MULTI-OBJECTIVE COMBINATORIAL OPTIMIZATION USING EVOLUTIONARY ALGORITHMS

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

BY

BURCU ÖZSAYIN

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

AUGUST 2009

Approval of the thesis:

MULTI-OBJECTIVE COMBINATORIAL OPTIMIZATION USING EVOLUTIONARY ALGORITHMS

submitted by **BURCU ÖZSAYIN** in partial fulfillment of the requirement for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

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

Prof. Dr. Nur Evin Özdemirel _____
Head of Department, **Industrial Engineering**

Prof. Dr. Murat Köksalan _____
Supervisor, **Industrial Engineering Dept., METU**

Examining Committee Members

Assoc. Prof. Dr. Yasemin Serin _____
Industrial Engineering Dept., METU

Prof. Dr. Murat Köksalan _____
Industrial Engineering Dept., METU

Prof. Dr. Nesim Erkip _____
Industrial Engineering Dept., Bilkent University

Assist. Prof. Dr. İsmail Serdar Bakal _____
Industrial Engineering Dept., METU

Assist. Prof. Dr. Banu Soylu _____
Industrial Engineering Dept., Erciyes University

Date: **12.08.2009**

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 : Burcu ÖZSAYIN

Signature :

ABSTRACT

MULTI-OBJECTIVE COMBINATORIAL OPTIMIZATION USING EVOLUTIONARY ALGORITHMS

Özsayın, Burcu

M.S., Department of Industrial Engineering

Supervisor: Prof. Dr. Murat Köksalan

August 2009, 110 pages

Due to the complexity of multi-objective combinatorial optimization problems (MOCO), metaheuristics like multi-objective evolutionary algorithms (MOEA) are gaining importance to obtain a well-converged and well-dispersed Pareto-optimal frontier approximation. In this study, of the well-known MOCO problems, single-dimensional multi-objective knapsack problem and multi-objective assignment problem are taken into consideration. We develop a steady-state and elitist MOEA in order to approximate the Pareto-optimal frontiers. We utilize a territory concept in order to provide diversity over the Pareto-optimal frontiers of various problem instances. The motivation behind the territory definition is to attach the algorithm the advantage of fast execution by eliminating the need for an explicit diversity preserving operator. We also develop an interactive preference incorporation mechanism to converge to the regions that are of special interest for the decision maker by interacting with him/her during the optimization process.

Keywords: Multi-objective Evolutionary Algorithms, Multi-objective Combinatorial Optimization, Preference Incorporation, Interactive Method

ÖZ

EVRİMSEL ALGORİTMA İLE ÇOK AMAÇLI KOMBİNATORYEL OPTİMİZASYON

Özsayın, Burcu
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Murat Köksalan

Ağustos 2009, 110 sayfa

Çok amaçlı kombinatoryel problemlerin karmaşıklığından dolayı, iyi yakınsanmış ve iyi dağılmış bir Pareto-optimal sınır yaklaşımı elde etmek için çok amaçlı evrimsel algoritmalar gibi üstsezgisel metodlar önem kazanmıştır. Bu çalışmada, belli başlı çok amaçlı kombinatoryel optimizasyon problemlerinden, tekboyutlu çok amaçlı sırt çantası problemi ve çok amaçlı atama problemi ele alınmaktadır. Pareto-optimal sınıra yaklaşmak için elitist ve kararlı durumda bir evrimsel algoritma geliştirilmektedir. Farklı problemlerin Pareto-optimal sınırları üzerinde çeşitlilik sağlayabilmek için, bölge kavramı kullanılmaktadır. Bölge tanımlamasının arkasındaki motivasyon, açık bir çeşitlilik koruma operatörünün kullanımını ortadan kaldırarak algoritmaya hızlı uygulama avantajını sağlamaktır. Ayrıca, karar verici için ilgi çekici olan Pareto-optimal sınır bölgelerini yakınsama amacıyla, karar verici ile optimizasyon aşamasında iletişim kurulmasını sağlayan tercihe dayalı interaktif bir mekanizma geliştirilmektedir.

Anahtar Kelimeler: Çok Amaçlı Evrimsel Algoritmalar, Çok Amaçlı Kombinatoryel Optimizasyon, Tercihe Dayalı, Etkileşimli Metod

To Dad

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my supervisor Professor Murat Köksalan for his guidance, encouragements, and persistent help throughout this study. I have dedicated myself to this study by the existence of his valuable presence.

I would like to thank İbrahim Karahan, whose work helped me to find my way in this study. He was always ready to help me whenever I needed his support.

I am grateful to my father who is not with me physically but always make me feel his excellent spirit in my heart. I want to thank my mom for her endless and unconditional love, and for being so strong to stand behind me all the time. Moreover, I am grateful to the most important person in my life, my brother Burak, for being the best brother one can ever have.

In addition, I would like to thank my dear friend Özge Baytürk for being my sister and for this wonderful friendship she dedicated to my life.

I am also thankful to my friends Aslı Gül Buğdacı, Gülşah Karakaya, and Diclehan Tezcaner. I have always felt their existence whenever I needed their ideas and support.

I would like to thank TÜBİTAK for supporting me financially during this study.

Lastly, I would like to thank Ercan Balkoç for providing me my second wing to make me fly. My life would not be as meaningful if he was not the second half of it. His encouragements, patience, and love assisted me to complete this thesis.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Multi-objective optimization problems (MOPs) cover most of the real world problems in which there exist many objectives that are in most occasions conflicting. There is not a single optimal solution, but there are many solutions which are better in some objectives and worse in others. The so called nondominated solutions show the trade-off between these objectives and compose the Pareto-optimal frontier. To end up with a single preferred nondominated solution, many mathematical modeling based approaches have been proposed. However, if the decision maker (DM) does not provide any preference information between the objectives, each objective is assumed to be of equal importance, which results in the equal importance of all solutions on the whole Pareto-optimal frontier. In such a case, the aim in MOPs is to construct the whole Pareto-optimal frontier and the decision making process is postponed to post-optimal stage.

A significant portion of MOPs are multi-objective combinatorial optimization (MOCO) problems. However, MOCO problems are in general considerably hard to solve problems. Due to the complexity of even the single objective combinatorial optimization problems, heuristic methods are becoming more and more important in this area of multi-objective optimization. When multi-objective combinatorial optimization (MOCO) problems come on the scene, one more but a very challenging issue emerges. This issue is the trade-off among the objective functions and the existence of many Pareto-optimal solutions one of which cannot be considered superior to another in the absence of any preference information. Consequently, metaheuristics like multi-objective evolutionary algorithms (MOEA) are gaining more importance to obtain a well-converged and well-dispersed Pareto-optimal frontier approximation.

Working with a population of solutions, multi-objective evolutionary algorithms (MOEAs) have gained a significant interest of operational researchers, especially in solving multi-objective optimization problems (MOP). Evolutionary algorithms are population-based metaheuristic optimization algorithms, which apply the principles of natural evolution to optimization. Moreover, MOEAs are not problem dependent, having the advantage of being suitable for a very general class of optimization problems. Therefore, having a population of solutions in every iteration, EAs are well-suited to approximate the Pareto-optimal frontier. However, there does not exist many MOEAs designed to solve MOCO problems. With appropriate adaptations, MOEAs constitute a good compromise on the way to solve MOCO problems with good approximation properties and acceptable computational complexity.

In the absence of information on the utility function of the DM, MOP techniques have two main objectives: ***Convergence to Pareto frontier*** and ***diversity of solutions over the entire Pareto frontier***. These two features provide the DM with a good representation of the Pareto-optimal solutions, that is, the nature of the Pareto-optimal frontier. As a third objective, the current research is mainly on algorithms that concentrate on the region of the Pareto-optimal frontier which is of special interest to the DM. That is, instead of generating the whole Pareto-optimal frontier, a specified region on the frontier can be generated.

We propose an elitist MOEA called Combinatorial Territory Defining Evolutionary Algorithm (C-TDEA) and apply it on two well-known combinatorial optimization (MOCO) problems: multi-objective knapsack problem and multi-objective assignment problem. The proposed MOEA is inspired by ***Territory Defining Evolutionary Algorithm (TDEA)*** that has been proposed by Karahan and Köksalan (2008). Moreover, we propose a preference-based version of the algorithm that concentrates on the preferred regions of the Pareto-optimal frontier and even obtains a single best solution for the DM in an interactive manner.

The thesis begins with presenting some definitions and related literature on MOEAs, preference incorporation in MOEAs, and MOCO problems in Chapter 2.

In Chapter 3, we introduce the general outline of C-TDEA and give the details of the algorithm. We test the performance of the proposed algorithm with respect to two well-known MOEAs on 2-, and 3-objective problems.

In chapter 4, we present the preference incorporated version of C-TDEA, called Interactive C-TDEA (IC-TDEA). The performance of this preference-based MOEA is also tested on the same 2- and 3-objective problems.

Finally, we provide some conclusive remarks on the contribution of our work and provide future research directions in Chapter 5.

CHAPTER 2

DEFINITIONS AND LITERATURE REVIEW

In this chapter, we provide the definitions that are used throughout this work and the related literature on multi-objective evolutionary algorithms and multi-objective combinatorial optimization problems. The definitions of the concepts related to this study are provided in Section 2.1. The literature review on multi-objective evolutionary algorithms and preference incorporation, and multi-objective combinatorial optimization problems are provided in Section 2.2.

2.1 DEFINITIONS

For further details on the introductory definitions below, see Steuer (1986) and Ehrgott (2000).

Multi-objective optimization is the process of optimizing multiple objective functions simultaneously. The optimization process may be subject to certain constraints.

Most multi-objective optimization problems (MOPs) have conflicting objectives. Therefore, such problems do not have a single solution optimizing all of the objective functions. The search is for finding the so called *nondominated solutions*. A nondominated solution is such a solution that no objective value can be improved without sacrificing from one or more of the remaining objectives.

We can formulate a *multi-objective optimization problem (MOP)* as follows:

$$\text{"Maximize"} \quad \mathbf{z} = \mathbf{f(x)} \tag{2.1}$$

$$\text{subject to} \quad \mathbf{x} \in \mathbf{X} \tag{2.2}$$

where,

$\mathbf{x} = (x_1, \ldots, x_N)^T$	decision variable vector
$\mathbf{X} \subseteq \mathcal{R}^N$	feasible decision space
$\mathbf{z} = (z_1, z_2, \ldots, z_M) = \mathbf{f(x)} = (f_1(x), \ldots, f_M(x))^T$	objective function vector
$\mathbf{Z} = \mathbf{f(X)}$	feasible objective space

Feasible decision space may be restricted by the following types of constraints:

$\mathbf{h(x)} = 0$	equality type constraints
$\mathbf{g(x)} \geq 0$	inequality type constraints

A *multi-objective combinatorial optimization (MOCO) problem* is an integer programming problem that is given by the following feasible set:

$$\text{"Maximize"} \quad \mathbf{z} = \mathbf{f(x)} = \mathbf{Cx} \tag{2.3}$$

$$\text{subject to} \quad \mathbf{Ax} \geq \mathbf{b} \tag{2.4}$$

$$\mathbf{x} \in \mathbf{Z}^N \tag{2.5}$$

where,

$\mathbf{x} = (x_1, \ldots, x_N)^T$	decision variable vector
$\mathbf{X} \subseteq \mathbf{Z}^N$	feasible decision space
$\mathbf{z} = (z_1, z_2, \ldots, z_M) = \mathbf{f(x)} = (f_1(x), \ldots, f_M(x))^T$	objective function vector
$\mathbf{Z} = \mathbf{f(X)}$	feasible objective space

C	matrix of objective function coefficients
A	matrix of constraint coefficients
b	matrix of constraint left-hand-sides

The entries of the matrices **C**, **A**, and **b** are generally integer numbers. The feasible decision space of a combinatorial optimization problem is assumed to be a finite set. MOCO problems are not linear optimization problems, and there exist unsupported efficient solutions which bring MOCO problems additional difficulty over multi-objective linear optimization problems. MOCO problems are in general NP-complete and #P-complete, and intractable; which correspond to the difficulty of finding a solution, difficulty due to large number of solutions, and the absence of efficient means of solving the problem, respectively. In fact, the existence of unsupported efficient solutions is the main reason of the computational complexity of MOCO problems, and the computational complexity grows very fast with the problem size (Ehrgott and Gandibleux, 2004).

Multi-objective knapsack problem (MOKP) is a binary programming problem that is characterized with maximization type objectives. The MOKP is NP-hard (Ehrgott, 2000) and the formulation is as follows:

$$
\begin{aligned}
&\textit{Maximize} \quad z_1 = \sum_{j=1}^{N} c_j^1 x_j \\
&. \\
&. \\
&. \\
&\textit{Maximize} \quad z_M = \sum_{j=1}^{N} c_j^M x_j \\
&\textit{subject to} \quad \sum_{i=1}^{N} a_j x_j \le b \\
&\qquad\qquad x_j \in \{0,1\}; j = 1,..,N
\end{aligned}
\tag{2.6}
$$

In MOKP, a decision must be made on which items to put into a capacitated knapsack, out of N available items. If item j is put into the knapsack, then the corresponding decision variable x_j takes the value 1; otherwise it takes the value 0. Item j has a weight of a_j, and the total item weights that are put into the knapsack should not exceed the knapsack capacity b. Moreover, each item j is attached objective coefficient c_j^m for each objective m, where each of the M objectives is to be maximized.

Multi-objective assignment problem (MOAP) can be formulated as a binary programming problem that is characterized with minimization type objectives. Out of N people and N jobs, a decision must be made on which person to assign to which job. The formulation is as follows:

$$
\begin{aligned}
&\textit{Minimize} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}^{1} x_{ij} \\
&. \\
&. \\
&. \\
&\textit{Minimize} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}^{M} x_{ij} \\
&\textit{subject to} \quad \sum_{i=1}^{N} x_{ij} = 1 \;\; ; \quad j = 1,..,N \\
&\qquad\qquad \sum_{j=1}^{N} x_{ij} = 1; i = 1,..,N \\
&\qquad\qquad x_{ij} \in \{0,1\}
\end{aligned}
\tag{2.7}
$$

where the decision variable x_{ij} takes the value 1 if person i is assigned to job j, and it takes the value 0 otherwise. Each person can be assigned to only one job, and each job can be assigned to only a single person. These two restrictions are handled by adding two corresponding constraints into the MOAP model. If job j is assigned to person i, this causes the corresponding cost c_{ij}^m for each objective

m. The aim is to minimize the associated total costs for each of the M objectives. The MOAP is NP-complete, #P-complete, and intractable (Ehrgott, 2000).

2.1.1 Dominance and Efficiency

Without loss of generality, the following definitions assume that the objectives are of maximization type.

Definition 2.1. Let $\mathbf{z^1}, \mathbf{z^2} \in \mathbf{Z}$ be two objective vectors. $\mathbf{z^1}$ is said to *dominate* $\mathbf{z^2}$ if $z_i^1 \geq z_i^2$ for all *i* and $z_j^1 > z_j^2$ for at least one *j*.

Definition 2.2. Let $\mathbf{z} \in \mathbf{Z}$ be an objective vector. If there is no $\mathbf{z'} \in \mathbf{Z}$ that dominates **z**, then **z** is said to be *nondominated.* If there exists at least one such $\mathbf{z'} \in \mathbf{Z}$ that dominates **z**, then **z** is said to be *dominated*. The set of all nondominated solutions is called the *nondominated set*.

Definition 2.3. Let $\mathbf{x} \in \mathbf{X}$ be a decision vector. Then, **x** is said to be *efficient* if there is no $\mathbf{x'} \in \mathbf{X}$ such that **f(x')** dominates **f(x).** If **f(x)** is dominated, then **x** is *inefficient*.

Definition 2.4. Let $\mathbf{z^1}, \mathbf{z^2} \in \mathbf{Z}$ be two objective vectors. Then, $\mathbf{z^1}$ *strictly dominates* $\mathbf{z^2}$ if $z_i^1 > z_i^2$ for all *i*.

Definition 2.5 Let $\mathbf{z} \in \mathbf{Z}$ be an objective vector. Then, **z** is said to be *weakly nondominated* if these is no $\mathbf{z'} \in \mathbf{Z}$ such that $z_i^{'} > z_i$ for all *i*. Nondominated set is a subset of the *weakly nondominated set* that may also contain dominated solutions.

Definition 2.6. Let $\mathbf{z'} \in \mathbf{Z}$ be an objective vector. Then, $\mathbf{z'}$ is said to be *convex dominated* if there exist objective vectors $\mathbf{z^1}, \mathbf{z^2}, \ldots, \mathbf{z^k} \in \mathbf{Z}$ and weights $\mu^1, \mu^2, \ldots, \mu^k \geq 0$ such that $\sum_{i=1}^{k} \mu^i = 1$ and $\sum_{i=1}^{k} \mu^i \mathbf{z}^i \geq \mathbf{z'}$.

Definition 2.7. Let $\mathbf{z'} \in \mathbf{Z}$ be an objective vector. Then, $\mathbf{z'}$ is said to be *unsupported nondominated* if it is convex dominated but nondominated. If the decision maker has a linear utility function, then the most preferred solution of the decision maker cannot be an unsupported nondominated solution, since these solutions cannot be obtained by maximizing a weighted-sum of objective functions.

Supported nondominated, unsupported nondominated, dominated, and weakly nondominated but dominated solutions are illustrated in Figure 2.1.

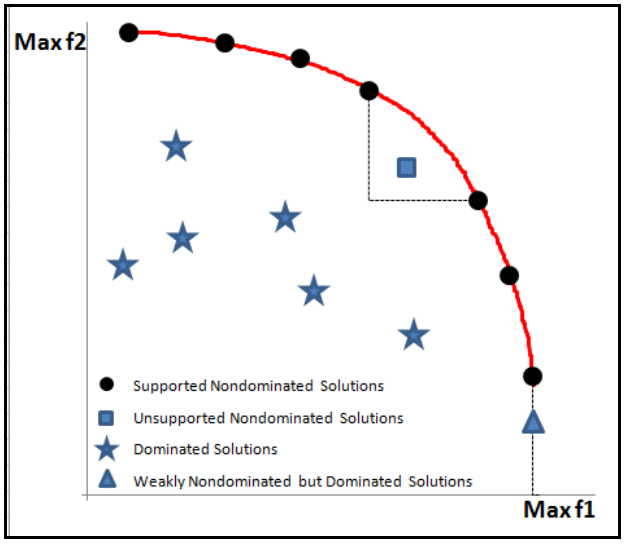

Figure 2.1 Solution Illustrations

2.1.2 Ideal and Nadir Objective Vectors

Definition 2.8. The *ideal objective vector* of a MOP is the vector of objective values $\mathbf{z^*} = \left[z_1^*, z_2^*, ..., z_i^* ..., z_M^*, \right]^T$, the i^{th} element of which corresponds to the optimal value of the i^{th} objective function. Ideal objective vector is constructed by optimizing each of the M objectives individually over the feasible region.

Definition 2.9. The *nadir objective vector* of a MOP is the vector of objective values $\mathbf{z^n} = \left[z_1^n, z_2^n, ..., z_i^n ..., z_M^n, \right]^T$, the i^{th} element of which corresponds to the worst value of the i^{th} objective function among all nondominated solutions.

Definition 2.10. For a MOP with M maximization-type objectives, a *payoff table* is an $M \times M$ matrix formed by using the decision vectors obtained while calculating the ideal objective vector. In the table, i^{th} row corresponds to the feasible objective vector at which the i^{th} objective function takes its optimal value. The nadir point can be estimated by using a payoff table. The smallest value in the j^{th} column gives an estimate for the j^{th} element of the nadir objective vector. However, there is no guarantee that this estimate is close to the correct nadir objective vector.

2.1.3 Favorable Weights

Definition 2.11. Favorable weights $\lambda^s = \{\lambda_1^s, \lambda_{2_2}^s, ..., \lambda_M^s\}$ of a solution s correspond to the weights that minimize the weighted Tchebycheff distance of the solution to the ideal objective vector f^*. Favorable weights of a solution are computed as follows (See Steuer 1986, p. 425):

$$\lambda_i = \begin{cases} \dfrac{1}{f_i^* - f_i}\left[\sum_{j=1}^{M} \dfrac{1}{f_j^* - f_j}\right]^{-1} & \text{if } f_j \neq f_j^* \text{ for all } j = 1,2,...,M \\ 1 & \text{if } f_i = f_i^* \\ 0 & \text{if } f_i \neq f_i^* \text{ but } \exists j \text{ such that } f_j = f_j^* \end{cases} \quad (2.8)$$

where f_i is the i^{th} objective value and f_i^* is the i^{th} element of the ideal objective vector. Tchebycheff favorable weights are illustrated in Figure 2.2.

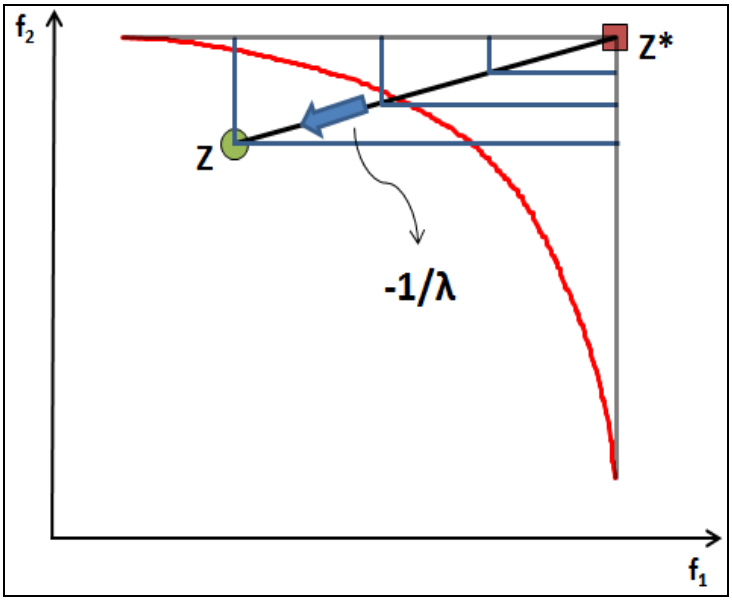

Figure 2.2 Tchebycheff Favorable Weights

2.1.4 Distance Metrics

L_q-metric is a function that defines the distance between two vectors **x, y** $\in \mathcal{R}^N$. The L_q-distance between vectors **x** and **y** is formulated as follows:

$$\|x - y\|_q = \left[\sum_{i=1}^{N} |x_i - y_i|^q\right]^{1/q} \quad (2.9)$$

If a weight vector w = [w_1, …, w_N] is provided, then the weighted L_q-distance between vectors **x** and **y** is calculated as follows:

$$\|x-y\|_{w,q} = \left[\sum_{i=1}^{N}(w_i|x_i-y_i|)^q\right]^{1/q} \tag{2.10}$$

Common weighted distance metrics are provided in Table 2.1.

Table 2.1 Common Weighted Distance Metrics

Name	Metric	Formula
Rectilinear	L_1	$\sum_{i=1}^{N} w_i\|x_i-y_i\|$
Euclidean	L_2	$\left[\sum_{i=1}^{N}(w_i\|x_i-y_i\|)^2\right]^{1/2}$
Tchebycheff	L_∞	$\max_i\left(w_i\|x_i-y_i\|\right)$

2.1.5 Definitions Related to Evolutionary Algorithms

For further details on the definitions below, see Deb (2001).

In an evolutionary algorithm, the solutions, the so called chromosomes, are assigned a goodness measure to evaluate that solution. This measure, which is defined in terms of the objective functions and the constraints, is called the *fitness* of that solution.

The main part of evolutionary algorithms consists of the genetic operators. The *selection operator* chooses the good parent solutions that mate to produce offspring solutions with the aim of duplicating the good solutions and eliminating the bad solutions. The most common selection operator, *tournament selection* operator creates tournaments between solutions and the best solution is chosen as parent.

Another genetic operator is the *crossover operator*, which mates the selected parent solutions, that is, exchanges some portion of the chromosomes between the two parents and creates new chromosomes, called offspring solutions.

The *mutation operator* changes some part on the chromosome of the offspring with the expectation of creating a better solution by performing a local search around the initial solution. Mutation operator also serves to provide diversity over the Pareto-optimal frontier.

Although it is possible to find multiple solutions during the early generations, it becomes extremely difficult to keep this diverse set of solutions throughout the evolutionary run. In an MOEA, to maintain the multiplicity of solutions till the final generation, an explicit *diversity-preserving operator* is needed. The operator may use different methods, but the aim is to provide a diverse set of solutions over the Pareto-optimal frontier.

In order to keep the previously found good solutions in the population and make use of them in the genetic operations, an *elite-preserving operator* can be used. This operator favors the good solutions, the so called elites, of a population by giving them the opportunity to survive among the subsequent generations. An evolutionary algorithm is said to be *elitist* if it utilizes an elite-preserving operator. In elitist MOEAs, a good solution obtained is never lost until a better solution is found.

In some evolutionary algorithms, an offspring is evaluated as soon as it is created. That is, after every offspring is created, that offspring is used to update the whole population. Such algorithms are called *steady-state MOEAs*.

Although the initial population of an MOEA can be constructed by randomly generated solutions, some non-random solutions can be put into the initial population that are thought to be good solutions for the considered problem. This

process is called *seeding*, and the placed non-random solutions are called *seed* solutions.

2.2 LITERATURE REVIEW

Working with a population of solutions, multi-objective evolutionary algorithms (MOEAs) constitute an important research area in multi-objective optimization. MOEAs provide fast convergence to Pareto-optimal frontier by evolving multiple solutions simultaneously. They also provide incorporation of various convergence and diversity operators, and even preference incorporation schemes. Deb (2001) and Coello et al. (2006) cover many aspects related to multi-objective evolutionary algorithms that are useful sources for the researchers.

Although most of the proposed algorithms in the MOEA literature focus on generating the entire Pareto-optimal frontier, there are many recent studies that aim to incorporate the preferences of the decision maker into the search process. Coello (2000), Chetković and Parmee (2002), and Rachmawati and Srinivasan (2006) classify the preference incorporation techniques used in MOEAs and explain the importance of preference incorporation.

The literature review section of this study is divided into three parts. First, the MOEAs approximating the entire efficient frontier without eliciting any preference information will be covered. Then, the preference incorporation techniques in the literature will be reviewed. Finally, we deal with multi-objective combinatorial optimization literature.

2.2.1 MOEAs that Approximate the Entire Efficient Frontier

Deb et al. (2002) have suggested the *Nondominated Sorting Genetic Algorithm 2 (NSGA-II)*. NSGA-II is an elitist algorithm that uses the idea of nondominated sorting. The solutions in the population are classified into nondomination levels

and each level is assigned its corresponding rank. The algorithm keeps an offspring population and a parent population. These two populations are combined after each generation and the population of the next generation is chosen among the members of the combined population. The selection is performed by considering the ranks of the solutions in the combined population such that, the solutions from the best ranked levels are selected. If the predetermined population size is hit such that all solutions from a particular rank cannot be accepted into the new population, the selection is made based on the crowding measure of the solutions at that particular rank. In order to provide diversity, sharing function method is used within each front, which makes the less crowded regions to have a higher probability for selection by degrading the fitness values of the solutions in the crowded regions.

Zitzler, Laumanns and Thiele (2002) have proposed an elitist multi-objective evolutionary algorithm, the *Strength Pareto Evolutionary Algorithm 2 (SPEA2)*, an improved version of the previous algorithm SPEA. SPEA2 maintains two coevolving populations, and copies the nondominated solutions of the regular population into the archive population. Each individual is assigned a strength which gives the number of individuals it dominates. The raw fitness of a solution becomes the sum of the strengths of the solutions that dominate it. By this way, SPEA2 prevents two solutions in the regular population to have the same raw fitness value if one of them dominates the other. In SPEA2, k^{th} nearest neighbor method is used as the density estimation technique. This density measure is added to the raw fitness value to obtain the fitness of a solution. Consequently, the solutions at the same raw fitness level can be discriminated according to their density value. SPEA2 is proposed with a constant archive size that does not vary throughout the genetic process. This is achieved by filling the remaining slots of the archive with the best dominated individuals if the number of nondominated individuals is less than the archive size. On the other hand, if the number of nondominated individuals is more than the archive size, an archive truncation procedure comes on the stage that utilizes the k-th nearest neighbor method.

The *Indicator-based Evolutionary Algorithm (IBEA)* is proposed by Zitzler and Künzli (2004) who have realized that most MOEAs use a property of the Pareto optimal set implicitly without taking other properties into consideration. This property may be to maximize the hypervolume dominated by the resulting Pareto-optimal frontier approximation, or to obtain an approximation that requires the minimal improvement to dominate the real Pareto-optimal frontier. The authors have proposed the usage of a binary performance measure, called an indicator, which can be obtained for any type of preference information. Once the indicator type is determined, it is used to calculate the fitness of each individual. In addition to the advantage of using any preference information about the realization of the Pareto frontier, IBEA does not also require any diversity preserving operator. The authors have proposed two binary quality indicators, additive ε-indicatior ($I_{\varepsilon+}$) and hypervolume indicator (I_{HD}). The algorithm suggests the design of more flexible MOEA techniques that respond to different types of preference information obtained from the DM.

In the study of Deb et al. (2005), the ε-domination concept has been utilized and a new steady-state algorithm called *"ε-Domination Based Multi-objective Evolutionary Algorithm (ε-MOEA)"* has been developed. The algorithm evolves two populations simultaneously, the EA population and the archive population. The archive population consists of the ε-nondominated solutions of the EA population. According to the ε-dominance concept, two solutions with a difference less than ε_i in the i-th objective are not nondominated to each other anymore. In ε-MOEA, the objective space is divided into hyper-boxes that are formed beginning from the minimum possible value of each objective. The archive selection procedure prevents two solutions to be in the same hyper-box. Consequently, the ε-dominance concept facilitates the reduction of the cardinality of the archive population, and provides the desired resolution for each of the objectives by obtaining a good diversity with a small computational time. However, there are two drawbacks associated with ε-MOEA. The first one is the absence of extreme solutions on the Pareto frontier which occurs due to the fact

that the extreme solutions usually become ε-dominated. The second drawback is related to the composition of the hyper-boxes. Instead of forming a hyper-box around each existing solution, the algorithm forms pre-specified hyper-boxes starting from the minimum possible value of each objective. This contradicts with the claim of the authors such that, it becomes possible for two solutions in neighboring hyper-boxes to be within ε_i in objective i. However, these drawbacks do not prevent ε-MOEA to be a sufficiently good compromise between convergence, diversity and computational efficiency.

Favorable Weight Based Evolutionary Algorithm (FWEA) of Soylu and Köksalan (2006) assigns its own weights to each of the individuals with respect to a weighted Tchebycheff distance function and these weights, called the favorable weights of the individual, are used to calculate its fitness score. According to the favorable weights mechanism, each individual contributes to convergence along its own favored direction so that it gains advantage over the other individuals. Moreover, the fitness values are assigned such that the underrepresented portions of the frontier are favored. After ranking the population according to their nondomination levels, the fitness values are adjusted such that, the fitness of any individual in a better frontier is at least an ε amount better than any individual in a worse frontier. Moreover, nearest neighbor information is incorporated into fitness calculation in order to maintain diversity. FWEA is a steady-state EA where only two offspring are generated in every generation and they replace two members of the population if the replacement rules are satisfied.

Territory Defining Evolutionary Algorithm (TDEA) which has been proposed by Karahan and Köksalan (2008) is a steady-state and elitist MOEA that has two coevolving populations, the regular population and the archive population. The archive population consists of the nondominated solutions of the regular population. The ε-domination concept used in ε-MOEA to update the archive population is replaced by the territory concept which is defined as the hyper-box enclosed by the territory size τ in all objective values of the solution. An offspring

is rejected to enter the archive if it violates the territory of an existing archive member that is nondominated with respect to the offspring. TDEA does not use a fitness function. The existence of territory defining property eliminates the need for an explicit diversity preserving operator resulting in high computational efficiency while providing diversity and convergence. The size of the territory provides a limit on the final cardinality of the archive population. Consequently, the final population size and the computational time increase as τ decreases.

2.2.2 Preference Incorporation in MOEAs

One of the first efforts to concentrate on a desired portion of the efficient frontier using MOEAs has been the attempt to solve *goal programming* problems. Deb (1999), realizing the difficulties associated to the classical goal programming procedure, has proposed a technique to solve such problems using an MOEA. He has suggested translating each goal to an objective that minimizes the deviation from the target. The modified goal programming problem is then solved using Nondominated Sorting Genetic Algorithm (NSGA). The multiple Pareto-optimal solutions found by NSGA correspond to different weights which are evaluated simultaneously. Consequently, the result is not a single solution corresponding to a certain weight vector as in the case of classical goal programming, but it is a region on the Pareto-optimal frontier dominated by the goal, where the distances from the goal is minimized for differing weight vectors.

Branke et al. (2001) has introduced an algorithm utilizing the trade-offs between objectives, in order to concentrate on the preferred region of the Pareto-optimal frontier. According to the introduced algorithm called *Guided MOEA (G-MOEA)*, the decision maker provides linear trade-offs between objectives instead of goals or weights. Then, the obtained linear utility functions with slopes as the maximum and minimum trade-off values are used to obtain a modified domination criterion. Following this new guided dominance scheme, nondominated sorting of the population is performed. Guided nondomination orients the population to the

desired part of the Pareto-optimal frontier since the whole Pareto-optimal frontier is no more nondominated according to the new nondomination scheme. In G-MOEA, diversity along the desired region of the frontier is provided by a modified fitness sharing approach.

Another study on preference incorporation has been performed by Chetković and Parmee (2002) in which fuzzy logic has been used to characterize the relative importance of objectives for the decision maker. *Binary preference relations* obtained from the decision maker are translated into qualitative values that can be used as the weights of the objectives. This transformation process is where the fuzzy preference relations come on the stage. From the binary preferences obtained, fuzzy preference matrices are formed from which objective weights can be obtained. Once the weights of the objectives are determined, the authors suggest different methods to reach the desired solutions on the Pareto-optimal frontier. First method is the "weighted sum-based optimization method" in which objectives are aggregated into a single objective function using the weights obtained. Moreover, a new domination criterion is introduced called "*weighted domination*". Using this new definition of nondomination, a weighted Pareto-optimal frontier is obtained which is a subset of the original Pareto-optimal frontier that is of interest for the decision maker.

Another study on *fuzzy preference relations* has been performed by Jin and Sendhoff (2002). Interval-based weights are incorporated into MOEA using a proposed approach called *"evolutionary dynamic weighted aggregation"*. Two different methods are proposed: Random weighted aggregation (RWA) and dynamic weighted aggregation (DWA). The weights are varied within the interval obtained using fuzzy preference relations.

Biased crowding distance introduced by Branke and Deb (2004) is based on the idea of *biased sharing approach* (Deb, 2001, pp 379-382). The approach is integrated in NSGA-II where nondominated sorting is used to ensure convergence

and crowding distance measure is used to provide diversity. However, the approach modifies the crowding distance measure by using the most probable linearly weighted utility function obtained from the decision maker. For the regions of the Pareto-optimal frontier which are approximately parallel to the projected utility function, the crowding distance becomes high, while the crowding distance becomes smaller for the regions that have a sharper slope with the projected utility function. Consequently, the solutions with high crowding distances cause the algorithm to focus on the region where the utility function is tangent to the frontier.

Reference Point Based Evolutionary Multi-objective Optimization approach has been proposed by Deb and Sundar (2006). The approach uses the concept of reference point methodology and without utilizing weight vector information, focuses on the region on the Pareto-optimal frontier that is close to the reference point. The result is a set of solutions corresponding to differing weight vectors to facilitate the understanding of the decision maker about the problem situation. By providing more than one reference points, the method can result in finding points in multiple regions of the Pareto-optimal frontier. The authors have developed reference-point-based NSGA-II (R-NSGA-II) by incorporating their approach into NSGA-II. The proposed algorithm differs in its crowding distance calculation in that, the crowding distance of a solution is determined according to its Euclidean distance to the reference point. The algorithm can focus on multiple regions of the Pareto frontier, and can obtain multiple solutions on each of these regions.

A study similar to that of Deb and Sundar (2006) has been proposed by Deb and Kumar (2007), which uses the *reference direction approach* instead of reference points. The approach is incorporated into an evolutionary algorithm method, *reference direction based NSGA-II (RD-NSGA-II)*. The algorithm chooses a set of points in the reference direction provided by the DM. For each of the points selected, the individual in the population having the minimum value of

achievement scalarizing function is determined. Then, this set of population members form the first nondominated front and the process goes on this way until all population members are assigned to their corresponding rank of nondomination. Reference direction procedure performs preference incorporation while sorting the population into nondomination classes. The procedure results in a region of the Pareto-optimal frontier where the reference direction is projected on. The reference direction method can also be extended to deal with multiple reference directions.

Phelps and Köksalan (2003) have proposed their own evolutionary metaheuristic that incorporate the preferences of the decision maker into the search process with the aim of solving combinatorial optimization problems. The *Interactive Evalutionary Metaheuristic (IEM)* proposed by the authors aims to end up with the preferred solution of the decision maker by interactively guiding the search effort. The partial preference information obtained from the decision maker in terms of binary preference relations between individuals are directly used in fitness calculations through finding the middlemost feasible weights. In each iteration of IEM, the estimated utility function is updated according to the preference information obtained from the decision maker and the individual having the best estimated utility value is considered as an incumbent. The incumbent is also updated in each iteration, until the decision maker is satisfied with the current incumbent. IEM aims to end up with a single solution of interest for the decision maker by interactively exploring the feasible space.

Another metaheuristic developed by Köksalan and Phelps (2007), called *evolutionary metaheuristic for approximating preference-nondominated solutions (EMAPS)*, gets partial preference information from the decision maker, not to end up with a single efficient solution, but to concentrate the search effort onto a region of the Pareto-optimal frontier. The algorithm is based on partial preference information case where the decision maker provides some restricted amount of information in qualitative terms a priori and this imprecise information is used to

concentrate the search effort to the desired regions of the Pareto-optimal frontier. The qualitative preference information is translated into constraints to restrict the weight space. The fitness value of an individual is assigned by taking into consideration the weight vector within the restricted weight space that gives the highest advantage to that individual and whether that individual has a potential to approximate the underrepresented portions of the preference-nondominated solution set of the decision maker.

Karahan and Köksalan (2008) have implemented a preference incorporation mechanism into TDEA. According to this mechanism, the size of the territories are altered such that, smaller territory sizes are determined for the preferred regions of the decision maker in order to obtain more solutions in these regions. The mechanism is used in two proposed versions of TDEA, the a priori version called *Preferred-Region TDEA (prTDEA)* and the interactive version called *Interactive TDEA (iTDEA).* In prTDEA, the decision maker specifies his/her preference regions a priori in terms of weight sets, where multiple regions are allowed. If an offspring is in a preferred region, it is evaluated using the territory size associated to that region. prTDEA obtains a better approximation of the preferred regions compared to TDEA. Karahan (2008) has also presented an interactive version of the algorithm, named as iTDEA. In this interactive version, the preferred region of the decision maker is approximated step by step during the optimization stage as information is obtained from the decision maker. For this purpose, interaction stages are determined at prespecified generations and a more focused and smaller preference region is estimated at each of these interaction stages. Preference incorporation results in better details in the regions of interest with higher computational efficiency.

2.2.3 Multi-objective Combinatorial Optimization

This section consists of an overview of multi-objective combinatorial optimization literature. We recommend the book written by Ehrgott (2000, pp 153-211) for further details on MOCO problems and the solution methods. Moreover, a comprehensive bibliography is provided by Ehrgott and Gandibleux (2000) on various MOCO problems.

Combinatorial optimization problems, having a finite number of feasible solutions, are special cases of integer programming problems. A MOCO problem is characterized by a specific set of constraints that give the problem its structure. Both the feasible objective space and the feasible decision space consist of a finite set of points for a MOCO problem (Ehrgott and Gandibleux, 2000). As Ehrgott (2000) mentions, MOCO problems do not have convex solution spaces and therefore there usually exist unsupported efficient solutions. In a MOCO problem, there are usually much more unsupported efficient solutions than supported ones and they highly contribute to the computational complexity of MOCO problems (Ehrgott and Gandibleux, 2000).

MOCO problems are generally NP-complete, #P-complete and intractable. That is, it becomes impossible to propose an efficient method to determine all of the Pareto-optimal solutions since the number of these solutions may be exponential in problem size (Ehrgott and Gandibleux, 2000).

The solution methods for MOCO problems are classified into two by Ehrgott and Gandibleux (2000): Exact methods and approximation methods. Exact Methods are *weighted sum scalarization method,* where all supported efficient solutions can be obtained; *compromise solution method; where* the Pareto-optimal frontier can be found theoretically by minimizing the distance to an ideal point; *goal programming; ranking method,* which is developed to solve two-objective problems for which the efficient frontier can be generated by finding the K-best

solutions between the ideal point and nadir point with respect to one of the objectives; and the *methods adapted from single objective combinatorial optimization*, like Branch-and-Bound and the Hungarian method for the single-objective assignment problem.. In the literature, 2-phase method is very popular and is addressed in several papers. In this method, the optimization process is divided into two phases. In the first phase, the supported efficient solutions are obtained, which can be easily found by weighted-sum scalarization of the objectives. In the second phase, the information obtained from the supported efficient solutions is used to find the unsupported efficient solutions. Przbylski et.al. (2007) have applied the two-phase method to solve bi-objective assignment problem, while Visée et.al. (1998) have applied the same method to bi-objective knapsack problem. In the second phase of the algorithm, Visée et.al. (1998) use Branch and Bound method to obtain the unsupported efficient solutios.

The approximation methods to solve MOCO problems are discussed in detail in a review paper by Ehrgott and Gandibleux (2004). Since MOCO problems are very hard to solve exactly, approximation methods are highly popular in MOCO literature. The heuristics that are developed to solve the single objective combinatorial optimization problems can generally be utilized to generate heuristics for their multiobjective counterparts. The authors divide the heuristic methods into two classes: the *local search methods* and the *population based methods*. They also overstrike that, different MOCO problems need problem-specific adaptations of the metaheuristics and the performance of the algorithm is highly affected from these adaptations.

Both single objective and multi-objective knapsack problems are NP-hard, while it is not yet known whether the problem is #P-complete or intractable (Ehrgott, 2000). The number of supported solutions grows linearly with the problem size while the number of unsupported solutions grows exponentially (Ehrgott and Gandibleux, 2000). There exist a heuristic approach to solve the knapsack problem, called the greedy method, that uses the weight ratios c_j^m / a_j. For the

single objective knapsack problem, the weight ratios are sorted in a non-increasing order, and the items are put into the knapsack in this order as long as the total weight of the items does not exceed the capacity of the knapsack. In order to solve the two-objective knapsack problem, the Branch and Bound method can be used (Ehrgott and Gandibleux, 2000).

When we come to MOAP, finding all efficient solutions of the problem is NP-complete, #P-complete, and intractable (Ehrgott, 2000). Ehrgott (2000) mentions that, athough it may take an exponential time, all Pareto-optimal solutions of two-objective assignment problem can be obtained by using "two-phase method". In fact, the literature on MOAP is generally restricted to the two-objective case. Since the constraint matrix of assignment problem is totally unimodular, the single-objective problem can be solved as a linear programming problem. The single-objective assignment problem can be solved by using the Hungarian method. However, Ehrgott (2000) mentions that, in multi-objective case, total unimodularity of the constraint matrix only serves to find the supported solutions.

Ehrgott and Gandibleux (2004) discuss the solution of MOCO problems by using MOEAs. They claim that, for the knapsack problem, seeding the initial population by greedy solutions or supported efficient solutions improves the performance of the algorithm. Jaszkiewicz (2002) has proposed a genetic local search approach to solve MOKP. The local search is provided by introducing a revised greedy heuristic that takes a weighted aggregation of objectives into the sorting process. Chu and Beasley (1998) have developed a genetic algorithm for the multi-dimensional knapsack problem. Both of these papers are good resources to see the representation of the knapsack problem in evolutionary algorithms.

Although there are many proposed MOEAs in the literature, we come across to a few MOEAs that are specially developed to solve MOCO problems. Moreover, such algorithms are generally designed to solve a specific MOCO problem. In this study, our aim is to fill this gap in the literature, and develop a general MOEA to

applicable to MOCO problems. While the general outline of our algorithm is applicable to any MOCO problem, special modifications are required to solve a specific MOCO problem. In order to demonstrate the required modifications, we deal with MOKP and MOAP, and show the detailed versions of the algorithm for these two problems. While the MOEA literature has started to address preference-based algorithms, there are a few algorithms that incorporate the preferences of the DM interactively. Our study also contributes to this part of the literature, since we propose a preference-based version of our algorithm that elicits preference information from the DM in an interactive manner. The interactive algorithm converges to the desired portion of the Pareto-optimal frontier, and is able to end up with a single solution to present to the DM.

CHAPTER 3

COMBINATORIAL TERRITORY DEFINING EVOLUTIONARY ALGORITHM (C-TDEA)

In this chapter, we present Combinatorial Territory Defining Evolutionary Algorithm (C-TDEA) that is developed to solve multi-objective combinatorial optimization problems. The algorithm is a modified version of Territory Defining Evolutionary Algorithm (TDEA) that has been proposed by Karahan and Köksalan (2008) as a steady-state and elitist multi-objective evolutionary algorithm. While TDEA is proposed to solve continuous and unconstrained optimization problems, we propose a modified algorithm to solve combinatorial optimization problems. TDEA has the aim of approximating the Pareto-optimal frontier while maintaining a good diversity of solutions over the frontier approximation. There are also preference incorporated versions of TDEA, preference region TDEA (pr-TDEA) and interactive TDEA (i-TDEA) (Karahan, 2008) where i-TDEA is addressed in Chapter 4. TDEA can maintain a uniform diversity in the final population by defining a territory around each individual solution which is not allowed to be violated by any other solution. In TDEA, there exist two coevolving populations: The regular population (P) and the archive population (A). While the regular population consists of both dominated and nondominated solutions, the archive population keeps the individuals, which are nondominated with respect to the regular population on hand and well-dispersed. That is, while the solutions in the archive population are nondominated with respect to each other, they also do not violate the territories of each other. Consequently, the archive population update procedure not only deals with the nondomination issue, but also the territory violation is checked. The territory defining property of TDEA therefore results in a diverse set of individuals in the archive population and provides computational advantage to the programmer.

Karahan and Köksalan (2008) define the territory of a solution X_p as follows: "The region within a distance τ of X_p in each objective among the regions that neither dominate nor is dominated by X_p". Defining a territory around an archive population member represents a forbidden region around that archive member which cannot be occupied by any other solution that competes for acceptance into the archive population. The territory of a solution is demonstrated in Figure 3.1.

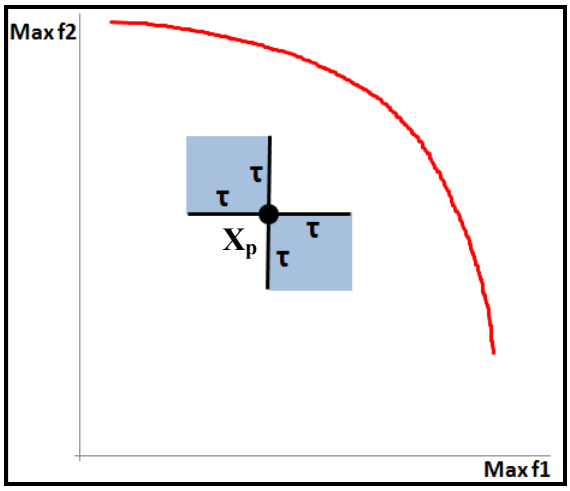

Figure 3.1 Illustration of Territory in Two-dimensional Space

The organization of this chapter is as follows: We present the general outline of C-TDEA in Section 3.1, where the general outline of the original TDEA is preserved as a skeleton. In C-TDEA, territory definition is mainly inherited from the original TDEA. However, since combinatorial optimization problems are substantially different from continuous optimization problems, most of the operators are modified and new operators are included into the algorithm. The general outline of C-TDEA is applicable to any MOCO problem. However, special modifications are required to solve a specific MOCO problem by using the proposed algorithm. In our study, we address multi-objective knapsack and assignment problems in order to illustrate the required modifications. We explain the steps of the algorithm for the multi-objective knapsack problem in detail in Section 3.2. Similarly, we explain the steps of the algorithm for the multi-

objective assignment problem in detail in Section 3.3. We explain the determination of the territory size τ in Section 3.4. Finally, we present the experimental results and comparisons in Section 3.5.

3.1 GENERAL OUTLINE OF C-TDEA

Below is the general outline of C-TDEA that is applicable to any MOCO problem:

1. Initialization: Set the initial population size $\overline{N}$, the territory size τ, and the maximum number of generations to iterate the algorithm T, such that the final population size is acceptable for the DM. Set the iteration counter $t = 0$.
2. Initialization of the Regular Population: Obtain $\overline{N}$ individuals to fill the initial regular population P(0) by creating $\overline{\overline{N}}$ seed solutions, and $\overline{N}$ - $\overline{\overline{N}}$ random individuals.
3. Initialization of the Archive Population: Initialize the archive population with the copies of the solutions in P(0) that are nondominated with respect to the solutions in P(0).
4. Parent Selection: Set $t \leftarrow t+1$. Choose one parent from the regular population, and one parent from the archive population
5. Recombination: Recombine the parents to create the offspring.
6. Mutation: Apply mutation to the offspring.
7. Repair and Improvement of the Offspring: If the offspring is infeasible, then repair the offspring such that it becomes feasible. Then, improve the offspring if there is place for improvement.
8. Acceptance into the Regular Population: Check whether the offspring satisfies the acceptance condition for the regular population. If it is accepted, insert it into P(t). Otherwise, go to step 10.

9. Acceptance into the Archive Population: If the offspring is accepted into P(t), check whether it satisfies the acceptance condition for the archive population. If it is accepted, insert it into A(t).
10. Stopping Condition: Stop if the pre-specified iteration limit T is hit (t=T), and report the archive population to the DM. Otherwise, go to step 4.

3.2 C-TDEA FOR MULTI-OBJECTIVE KNAPSACK PROBLEM

3.2.1 Initialization of the Regular Population

In the original version of TDEA, the initial regular population P(0) is randomly created until the regular population size $\overline{N}$ is reached and no seeding is utilized. In C-TDEA, for an M-objective knapsack problem, $\overline{\overline{N}} = \sum_{m=1}^{m=M-1} \binom{M}{m}$ high-quality solutions are seeded into the initial population. $\overline{N}$ should be set carefully for large number of objectives since $\overline{\overline{N}}$ increases with the number of objectives. We set the formulation of $\overline{\overline{N}}$ for small number of objectives and utilize this formulation with such problems. Therefore, the value of $\overline{\overline{N}}$ does not constitute any problem in this study. The remaining $\overline{N} - \overline{\overline{N}}$ solutions are still randomly generated such that, each item is put into the knapsack with probability 0.5. The solutions are represented by binary chromosomes where each gene of the chromosome corresponds to an item. The value in a gene is 1 if the corresponding item is put into the knapsack; and the value in that gene is 0 if the item is not put into the knapsack. However, a randomly filled solution may violate the knapsack capacity and become infeasible. Consequently, all of the solutions in P(0) are checked for feasibility and the infeasible solutions are repaired by removing items from the knapsack. Moreover, each of the initial population members are checked for improvement such that, items are placed into the solutions which have available space until the knapsack capacity does not allow any more items to be

put into the knapsack. These greedy algorithm based repair-improvement procedures are discussed in Section 3.2.5.

The remaining $\overline{\overline{N}}$ slots of the initial population P(0) is filled with the high-quality seed solutions. M of these $\overline{\overline{N}}$ seed solutions are the approximately optimal solutions of the single objective knapsack problems when we consider each of the M objectives separately. That is, for a 2-objective knapsack problem, these two solutions are the approximations of the two extreme points of the Pareto-optimal frontier. These M seed solutions are obtained heuristically by using the greedy algorithm on each of the single objective knapsack problems. That is, for the seed solution which aims to approximate the efficient solution with the best value of objective *i*, the items are sorted in the non-increasing order of their contribution to objective *i* per unit capacity; c_{ij} / a_j where *j* corresponds to the items. Afterwards, the knapsack is filled in order of decreasing ratios as long as the knapsack capacity is not violated. Then, the objectives are grouped in sets of 2 considering every possible combination, and $\binom{M}{2}$ seed solutions are obtained, for each case giving each objective equal Tchebyceff weights. This procedure continues until these combination sets cover each possible combination of objectives from size 2 to M-1. This seeding procedure is given below for a combination set of size *l*:

1. Set iteration counter k = 0, and $z_{10} = z_{20} = \ldots = z_{l0} = 0$.
2. For each item *j* that is not in the current knapsack and that does not violate the knapsack capacity, calculate:

   $$Min\left\{ z_{1k} + \frac{c_{1j}}{a_j}, z_{2k} + \frac{c_{2j}}{a_j}, \ldots, z_{lk} + \frac{c_{lj}}{a_j} \right\} = \frac{c_{tj}}{a_j}$$

   If there is no such item, go to step 4.

3. Set k←k+1. Find $\underset{j}{Max}\left\{\frac{c_{tj}}{a_j}\right\}=\frac{c_{tr}}{a_r}$. Insert item r into the current knapsack.

   Set $z_{1k}=z_{1(k-1)}+\frac{c_{1r}}{a_r},\ldots,z_{lk}=z_{l(k-1)}+\frac{c_{lr}}{a_r}$. If there is place in the current knapsack, go to Step 2.
4. Stop the item acceptance procedure and insert the solution into P(0).

Consequently, repeating the above procedure for l = 2,…, M-1, the initial population is heuristically seeded with $\overline{\overline{N}}=\sum_{m=1}^{m=M-1}\binom{M}{m}$ solutions.

3.2.2 Parent Selection

The parent selection procedure of C-TDEA is similar to that of original TDEA. The parent from the regular population is selected by binary tournament selection. Two solutions are randomly picked from the regular population and they are tested for dominance. If one dominates the other, dominating solution is determined as the first parent. If the solutions are nondominated to each other, one of them is selected randomly as the first parent.

Since we cannot consider any dominance relation within the archive population consisting of relatively nondominated solutions, the second parent is selected from the archive randomly.

3.2.3 Crossover and Mutation

The crossover operator used is the Uniform crossover operator. In each generation, two offspring are generated by a single crossover. For each gene, child 1 inherits from parent 1 with probability 0.5 and inherits from parent 2 with

again probability 0.5. If child 1 inherits a gene from parent 1, then child 2 inherits that gene from parent 2 and vice versa.

On the other hand, the mutation operator used is single bit flip mutation operator. If we determine the mutation probability as p_m, a randomly selected gene of the child is flipped with probability p_m.

3.2.4 Scaling

The original TDEA uses the sigmoid function scaling (Soylu and Köksalan, 2006). In sigmoid scaling, all objective values are scaled in [0, 1] interval. The important point is that, while the objective values between the ideal and nadir point are linearly scaled, objective values beyond the nadir point are scaled by using a sigmoid function. The aim is to scale the nondominated range into a higher portion of the [0, 1] range. However, this sigmoid function scaling is formulated for minimization problems. Instead of using that sigmoid function by converting the maximization type objectives of the knapsack problem into minimization type objectives, a new sigmoid scaling function has been proposed for maximization type problems. Below is the formula of this sigmoid scaling function:

$$\hat{f}_i = \begin{cases} \psi_{f_i} = e^{f_i/\lambda} - 1 & \text{if} \quad f_i \le f_i^{nadir} \\ \psi_{nadir} + \dfrac{1-\psi_{nadir}}{f_i^{ideal} - f_i^{nadir}} * (f_i - f_i^{nadir}) & \text{otherwise} \end{cases} \tag{3.1}$$

In this formula, f_i stands for the i^{th} objective value of the solution to be scaled, and ψ_{fi} stands for the scaled value of the objective if the objective value is smaller than the nadir point value f_i^{nadir}. In this formula, there exists a parameter λ which controls the slope of the sigmoid function. The value of this parameter is determined by trial-and-error such that, the nadir point is scaled to 0.1. If the

above formulation is analyzed, it can also be seen that the ideal point is scaled to the value 1 in any case. Consequently, the value of λ determines the scaled range of the efficient range, and this range is approximated to 0.9 in all of the studies discussed in this report by appropriately setting λ. The shape of this sigmoid function is illustrated in Figure 3.2.

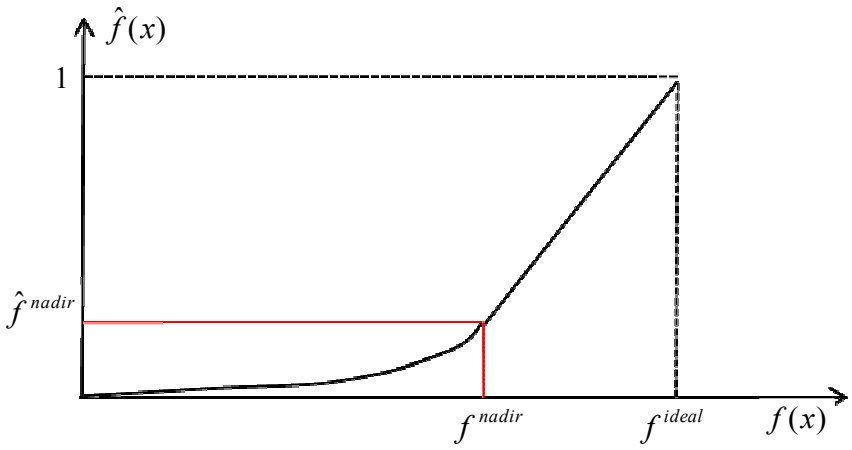

Figure 3.2 Sigmoid and Linear Function Scaling for Maximization Problems

3.2.5 Repair and Improvement of the Solutions

Repair and improvement are two procedures that are used on both the initial population and the offspring. The discussion from now on is performed on the offspring; however the initial population members are also subject to these two procedures in the same manner as the offspring.

The common property of these two procedures is that, a modified greedy algorithm is utilized in the direction of the Tchebycheff favorable weights of the solution. These directions can be seen in Figure 3.3. The motivation behind using favorable weights issue is the fact that each solution on the Pareto-optimal frontier has a unique weight set which makes its weighted Thebycheff distance to

the ideal point minimum compared to all other feasible solutions. Therefore, by emptying and filling the knapsack of a solution with the aim of minimizing its Tchebycheff distance to the ideal point in the direction of its favorable weights, we can keep the original direction of the individual in which it contributes to the convergence. Our aim is not only to converge to the Pareto-optimal frontier, but also make each solution to converge to the region on the frontier that is represented by the Tchebycheff favorable weights of that solution. Therefore, once we determine the repair and improvement directions of a solution with respect to its favorable weights, we try to keep the repaired and/or improved solution as close as possible to this direction of the original solution.

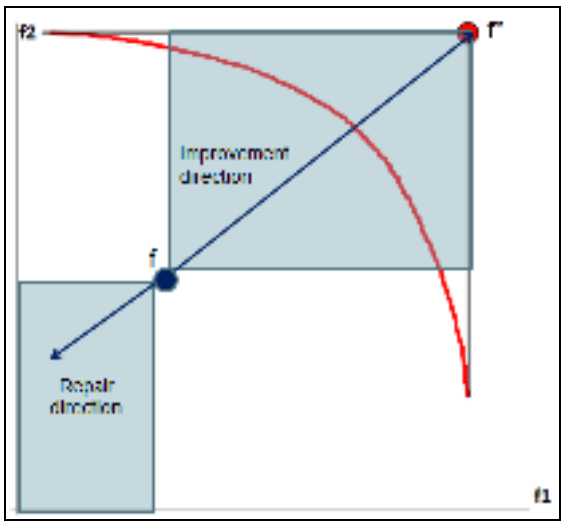

Figure 3.3 C-TDEA Repair and Improvement Directions

Before carrying out repair and improvement procedures, the weighed Tchebycheff distance of the initial point z to the ideal point z^* is calculated as follows where λ_i is the favorable weight of the solution z with respect to objective i:

$$\lambda_k(\bar{z}_k^* - \bar{z}_k) = \underset{i \in M}{Max}\{\lambda_i(\bar{z}_i^* - \bar{z}_i)\} \tag{3.2}$$

where $\bar{z}_i$ is the scaled value of z_i.

In the following subsections, we will show how this distance changes after removal of an item from the knapsack or addition of an item to the knapsack.

3.2.5.1 Repair of the Infeasible Solutions

If an offspring or an initial population member is infeasible, then the following procedure is carried out:

1. Calculate the favorable weights, λ_i's, of the solution.
2. Calculate Tchebycheff distance after moving out item *j* from the knapsack for each item *j* which is in the current knapsack:

$$d_j = \underset{i \in M}{Max} \left\{ \lambda_i \left(\bar{z}_i^* - \left(\overline{z_i - c_{ij}} \right) \right) \right\} \tag{3.3}$$

3. Calculate d_j / a_j for each item *j* which is in the current knapsack, where a_j is the constraint coefficient of item *j*.
4. Find $\text{Min}_j \{ d_j / a_j \} = d_k / a_k$
5. Discard item x_k from the knapsack.
6. If the new solution is infeasible, turn back to Step 1 and repeat the same procedure using the favorable weights of the new solution until a feasible solution is obtained. Otherwise, finish the repair procedure.

3.2.5.2 Improvement of the Solutions – Genetic Local Search

If an initial population member or an offspring is feasible or becomes feasible after the repair procedure, it is evaluated for improvement. The procedure is as follows:

1. Calculate the favorable weights, λ_is, of the solution.
2. For each item *j* which is not in the current knapsack and addition of which does not violate the knapsack capacity, calculate Tchebycheff distance after placing item *j* into the knapsack:

$$d_j = \underset{i \in M}{Max} \left\{ \lambda_i (\bar{z}_i^* - (\overline{z_i + c_{ij}}) \right\} \tag{3.4}$$

3. Calculate $(1-d_j) / a_j$ for each item *j* which is not in the current knapsack and addition of which does not violate the knapsack capacity, where a_j is the constraint coefficient of item *j*.
4. Find $\text{Max}_j \{ (1-d_j) / a_j \} = (1-d_k) / a_k$
5. Place item x_k into the knapsack.
6. If there is place for improvement of the new solution, turn back to Step 1 and repeat the same procedure using the favorable weights of the new solution until there is no place for improvement. Otherwise, finish the improvement procedure.

While calculating the Tchebycheff distance d_j, both of the objective values are scaled, where the scaled ideal objective value, $\bar{z}_i^*$, is equal to 1. Therefore, the value of d_j is between 0 and 1. In step 3 of the above improvement procedure, we subtract d_j from 1 and calculate $(1-d_j) / a_j$, since we need the item with the smallest d_j and a_j values in Step 4 of the procedure.

3.2.6 Regular and Archive Population Updates

C-TDEA inherits the regular population update procedure from the original version of TDEA. After the offspring is repaired if necessary and improved if possible, it is evaluated for acceptance into the regular population. The offspring is tested against each regular population member for dominance. If the offspring is dominated by at least one regular population member, then the offspring is

rejected. On the other hand, if the offspring is not dominated by any of the population members, then one of the solutions that is dominated by the offspring is removed from the regular population if any. If no such dominated solutions exist, then an individual is removed randomly and the offspring is inserted into the regular population.

The important issue at this point is that, regular population acceptance is a prerequisite of evaluation for archive acceptance. If the solution is not accepted into the regular population, it is not even evaluated for acceptance into the archive and it is directly rejected. On the other hand, if the offspring is accepted into the regular population, the archive update procedure starts.

Although the general outline of the archive acceptance procedure is similar to the original TDEA, some modifications are performed in this step. Below is the outline of the archive acceptance procedure:

Stage 1:

Test the offspring against each individual of the archive population. Mark the individuals that are dominated by the offspring. If the offspring is dominated by at least one archive member, it is rejected. If the offspring is not rejected, go to Stage 2.

Stage 2:

1. Remove all of the marked individuals from the archive population.
2. Accept and insert the offspring into the archive if the archive is empty before the insertion, stop. Continue otherwise.
3. Calculate the Tchebyceff distance of the offspring to each individual of the archive.
4. Find the closest individual s_i^* to the offspring in terms of Tchebycheff distance: δ.
5. Accept and insert the offspring into archive if $\delta \geq \tau$. Otherwise, reject the offspring.

The modification in archive acceptance procedure is at the third step of Stage 2 where the nearest neighbor of the offspring is found by using Tchebycheff distance metric. In the original version of TDEA, rectilinear distance is used to find the nearest neighbor of the offspring, and then the Tchebycheff distance of the offspring to that point is compared to the territory size τ. However, the utilization of rectilinear distance to obtain the closest solution of the offspring may result in violation of the claim that there exists only one solution in each hyper-box defined by the territories of the existing solutions.

This violation can be proven by a counter-example in which, although there exists a solution violating the territory of the offspring, by using rectilinear distance, a solution that does not violate its territory is found to be its closest solution. The counter-example is illustrated in Figure 3.4. In the figure, c is the offspring solution and the solutions s1 and s2 are two existing solutions that are evaluated to be the closest solution to the offspring. The rectilinear distance between offspring c and the solutions s1 and s2 are 7 and 8 units respectively. Assuming that there does not exist any other solution that is closer to the offspring than these two solutions, the closest solution to the offspring becomes s1 if we consider rectilinear distance. On the other hand, the Tchebycheff distance between c and s1 is 5 units. If τ is determined to be 4.5 units as in the figure, the territory of s1 is found to be preserved, and the offspring is accepted into the archive population. However, having a Tchebycheff distance of 4 units to the offspring, the territory of s2 is violated by the offspring. Therefore, the use of rectilinear distance to determine the closest neighbor may fail to evaluate a solution the territory of which is violated by the offspring, if the solution has a relatively high rectilinear distance to the offspring.

Figure 3.4 illustrates the motivation behind the modification that has been performed in archive acceptance procedure.

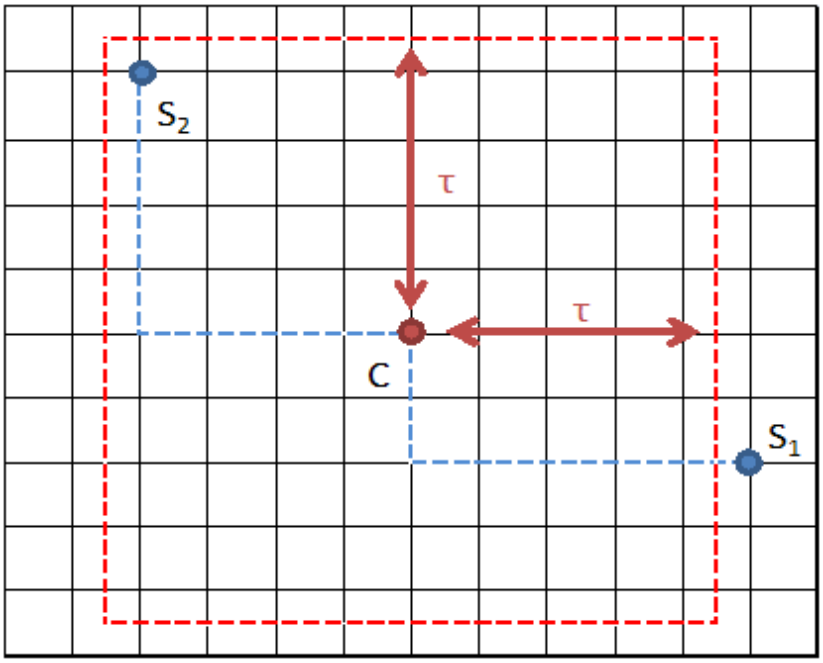

Figure 3.4 Violation of territories when rectilinear distance is used to determine the closest neighbor of the offspring

3.3 C-TDEA FOR MULTI-OBJECTIVE ASSIGNMENT PROBLEM

3.3.1 Initialization of the Regular Population

As in the case of multi-objective knapsack problem, $\overline{\overline{N}} = \sum_{m=1}^{m=M-1} \binom{M}{m}$ high-quality solutions are seeded into the initial population for multi-objective assignment problem. The remaining $\overline{N} - \overline{\overline{N}}$ solutions are randomly generated. The solutions are represented by integer-valued chromosomes where each gene of the chromosome corresponds to a person, while the value kept in that gene corresponds to the job assigned to that person. For instance, the gene (2 3 1) represents the case where person 1 is assigned to job 2, person 2 is assigned to job 3, and person 3 is assigned to job 1. In contrast to the knapsack problem, a randomly created solution is not allowed to be infeasible in the assignment problem case. That is, the random assignment of jobs to the people starts from the

first gene, that is, the first person, and goes on until all people are assigned their corresponding jobs. However, a previously assigned job is not allowed to be re-assigned to another person during this iterative process. Consequently, the constraints of the assignment problem, each person assigned to a single job and each job assigned to a single person, are provided by the collaboration of the chromosome representation scheme and the random generation process. Since, there is no need for a feasibility check procedure, each member of the initial population is directly checked for improvement such that, each person tries to change his assigned job with the people that come after that person in the chromosome representation. This improvement procedure is discussed in Section 3.3.5.

Due to the fact that the single objective assignment problem can be solved as a linear programming problem in polynomial time, the seed solutions are not determined heuristically, but they are found optimally by mathematical modeling. Again, M of the $\overline{\overline{N}}$ seed solutions are the optimal solutions of the single objective assignment problems when we consider each of the M objectives separately. The procedure of obtaining a combination set of objectives that cover each possible combination of objectives from size 2 to M-1 is performed in the same manner as the knapsack problem. By giving equal Tchebyceff weights to each of the objectives in the combination, the following linear programming model is solved for a combination of size 2:

$$
\begin{aligned}
&\textit{Minimize} \quad \varepsilon \\
&\textit{subject to} \quad \sum_{i=1}^{n} x_{ij} = 1; j = 1,..,n \\
&\qquad \sum_{j=1}^{n} x_{ij} = 1; i = 1,..,n \\
&\qquad f_1^* - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^1 x_{ij} \le \varepsilon \\
&\qquad f_2^* - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^2 x_{ij} \le \varepsilon \\
&\qquad x_{ij} \in \{0,1\}
\end{aligned}
\tag{3.5}
$$

Solving the above model repeatedly for combination sizes of l = 2,…, M-1, the initial population is heuristically seeded with $\overline{\overline{N}} = \sum_{m=1}^{m=M-1} \binom{M}{m}$ solutions.

3.3.2 Parent Selection

The parent selection procedure is similar to that of C-TDEA for the knapsack problem. The parent from the regular population is selected by binary tournament selection, while the second parent is selected from the archive randomly.

3.3.3 Crossover and Mutation

The crossover operator used is the Cycle crossover (CX) operator (Larranaga et.al., 1999). In each generation, two offspring are generated by a single crossover. The working principle of the operator lies on the fact that, each gene position of the offspring is inherited from one of the parents. That is, each person in the offspring is assigned to the same job of parent 1 or parent 2.

At the beginning, a gene position g is selected randomly as the cycle initialization position. One of the offspring inherits that gene value from the first parent, and the second offspring inherits the gene value from the second parent. For the first offspring, the gene value inherited from the first parent cannot be inherited from the second parent once more, consequently, according to the working principle of the operator, the position on the second parent whose value is inherited from the first parent is found, and that gene position is again inherited from the first parent. This cycle continues until the beginning gene value is hit. Then, a new cycle is started from a position which is not yet assigned, and these cycles continue until all genes of the offspring are assigned their corresponding inherited values. At each cycle, the parents from which the genes are inherited are exchanged between the two offspring.

Figure 3.5 illustrates this crossover process on a single offspring for an assignment problem of size eight. In the example, the first cycle starts at the first gene of Parent 1, which is selected randomly, and the offspring inherits that gene position from Parent 1. The cycle continues by searching the value inherited (1) in Parent 2. The gene value occurs at the third gene position of Parent 2. Since this gene position cannot be inherited from Parent 2, the crossover operator inherits the third gene position of the offspring from Parent 1. When we search for this inherited value (3) in Parent 2, we come back to the first gene location, and the first cycle terminates. Then, the second cycle starts from a randomly selected gene position that is not currently assigned. But this time, the inheritance process works for Parent 2. These cycles continue in the same manner until the offspring in Figure 3.5 is obtained.

The mutation operator used is exchange mutation (EM) operator. If we determine the mutation probability as p_m, two randomly selected genes exchange their values with probability p_m.

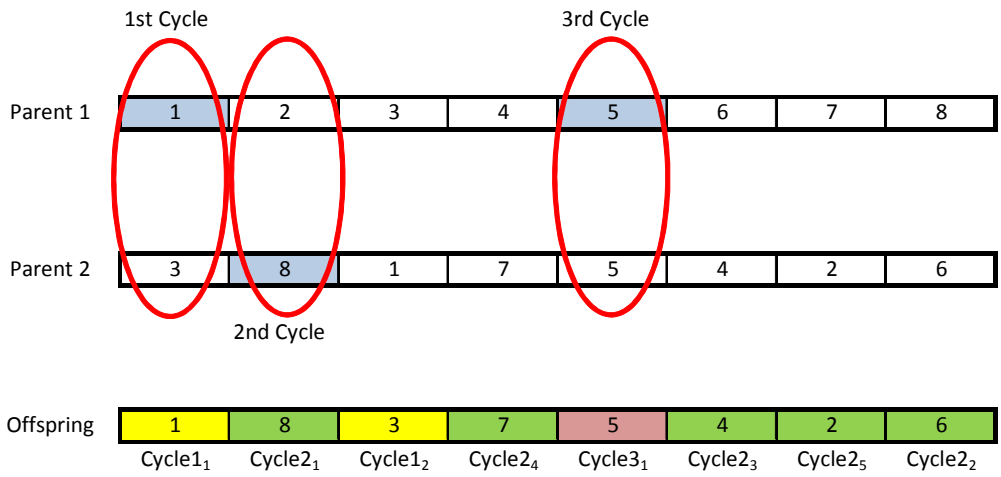

Figure 3.5 Cycle Crossover

3.3.4 Scaling

Since the objectives of the multi-objective assignment problem is of minimization type, sigmoid function scaling used in the original TDEA, which has been formulated for minimization problems, is used (Soylu and Köksalan, 2008). Below is this sigmoid scaling function:

$$\hat{f}_i = \begin{cases} \psi_{f_i} = \left(\dfrac{1}{1+e^{-\frac{\bar{f}_i}{\lambda}}} - 0.5 \right) * 2 & \text{if} \quad f_i \geq f_i^{nadir} \\ \dfrac{\psi_{nadir}}{\bar{f}_i^{nadir}} * \bar{f}_i & \text{otherwise} \end{cases} \tag{3.6}$$

In the formula, $\bar{f}_i$ and $\bar{f}_i^{nadir}$ are the values of the objective value of the solution to be scaled, and the nadir point, respectively that are shifted to scale the ideal point to the value of 0 in any problem. ψ_{fi} stands for the scaled value of the objective if the objective value is greater than the nadir point value, f_i^{nadir}. The value of the parameter λ that controls the slope of the sigmoid function is

determined by trial-and-error such that, the nadir point is scaled to take the value of 0.9. The shape of this sigmoid function is illustrated in Figure 3.6.

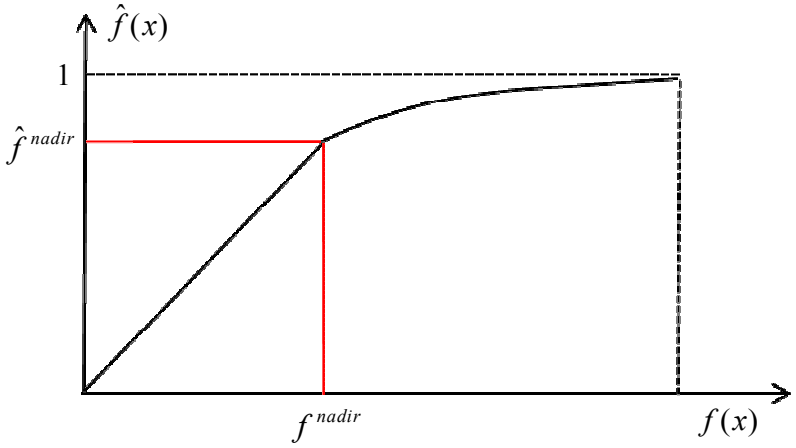

Figure 3.6 Sigmoid and Linear Function Scaling for Minimization Problems

The sigmoid function scaling procedure for minimization problems is as follows:

1. Shift the objective value and nadir point to set the ideal point to 0 as follows: $\bar{f}_i = f_i - f_i^*$ and $\bar{f}_i^{nadir} = f_i^{nadir} - f_i^*$.
2. Determine the value of the parameter λ by trial-and-error such that the nadir point is scaled to approximately 0.9.
3. Find the scaled objective value $\hat{f}_i$ using the above sigmoid scaling function.

3.3.5 Improvement of the Solutions

Improvement procedure is performed on both the initial population and the offspring. We will discuss the procedure on the offspring; however the initial population members are also subject to the same improvement procedure.

The procedure is as follows:

1. Set first iteration counter n =1. Start with the n^{th} gene corresponding to the first person.
2. Set second iteration counter $m = n+1$.
3. Exchange the jobs of person m with the job of person n. If the newly obtained solution dominates the initial solution, keep the jobs exchanged and go to Step 4. Otherwise, undo the exchange, set $m \leftarrow m+1$, and repeat this step. If m=M+1, go to Step 4.
4. Set $n \leftarrow n+1$ and go back to Step 2. If n = M, stop.

According to this procedure, at most $\frac{N \cdot (N-1)}{2}$ comparisons are made. In the initial generations, the comparison number to obtain an improved solution is comparatively smaller. Because the solutions are far away from the Pareto-optimal frontier, and the comparisons are more promising to end up with an exchange. However, since the solutions converge to the Pareto-optimal frontier as the generations proceed, it becomes harder to find an exchange location to improve a solution. Consequently, this improvement procedure is held until a predetermined small number of generations that is determined by preliminary runs.

3.3.6 Regular and Archive Population Updates

The regular population update procedure explained for the knapsack problem is used for the assignment problem as well. After the offspring is improved if possible, it is evaluated for acceptance into the regular population.

Again, the regular population acceptance constitutes the prerequisite for archive acceptance procedure. The archive acceptance procedure is as follows:

Stage 1:

Test the offspring against each individual of the archive population. Mark the individuals that are dominated by the offspring. If the offspring is dominated by at least one archive member, it is rejected. If the offspring is not rejected, go to Stage 2.

Stage 2:

1. Remove all of the marked individuals from the archive population.
2. Accept and insert the offspring into the archive if the archive is empty before the insertion, stop. Continue otherwise.
3. Calculate the Tchebyceff distance of the offspring to each individual of the archive.
4. Find the closest individual s_i* to the offspring in terms of Tchebycheff distance: δ.
5. Accept and insert the offspring into archive if $\delta \geq \tau$. Otherwise, reject the offspring.

3.4 DETERMINATION OF τ

Territory size τ determines a hypervolume around each archive member that is forbidden to be occupied by any other solutions within the archive. It is known that the total nondominated hypervolume is limited (Karahan and Köksalan, 2008). Consequently, territory size has a significant effect on the maximum number of solutions that can enter the archive population and the final archive population size. Reducing τ is expected to increase the final population size. Moreover, a single τ value may result in different final population sizes in different problems since the size of nondominated hypervolume may change from problem to problem with respect to the shape of the efficient frontier. Consequently, the τ values for each problem instance are obtained by trial-and-error in order to obtain approximately the desired number of solutions in the final population as in the case of original TDEA.

3.5 SIMULATION RUNS AND COMPARISONS

We test the performance of C-TDEA on randomly generated 2- and 3-objective problem instances of multi-objective knapsack and multi-objective assignment problems. We also compare the performance of the algorithm against two well-known MOEAs, NSGA-II and ε-MOEA. In order to perform the analyses in a more fair condition, the seeded versions of NSGA-II and ε-MOEA are also considered, in which the seeding mechanism of C-TDEA is utilized.

For the single-dimensional multi-objective knapsack problem, we use binary representation of the chromosomes. We randomly generate 200-item and 750-item problem instances for both 2- and 3-objective cases. The objective coefficients and the knapsack coefficients are randomly generated from a discrete uniform distribution between the values 20 and 100. The objective coefficients for each objective and the knapsack coefficients are assumed to be independently distributed. For each problem instance, the knapsack capacity is set to half the total weight of all items in order to make the problem harder. Since each item is put into the knapsack with probability 0.5 in the initialization stage, the knapsack capacity is efficiently utilized by decreasing the number of repair and improvements operations. This initialization probability can be revised according to the ratio of the knapsack capacity to the total item weight such that the repair-improvement operators can be used more efficiently.

For the multi-objective assignment problem, the chromosomes are represented as integer numbers that take values between 1 and the problem size. We randomly generate 50×50 and 100×100 problem instances for each of 2- and 3-objective assignment problem cases. The objective coefficients are generated randomly and independently from a discrete uniform distribution between the values 20 and 100.

For both multi-objective knapsack and multi-objective assignment problems, the crossover operators are used with 1.0 probability of crossover, while the mutation operators are used with 0.90 mutation probability.

The initial population sizes are determined with respect to the number of objectives and the size of the problem. For the multi-objective knapsack problem, the problem size is determined by the knapsack size, while the number of person-job combinations set the problem size of the multi-objective assignment problem. For the multi-objective knapsack problem, in order to determine the population size corresponding to a certain knapsack size, the suggestions in Zitzler and Thiele (1999) are used. The population sizes that are used in the knapsack problem are also utilized for the assignment problem instances correspondingly.

The ideal points of the problems are obtained by Cplex solver on Gams 22.3. Moreover, the whole Pareto-optimal frontiers are obtained for 2-objective test problems on Gams 22.3 by using ε -constraint method. The formulation is given for a 2-objective knapsack instance as follows:

$$
\begin{aligned}
&Max \quad \sum_j c_{1j}x_j + 0.0001 * \sum_j c_{2j}x_j \\
&s.to \\
&\quad \sum_j a_j x_j \le b \\
&\quad \sum_j c_{2j}x_j \ge \varepsilon \\
&\quad x_j \in \{0,1\}
\end{aligned}
\tag{3.7}
$$

The above model has been solved on Gams 22.3 by increasing the ε value within the range of the nadir and ideal points of the ε-constrained objective in order to ensure that the whole Pareto-optimal frontier is generated. However, in order to decrease the computational effort, the ε value is increased to [1+(value of the ε -constrained objective in the last obtained efficient solution)] in each iteration.

Although the whole Pareto-optimal frontiers have not been generated for 3-objective test problems, ε-constraint method have been applied to those problems by dividing the Pareto-optimal frontier to 100×100 grids and search these grids. That is, the ε value corresponding to objective m is not updated by increments of 1, but it is updated as $\varepsilon = \left\lfloor (f^m_{Ideal} - f^m_{Nadir})/100 \right\rfloor$ in each iteration. The formulation of ε-constrained method for the 3-objective knapsack problem is given as follows:

$$
\begin{aligned}
&Max \quad \sum_j c_{1j} x_j + 0.0001 * \sum_j c_{2j} x_j + 0.0001 * \sum_j c_{3j} x_j \\
&s.to \\
&\quad \sum_j a_j x_j \le b \\
&\quad \sum_j c_{2j} x_j \ge \varepsilon_2 \\
&\quad \sum_j c_{3j} x_j \ge \varepsilon_3 \\
&\quad x_j \in \{0,1\}
\end{aligned}
\tag{3.8}
$$

For all of the problem instances, the value of the scaling parameters λ_m corresponding to objective *m* is set by trial-and-error in order to scale the corresponding nadir point to 0.1 for the knapsack problem, and to 0.9 for the assignment problem.

We run the algorithm 10 times for each problem instance by providing different seeds to the random number generator. Determination of the number of function evaluations and the mutation rate of 0.9 has been inspired by the settings of Köksalan and Karahan (2008). Here, the number of function evaluations refers to the total number of offspring evaluated throughout the simulation run. For TDEA and ε-MOEA which do not have constant population sizes, we perform preliminary runs to set the parameter values to obtain the same final population size as that of NSGA-II.

The parameter settings for the knapsack problem can be seen in Table 3.1, while the parameter settings for the assignment problem are provided in Table 3.2.

Table 3.1 Test Parameters for the Multi-objective Knapsack Problem

Number of Objectives	Knapsack Capacity	Initial Population Size	Function Evaluations	Replications
2	200	200	80000	10
	750	300	120000	10
Number of Objectives	Knapsack Capacity	Initial Population Size	Function Evaluations	Replications
3	200	250	100000	10
	750	400	160000	10

Table 3.2 Test Parameters for the Multi-objective Assignment Problem

Number of Objectives	Problem Size	Initial Population Size	Function Evaluations	Replications
2	50×50	200	80000	10
	100×100	300	120000	10
Number of Objectives	Problem Size	Initial Population Size	Function Evaluations	Replications
3	50×50	250	100000	10
	100×100	400	160000	10

The algorithm is implemented in C++ programming language. The NSGA-II and ε-MOEA codes are taken from the website of Kanpur Genetic Algorithms Laboratory (http://www.iitk.ac.in/kangal/codes.shtml). We build the codes with Microsoft Visual C++ 2008 Express Edition. All computational tests are made on an Intel Core 2 Duo 2.0 GHz, 2 GB RAM computer running Microsoft Windows Vista.

3.5.1 Performance Metrics

We compare the performances of the algorithms by using two performance metrics. The first metric is the Hypervolume metric (Zitzler and Thiele, 1998), which gives the volume of the total objective space dominated by the final population P with respect to a reference point W. Hypervolume metric serves to measure both the convergence and diversity of the final population P. A larger value of the metric is desirable.

Hypervolume indicator can be formulated as follows:

$$Hypervolume = \bigcup_{i=1}^{|P|} V_i^W \tag{3.9}$$

where V_i is the volume of the objective space dominated by solution $i \in$ P with respect to the reference point W. The metric is illustrated in Figure 3.7, where the cross-hatched region corresponds to the hypervolume dominated by the final population P with respect to the reference point W where both objectives are to be maximized.

For both multi-objective knapsack and assignment problems, we use the nadir point as the reference point. Therefore, the extreme points of the Pareto-optimal frontier do not contribute to the Hypervolume metric. Moreover, if an obtained solution is worse than the reference point in any of the objectives, the contribution of that point to the Hypervolume measure is 0.

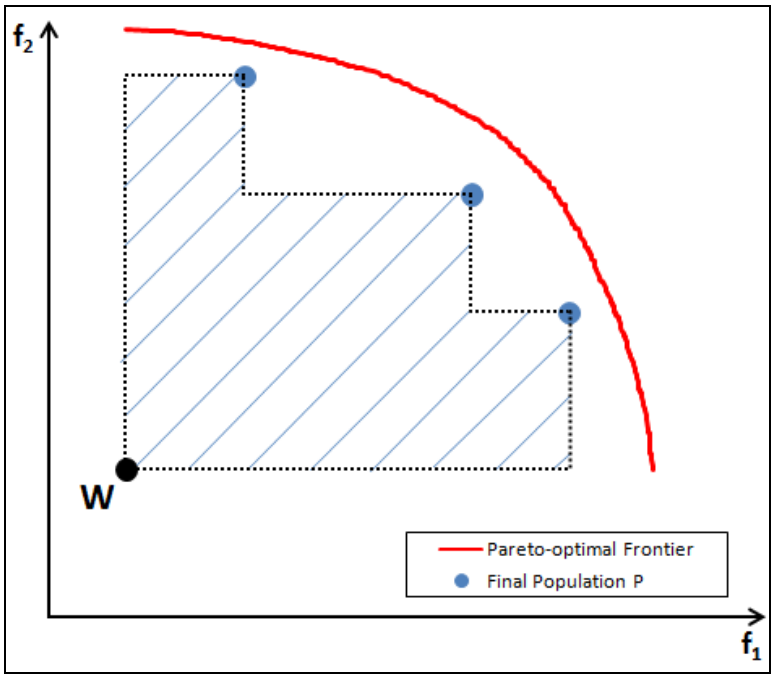

Figure 3.7 Illustration of Hypervolume metric

The second performance metric is the Additive I-Epsilon metric (Zitzler and Kunzli, 2004), which gives the minimum distance by which the final population P needs to be improved in each objective such that the Pareto-optimal frontier is weakly dominated. The metric can be formulated for a minimization-type problem as follows:

$$I_{\varepsilon}^{+}(P,S) = \underset{\varepsilon}{Min}\left\{\forall x^* \in S \quad \exists x \in P: \quad f_i(x) - \varepsilon \le f_i(x^*) \quad for \quad \forall i \in M\right\} \quad (3.10)$$

where S is the Pareto-optimal set. A smaller value of this metric is desirable.

We calculate the performance of the algorithm for each problem in terms of Hypervolume (H) and Additive I-Epsilon metric (I_{ε}^{+}). We assume that the sample means of the metrics are normally distributed By Central Limit Theorem. After

calculating the sample means, $\bar{x}_H$ and $\bar{x}_{I_\varepsilon^+}$, and sample standard deviations, s_H and $s_{I_\varepsilon^+}$, of these metrics, we test the following hypothesis at 95% significance level (Karahan, 2008):

$$\begin{aligned} H_0 &: \mu_{pm}^T = \mu_{pm}^C \\ H_1 &: \mu_{pm}^T \neq \mu_{pm}^C \end{aligned} \tag{3.11}$$

where pm is the performance metric utilized. This hypothesis checks whether the difference between the metric values of C-TDEA and the contender algorithms is statistically significant for both metrics. At the end of the simulation runs, we present the estimated difference between the metric means of TDEA and the contender algorithm, the p-value corresponding to the statistical test, and the winner of the hypothesis test. The metric results of the Pareto-optimal frontiers are also provided.

3.5.2 Multi-objective Knapsack Problem Instances

The figures provided in this section show the final archive populations for C-TDEA and ε-MOEA, and the nondominated solutions of the final population for NSGA-II. The entire final populations of ε-MOEA and NSGA-II including the dominated solutions are provided in Appendix A.

2-objective 200-item Knapsack Problem

Figure 3.8 shows that, while C-TDEA successfully converges to and provides diversity over the Pareto-optimal frontier, NSGA-II and ε-MOEA fail to converge. Indicator values in Table 3.3 also show that C-TDEA has much better convergence and diversity than both of the algorithms in terms of the metric values. The hypervolume metric value for both NSGA-II and ε-MOEA are 0, since all solutions in the final populations of both of the algorithms are dominated

by the reference point. This situation is valid for all multi-objective knapsack problem instances handled in this study. At 95% significance level, C-TDEA is statistically better than NSGA-II and ε-MOEA in both metrics. The seeded versions of NSGA-II, *NSGA-II (s)*, and ε-MOEA, *ε-MOEA (s)*, are also considered. However, from both metric values and statistical results in Tables 3.3 and 3.4; and the final populations obtained in Figure 3.8, we see that seeding has very little effect on the performances of the algorithms, and does not provide the algorithms to converge to the Pareto-optimal frontier. The two algorithms can only maintain the seed solutions as the nondominated solutions of their final populations. Moreover, C-TDEA still outperforms the seeded algorithms. On the other hand, the repair-improvement mechanisms of C-TDEA provide the algorithm fine convergence and diversity properties, but in expense of increased computational times compared to NSGA-II and ε-MOEA.

Table 3.3 Indicator Results for the 2-objective 200-item KP

	I-Epsilon		**Hypervolume**			
Algorithm	$\bar{x}_{I_\varepsilon^+}$	$s_{I_\varepsilon^+}$	$\bar{x}_H$	$\bar{x}_H / \bar{x}_H^{Pareto}$	s_H	**Duration (min)**
C-TDEA	42.7	4.3	0.7740	0.9685	0.0022	0.6892
NSGA-II	2194.2	73.1	0.0000	0.0000	-	0.2790
NSGA-II (s)	1201.0	-	0.0712	0.0891	-	0.2535
ε-MOEA	1668.8	40.5	0.0000	0.0000	-	0.2192
ε-MOEA (s)	1201.0	-	0.0712	0.0891	-	0.1935
Pareto Front	0.0	-	0.7992	1.0000	-	-

Table 3.4 Test Results for the 2-objective 200-item KP

$H_0 : \mu_{pm}^T = \mu_{pm}^C$ vs. $H_1 : \mu_{pm}^T \neq \mu_{pm}^C$						
	I-Epsilon			**Hypervolume**		
Contender	$\Delta_{I_\varepsilon^+}$	P-Value	Winner	Δ_H	P-Value	Winner
NSGA-II	-2151.5	0.0000	C-TDEA	0.7740	-	C-TDEA
NSGA-II (s)	-1158.3	-	C-TDEA	0.7028	-	C-TDEA
ε-MOEA	-1626.1	0.0000	C-TDEA	0.7740	-	C-TDEA
ε-MOEA (s)	-1158.3	-	C-TDEA	0.7028	-	C-TDEA

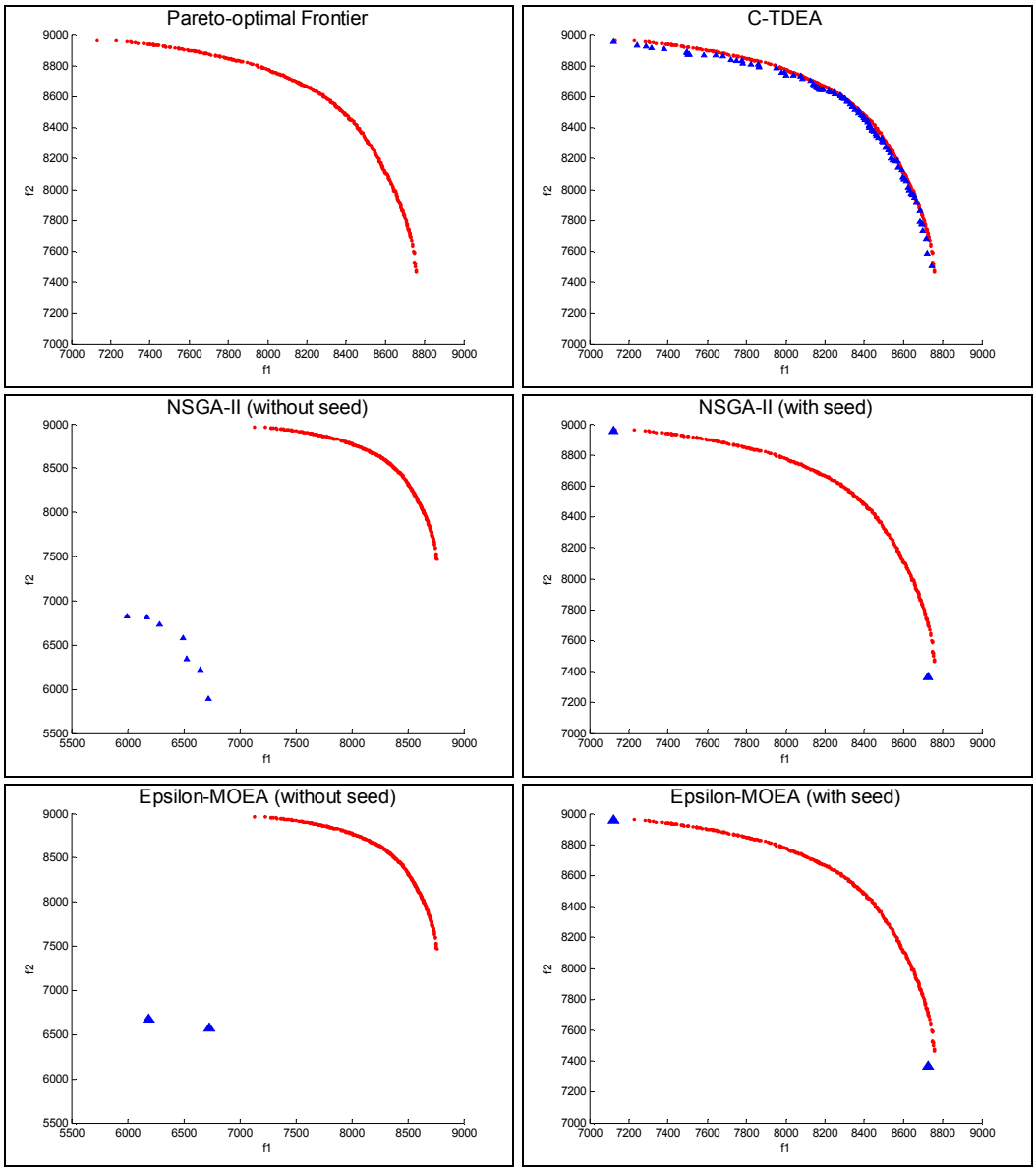

Figure 3.8 Plots for the 2-objective 200-item KP

2-objective 750-item Knapsack Problem

In terms of indicator values, we obtain similar results compared to 200-item knapsack problem (Table 3.5). NSGA-II and ε-MOEA again fail to converge to the Pareto-optimal frontier. All solutions obtained by the contender algorithms are dominated by the nadir point, resulting in a hypervolume metric value of 0 for

both algorithms. We can see in Table 3.6 that, the other two algorithms are outperformed by C-TDEA in terms of both metrics. As we can see in Figure 3.9, the final populations that are presented to the DM show similar behavior to 200-item problem. The seeded versions of the contender algorithms can only present the seed solutions as the final nondominated solutions (Figure 3.9) and they are again outperformed by C-TDEA in terms of both metrics (Table 3.6).

Table 3.5 Indicator Results for the 2-objective 750-item KP

	I-Epsilon		**Hypervolume**			
Algorithm	$\bar{x}_{I_\varepsilon^+}$	$s_{I_\varepsilon^+}$	$\bar{x}_H$	$\bar{x}_H / \bar{x}_H^{Pareto}$	s_H	**Duration (min)**
C-TDEA	196.4	12.4	0.7727	0.9371	0.0014	6.9823
NSGA-II	8525.3	518.0	0.0000	0.0000	-	1.2865
NSGA-II (s)	3652.0	-	0.0082	0.0099	-	1.1911
ε-MOEA	7393.0	476.7	0.0000	0.0000	-	0.2868
ε-MOEA (s)	3652.0	-	0.0082	0.0099	-	0.2899
Pareto Front	0.0	-	0.8246	1.0000	-	-

Table 3.6 Test Results for the 2-objective 750-item KP

$H_0 : \mu_{pm}^T = \mu_{pm}^C$ vs. $H_1 : \mu_{pm}^T \neq \mu_{pm}^C$						
	I-Epsilon			**Hypervolume**		
Contender	$\Delta_{I_\varepsilon^+}$	P-Value	Winner	Δ_H	P-Value	Winner
NSGA-II	-8328.9	0.0000	C-TDEA	0.7727	-	C-TDEA
NSGA-II (s)	-3455.6	-	C-TDEA	0.7645	-	C-TDEA
ε-MOEA	-7196.6	0.0000	C-TDEA	0.7727	-	C-TDEA
ε-MOEA (s)	-3455.6	-	C-TDEA	0.7645	-	C-TDEA

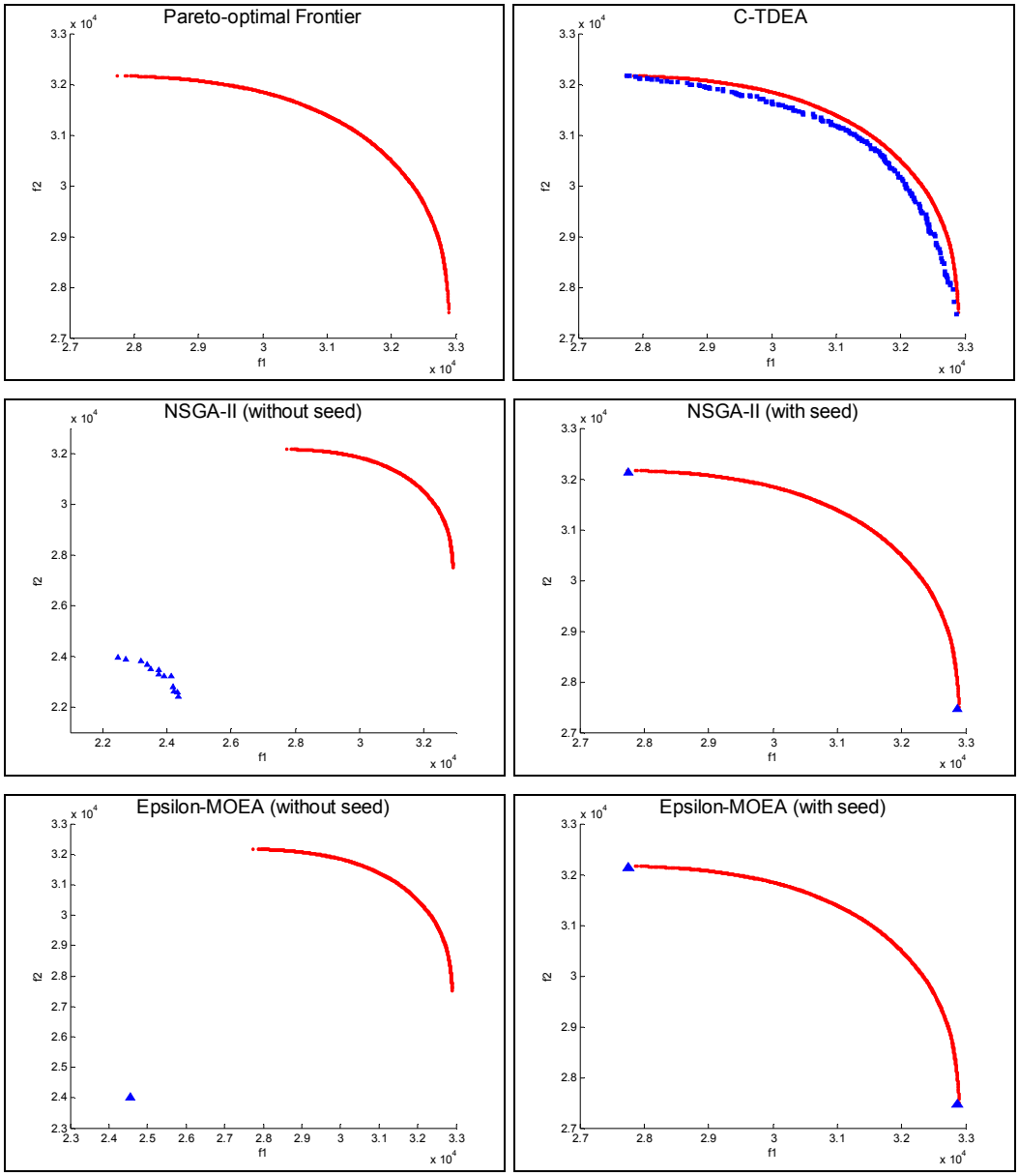

Figure 3.9 Plots for the 2-objective 750-item KP

3-objective 200-item Knapsack Problem

According to the performance metric values and the test results provided in Table 3.7 and 3.8 respectively, C-TDEA outperforms NSGA-II and ε-MOEA, and also their seeded versions in terms of both performance metrics. In Figure 3.10, it is seen that C-TDEA maintains convergence and diversity, while it loses solutions

towards the three edges of the Pareto-optimal frontier. On the other hand, NSGA-II and ε-MOEA cannot even provide convergence to the Pareto-optimal frontier, and they again end up with a hypervolume metric value of 0. The seeded versions of the algorithms are also outperformed by C-TDEA in terms of both metrics, by being able to maintain only the 6 seed solutions in their final nondominated sets.

Table 3.7 Indicator Results for the 3-objective 200-item KP

	I-Epsilon		**Hypervolume**			
Algorithm	$\bar{x}_{I_\varepsilon^+}$	$s_{I_\varepsilon^+}$	$\bar{x}_H$	$\bar{x}_H / \bar{x}_H^{Pareto}$	s_H	**Duration(min)**
C-TDEA	148.2	3.3	0.5244	0.8465	0.0040	1.5807
NSGA-II	1987.3	11.5	0.0000	0.0000	-	0.4482
NSGA-II (s)	604.0	-	0.2581	0.4166	-	0.4004
ε-MOEA	1487.6	57.9	0.0000	0.0000	-	0.3401
ε-MOEA (s)	604.0	-	0.2581	0.4166	-	0.3305
Pareto Front	0.0	-	0.6195	1.0000	-	-

Table 3.8 Test Results for the 3-objective 200-item KP

$H_0 : \mu_{pm}^T = \mu_{pm}^C$ vs. $H_1 : \mu_{pm}^T \neq \mu_{pm}^C$

	I-Epsilon			**Hypervolume**		
Contender	$\Delta_{I_\varepsilon^+}$	P-Value	Winner	Δ_H	P-Value	Winner
NSGA-II	-1839.1	0.0000	C-TDEA	0.5244	0.0000	C-TDEA
NSGA-II (s)	-455.8	-	C-TDEA	0.2663	-	C-TDEA
ε-MOEA	-1339.4	0.0000	C-TDEA	0.5244	0.0000	C-TDEA
ε-MOEA (s)	-455.8	-	C-TDEA	0.2663	-	C-TDEA

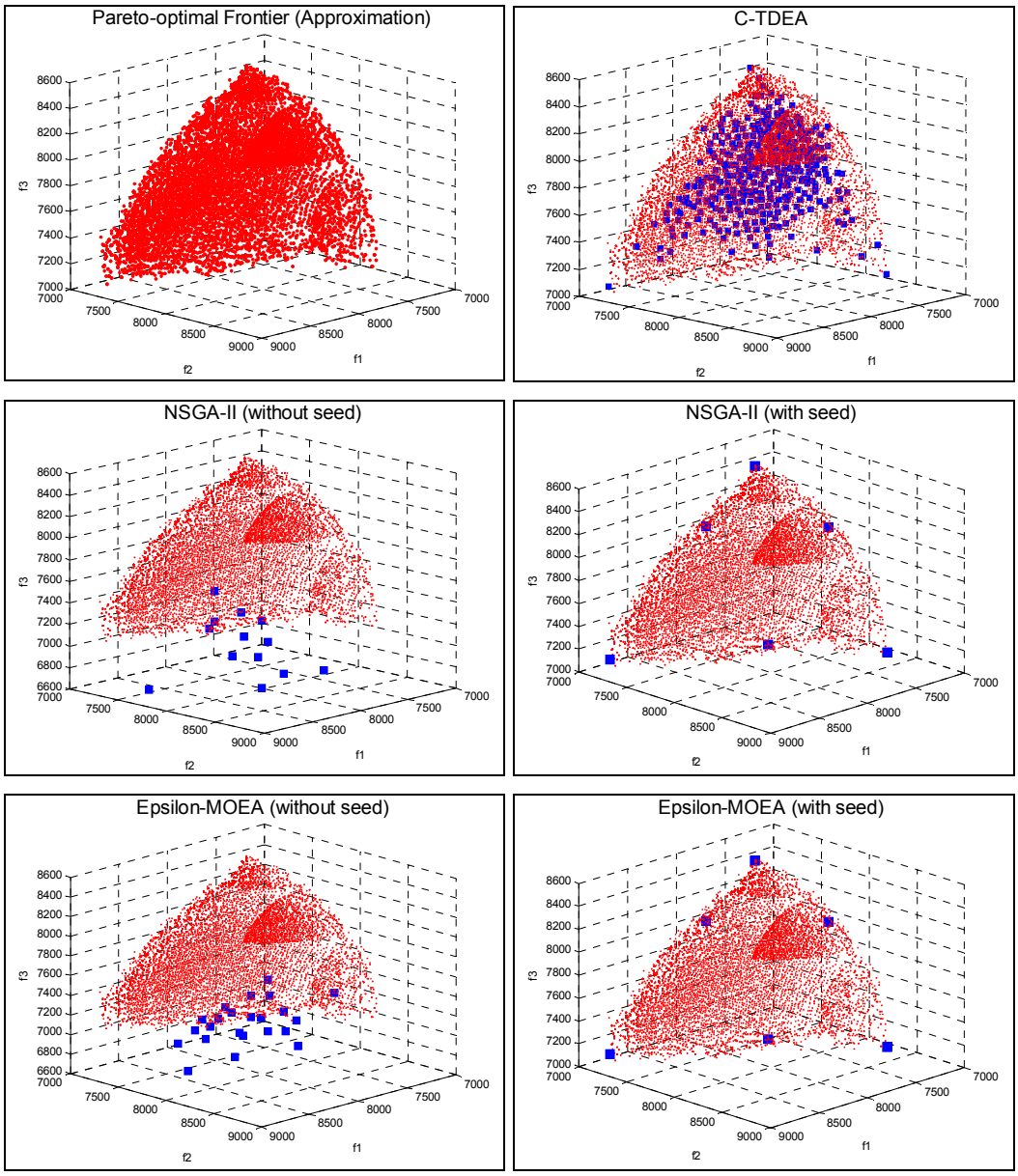

Figure 3.10 Plots for the 3-objective 200-item KP

3-objective 750-item Knapsack Problem

In this 3-objective knapsack problem instance, although C-TDEA outperforms both NSGA-II and ε-MOEA in terms of both performance metrics (see Table 3.9 and Table 3.10), it loses to provide diversity over the Pareto-optimal frontier. Figure 3.11 shows that, while the middle-most section of the frontier is well-

converged, the 3 sides corresponding to the respective 2-objective Pareto-optimal frontiers are lost by C-TDEA during the simulation run. The two contenders of C-TDEA cannot even converge to the Pareto-optimal frontier. The seeded NSGA-II and ε-MOEA can only present the 6 seed solutions, and they are again outperformed by C-TDEA in terms of both performance metrics.

Table 3.9 Indicator Results for the 3-objective 750-item KP

	I-Epsilon		**Hypervolume**			
Algorithm	$\bar{x}_{I_\varepsilon^+}$	$s_{I_\varepsilon^+}$	$\bar{x}_H$	$\bar{x}_H / \bar{x}_H^{Pareto}$	s_H	**Duration (min)**
C-TDEA	549.6	22.3	0.5135	0.7984	0.0114	16.0037
NSGA-II	8803.3	126.5	0.0000	0.0000	-	1.5595
NSGA-II (s)	2325.0	-	0.2969	0.4616	-	1.5453
ε-MOEA	7552.6	197.0	0.0000	0.0000	-	0.4951
ε-MOEA (s)	2325.0	-	0.2969	0.4616		0.4576
Pareto Front	0.0	-	0.6432	1.0000	-	-

Table 3.10 Test Results for the 3-objective 750-item KP

$H_0 : \mu_{pm}^T = \mu_{pm}^C$ vs. $H_1 : \mu_{pm}^T \neq \mu_{pm}^C$						
	I-Epsilon			**Hypervolume**		
Contender	$\Delta_{I_\varepsilon^+}$	P-Value	Winner	Δ_H	P-Value	Winner
NSGA-II	-8253.7	0.0000	C-TDEA	0.5135	-	C-TDEA
NSGA-II (s)	-1775.4	-	C-TDEA	0.2166	-	C-TDEA
ε-MOEA	-7003.0	0.0000	C-TDEA	0.5135	-	C-TDEA
ε-MOEA (s)	-1775.4	-	C-TDEA	0.2166	-	C-TDEA

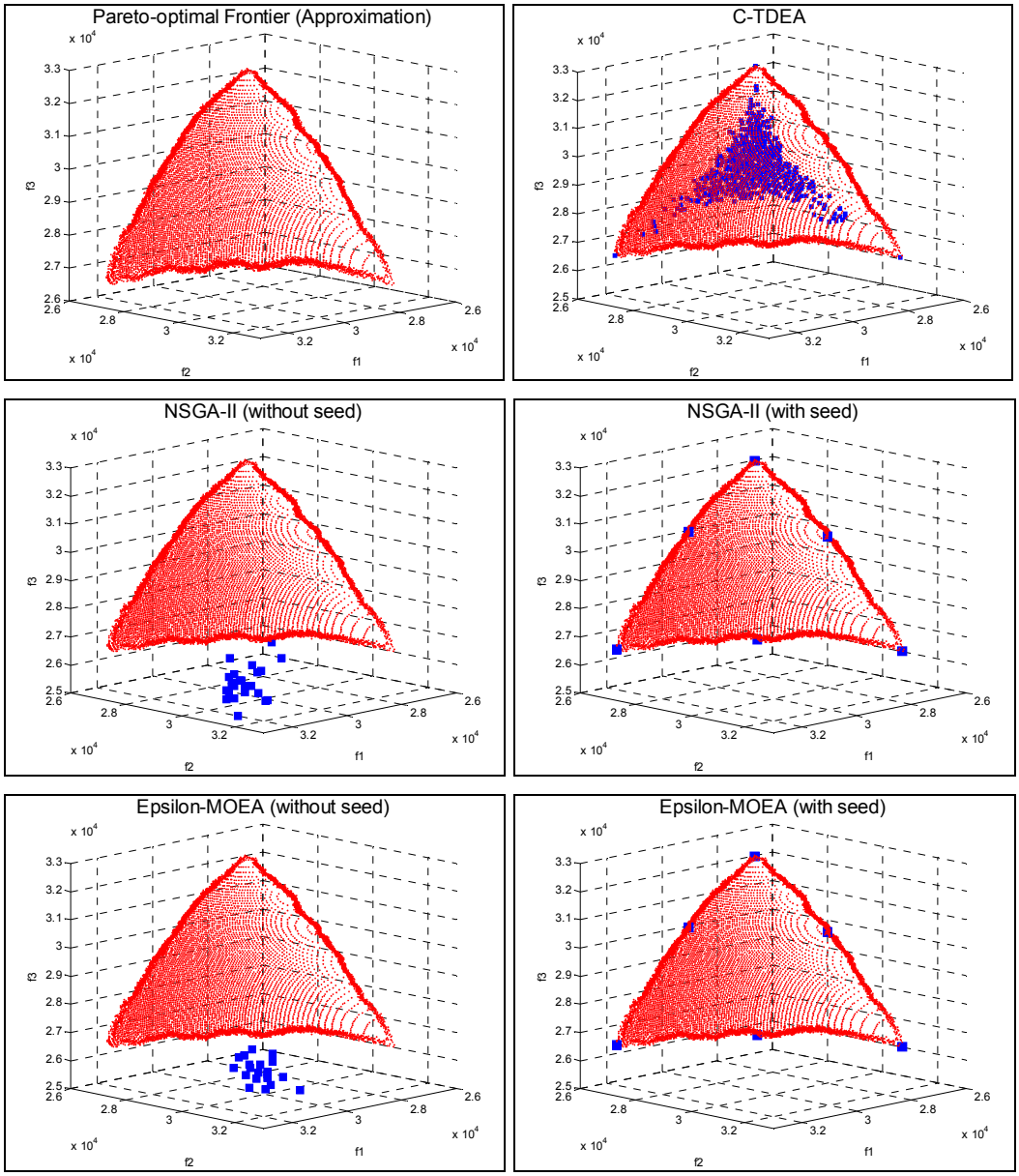

Figure 3.11 Plots for the 3-objective 750-item KP

3.5.3 Multi-objective Assignment Problem Instances

The figures provided in this section show the final archive populations for C-TDEA and ε-MOEA, and the nondominated solutions of the final population for NSGA-II. The entire final populations of ε-MOEA and NSGA-II including the dominated solutions are provided in Appendix B.

2-objective N=50 Assignment Problem

While C-TDEA successfully converges to the Pareto-optimal frontier, NSGA-II and ε-MOEA fail to converge as seen in Figure 3.12. Indicator values in Table 3.11 also show that C-TDEA has much better convergence and diversity than both of the algorithms in terms of the metric values. At 95% significance level, C-TDEA is statistically better than NSGA-II and ε-MOEA in both metrics. The seeded versions of NSGA-II and ε-MOEA are also considered as in the case of knapsack problem. In Figure 3.12, we see that seeding improves the performance of the contender algorithms to converge to the Pareto-optimal frontier, in contrast to the case for the MOKP. However, C-TDEA still outperforms the seeded versions of the algorithms, as seen in Table 3.12. On the other hand, the territory definition and the repair-improvement mechanisms of C-TDEA provide the algorithm both convergence and diversity, but in expense of increased computational times compared to NSGA-II and ε-MOEA.

Table 3.11 Indicator Results for the 2-objective N=50 AP

	I-Epsilon		**Hypervolume**			
Algorithm	$\bar{x}_{I_\varepsilon^+}$	$s_{I_\varepsilon^+}$	$\bar{x}_H$	$\bar{x}_H / \bar{x}_H^{Pareto}$	s_H	**Duration (min)**
C-TDEA	31.6	3.6	0.8992	0.9841	0.0023	0.8631
NSGA-II	1221.3	27.5	0.0800	0.0876	0.0131	0.2387
NSGA-II (s)	562.2	2.8	0.5685	0.3313	0.0038	0.2067
ε-MOEA	841.0	11.8	0.2715	0.2971	0.0214	0.1786
ε-MOEA (s)	731.4	5.7	0.4812	0.5735	0.0077	0.1242
Pareto Front	0.0	-	0.9137	1.0000	-	-

Table 3.12 Test Results for the 2-objective N=50 AP

$H_0 : \mu_{pm}^T = \mu_{pm}^C$ vs. $H_1 : \mu_{pm}^T \neq \mu_{pm}^C$						
	I-Epsilon			**Hypervolume**		
Contender	$\Delta_{I_\varepsilon^+}$	P-Value	Winner	Δ_H	P-Value	Winner
NSGA-II	-1189.7	0.0000	C-TDEA	0.8192	0.0000	C-TDEA
NSGA-II (s)	-530.6	0.0000	C-TDEA	0.3307	0.0000	C-TDEA
ε-MOEA	-809.4	0.0000	C-TDEA	0.6276	0.0000	C-TDEA
ε-MOEA (s)	-699.8	0.0000	C-TDEA	0.4180	0.0000	C-TDEA

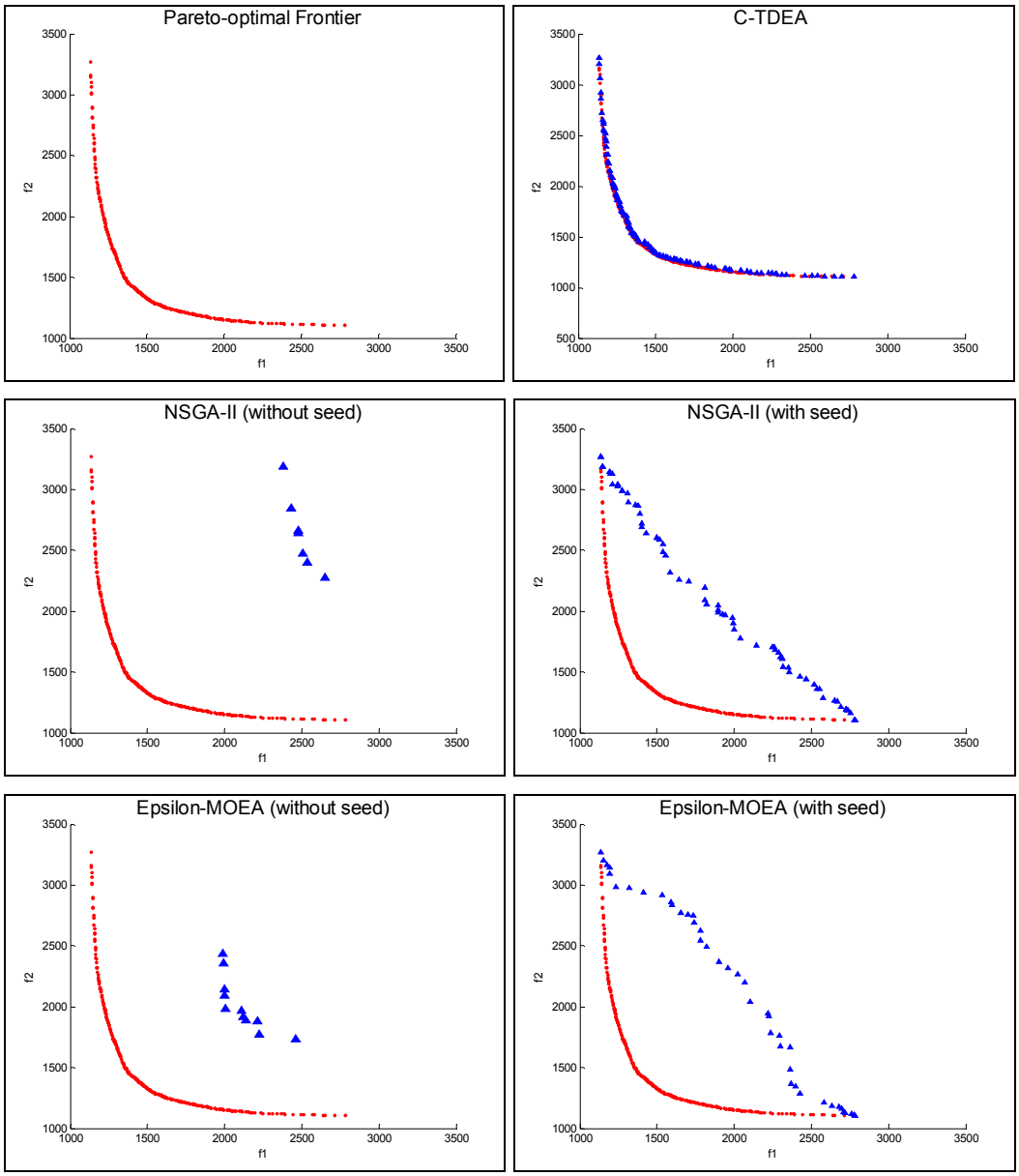

Figure 3.12 Plots for the 2-objective N=50 AP

2-objective N=100 Assignment Problem

Both NSGA-II and ε-MOEA fail to converge to the Pareto-optimal frontier of this 2-objective assignment problem (see Table 3.13). We can see in Table 3.14 that, the other two algorithms are outperformed by C-TDEA in terms of both metrics. Although the seeded NSGA-II and ε-MOEA perform better than their original versions, they are still outperformed by C-TDEA in terms of both performance metrics. The final populations obtained by each of the algorithms are provided over the Pareto-optimal frontier in Figure 3.13.

Table 3.13 Indicator Results for the 2-objective N=100 AP

	I-Epsilon		**Hypervolume**			
Algorithm	$\bar{x}_{I_\varepsilon^+}$	$s_{I_\varepsilon^+}$	$\bar{x}_H$	$\bar{x}_H / \bar{x}_H^{Pareto}$	s_H	**Duration (min)**
C-TDEA	105.2	3.4	0.8995	0.9665	0.0014	5.2318
NSGA-II	3001.7	7.0	0.0081	0.0087	0.0029	0.8003
NSGA-II (s)	1811.6	3.6	0.3762	0.4042	0.0016	0.5366
ε-MOEA	2181.0	9.9	0.1564	0.1681	0.0071	0.4116
ε-MOEA (s)	2094.2	5.2	0.3488	0.3748	0.0056	0.3657
Pareto Front	0.0000	-	0.9307	1.0000	-	-

Table 3.14 Test Results for the 2-objective N=100 AP

$H_0 : \mu_{pm}^T = \mu_{pm}^C$ vs. $H_1 : \mu_{pm}^T \neq \mu_{pm}^C$						
	I-Epsilon			**Hypervolume**		
Contender	$\Delta_{I_\varepsilon^+}$	P-Value	Winner	Δ_H	P-Value	Winner
NSGA-II	-2896.5	0.0000	C-TDEA	0.8914	0.0000	C-TDEA
NSGA-II (s)	-1706.4	0.0000	C-TDEA	0.5233	0.0000	C-TDEA
ε-MOEA	-2075.8	0.0000	C-TDEA	0.7431	0.0000	C-TDEA
ε-MOEA (s)	-1989.0	0.0000	C-TDEA	0.5507	0.0000	C-TDEA

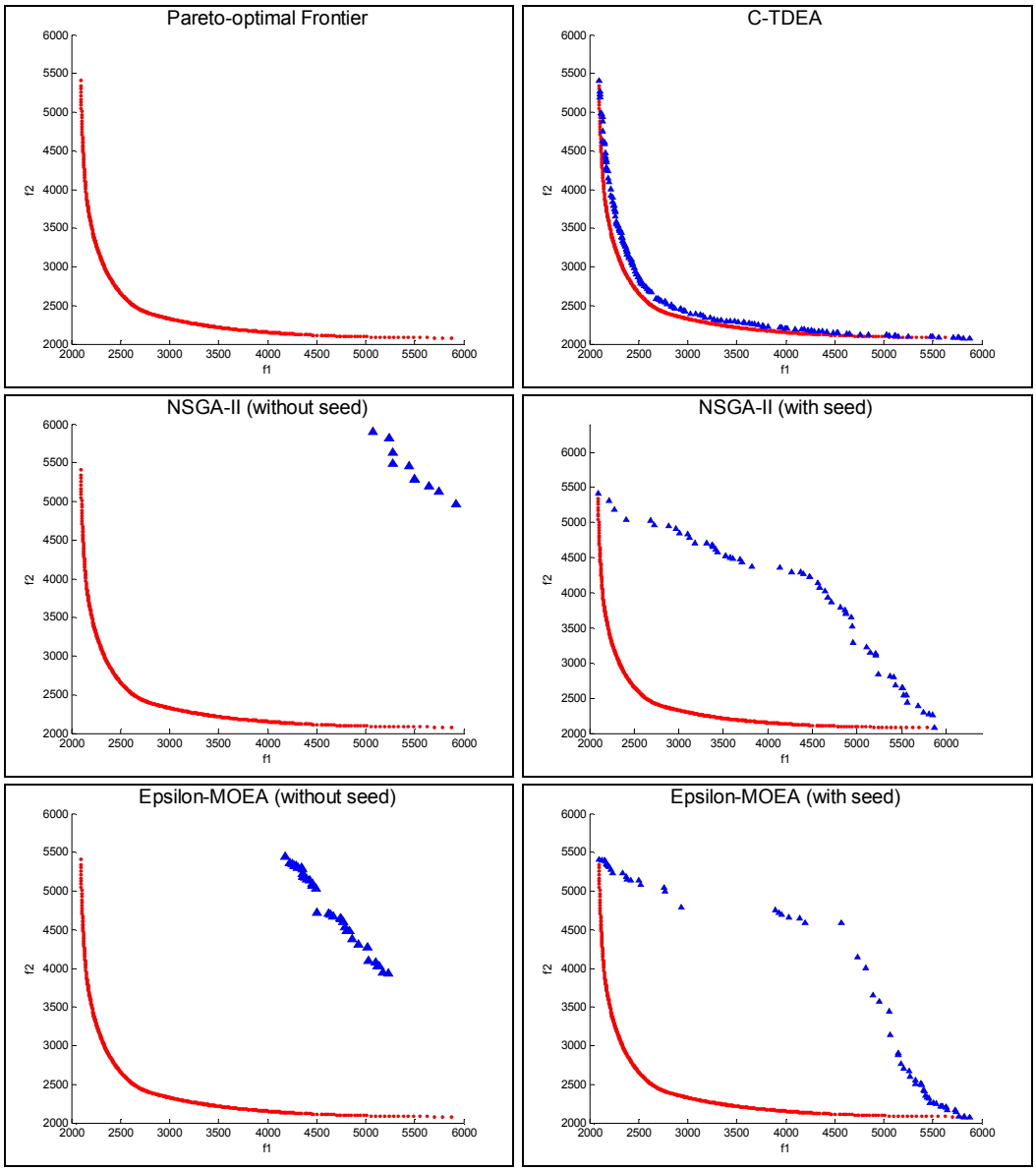

Figure 3.13 Plots for the 2-objective N=100 AP

3-objective N=50 Assignment Problem

According to the performance metric values and the test results provided in Table 3.15 and 3.16 respectively, C-TDEA outperforms the other two contender algorithms in terms of both performance metrics. In Figure 3.14, it is seen that C-TDEA maintains convergence and diversity over the Pareto-optimal frontier. On

the other hand, NSGA-II and ε-MOEA can only converge to the middle part of the Pareto-optimal frontier. The seeded versions of the contender algorithms provide better diversity over the Pareto-optimal frontier, and they converge better by the assistance of the seed solutions. However, C-TDEA again outperforms the seeded algorithms in terms of both performance metrics (Table 3.16).

Table 3.15 Indicator Results for the 3-objective N=50 AP

	I-Epsilon		**Hypervolume**			
Algorithm	$\bar{x}_{I_\varepsilon^+}$	$s_{I_\varepsilon^+}$	$\bar{x}_H$	$\bar{x}_H / \bar{x}_H^{Pareto}$	s_H	**Duration (min)**
C-TDEA	144.8	10.6	0.6873	0.8797	0.0025	2.5817
NSGA-II	1225.3	29.2	0.0320	0.0410	0.0003	0.3048
NSGA-II (s)	499.0	24.9	0.4561	0.5838	0.0003	0.2764
ε-MOEA	980.5	26.4	0.1086	0.1390	0.0064	0.2404
ε-MOEA (s)	606.7	25.8	0.4281	0.5479	0.0007	0.2287
Pareto Front	0.0000	-	0.7813	1.0000	-	-

Table 3.16 Test Results for the 3-objective N=50 AP

$H_0 : \mu_{pm}^T = \mu_{pm}^C$ vs. $H_1 : \mu_{pm}^T \neq \mu_{pm}^C$						
	I-Epsilon			**Hypervolume**		
Contender	$\Delta_{I_\varepsilon^+}$	P-Value	Winner	Δ_H	P-Value	Winner
NSGA-II	-1080.5	0.0000	C-TDEA	0.6553	0.0000	C-TDEA
NSGA-II (s)	-354.2	0.0000	C-TDEA	0.2312	0.0000	C-TDEA
ε-MOEA	-835.7	0.0000	C-TDEA	0.5787	0.0000	C-TDEA
ε-MOEA (s)	-461.9	0.0000	C-TDEA	0.2592	0.0000	C-TDEA

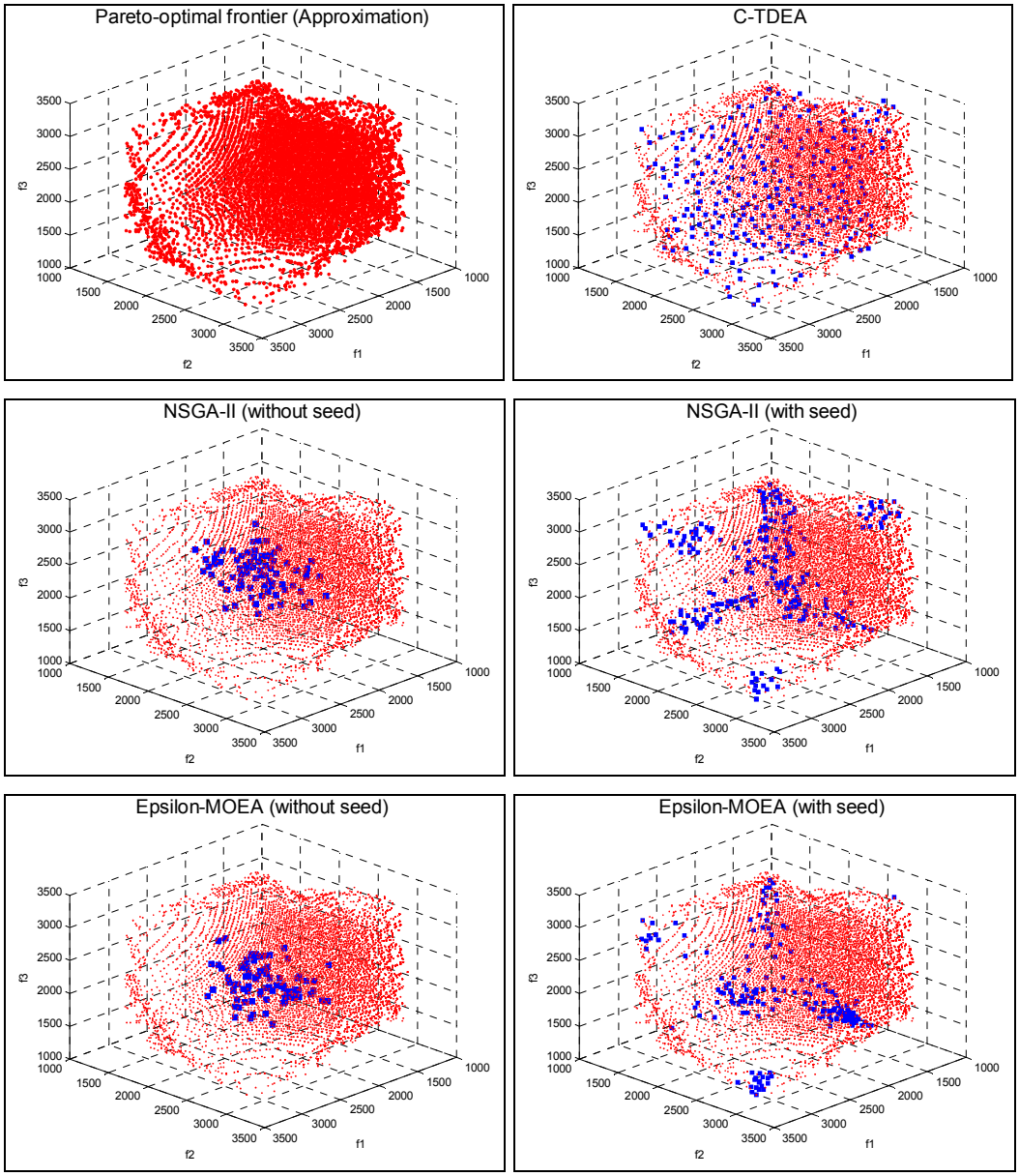

Figure 3.14 Plots for the 3-objective N=50 AP

3-objective N=100 Assignment Problem

In this problem instance, C-TDEA outperforms both NSGA-II and ε-MOEA in terms of both performance metrics (see Table 3.17 and Table 3.18). The algorithm provides diversity over the Pareto-optimal frontier as seen in Figure

3.15. The two contenders of C-TDEA can only converge to the middle portion of the Pareto-optimal frontier. C-TDEA also outperforms the seeded versions of the contenders. However, it should be mentioned that the performances of the contenders have substantially increased by the inclusion of the seed solutions.

Table 3.17 Indicator Results for the 3-objective N=100 AP

	I-Epsilon		**Hypervolume**			
Algorithm	$\bar{x}_{I_\varepsilon^+}$	$s_{I_\varepsilon^+}$	$\bar{x}_H$	$\bar{x}_H / \bar{x}_H^{Pareto}$	s_H	**Duration (min)**
C-TDEA	399.2	13.9	0.6951	0.8364	0.0024	16.3724
NSGA-II	2924.4	20.3	0.0203	0.0244	0.0005	0.9492
NSGA-II (s)	1549.9	12.7	0.3795	0.4566	0.0004	0.6204
ε-MOEA	2483.2	16.8	0.0555	0.0668	0.0022	0.7562
ε-MOEA (s)	1916.8	15.2	0.3198	0.3848	0.0006	0.6017
Pareto Front	0.0000	-	0.8311	1.0000	-	-

Table 3.18 Test Results for the 3-objective N=100 AP

$H_0 : \mu_{pm}^T = \mu_{pm}^C$ vs. $H_1 : \mu_{pm}^T \neq \mu_{pm}^C$						
	I-Epsilon			**Hypervolume**		
Contender	$\Delta_{I_\varepsilon^+}$	P-Value	Winner	Δ_H	P-Value	Winner
NSGA-II	-2525.2	0.0000	C-TDEA	0.6748	0.0000	C-TDEA
NSGA-II (s)	-1150.7	0.0000	C-TDEA	0.3156	0.0000	C-TDEA
ε-MOEA	-2084.0	0.0000	C-TDEA	0.6396	0.0000	C-TDEA
ε-MOEA (s)	-1517.6	0.0000	C-TDEA	0.3753	0.0000	C-TDEA

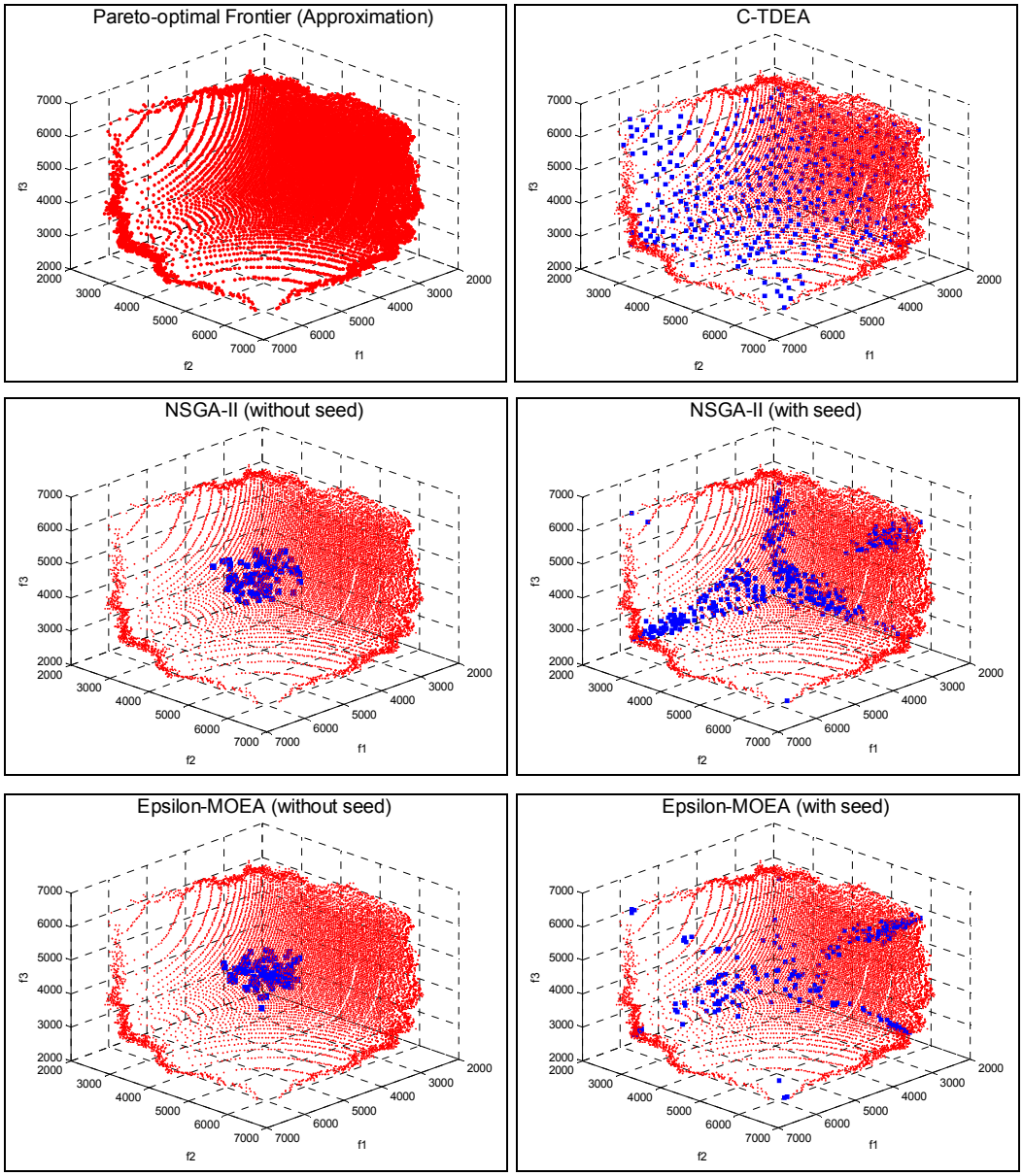

Figure 3.15 Plots for the 3-objective N=100 AP

3.5.4 Discussions

In this section, we show that C-TDEA can converge to and provide diversity on the Pareto-optimal frontiers of all of the test problems. C-TDEA outperforms both NSGA-II and ε-MOEA in terms of both Hypervolume metric and I-Epsilon metric in each of the 8 test problems. Even the seeded contender algorithms cannot come close to the performance of C-TDEA. However, the advantage obtained by territory definition is eliminated substantially in C-TDEA. That is, although territory definition provides fast execution, the repair and improvement procedures used in C-TDEA blankets this advantage. Therefore, less time requiring repair and improvement procedures can be developed to overcome this problem.

CHAPTER 4

INTERACTIVE COMBINATORIAL TERRITORY DEFINING EVOLUTIONARY ALGORITHM (IC-TDEA)

Multi-objective evolutionary algorithms end up with a population of solutions as an approximation of the Pareto-optimal frontier. However, in most cases the DM is interested in obtaining a single solution, or a very small set of solutions to select from. Rachmawati and Srinivasan (2006) have illustrated preferences as "the basis of tie-breaking between solutions in the Pareto optimal set". Therefore, we now focus on the issue of preference incorporation to present a preference-based multi-objective evolutionary algorithm.

In this chapter, we present Interactive Combinatorial Territory Defining Evolutionary Algorithm (IC-TDEA) that is developed to end up with the preferred solutions of the DM in an interactive manner. The algorithm is a modified version of Interactive Territory Defining Evolutionary Algorithm (iTDEA) that has been proposed by Karahan (2008).

Influenced by the advantages of preference incorporation into MOEAs, Karahan (2008) has also presented the interactive version of TDEA, named as iTDEA. In this interactive version, the preferred region of the DM is approximated step by step during the optimization stage as information is obtained from the DM. For this purpose, interaction stages are determined at prespecified generations, and a more focused and smaller preference region is estimated at each of these interaction stages. The first focus region is determined as the whole Pareto frontier and it is assigned the highest τ value. The corresponding τ value of the final focus region (the smallest τ value) is also prespecified. The subsequent regions are estimated during the optimization process and a newer region is

smaller and has a smaller τ value than the previous ones. The offspring is in a preferred region if its favorable weights are covered by the weight sets of that preferred region and it is evaluated using the τ associated to that region. Otherwise, it is evaluated using the τ value of the remaining region. Since a preferred region is specified by its weight set, Karahan (2008) has demonstrated a procedure to obtain these weight sets and also to determine the values of the corresponding τ values. IC-TDEA utilizes the same weight set and τ calculations that Karahan (2008) has proposed in iTDEA. The weight set of the next preference region is obtained by constructing an interval around the favorable weights of the best solution of the DM obtained at the interaction stage. This best solution is selected by the DM from a representative set of solutions from the previous focus region such that, this representative set is composed of the farthest ε-nondominated solutions obtained by a filtering procedure. In fact, the only difference of iTDEA from TDEA is in the archive acceptance procedure such that, before checking for territory violation, the favorable weights of the offspring are calculated and the corresponding preference region of the offspring is found to obtain the τ value of territory violation check. After determining the regions that contain the favorable weights of the offspring, the offspring is evaluated with respect to the smallest of the corresponding τ values. In iTDEA, the determination of the interaction stages should be attached enough importance for the correct functioning of the algorithm in order to provide convergence to the Pareto-optimal frontier. This interactive preference incorporation scheme results in better details in the regions of interest with higher computational efficiency.

This chapter is organized as follows: We present the general outline of IC-TDEA that is common for all MOCO problems in Section 4.1. We explain the modifications of the algorithm for the knapsack problem in detail in Section 4.2. Similarly, we explain the modifications of the algorithm for the assignment problem in detail in Section 4.3. Finally, we present the experimental results and comparisons in Section 4.4.

4.1 GENERAL OUTLINE OF IC-TDEA

Below is the general outline of the algorithm that is applicable to any MOCO problem:

1. Initialization: Set the initial regular population size $\overline{N}$, the starting territory size τ_0, the final territory size τ_H for the final preference region, the maximum number of generations to iterate the algorithm *T,* such that the final population size is acceptable for the DM. Set the number of interactions *H* according to the availability of the DM. Set the iteration counter $t = 0$ and interaction counter h=1. Set the first preference region R_0 as the whole Pareto-optimal frontier. Set the initial reference point $f_0^{\mathrm{Re}\,f}$ as the ideal point f^*. Set the interaction generations $G_1, \ldots, G_H$
2. Initialization of the Regular Population: Obtain $\overline{N}$ individuals to fill the initial regular population P(0) by creating $\overline{\overline{N}}$ seed solutions, and $\overline{N}$ - $\overline{\overline{N}}$ random individuals.
3. Initialization of the Archive Population: Initialize the archive population with the copies of the solutions in P(0) that are nondominated with respect to the solutions in P(0).
4. Parent Selection: Set $t \leftarrow t + 1$. Choose one parent from the regular population, and one parent from the archive population.
5. Recombination: Recombine the parents to create the offspring.
6. Mutation: Apply mutation to the offspring.
7. Repair and Improvement of the Offspring: If the offspring is infeasible, then repair the offspring such that it becomes feasible. Then, improve the offspring if there is place for improvement.
8. Acceptance into the Regular Population: Check whether the offspring satisfies the acceptance condition for the regular population. If it is accepted, insert it into P(t). Otherwise, go to step 10.

9. Acceptance into the Archive Population: If the offspring is accepted into P(t), check whether it satisfies the acceptance condition for the archive population. If it is accepted, insert it into A(t).
10. Interaction: If $t = G_h$, stop for interaction with the DM. Present a filtered and well-dispersed sample of solutions to the DM. Set the solution preferred by the DM as the reference point f_h^{Ref}. Determine the new preferred region R_h around f_h^{Ref} and set the corresponding territory size τ_h. Set $h \leftarrow h+1$. Otherwise, go to Step 11.
11. Stopping Condition: Stop if the pre-specified iteration limit T is hit ($t=T$), and report the archive population, or a well-dispersed small set of solutions from the archive population to the DM. Otherwise, go to step 4.

In the interaction stage, the DM may choose more than one solutions. This situation can be handled by the algorithm after some revisions to deal with multiple reference points. However, we deal with the single reference point case in this study.

4.2 IC-TDEA FOR MULTI-OBJECTIVE KNAPSACK PROBLEM

In IC-TDEA, the repair-improvement procedure is revised for the offspring to serve for convergence to the desired region on the Pareto-optimal frontier. The procedure is the same as C-TDEA for the initial population members. The revised repair-improvement procedure for an offspring between interaction stages G_h and G_{h+1} is as follows:

1. Check whether the offspring is dominated by the reference point f_h^{Ref}. If the offspring is dominated by f_h^{Ref}, go to Step 2. Otherwise, go to Step 3.
2. Check whether the offspring is within the last preferred region R_h. If the offspring is within R_h, then go to Step 3. Otherwise go to Step 4.

3. If the offspring is infeasible, repair the offspring in the direction of the ideal point f^*. If there is place for further improvement, improve the offspring in the direction of f^*.
4. If the offspring is infeasible, repair the offspring in the direction of the ideal point $f_h^{\mathrm{Re}f}$. If there is place for further improvement, improve the offspring in the direction of $f_h^{\mathrm{Re}f}$.

The repair-improvement in the direction of the ideal point f^* is the original repair-improvement procedure used in C-TDEA. However, in order to guide a solution in the direction of a dominating point except the ideal point, a modification in the original procedure is required. In the original repair-improvement procedure, instead of the favorable weights of the solutions, the so called "reference weights" are utilized (Equation 4.1). The formulation of the reference weights is a modified version of the favorable weight formulation, where the function values of the reference point are used instead of the function values of the ideal point.

$$\lambda_i^{REF} = \begin{cases} \dfrac{1}{f_i^{REF} - fi}\left[\sum_{j=1}^{M} \dfrac{1}{f_j^{REF} - f_j}\right]^{-1} & if \quad f_j \neq f_j^* \quad for \quad all \quad j = 1,2,...,M \\ 1 & if\ f_i = f_i^{REF} \\ 0 & if \quad f_i \neq f_i^{REF} \quad but \quad \exists j \quad such \quad that \quad f_j = f_j^{REF} \end{cases} \tag{4.1}$$

Therefore, if an offspring is infeasible and it is dominated by the reference point $f_h^{\mathrm{Re}f}$ but it is not in the last preferred region R_h, the following repair procedure is carried out:

1. Calculate the reference weights, λ_i^{REF} s, of the solution.
2. Calculate Tchebycheff distance after moving out item *j* from the knapsack for each item *j* which is in the current knapsack:

$$d_j = \underset{i \in M}{Max} \left\{ \lambda_i^{REF} \left(\bar{z}_i^* - (\overline{z_i - c_{ij}}) \right) \right\} \tag{4.2}$$

3. Calculate d_j / a_j for each item j which is in the current knapsack, where a_j is the constraint coefficient of item j.
4. Find $\text{Min}_j \{ d_j / a_j \} = d_k / a_k$
5. Discard item x_k from the knapsack.
6. If the new solution is infeasible, turn back to Step 1 and repeat the same procedure using the favorable weights of the new solution until a feasible solution is obtained. Otherwise, finish the repair procedure.

On the other hand, if an offspring is feasible or it becomes feasible after the repair procedure, it is evaluated for the improvement procedure. If the offspring is dominated by the reference point f_h^{Ref} but it is not in the last preferred region R_h, the following procedure is utilized:

1. Calculate the reference weights, λ_i^{REF} s, of the solution.
2. For each item *j* which is not in the current knapsack and addition of which does not violate the knapsack capacity, calculate Tchebycheff distance after placing item *j* into the knapsack:

$$d_j = \underset{i \in M}{Max} \left\{ \lambda_i^{REF} \left(\bar{z}_i^* - (\overline{z_i + c_{ij}}) \right) \right\} \tag{4.3}$$

3. Calculate $(1-d_j) / a_j$ for each item *j* which is not in the current knapsack and addition of which does not violate the knapsack capacity, where a_j is the constraint coefficient of item *j*.

4. Find $\text{Max}_j \{ (1-d_j) / a_j \} = (1-d_k) / a_k$
5. Place item x_k into the Knapsack.

If there is place for improvement of the new solution, turn back to Step 1 and repeat the same procedure using the favorable weights of the new solution until there is no place for improvement. Otherwise, finish the improvement procedure. The procedure is illustrated in Figure 4.1. Region 1 to the southwest of f_h^{Ref} corresponds to the objective space that is dominated by the reference point f_h^{Ref}, while Region 2 corresponds to the last preferred region R_h that is constructed around f_h^{Ref}.

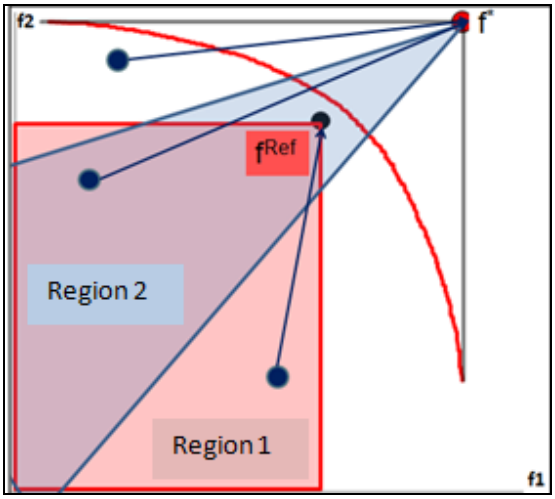

Figure 4.1 IC-TDEA Repair and Improvement Procedure for MOKP

Although the aim is to converge to the region that is preferred by the DM, we should still continue to search the whole Pareto-optimal frontier for other solutions that may be interesting for the DM. The reason is that, the DM does not know his utility function beforehand, and any region on the Pareto-optimal frontier that is not searched yet may be of interest for the DM. Consequently, the solutions that are not dominated by the reference point are used to serve for exploration over the whole Pareto-optimal frontier, and they are subject to the

ideal point based repair-improvement procedure. On the other hand, the solutions that are dominated by the reference point have the potential to be repaired and improved in the direction of the reference point. However, the solutions that are in the last preferred region serve to convergence to the preferred region by using their own favorable weights that are covered by the weight range of this preferred region. Since the solutions that are dominated by the reference point $f_h^{\mathrm{Re}f}$ but are not in the last preferred region R_h cannot guarantee the convergence to the preferred region by directing them to the ideal point, those solutions are repaired and improved in the direction of the reference point $f_h^{\mathrm{Re}f}$.

4.3 IC-TDEA FOR THE MULTI-OBJECTIVE ASSIGNMENT PROBLEM

As in the case of the knapsack problem, the offspring improvement procedure of the assignment problem is also revised in order to focus on the desired region on the Pareto optimal frontier. The procedure is the same as in C-TDEA for the initial population members. The revised improvement procedure for an offspring between interaction stages G_h and G_{h+1} is as follows:

1. Check whether the offspring is dominated by the reference point $f_h^{\mathrm{Re}f}$. If the offspring is dominated by $f_h^{\mathrm{Re}f}$, go to Step 2. Otherwise, go to Step 3.
2. Check whether the offspring is within the last preferred region R_h. If the offspring is within R_h, then go to Step 4. Otherwise go to Step 5.
3. Check for improvement of the offspring based on domination criterion.
4. Check for improvement of the offspring based on domination criterion without allowing the offspring to get out of R_h.
5. Check for improvement of the offspring in the direction of $f_h^{\mathrm{Re}f}$ until it is covered by R_h. Then continue to check for improvement of the offspring based on domination criterion without allowing the offspring to get out of R_h.

The improvement based on domination criterion is the original improvement procedure used in C-TDEA. That is, the jobs are exchanged among the people if and only if the revised solution dominates the original solution. However, in order to guide the solution to the preferred region, original improvement procedure should be revised. For this aim, we propose a new criterion, which is the Euclidean distance to the reference point $f_h^{\mathrm{Re}f}$. In the original improvement procedure, instead of taking the job exchange criterion as domination, we check whether the revised solution has a smaller Euclidean distance to $f_h^{\mathrm{Re}f}$ or not. Therefore, the jobs are exchanged if and only if the new solution is closer to $f_h^{\mathrm{Re}f}$ in terms of Euclidean distance.

The procedure is illustrated in Figure 4.2. The solutions that are subject to domination based improvement are illustrated as being directed to the ideal point. On the other hand, for the solutions that are improved by using the Euclidean distance to the reference point criterion, the improvement is shown to be directed to the reference point $f_h^{\mathrm{Re}f}$. The motivation behind this improvement scheme is similar to the knapsack problem, which has been discussed in Section 4.2.

If an offspring is dominated by the reference point $f_h^{\mathrm{Re}f}$ but it is not in the last preferred region R_h, the following improvement procedure is carried out:

1. Set first iteration counter n =1. Start at gene n that corresponds to the first person.
2. Set second iteration counter $m = n+1$. If the solution is not within the last preferred region R_h, then go to Step 3, otherwise go to Step 4.
3. Exchange the jobs of person m with the job of person n. If the newly obtained solution has a smaller Euclidean distance to $f_h^{\mathrm{Re}f}$ than the initial solution, keep the jobs exchanged and go to Step 5. Otherwise, undo the exchange, set $m \leftarrow m+1$, and repeat this step. If m=M+1, go to Step 5.

4. Exchange the jobs of person m with the job of person n. If the newly obtained solution dominates the initial solution and it is within the last preferred region R_h, keep the jobs exchanged and go to Step 5. Otherwise, undo the exchange, set $m \leftarrow m+1$, and repeat this step. If m=M+1, go to Step 5.
5. Set $n \leftarrow n+1$ and go back to Step 2. If n = M, stop.

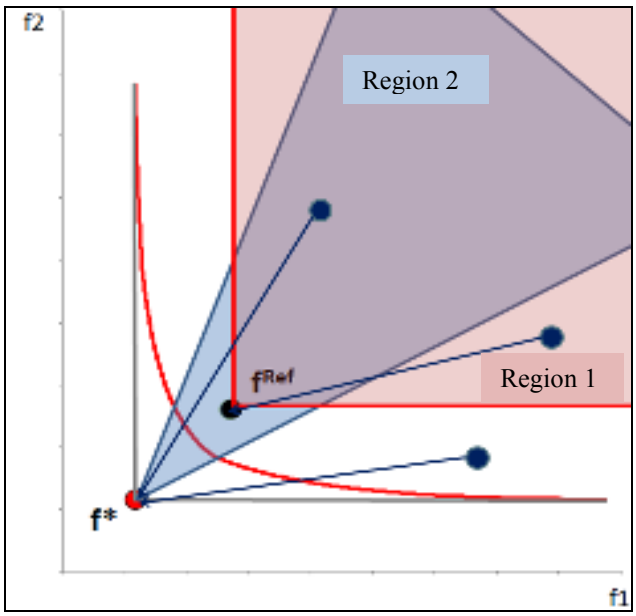

Figure 4.2 IC-TDEA Improvement Procedure for MOAP

For the same reason as in the case of C-TDEA, this improvement procedure is held until a predetermined small number of generations that is determined by preliminary runs.

4.4 SIMULATION RUNS AND COMPARISONS

The computational experiments are performed on 2- and 3-objective 750-item single-dimensional knapsack problem instances; and 2- and 3-objective 100×100 assignment problem instances. We assume that the DM's preferences can be represented by a Tchebycheff utility function of the following form (Karahan, 2008):

$$U = \underset{z \in Z}{Min} \quad \underset{i=1,\ldots,M}{Max} \left[w_i \left| f_i^* - f_i \right| \right] \tag{4.4}$$

where f_i is the value of the i^{th} objective, f_i^* is the i^{th} element of the ideal objective vector, and w_i is the weight of the i^{th} objective. For each of the 4 problem instances, we choose three different utility functions to test the capability of the algorithm to converge to the different regions of the Pareto-optimal frontiers. Moreover, we test the performance of the algorithm with 4 and 6 interactions (Karahan, 2008), in order to see the effect of the number of interactions to focus on the desired region. The simulation runs are replicated 10 times. The parameters used in the simulation runs are provided in Table 4.1.

Table 4.1 Interactive Test Parameters

	2 * 750 Knapsack Instance	3 * 750 Knapsack Instance	2 * 100×100 Assignment Instance	3 * 100×100 Assignment Instance
Test 1 Weights, w	(0.5, 0.5)	(0.33, 0.33, 0.33)	(0.5, 0.5)	(0.33, 0.33, 0.33)
Test 2 Weights, w	(0.8, 0.2)	(0.7, 0.2, 0.1)	(0.8, 0.2)	(0.7, 0.2, 0.1)
Test 3 Weights, w	(0.3, 0.7)	(0.2, 0.3, 0.5)	(0.3, 0.7)	(0.2, 0.3, 0.5)
Interactions	4, 6	4, 6	4, 6	4, 6
Population Size	300	400	300	400
τ_0	0.1	0.1	0.1	0.1
τ_H	0.0001	0.0001	0.0001	0.0001
Function evaluations	120000	160000	120000	160000
Replications	10	10	10	10

In each interaction stage, 2M solutions are filtered from the archive population, where M is the number of objectives, and presented to the DM to set the new reference point. However, in the first interaction, 4M solutions are presented since this is the first time preference information is elicited from the DM and it has a very significant effect to direct the search process to the correct region of the Pareto-optimal frontier. In the end of the simulation run, we present the best utility solution among the filtered 2M solutions from the final archive population, and the solution having the best utility among all archive members without any filtering.

In order to test the performance of the algorithm, we calculate the deviation of utility of the obtained solution from that of the true optimal solution:

$$Absolute\ Deviation = U - U^* \tag{4.5}$$

We also calculate the percent deviation to scale the obtained deviation value as follows:

$$Relative\ Deviation = \frac{U - U^*}{U^w - U^*} \tag{4.6}$$

where U is the utility value of the solution obtained by the algorithm, U^* is the utility value of the true optimal solution, and U^w is the utility value of the worst solution out of all nondominated solutions that are known. This performance metric is taken from Karahan (2008).

4.4.1 Multi-objective Knapsack Problem

2-objective 750-item Knapsack Problem

Test 1: In the first test, the best solution is obtained at the middle of the Pareto-optimal frontier by attaching equal weights to each of the objectives. Table 4.2 shows the results found for the cases where the final population is filtered or unfiltered before presenting the population to the DM.

Table 4.2 Interactive Test 1 Results for 2-objective 750-item KP

Solution	Inter.	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Duration (min)
Unfilter	4	669.2	5.996	631.00	2580.0	38.15	1.9574%	7.4882
Filter	4	671.3	6.084	631.00	2580.0	40.25	2.0652%	7.4882
Unfilter	6	666.4	5.116	631.00	2580.0	35.35	1.8138%	7.6622
Filter	6	666.4	5.116	631.00	2580.0	35.35	1.8138%	7.6622

In Figure 4.3, we observe that better convergence to the desired regions can be obtained by increasing the number of interactions. Moreover, the final preferred region gets narrower when more interactions are performed with the DM. Figure 4.3 shows that IC-TDEA successfully converges to the desired portion of the Pareto-optimal frontier. Although the deviation from the optimal solution increases when the final population is filtered for the 4-interaction case, this disadvantage of filtering is eliminated with the increasing number of interactions. For the 6-interaction case, the filtering process applied to the final population is able to obtain the same solution as the unfiltered case. Since filtering is important to decrease the decision making effort of the DM, this process may become advantageous by increasing the number of interactions with the DM.

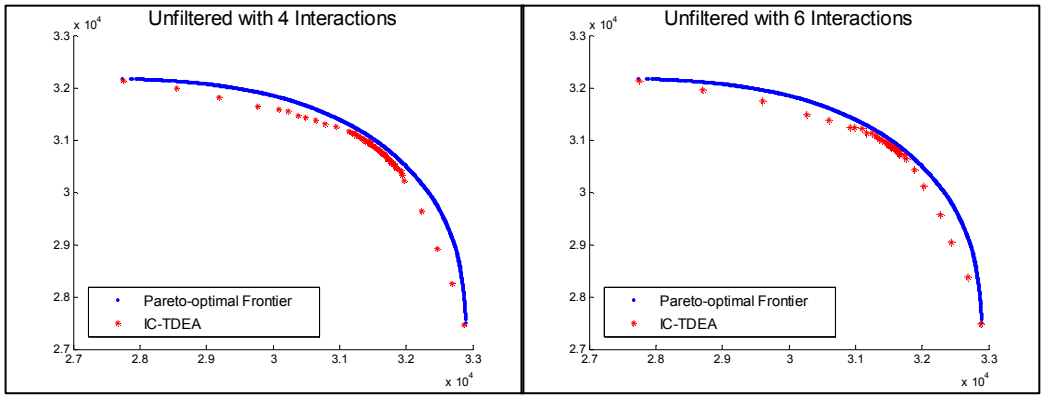

Figure 4.3 Interactive Test 1 Plots of 2-objective 750-item KP

Test 2: In this test, the DM favors Objective 1 much more than Objective 2. This test is utilized to test the performance of IC-TDEA is case of such a bias to deal with on the Pareto-optimal frontier. Table 4.3 shows that, the performance of the algorithm is similar to that in Test 1. While the filtered and unfiltered results are similar to each other, 6-interaction case performs much better than 4-interaction case. The results of the runs are presented in Figure 4.4.

Table 4.3 Interactive Test 2 Results for 2-objective 750-item KP

Solution	Inter.	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Duration (min)
Unfilter	4	481.6	2.278	437.6	4128.0	44.04	1.1934%	7.7825
Filter	4	482.9	3.621	437.6	4128.0	45.28	1.2270%	7.7825
Unfilter	6	477.4	5.356	437.6	4128.0	39.84	1.0796%	7.8313
Filter	6	478.3	6.267	437.6	4128.0	40.68	1.1023%	7.8313

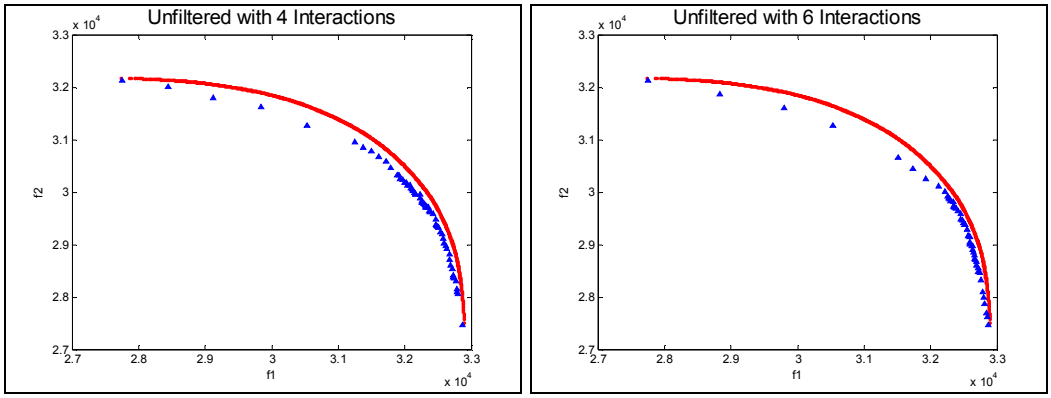

Figure 4.4 Interactive Test 2 Plots of 2-objective 750-item KP

Test 3: In this test, in contrast to Test 2, the DM favors Objective 2. Table 4.4 shows the deviations from the optimal point, and we see that the algorithm is able to converge to the region of the optimal point as in the case of previous two tests. Surprisingly, the algorithm works better in 4-interaction case. That is, the remaining interactions fail to guide the algorithm to the right direction within the narrower preferred regions. The final populations of the 4- and 6-interaction cases are presented in Figure 4.5.

Table 4.4 Interactive Test 3 Results for 2-objective 750-item KP

Solution	Inter.	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Duration (min)
Unfilter	4	601.2	3.586	559.3	3262.0	41.90	1.5503%	8.3947
Filter	4	604.8	4.133	559.3	3262.0	45.50	1.6835%	8.3947
Unfilter	6	608.7	3.867	559.3	3262.0	49.40	1.8278%	8.0714
Filter	6	611.3	4.101	559.3	3262.0	52.00	1.9240%	8.0714

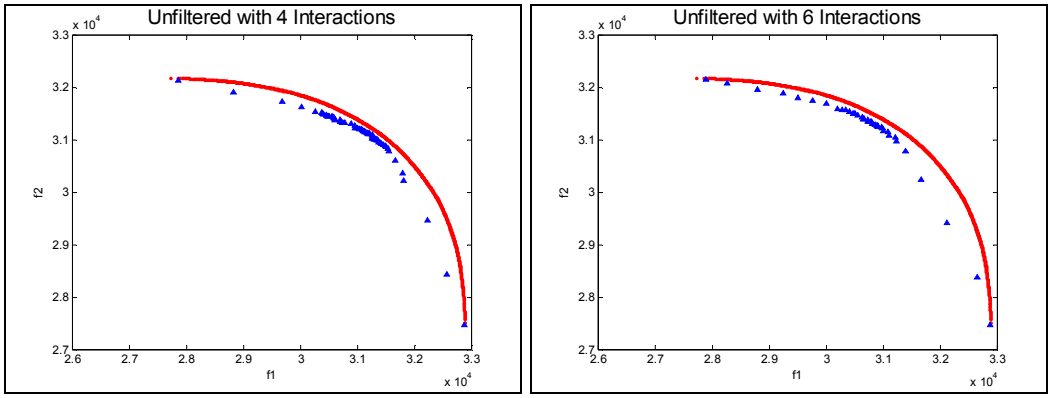

Figure 4.5 Interactive Test 3 Plots of 2-objective 750-item KP

3-objective 750-item Knapsack Problem

Test 1: In 3-objective test problem runs, we see that the differences between 4- and 6-interaction cases; and the filtered and unfiltered cases are more obvious. As seen in Table 4.5, standard deviations of the filtered cases are higher than the unfiltered cases. Moreover, the standard deviation gets smaller as we increase the number of interactions from 4 to 6. Although the algorithm converges to the middle of the Pareto-optimal frontier, where all objectives are approximately equally weighted, the performance of the algorithms worsens as we increase the number of objectives. We can follow the preferred regions in Figure 4.6 for both 4- and 6-interaction cases.

Table 4.5 Interactive Test 1 Results for 3-objective 750-item KP

Sol.	Int.	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Duration (min)
Unfilt.	4	656.70	6.561	605.22	1856.58	51.48	4.1139%	17.7138
Filt.	4	687.10	6.975	605.22	1856.58	81.84	6.5401%	17.7138
Unfilt.	6	640.20	5.455	605.22	1856.58	34.98	2.7954%	17.8944
Filt.	6	648.12	5.983	605.22	1856.58	42.90	3.4283%	17.8944

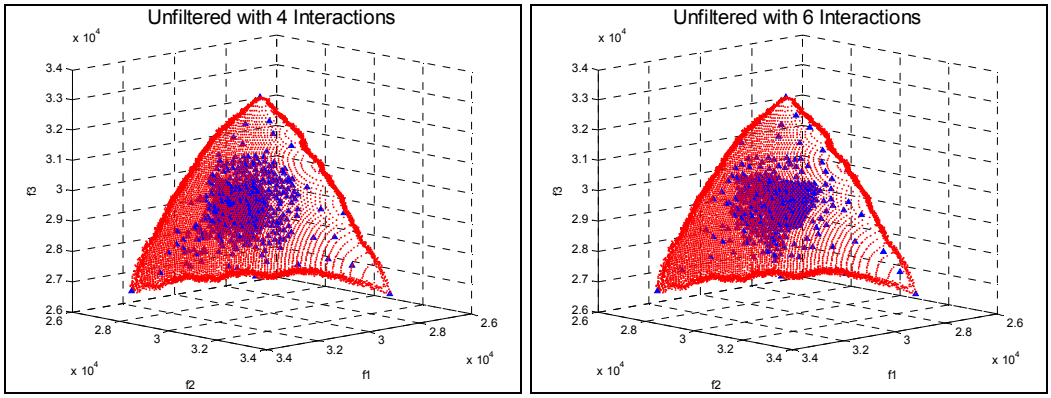

Figure 4.6 Interactive Test 1 Plots of 3-objective 750-item KP

Test 2: In this test, Objective 1 is favored more than the remaining two objectives. We can see in Table 4.6 that, the performance of the algorithm is not much affected by this bias, since the performance metric results are similar to Test 1. The plot of the final population is presented in Figure 4.7.

Table 4.6 Interactive Test 2 Results for 3-objective 750-item KP

Sol.	Inter.	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Duration (min)
Unfilt.	4	556.4	6.432	432.2	3810.1	124.20	3.6768%	16.8509
Filt.	4	556.5	6.520	432.2	3810.1	124.30	3.6798%	16.8509
Unfilt.	6	515.6	5.651	432.2	3810.1	83.40	2.4690%	17.6166
Filt.	6	518.3	5.412	432.2	3810.1	86.10	2.5489%	17.6166

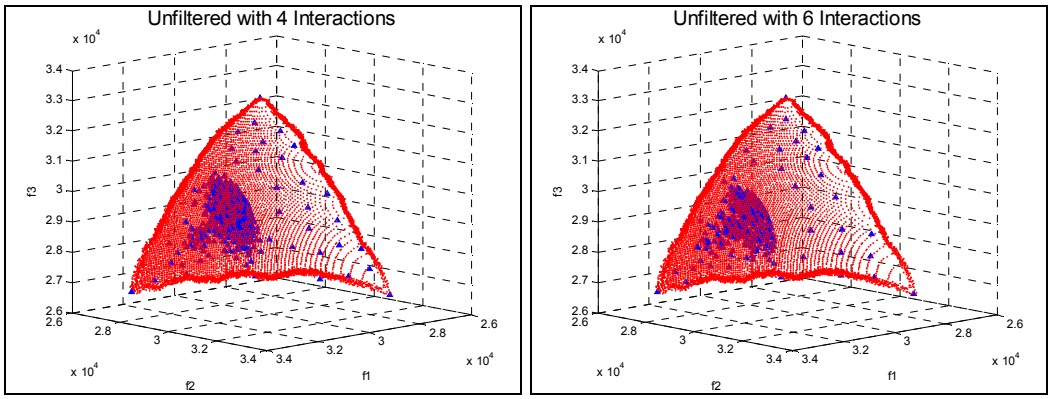

Figure 4.7 Interactive Test 2 Plots of 3-objective 750-item KP

Test 3: In this test, the bias is for the third objective, which has been finely captured by IC-TDEA, as seen in Figure 4.8. Table 4.7 shows that, the algorithm shows its best 3-objective knapsack performance in this test. The final population plots are provided in Figure 4.8, where the obtained solutions have converged to the region of higher Objective 3 values.

Table 4.7 Interactive Test 3 Results for 3-objective 750-item KP

Sol.	Int.	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Duration (min)
Unfilt.	4	636.0	6.712	581.50	2813.0	54.50	2.4423%	16.9614
Filt.	4	650.5	6.928	581.50	2813.0	69.00	3.0921%	16.9614
Unfilt.	6	633.0	5.816	581.50	2813.0	494.46	2.3079%	17.9326
Filt.	6	639.9	6.025	581.50	2813.0	58.40	2.6171%	17.9326

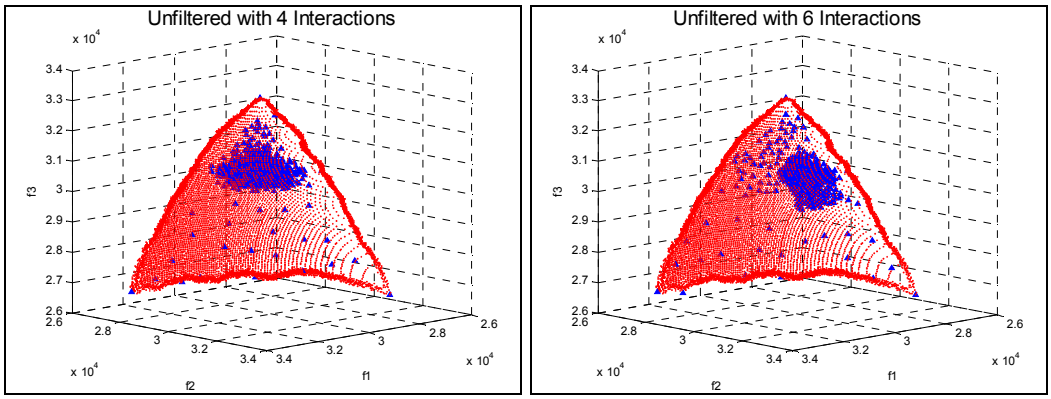

Figure 4.8 Interactive Test 3 Plots of 3-objective 750-item KP

4.4.2 Multi-objective Assignment Problem

2-objective 100×100 Assignment Problem

Test 1: In the first 2-objective assignment problem test, the best solution is obtained at the middle of the Pareto-optimal frontier. Table 4.8 shows the results found for the cases where the final population is filtered or unfiltered. In Figure 4.9, we observe that, limiting the improvement procedure to a prespecified number of generations hinders the final population to finely converge to the Pareto optimal frontier. However, IC-TDEA is still successful in converging to the desired portion of the Pareto-optimal frontier. The final preferred region gets narrower as the number of interactions is increased.

Table 4.8 Interactive Test 1 Results for 2-objective N=100 AP

Sol.	Inter.	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Duration (min)
Unfilt.	4	441.5	5.316	241.0	1889.5	200.50	12.1626%	6.5889
Filt.	4	445.5	6.478	241.0	1889.5	204.50	12.4052%	6.5889
Unfilt.	6	431.5	5.102	241.0	1889.5	190.50	11.5560%	6.6862
Filt.	6	436.5	5.411	241.0	1889.5	195.50	11.8593%	6.6862

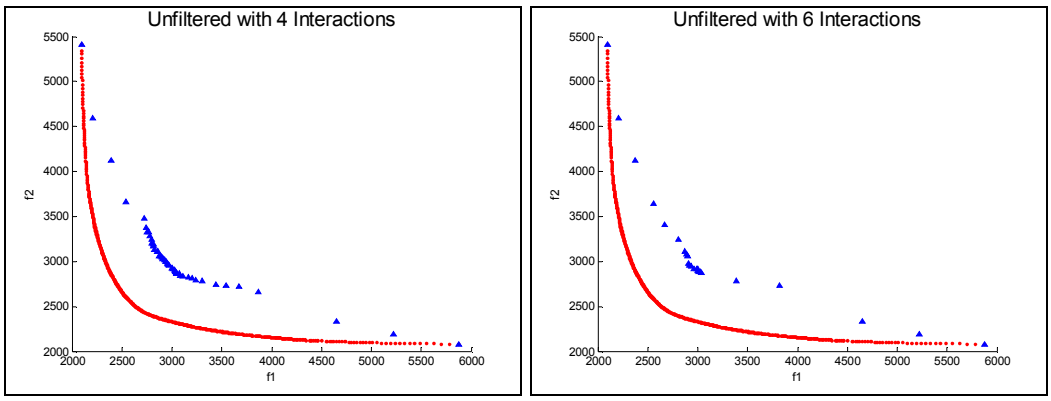

Figure 4.9 Interactive Test 1 Plots of 2-objective N=100 AP

Test 2: In this test, the DM favors the first objective. The performance of the algorithm is similar to that in Test 1 (see Table 4.9). 6-interaction case performs much better than 4-interactions case, encouraging more interactions with the DM. The results of the runs are presented in Figure 4.10.

Table 4.9 Interactive Test 2 Results for 2-objective N=100 AP

Sol.	Int.	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Duration (min)
Unfilt.	4	322.4	6.001	188.4	3023.2	134.00	4.7270%	6.3848
Filt.	4	330.4	6.220	188.4	3023.2	142.00	5.0092%	6.3848
Unfilt.	6	308.4	5.885	188.4	3023.2	120.00	4.2331%	6.5046
Filt.	6	310.5	5.999	188.4	3023.2	122.10	4.3072%	6.5046

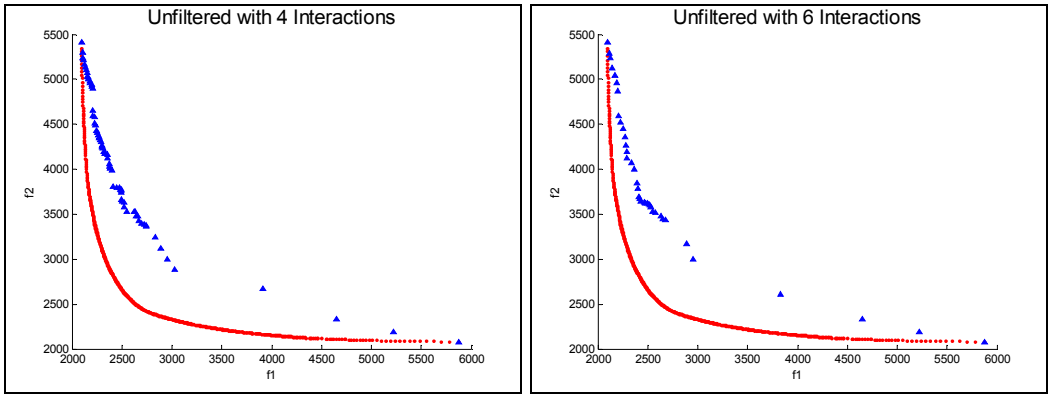

Figure 4.10 Interactive Test 2 Plots of 2-objective N=100 AP

Test 3: This time, the DM's preference is in favor of Objective 2. In Table 4.10, the performance metric results show that the algorithm is able to converge to the region of the optimal point. The final populations of the 4- and 6-interaction cases are presented in Figure 4.11.

Table 4.10 Interactive Test 3 Results for 2-objective N=100 AP

Sol.	Int.	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Duration (min)
Unfilt.	4	422.8	5.682	216.9	2331.7	205.90	9.7361%	6.6737
Filt.	4	424.6	5.825	216.9	2331.7	207.70	9.8213%	6.6737
Unfilt.	6	399.6	4.921	216.9	2331.7	182.70	8.6391%	6.5720
Filt.	6	405.8	5.127	216.9	2331.7	188.90	8.9323%	6.5720

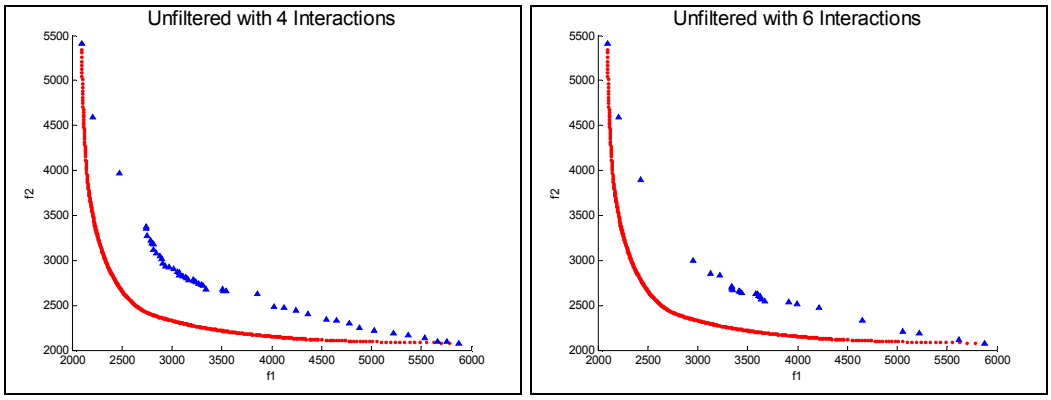

Figure 4.11 Interactive Test 3 Plots of 2-objective N=100 AP

3-objective 100×100 Assignment Problem

Test 1: As seen in Table 4.11, the performance of the algorithms gets worse as we increase the number of objectives. However, the algorithm is still able to converge to the middle of the Pareto-optimal frontier, the preferred region of this test. We can see the final preferred region in Figure 4.12 for both 4- and 6-interaction cases.

Table 4.11 Interactive Test 1 Results for 3-objective N=100 AP

Sol	Int	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Durat. (min)
Unfilt	4	640.20	10.101	332.31	1428.57	307.89	28.0855%	10.7785
Filt	4	648.12	11.226	332.31	1428.57	315.81	28.8079%	10.7785
Unfilt	6	552.59	9.873	332.31	1428.57	220.28	20.0938%	10.9896
Filt	6	552.10	10.118	332.31	1428.57	219.78	20.0482%	10.9896

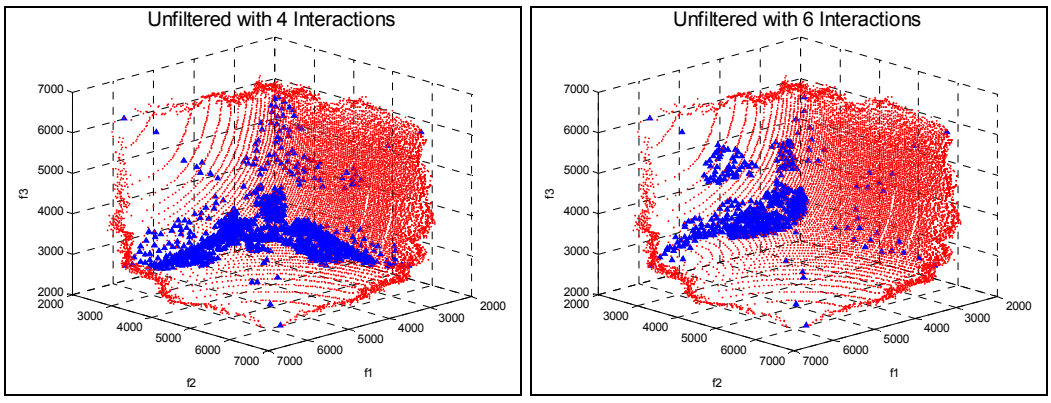

Figure 4.12 Interactive Test 1 Plots of 3-objective N=100 AP

Test 2: In this test, Objective 1 is favored more than the other two objectives. We can see in Table 4.12 that, the performance of the algorithm is much better compared to the performance metric results of Test 1. That is, the bias introduced in favor of an objective has improved the performance of the algorithm. The plot of the final population is presented in Figure 4.13.

Table 4.12 Interactive Test 2 Results for 3-objective N=100 AP

Sol	Int	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Durat. (min)
Unfilt	4	348.60	7.506	230.30	3030.30	118.30	4.2250%	10.4281
Filt	4	358.90	7.861	230.30	3030.30	128.60	4.5929%	10.4281
Unfilt	6	340.85	6.855	230.30	3030.30	110.55	3.9482%	10.3104
Filt	6	349.30	7.004	230.30	3030.30	119.00	4.2500%	10.3104

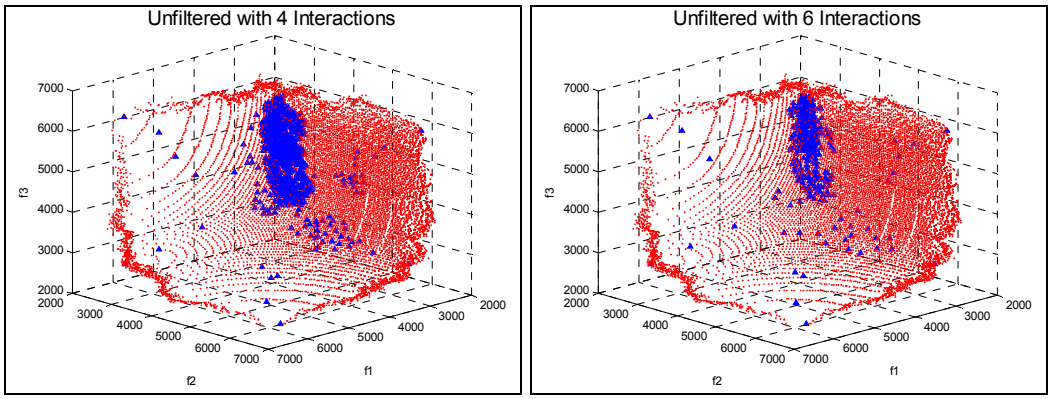

Figure 4.13 Interactive Test 2 Plots of 3-objective N=100 AP

Test 3: In this test, the bias is for the third objective as seen in Figure 4.14. Table 4.13 shows the deviation results from the optimal point of this test. The algorithm is able to converge to the preferred region of the DM. The final population plot is provided in Figure 4.14, where the obtained solutions have converged to the region of lower Objective 3 values. In this test, 4-interaction case performs better than 6-interaction case which shows that the population is not guided well during the last two interaction stages.

Table 4.13 Interactive Test 3 Results for 3-objective N=100 AP

Sol	Int	Mean	Std. Dev.	Opt.	Worst	Abs. Dev.	Rel. Dev.	Dur. (min)
Unfilt.	4	541.50	9.992	308.00	2159.50	233.50	12.6114%	10.5350
Filt.	4	549.90	11.864	308.00	2159.50	241.90	13.0651%	10.5350
Unfilt.	6	543.80	7.975	308.00	2159.50	235.80	12.7356%	10.0921
Filt.	6	551.40	8.112	308.00	2159.50	243.40	13.1461%	10.0921

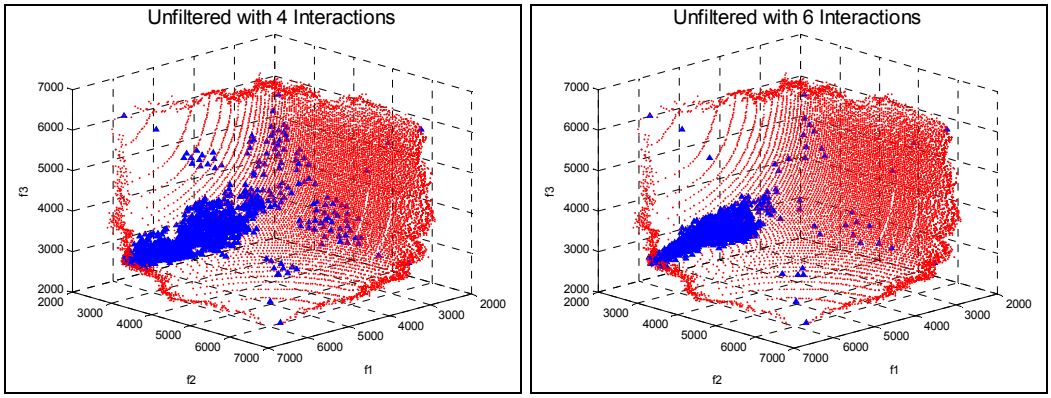

Figure 4.14 Interactive Test 3 Plots of 3-objective N=100 AP

4.4.3 Discussions

IC-TDEA can converge to the final preference region of the DM in an interactive manner. We show this on various 2- and 3-objective test problems and by using different utility functions for the DM. The algorithm can converge to the region of interest in all of the test problems. As we increase the number of interactions with the DM, we can obtain a more realistic final preference region approximation. Since the filtering procedure is seen to mislead the process, the filtering method can be revised, which has also been suggested by Karahan (2008).

CHAPTER 5

CONCLUSIONS

In this study, we propose a multi-objective evolutionary algorithm (MOEA), the Combinatorial Territory Defining Evolutionary Algorithm (C-TDEA) to solve multi-objective combinatorial optimization (MOCO) problems. We handle the single-dimensional multi-objective knapsack problem (MOKP) and multi-objective assignment problem (MOAP). We introduce a new repair-improvement procedure that serves both convergence and diversity at the same time.

We test the performance of C-TDEA against well-known MOEAs in the literature on 2-, and 3-objective randomly generated problem instances of MOKP and MOAP. We discuss how the territory defining property of Karahan and Köksalan (2008) facilitates the solution of MOCO problems by providing fast execution. Moreover, the repair and improvement mechanisms, and the other modifications in the original algorithm are realized to help in providing convergence and preserving diversity for the selected MOCO problems. We observe that C-TDEA performs well on Hypervolume and I-Epsilon performance metrics in all problems. In all of the problems, it outperforms other algorithms in both metrics.

We also propose a preference incorporation mechanism that focuses on the regions that are of special interest for the decision maker. We propose an interactive version of C-TDEA called Interactive Combinatorial Territory Defining Evolutionary Algorithm (IC-TDEA). Assuming that the decision maker does not have any idea on his preferred region at the beginning of the search, IC-TDEA guides the search by eliciting information from the decision maker in an interactive manner and moves the population towards the regions of interest. In computational tests, we observe that the interactive version of the algorithm is able to converge to the regions that are preferred by the decision maker.

There is a limited number of multi-objective evolutionary algorithms available in the literature that deal with combinatorial optimization problems. This study aims to contribute to fill this gap in the MCDM literature. The algorithm obtains diversity in the final population. The territory defining property and the repair-improvement procedure serves for diversity and these properties are also utilized to incorporate the preferences of the decision maker into the search process.

Although the repair-improvement procedures serve to diversity substantially, they cast shadow on the fast execution advantage of the territory defining property. For instance, the improvement mechanism used in the assignment problem is time consuming and we have to limit the algorithm to make improvements in only a few initial populations. In a future research, the repair-improvement procedures need to be revised to converge to the Pareto-optimal frontier within shorter computational time.

We test the algorithms for only multi-objective knapsack and assignment problems. In order to show the generality of the algorithm, it needs to be tested on other MOCO problems as a future research direction.

In this study, we test the performance of the algorithms on only 2- and 3-objective test problems. It remains as a future research to test the algorithms on problems with higher number of objectives in order to better assess on the performance of our algorithms.

REFERENCES

Branke, J., Deb, K. 2004. Integrating User Preferences into Evolutionary Multi-Objective Optimization. *Knowledge Incorporation in Evolutionary Computation* 461-477.

Branke, J., Kauβler, T., Schmeck, H. 2001. Guidance in Evolutionary Multi-objective Optimization. *Advances in Engineering Software* **32** 499-507.

Chetković, D., Parmee, I.C. 2002. Preferences and Their Application in Evolutionary Multi-objective Optimization. *IEEE Transactions on Evolutionary Computation* **6** 42-57.

Chu, P.C., Beasley J.E. 1998. A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics* **4**, 63-86.

Coello Coello, C. A. 2000. Handling Preferences in Evolutionary Multi-objective Optimization: A Survey. 2000 *Congress Evolutionary Comput*. **1** 30-37.

Coello Coello, C. A., Lamont, G. B., Veldhuizen, D. A. V. 2006. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer.

Deb, K. 1999. Solving Goal Programming Problems Using Multi-Objective Genetic Algorithms. *In Proceedings of Congress on Evolutionary Computation* 77-84.

Deb, K. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms.* Wiley, Chichester, UK.

Deb, K., Kumar, A. 2007. Interactive Evolutionary Multi-Objective Optimization and Decision-Making using Reference Direction Method. *In Proceedings of the Genetic and Evolutionary Computation Conference* 781-788.

Deb, K., Mohan, M., Mishra, S. 2005. Evaluating the Є-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. *Evolutionary Computation* **13(4),** 501-525.

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(2)** 182–197.

Deb, K., Sundar, J. 2006. Reference Point Based Multi-Objective Optimization Using Evolutionary Algorithms. *In Proceedings of the Genetic and Evolutionary Computation Conference* 635-642.

Ehrgott, M. 2000. *Multicriteria Optimization*. Springer Verlag, Berlin.

Ehrgott, M., Gandibleux, X. 2000. A survey and Annotated Bibliography of Multi-objective Combinatorial Optimization. *OR Spektrum* **22,** 425-460.

Ehrgott, M., Gandibleux, X. 2004. Approximate Solution Methods for Multi-objective Combinatorial Optimization. *Top* **12:1,** 1-90.

Jaszkiewicz, A. 2002. Genetic Local Search for Multi-Objective Combinatorial Optimization. *European Journal of Operational Research* **137**, 50-71.

Jin, Y., Sendhoff, B. 2002. Incorporation of Fuzzy Preferences into Evolutionary Multi-objective Optimization. *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, Vol.1 Singapore* 26-30.

Karahan, İ. 2008. Preference-Based Flexible Multi-objective Evolutionary Algorithms. *M.S. Thesis, Industrial Engineering Department, Middle East Technical University*.

Karahan, İ., Köksalan, M. 2008. Territory Defining Evolutionary Algorithm and Preference Incorporation.

Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S. 1999. Genetic Algorithms for Travelling Salesman Problem: A Review of Representations and Operators. *Artifical Intelligence Review* **13** 129-170.

Phelps, S., Köksalan, M. 2003. An Interactive Evolutionary Metaheuristic for Multi-objective Combinatorial Optimization. *Management Sci.* ***49*** 1726-1738.

Köksalan, M., Phelps, S. 2007. An Evolutionary Metaheuristic for Approximating Preference-Nondominated Solutions. *Informs Journal on Computing* **19** 291-301

Przybylski, A., Gandibleux, X., Ehrgott, M. 2007. Two Phase Algorithms for the Bi-objective Assignment Problem. *European Journal of Operational Research* **185,** 509-533.

Rachmawati, L., Srinivasan, D. 2006. Preference Incorporation in Multi-objective Evolutionary Algorithms: A Survey. *IEEE Congress on Evolutionary Computation* 962-968

Soylu, B., Köksalan, M. 2006. A Favorable Weight Based Evolutionary Algorithm for Multiple Criteria Problems. *Technical Report, for the coming IEEE Transaction on Evolutionary Computation.*

Steur, R. E. 1986. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley, New York.

Visée, M., Teghem, J., Pirlot, M., Ulungu, E.L. 1998. Two-phases Method and Branch and Bound Procedures to Solve the Bi-objective Knapsack Problem. *Journal of Global Optimization* **12,** 139-155.

Zitzler, E., Künzli, S. 2004. Indicator-Based Selection in Multi-objective Search. *Lecture Notes in Computer Science* **3242** 832-842.

Zitzler, E., Laumanns, M., Thiele, L. 2002. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. 95-100.

Zitzler, E., Thiele, L. 1998. Multi-objective Optimization Using Evolutionary Algorithms - A Comparative Case Study. *Parallel Problem Solving from Nature* **5** 292–301.

Zitzler, E., Thiele, L. 1999. Multi-objective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation* **Vol. 3, No 4** 257-271.

APPENDIX A

DETAILED PLOTS OF NSGA-II AND ε-MOEA FOR MULTI-OBJECTIVE KNAPSACK PROBLEM

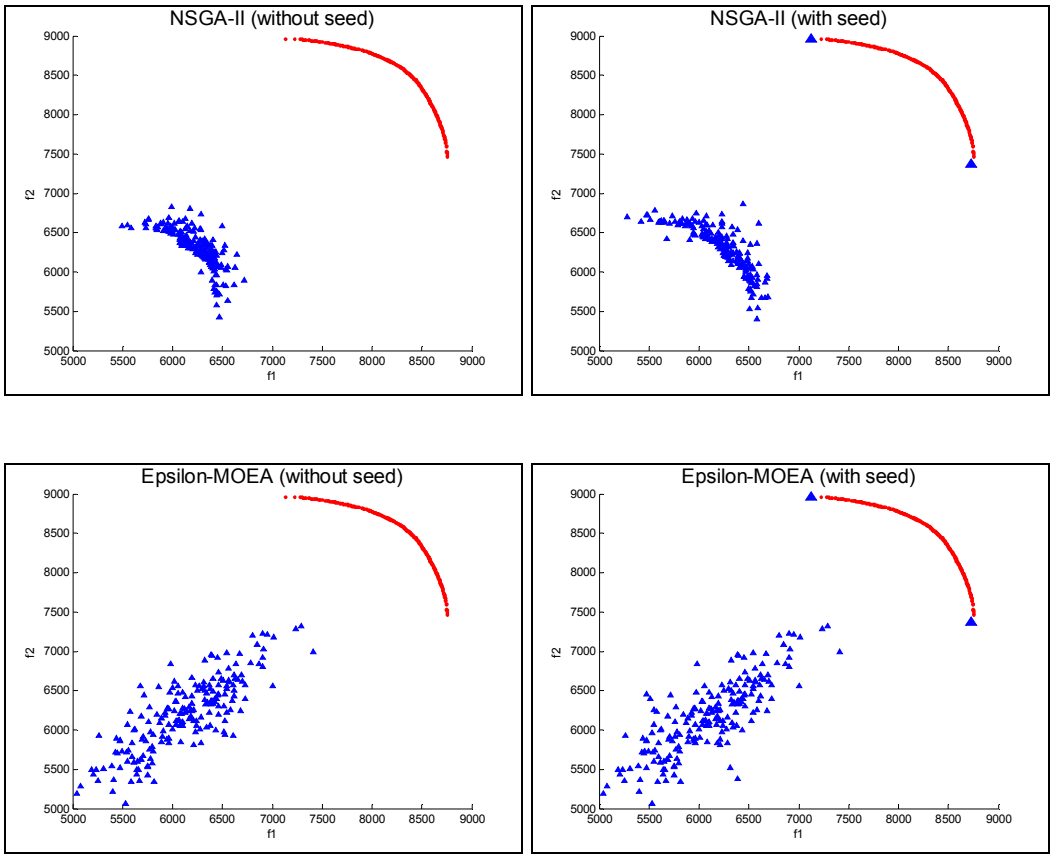

Figure A.1 Contender Plots for the 2-objective 200-item KP

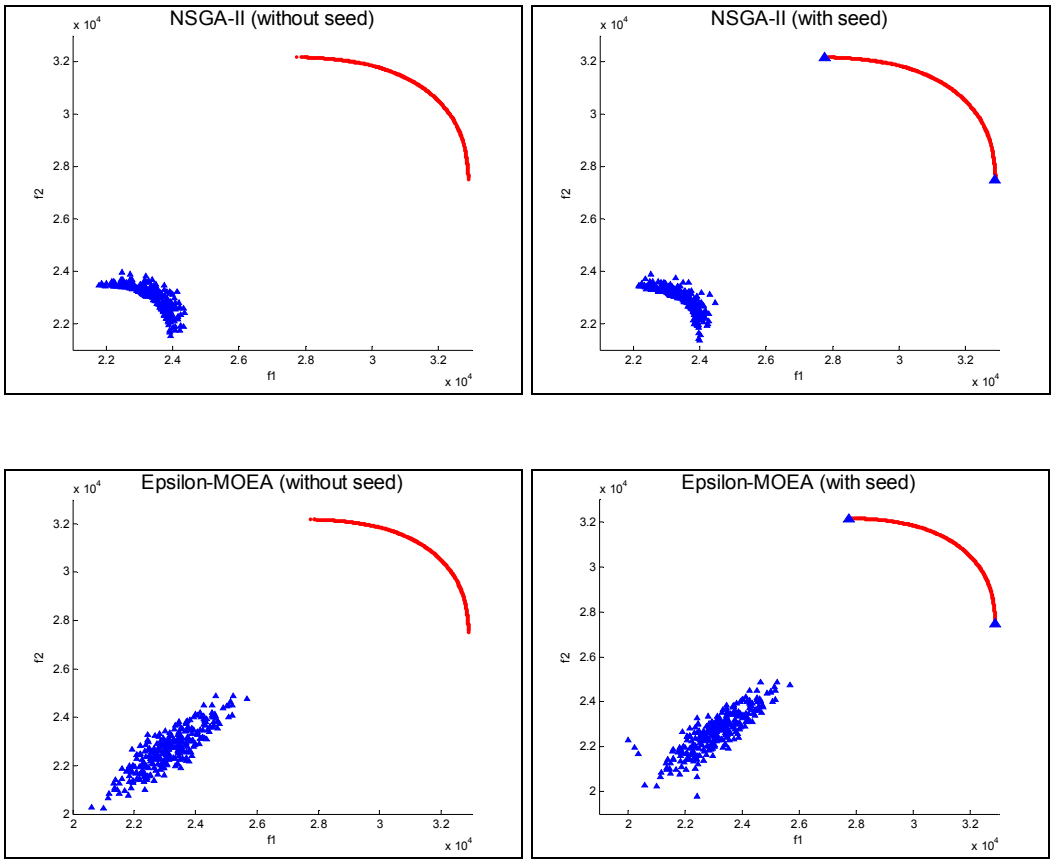

Figure A.2 Contender Plots for the 2-objective 750-item KP

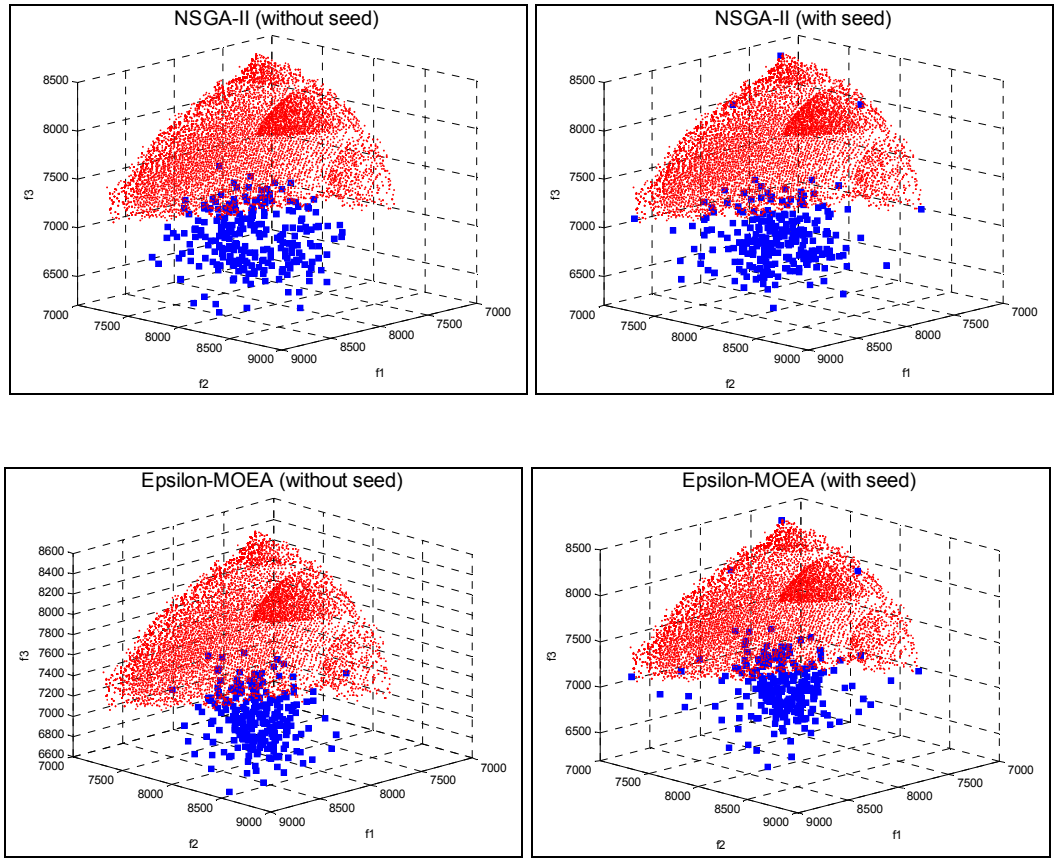

Figure A.3 Contender Plots for the 3-objective 200-item KP

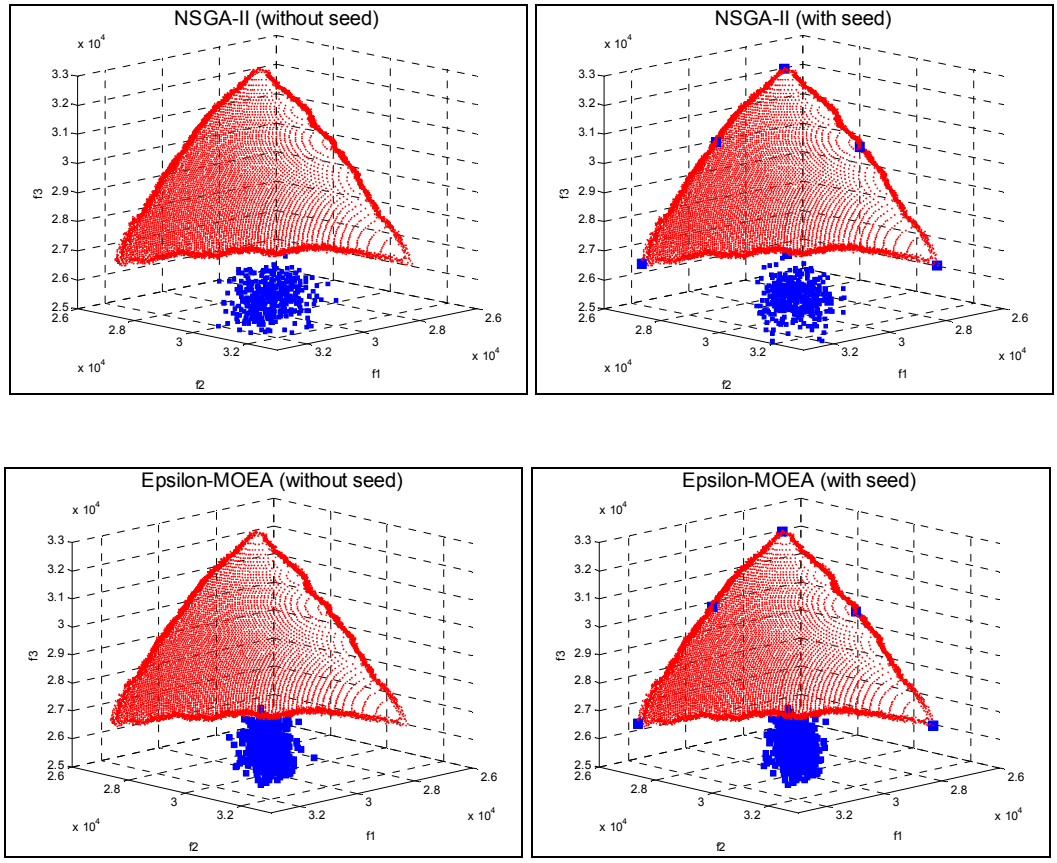

Figure A.4 Contender Plots for the 3-objective 750-item KP

APPENDIX B

DETAILED PLOTS OF NSGA-II AND ε-MOEA FOR MULTI-OBJECTIVE ASSIGNMENT PROBLEM

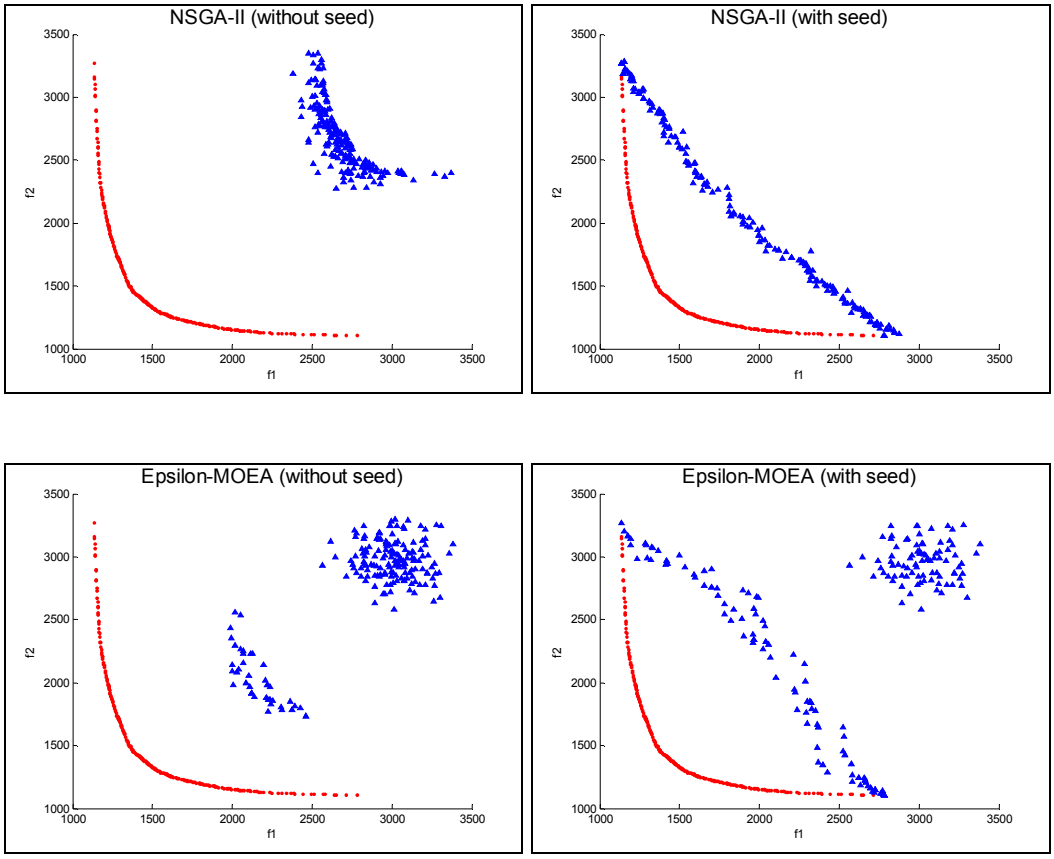

Figure B.1 Contender Plots for the 2-objective N=50 AP

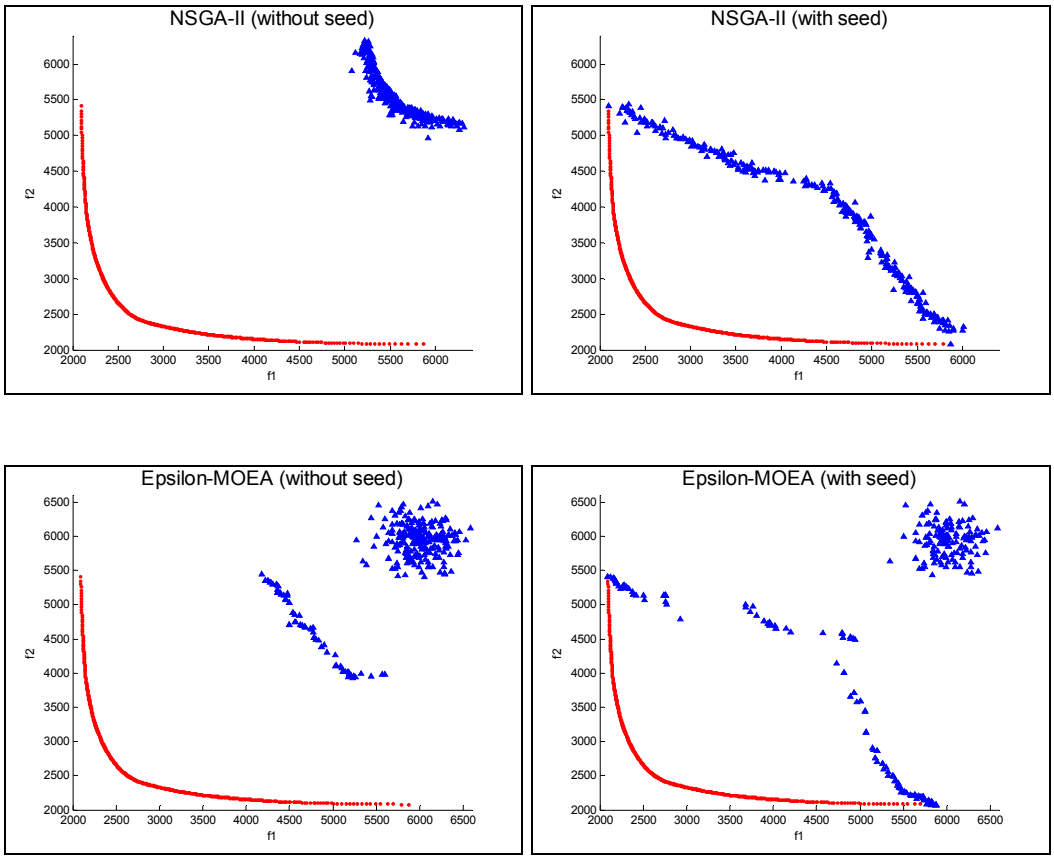

Figure B.2 Contender Plots for the 2-objective N=100 AP

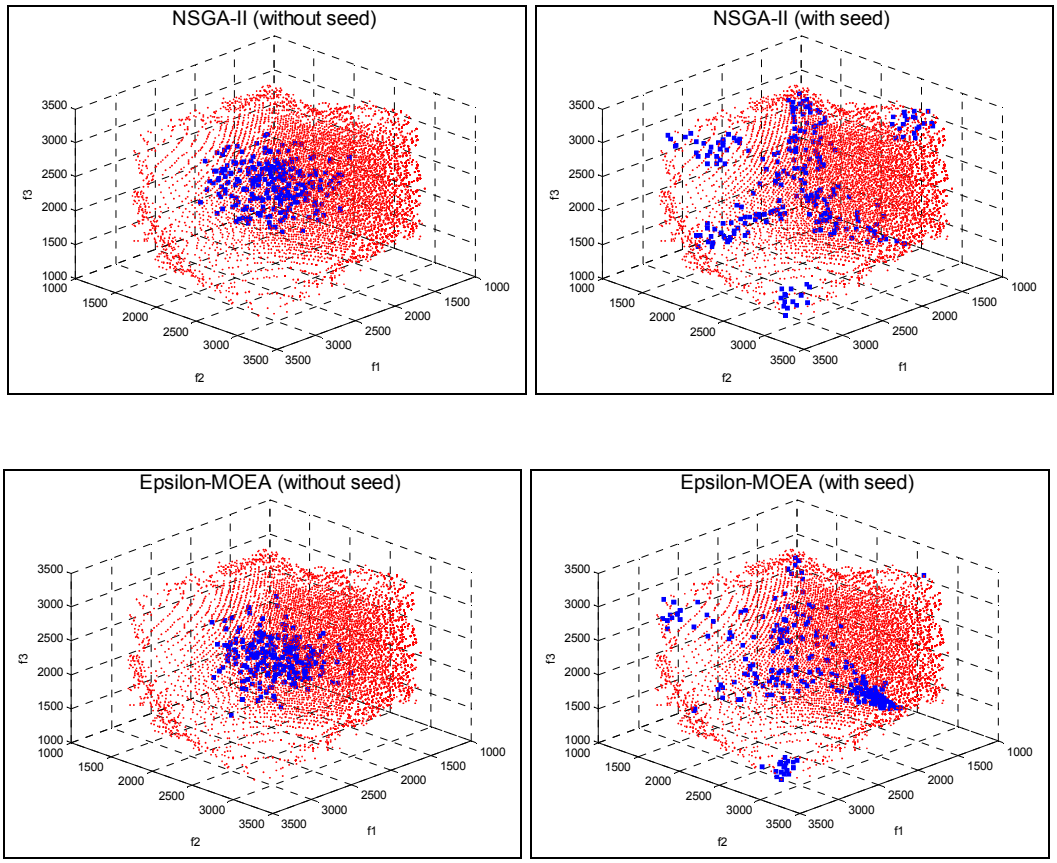

Figure B.3 Contender Plots for the 3-objective N=50 AP

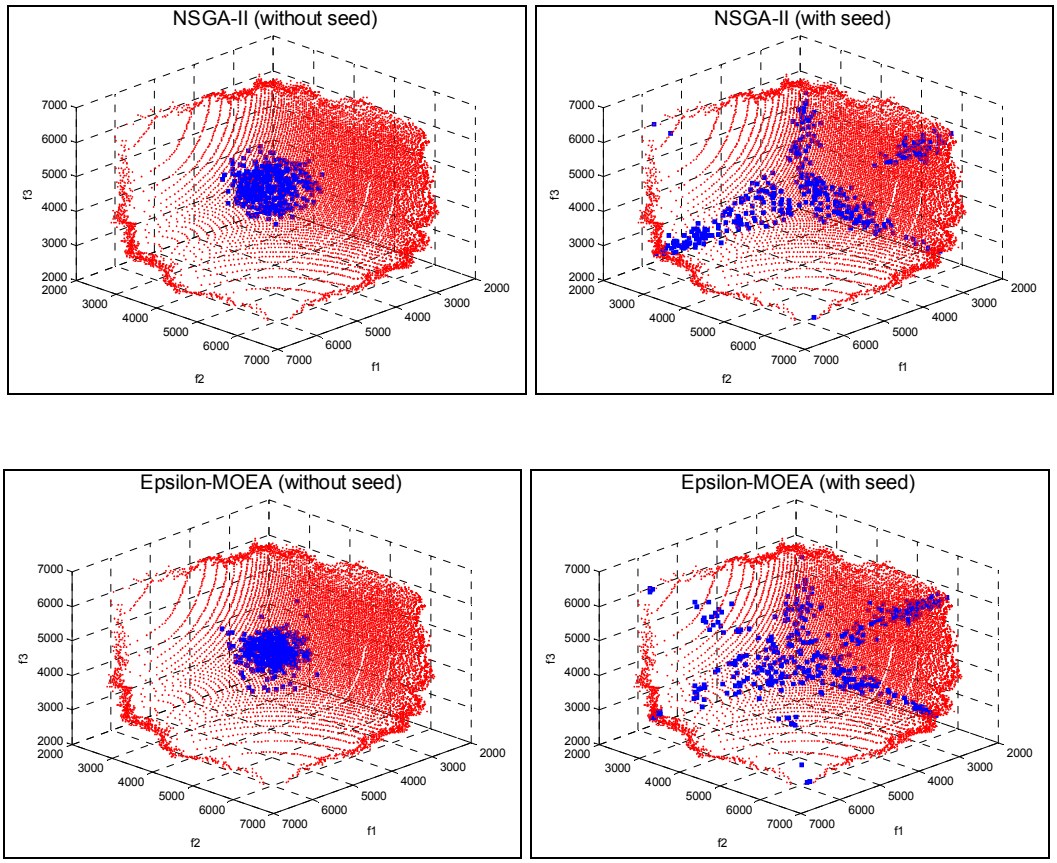

Figure B.4 Contender Plots for the 3-objective N=100 AP