	0.0103	2452	3
Problem Set 4				
Minimum	10	0.0173	1785	1
Maximum	11	0.0720	2520	3
Std.Dev.	0.55	0.0232	320.30	1.10
Average	10.60	0.0472	2354.60	1.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	11	0.0693	2514	1
2	10	0.0173	2517	3
3	10	0.0373	2437	3
4	11	0.0400	1785	1
5	11	0.0720	2520	1

Table 24. Problem type 1 modified NSGA-II linear utility function results (cont.)

Problem Set 5				
Minimum	8	0.2323	2001	0
Maximum	12	0.3671	2563	1
Std.Dev.	1.87	0.0665	298.91	0.55
Average	10.00	0.3192	2219.80	0.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	9	0.3671	2001	0
2	12	0.2624	2531	1
3	9	0.3671	2002	0
4	8	0.3671	2002	0
5	12	0.2323	2563	1

Table 25. Problem type 1 modified NSGA-II chebyshev utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	8	0.0000	2001	0
Maximum	13	0.5138	3036	5
Std.Dev.	1.50	0.1739	251.97	1.16
Average	10.60	0.1741	2395.68	1.44
Problem Set 1				
Minimum	10	0.0000	2394	1
Maximum	12	0.0975	3036	5
Std.Dev.	0.89	0.0506	267.84	1.41
Average	10.60	0.0548	2666.80	3.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	12	0.0000	3036	5
2	10	0.0975	2537	3
3	10	0.0975	2512	1
4	11	0.0000	2855	3
5	10	0.0790	2394	3

Table 25. Problem type 1 modified NSGA-II chebyshev utility function results (cont.)

Problem Set 2				
Minimum	12	0.3498	2148	1
Maximum	13	0.5138	2521	1
Std.Dev.	0.55	0.0898	159.99	0.00
Average	12.60	0.4482	2432.00	1.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	12	0.3498	2521	1
2	12	0.5138	2506	1
3	13	0.3498	2148	1
4	13	0.5138	2470	1
5	13	0.5138	2515	1
Problem Set 3				
Minimum	8	0.1209	2008	1
Maximum	10	0.2959	2382	2
Std.Dev.	0.71	0.0783	147.78	0.45
Average	9.00	0.2609	2267.20	1.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	8	0.2959	2008	1
2	9	0.2959	2310	1
3	9	0.1209	2382	2
4	10	0.2959	2317	1
5	9	0.2959	2319	1
Problem Set 4				
Minimum	9	0.0000	2002	0
Maximum	11	0.1750	2519	3
Std.Dev.	1.10	0.0732	214.91	1.10
Average	10.20	0.0700	2370.20	1.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	11	0.0875	2515	1
2	11	0.0000	2448	1
3	11	0.0875	2519	1
4	9	0.0000	2367	3
5	9	0.1750	2002	0

Table 25. Problem type 1 modified NSGA-II chebyshev utility function results (cont.)

Problem Set 5				
Minimum	9	0.0119	2001	0
Maximum	12	0.0744	2547	2
Std.Dev.	1.52	0.0342	266.73	0.84
Average	10.60	0.0369	2242.20	0.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	9	0.0744	2002	0
2	12	0.0119	2504	1
3	11	0.0119	2157	1
4	9	0.0744	2001	0
5	12	0.0119	2547	2

Table 26. Problem type 2 modified NSGA-II linear utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	11	0.0014	2014	0
Maximum	17	0.3422	2782	4
Std.Dev.	1.70	0.1365	280.33	1.03
Average	14.84	0.1291	2364.72	1.32
Problem Set 1				
Minimum	13	0.3422	2033	0
Maximum	16	0.3422	2162	0
Std.Dev.	1.30	0.0000	50.86	0.00
Average	15.20	0.3422	2075.20	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	15	0.3422	2073	0
2	16	0.3422	2062	0
3	16	0.3422	2162	0
4	16	0.3422	2033	0
5	13	0.3422	2046	0

Table 26. Problem type 2 modified NSGA-II linear utility function results (cont.)

Problem Set 2				
Minimum	14	0.0014	2014	1
Maximum	17	0.0036	2526	2
Std.Dev.	1.14	0.0009	234.17	0.45
Average	15.40	0.0027	2258.00	1.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	15	0.0025	2249	1
2	15	0.0025	2041	1
3	16	0.0014	2460	1
4	17	0.0036	2526	2
5	14	0.0036	2014	1
Problem Set 3				
Minimum	12	0.0121	2039	2
Maximum	17	0.0132	2756	4
Std.Dev.	2.07	0.0005	377.25	0.89
Average	14.60	0.0123	2453.60	2.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	13	0.0121	2039	2
2	12	0.0121	2045	2
3	15	0.0121	2756	4
4	16	0.0121	2673	2
5	17	0.0132	2755	3
Problem Set 4				
Minimum	11	0.0454	2069	1
Maximum	14	0.0785	2536	3
Std.Dev.	1.10	0.0167	185.96	0.84
Average	12.80	0.0652	2394.80	1.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	13	0.0782	2440	1
2	13	0.0754	2449	1
3	14	0.0454	2536	2
4	11	0.0785	2069	3
5	13	0.0487	2480	2

Table 26. Problem type 2 modified NSGA-II linear utility function results (cont.)

Problem Set 5				
Minimum	15	0.1942	2541	1
Maximum	17	0.2665	2782	1
Std.Dev.	0.84	0.0396	105.27	0.00
Average	16.20	0.2231	2642.00	1.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	17	0.2665	2541	1
2	15	0.2665	2542	1
3	16	0.1942	2636	1
4	16	0.1942	2709	1
5	17	0.1942	2782	1

Table 27. Problem type 2 modified NSGA-II chebyshev utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	11	0.0556	2015	0
Maximum	19	0.5247	3594	3
Std.Dev.	2.37	0.1485	347.60	0.87
Average	14.84	0.2192	2400.12	1.00
Problem Set 1				
Minimum	13	0.1300	2033	0
Maximum	17	0.1300	2162	0
Std.Dev.	1.58	0.0000	50.86	0.00
Average	15.00	0.1300	2075.20	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	14	0.1300	2073	0
2	17	0.1300	2062	0
3	16	0.1300	2162	0
4	15	0.1300	2033	0
5	13	0.1300	2046	0

Table 27. Problem type 2 modified NSGA-II chebyshev utility function results (cont.)

Problem Set 2				
Minimum	17	0.3447	2517	1
Maximum	19	0.5247	2740	2
Std.Dev.	0.71	0.0636	96.82	0.55
Average	18.00	0.4347	2580.80	1.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	19	0.4347	2740	1
2	17	0.5247	2517	1
3	18	0.4347	2521	2
4	18	0.4347	2519	1
5	18	0.3447	2607	2
Problem Set 3				
Minimum	13	0.3041	2332	1
Maximum	15	0.3889	2625	3
Std.Dev.	0.71	0.0440	126.42	1.00
Average	14.00	0.3408	2463.40	2.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	14	0.3111	2477	3
2	15	0.3041	2625	3
3	13	0.3889	2343	1
4	14	0.3889	2332	1
5	14	0.3111	2540	2
Problem Set 4				
Minimum	12	0.1167	2031	1
Maximum	19	0.1202	3594	1
Std.Dev.	2.83	0.0016	577.57	0.00
Average	14.00	0.1174	2629.40	1.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	13	0.1167	2509	1
2	13	0.1167	2530	1
3	12	0.1202	2031	1
4	19	0.1167	3594	1
5	13	0.1167	2483	1

Table 27. Problem type 2 modified NSGA-II chebyshev utility function results (cont.)

Problem Set 5				
Minimum	11	0.0556	2015	0
Maximum	16	0.0858	2597	1
Std.Dev.	2.17	0.0162	304.03	0.55
Average	13.20	0.0732	2251.80	0.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	12	0.0858	2015	0
2	15	0.0556	2572	1
3	16	0.0556	2597	1
4	12	0.0833	2030	1
5	11	0.0858	2045	0

Table 28. Problem type 3 modified NSGA-II linear utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	16	0.0008	2032	0
Maximum	28	0.3957	3560	2
Std.Dev.	3.32	0.1376	333.41	0.54
Average	18.92	0.1370	2247.12	0.28
Problem Set 1				
Minimum	16	0.3336	2055	0
Maximum	28	0.3957	3560	1
Std.Dev.	5.22	0.0334	628.59	0.45
Average	18.80	0.3592	2457.60	0.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	16	0.3371	2055	0
2	28	0.3336	3560	1
3	18	0.3957	2385	0
4	16	0.3957	2120	0
5	16	0.3336	2168	0

Table 28. Problem type 3 modified NSGA-II linear utility function results (cont.)

Problem Set 2				
Minimum	16	0.0008	2045	0
Maximum	25	0.0015	2666	2
Std.Dev.	3.58	0.0004	288.69	0.89
Average	19.60	0.0012	2264.00	0.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	25	0.0008	2666	1
2	17	0.0014	2087	0
3	21	0.0008	2476	2
4	19	0.0015	2046	0
5	16	0.0014	2045	0
Problem Set 3				
Minimum	18	0.0375	2096	0
Maximum	25	0.0455	2626	1
Std.Dev.	2.59	0.0036	260.83	0.55
Average	21.80	0.0391	2318.20	0.60
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	25	0.0375	2579	1
2	18	0.0455	2143	0
3	23	0.0375	2626	1
4	21	0.0375	2096	1
5	22	0.0375	2147	0
Problem Set 4				
Minimum	17	0.0612	2073	0
Maximum	18	0.0833	2142	0
Std.Dev.	0.55	0.0099	24.88	0.00
Average	17.60	0.0656	2110.00	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	18	0.0612	2142	0
2	18	0.0612	2105	0
3	17	0.0833	2113	0
4	18	0.0612	2073	0
5	17	0.0612	2117	0

Table 28. Problem type 3 modified NSGA-II linear utility function results (cont.)

Problem Set 5				
Minimum	16	0.2014	2032	0
Maximum	17	0.2506	2131	0
Std.Dev.	0.45	0.0255	42.06	0.00
Average	16.80	0.2199	2085.80	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	17	0.2014	2082	0
2	16	0.2506	2032	0
3	17	0.2450	2131	0
4	17	0.2014	2124	0
5	17	0.2014	2060	0

Table 29. Problem type 3 modified NSGA-II chebyshev utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	16	0.0363	2032	0
Maximum	29	0.3440	3484	2
Std.Dev.	3.41	0.1040	328.98	0.58
Average	19.72	0.1749	2359.80	0.40
Problem Set 1				
Minimum	16	0.0896	2110	0
Maximum	19	0.1437	2582	1
Std.Dev.	1.52	0.0227	200.01	0.55
Average	17.40	0.1116	2324.80	0.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	19	0.0896	2465	1
2	19	0.1175	2582	1
3	16	0.1175	2110	0
4	17	0.1437	2309	0
5	16	0.0896	2158	0

Table 29. Problem type 3 modified NSGA-II chebyshev utility function results (cont.)

Problem Set 2				
Minimum	17	0.2593	2139	0
Maximum	24	0.3172	2669	1
Std.Dev.	2.74	0.0317	253.91	0.55
Average	21.00	0.2825	2436.20	0.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	20	0.3172	2185	0
2	24	0.2593	2669	1
3	17	0.3172	2139	0
4	23	0.2593	2629	1
5	21	0.2593	2559	0
Problem Set 3				
Minimum	21	0.2970	2453	0
Maximum	25	0.3440	2601	1
Std.Dev.	1.58	0.0210	63.94	0.45
Average	23.00	0.3064	2567.20	0.80
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	21	0.2970	2453	0
2	25	0.2970	2591	1
3	22	0.3440	2595	1
4	24	0.2970	2596	1
5	23	0.2970	2601	1
Problem Set 4				
Minimum	16	0.1000	2032	0
Maximum	19	0.1265	2197	0
Std.Dev.	1.10	0.0140	65.72	0.00
Average	17.80	0.1153	2122.20	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	18	0.1250	2114	0
2	16	0.1250	2094	0
3	18	0.1000	2197	0
4	18	0.1265	2032	0
5	19	0.1000	2174	0

Table 29. Problem type 3 modified NSGA-II chebyshev utility function results (cont.)

Problem Set 5				
Minimum	16	0.0363	2042	0
Maximum	29	0.0731	3484	2
Std.Dev.	5.41	0.0153	634.97	0.89
Average	19.40	0.0586	2348.60	0.40
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	18	0.0731	2067	0
2	17	0.0552	2092	0
3	17	0.0552	2042	0
4	29	0.0363	3484	2
5	16	0.0731	2058	0

Table 30. Problem type 4 modified NSGA-II linear utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	20	0.0006	2063	0
Maximum	37	0.3401	3890	1
Std.Dev.	4.08	0.1315	376.50	0.28
Average	24.36	0.1572	2279.80	0.08
Problem Set 1				
Minimum	20	0.2950	2120	0
Maximum	25	0.3401	2407	0
Std.Dev.	1.92	0.0235	119.49	0.00
Average	22.20	0.3217	2211.00	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	20	0.3381	2144	0
2	22	0.2950	2141	0
3	21	0.3381	2120	0
4	25	0.3401	2407	0
5	23	0.296	2243	0

Table 30. Problem type 4 modified NSGA-II linear utility function results (cont.)

Problem Set 2				
Minimum	22	0.0006	2092	0
Maximum	26	0.0011	2586	0
Std.Dev.	1.67	0.0003	209.03	0.00
Average	24.60	0.0009	2321.80	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	24	0.0011	2286	0
2	25	0.0006	2092	0
3	26	0.0011	2586	0
4	22	0.0006	2163	0
5	26	0.0011	2482	0
Problem Set 3				
Minimum	21	0.0165	2109	0
Maximum	37	0.0228	3890	1
Std.Dev.	7.06	0.0028	776.60	0.45
Average	24.40	0.0216	2504.20	0.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	21	0.0228	2125	0
2	21	0.0228	2109	0
3	21	0.0228	2148	0
4	37	0.0165	3890	1
5	22	0.0228	2249	0
Problem Set 4				
Minimum	23	0.1706	2063	0
Maximum	35	0.1876	2605	1
Std.Dev.	4.69	0.0073	240.20	0.45
Average	27.00	0.1746	2175.40	0.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	27	0.1713	2075	0
2	25	0.1720	2069	0
3	25	0.1713	2065	0
4	35	0.1706	2605	1
5	23	0.1876	2063	0

Table 30. Problem type 4 modified NSGA-II linear utility function results (cont.)

Problem Set 5				
Minimum	22	0.2601	2068	0
Maximum	28	0.2943	2525	0
Std.Dev.	2.51	0.0151	190.25	0.00
Average	23.60	0.2672	2186.60	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	22	0.2601	2120	0
2	23	0.2943	2106	0
3	23	0.2601	2114	0
4	28	0.2601	2525	0
5	22	0.2616	2068	0

Table 31. Problem type 4 modified NSGA-II chebyshev utility function results

Problems Average				
	Runtime (sec)	**Utility Dev. (abs)**	**Generations**	**Questions**
Minimum	20	0.0778	2046	0
Maximum	29	0.2914	2580	1
Std.Dev.	2.26	0.0613	112.84	0.20
Average	23.52	0.1620	2164.24	0.04
Problem Set 1				
Minimum	21	0.1218	2113	0
Maximum	26	0.1433	2580	1
Std.Dev.	1.87	0.0113	194.29	0.45
Average	23.00	0.1347	2266.80	0.20
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	22	0.1424	2147	0
2	26	0.1433	2580	1
3	23	0.1433	2163	0
4	21	0.1228	2113	0
5	23	0.1218	2331	0

Table 31. Problem type 4 modified NSGA-II chebyshev utility function results (cont.)

Problem Set 2				
Minimum	22	0.1913	2149	0
Maximum	26	0.2359	2323	0
Std.Dev.	1.87	0.0191	72.51	0.00
Average	24.00	0.2018	2220.20	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	23	0.2359	2149	0
2	22	0.1952	2155	0
3	26	0.1913	2323	0
4	26	0.1932	2254	0
5	23	0.1932	2220	0
Problem Set 3				
Minimum	20	0.2194	2075	0
Maximum	22	0.2914	2214	0
Std.Dev.	0.71	0.0302	56.54	0.00
Average	21.00	0.2485	2117.60	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	21	0.2914	2075	0
2	21	0.2194	2121	0
3	20	0.2194	2093	0
4	22	0.2562	2214	0
5	21	0.2562	2085	0
Problem Set 4				
Minimum	23	0.1259	2078	0
Maximum	29	0.1619	2138	0
Std.Dev.	2.24	0.0127	28.79	0.00
Average	26.00	0.1443	2108.60	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	27	0.1443	2108	0
2	29	0.1259	2137	0
3	26	0.1443	2138	0
4	25	0.1619	2082	0
5	23	0.1451	2078	0

Table 31. Problem type 4 modified NSGA-II chebyshev utility function results (cont.)

Problem Set 5				
Minimum	22	0.0778	2046	0
Maximum	25	0.0909	2160	0
Std.Dev.	1.34	0.0058	41.95	0.00
Average	23.60	0.0806	2108.00	0.00
Run	**Runtime (sec)**	**Utility Dev. (abs)**	**Generations**	**Questions**
1	23	0.0909	2118	0
2	23	0.0778	2122	0
3	22	0.0784	2046	0
4	25	0.0778	2094	0
5	25	0.0784	2160	0