AN EVOLUTIONARY ALGORITHM FOR MULTIPLE CRITERIA PROBLEMS

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

BY

BANU SOYLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
INDUSTRIAL ENGINEERING

JANUARY 2007

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

_____

Prof. Dr. Çağlar Güven
Head of Department

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

_____

Prof. Dr. Murat Köksalan
Supervisor

Examining Committee Members

Prof. Dr. Nur Evin Özdemirel	(METU, IE)	_____
Prof. Dr. Murat Köksalan	(METU, IE)	_____
Prof. Dr. Selim Aktürk	(Bilkent Unv., IE)	_____
Assoc. Prof. Dr. Canan Sepil	(METU, IE)	_____
Asst. Prof. Dr. Esra Karasakal	(METU, IE)	_____

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

Name, Last name : Banu SOYLU

Signature :

ABSTRACT

AN EVOLUTIONARY ALGORITHM FOR MULTIPLE CRITERIA PROBLEMS

SOYLU, Banu

Ph.D., Department of Industrial Engineering
Supervisor: Prof. Dr. Murat KÖKSALAN

January 2007, 156 pages

In this thesis, we develop an evolutionary algorithm for approximating the Pareto frontier of multi-objective continuous and combinatorial optimization problems. The algorithm tries to evolve the population of solutions towards the Pareto frontier and distribute it over the frontier in order to maintain a well-spread representation. The fitness score of each solution is computed with a Tchebycheff distance function and non-dominating sorting approach. Each solution chooses its own favorable weights according to the Tchebycheff distance function. Some seed solutions at initial population and a crowding measure also help to achieve satisfactory results.

In order to test the performance of our evolutionary algorithm, we use some continuous and combinatorial problems. The continuous test problems taken from

the literature have special difficulties that an evolutionary algorithm has to deal with. Experimental results of our algorithm on these problems are provided.

One of the combinatorial problems we address is the multi-objective knapsack problem. We carry out experiments on test data for this problem given in the literature.

We work on two bi-criteria p-hub location problems and propose an evolutionary algorithm to approximate the Pareto frontiers of these problems. We test the performance of our algorithm on Turkish Postal System (PTT) data set (TPDS), AP (Australian Post) and CAB (US Civil Aeronautics Board) data sets.

The main contribution of this thesis is in the field of developing a multi-objective evolutionary algorithm and applying it to a number of multi-objective continuous and combinatorial optimization problems.

Keywords: Multi-objective Evolutionary Algorithms, Multi-objective Optimization, Multi-objective Combinatorial Optimization.

ÖZ

ÇOK KRITERLİ PROBLEMLER İÇİN EVRİMCİ BİR ALGORİTMA

SOYLU, Banu

Doktora, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Murat KÖKSALAN

Ocak 2007, 156 sayfa

Bu tezde çok amaçlı sürekli ve birleşi en iyileme problemlerinin Pareto sınırına yaklaşmak için evrimci bir algoritma geliştirdik. Algoritma çözümler kümesini, Pareto sınıra doğru ilerletmeye ve iyi bir yayılım sağlamak için Pareto sınırı boyunca dağıtmaya çalışmaktadır. Her çözümün uygunluk değeri Tchebycheff uzaklık ölçüsü ve baskın olmayan sıralama yaklaşımı ile hesaplanmaktadır. Her çözüm Tchebycheff uzaklık ölçüsüne göre kendisi için en uygun ağırlıkları seçer. Başlangıç kümesine konulan bazı kaliteli çözümler ve kalabalıklık ölçüsü de yeterli sonuçlar almak için yardımcı olur.

Evrimci algoritmamızın performansını test etmek için bazı sürekli ve birleşi problemlerini kullandık. Literatürden alınan sürekli test problemleri evrimci algoritmaların ilgilenmesi gereken özel zorluklara sahiptir. Algoritmamızın bu problemler üzerindeki deneysel sonuçları verilmiştir.

Üzerinde çalıştığımız birleşi problemlerinden biri çok amaçlı sırt çantası problemidir. Literatürde bu problem için verilen test verisi üzerinde deney yaptık.

İki tane iki amaçlı p merkez üssü yerleştirme problemi üzerinde çalıştık ve bu problemlerin Pareto sınırına yaklaşmak için evrimci bir algoritma önerdik. Algoritmamızın performansını Türk Posta Sistemi (PTT) verisi (TPDS), AP (Avustralya Postası) ve CAB (ABD Sivil Havacılık Kurulu) verileri üzerinde denedik.

Bu tezin temel katkısı çok kriterli evrimci bir algoritma geliştirme ve onu çok kriterli sürekli ve birleşi en iyileme problemlerine uygulama alanındadır.

Anahtar kelimeler: Çok Amaçlı Evrimci Algoritmalar, Çok Amaçlı En İyileme, Çok Amaçlı Birleşi En İyileme.

To my family

and

To the memory of 21 students of Erciyes University and Niğde University who died in a terrific accident on October 24th, 1997.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Prof. Dr. Murat Köksalan for their valuable and patient supervision and continual support through the course of this study. I feel myself very lucky to have the chance of working with such an excellent supervisor.

I am also grateful to Prof. Dr. Meral Azizoğlu and Prof. Dr. Nur Evin Özdemirel for their encouragement and morale support that re-established my confidence many times throughout this study.

I would like to thank the jury members of my PhD committee who monitored this study and took effort in reading and providing me valuable comments on this thesis: Prof. Dr. Nur Evin Özdemirel, Prof. Dr. Selim Aktürk, Assoc. Prof. Dr. Canan Sepil and Asst. Prof. Dr. Esra Karasakal.

I would like to express my deepest thanks to my mother Kadriye Soylu, my father Ömer Osman Soylu and my sister Aysu Soylu for their continued patience and understanding. The completion of this task would not have been possible without their endless love and faith in me. I am also grateful to my grand mother Sabire Kazan; thanks for caring me all the time and for being a wonderful friend.

I am very grateful to my friends Leman Esra Dolgun, Selin Özpeynirci, Banu Lokman and Tülin İnkaya for their help, consideration and cheerful presence. Although their names do not appear individually here, I am grateful to all my professors, my friends and my colleagues in IE Department. I am very happy and proud of meeting and working with them.

Finally, my special thanks go to my dear friends Yunus Ege, Hayriye Sundu, Sibel Korkmazgil, Şule Okuroğlu, Nuray Ateş, Nazife Erkurşun and Aysun Sertiç: Thank you all for your everlasting support, continuous motivation, consideration and cheerful talks. I will always remember the wonderful time spent together with you.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURE

CHAPTER 1

INTRODUCTION

Many real life problems are actually Multiple Criteria Problems (MCPs). There are successful applications of evolutionary algorithms (EAs) to many MCPs, such as multiple criteria knapsack, spanning tree, assignment, location problems etc. They also find their applications into continuous optimization problems such as design problems, chemical processes etc. EAs are inspired by the idea of Darwin's natural evolution. They maintain a population of solutions at each iteration, instead of a single solution. This is one of the main differences between EAs and other heuristic approaches. In multiple criteria problems, this property of EAs has been used to obtain multiple efficient solutions in a single run.

While designing a new multi-objective EA (MOEA), one of the main issues is related with the performance of the algorithm. Many algorithms have been proposed and new algorithms keep being developed. While choosing the appropriate performance measures and evaluating the performance of MOEAs, both diversity of solutions and their proximity to the efficient frontier should be considered.

In this thesis, our aim is to propose new multi-objective EAs for MCPs. The EAs presented in this study have a different perspective mainly based on favorable weight and seeding mechanisms. Similar to many other multi-objective approaches, our aim is to approximate the Pareto frontier and distribute the solutions evenly over the Pareto frontier. Non-dominated sorting concept of Goldberg (1989) and a crowding measure also help to achieve these goals. We evaluate the performance of the algorithms by using different measures.

The outline of the chapters is given below.

The thesis includes six chapters. In Chapter 2, we present a brief overview of the literature related with MOEAs. Note that Chapters 4 and 5 have also their own specific literature review sections.

In Chapter 3, we introduce FWEA (Favorable Weight based EA), for approximating the Pareto frontier of MCPs. In this chapter, we compare the performance of FWEA with NSGAII, which is one of the well-known benchmarks in the literature, and provide the results of experiments on 2, 3 and 5 criteria continuous test problems.

In Chapter 4, we address two bi-criteria uncapacitated multiple allocation p-hub location problems (Bp-HLP1 and Bp-HLP2). We evaluate location and allocation decisions of these problems separately and propose FWEA_loc and FWEA_alloc algorithms to approximate the efficient hub locations and efficient allocation structures respectively. We test the performances of these algorithms on Turkish Postal System (PTT) data set (TPDS), AP (Australian Post) and CAB (US Civil Aeronautics Board) data sets.

In Chapter 5, we consider the multi-objective knapsack problem (MOKP). We adapt FWEA to approximate the Pareto frontier of MOKP. We call the new EA as FWEA_KP. We compare the performance of FWEA_KP with EMAPS, NSGAII and SPEA2 on test data for MOKP.

In Chapter 6, we conclude by summarizing the contribution of this thesis and suggesting future research directions.

CHAPTER 2

SOME DEFINITIONS AND LITERATURE REVIEW

In multiple criteria optimization, conflicting objectives are evaluated over a set of feasible solutions. Due to trade-offs between multiple objective functions, a unique feasible solution optimizing all the objective functions simultaneously is unusual. So, one has to deal with a large number of efficient solutions. A solution is said to be efficient (Pareto) if there does not exist any other solution performing at least as well in terms of all objectives and better in terms of at least one objective. All efficient solutions to the problem define the efficient (Pareto) frontier. Let x_j, j=1,2,…,n be the j^{th} decision variable. Let X be the set of feasible solutions and $f_k(x)$, k = 1,2, … ,m be the value of k^{th} objective for solution $x = (x_1, x_2,\ldots, x_n) \in X$. Each solution x in the decision variable space corresponds to a point $z = (z_1, z_2,\ldots, z_m) \in Z$ in the criteria space. According to this, a MCP can be formulated as:

$$\text{"Max"}\{f_1(x), f_2(x),\ldots, f_m(x)\}$$

subject to $x \in X$

where we have m (≥ 2) objective functions $f_k(x): \Re^n \longrightarrow \Re$. Thus, a MCP consists of n decision variables and m objective functions. If we know an $x' \in X$ such that $f_k(x') \geq f_k(x)$ for all k=1,2, … , m, with at least one strict inequality, then we can say that x' dominates x. Note that here we assume all criteria are maximization type. In the minimization case, opposite is true for dominance.

The components (z_k^*) of the ideal criterion vector shown in Figure 2.1 are obtained by maximizing each of the objective functions individually, i.e., by solving

Maximize $f_k(x)$ (for k = 1, 2, … , m)

Subject to $x \in X$,

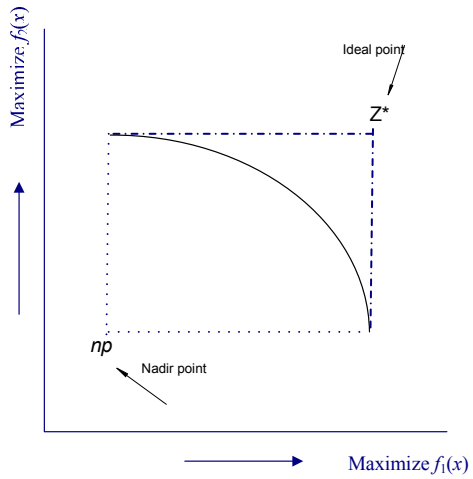

Figure 2. 1 Illustration of Ideal and Nadir point

The components of the nadir criterion vector, i.e. the lower bounds (for "Max" type problems) of the Pareto frontier, can be estimated from payoff table (Table 2.1). One should be aware that these estimates could be poor estimates and may affect the performance of an approach that relies too much on the quality of nadir point estimation. Let z_{ik} be the value of objective *i* for the decision variables that maximize objective *k* and z_{ik} is equal to the ideal value of objective *k* for *i=k*.

Table 2. 1 Computation of nadir criterion vector for "Max" type problems

Criterion i \ k	1	2	.	.	m
1	z_1^*	z_{21}	.	.	z_{m1}
2	z_{12}	z_2^*	.	.	z_{m2}
.	.	.	.	.	.
.	.	.	.	.	.
m	z_{1m}	z_{2m}	.	.	z_m^*

In Table 2.1, the components of the nadir criterion vector np_k is equal to $\underset{k}{Min}\{z_{ik}\}$.

Some Definitions

Metrics measure the distance between two points (**a** and **b**) in the space. An L_q-metric for the points **a** and **b** is computed as follows:

$$\|\mathbf{a}-\mathbf{b}\|_q = \left[\sum_{k=1}^{m}\left|a_k - b_k\right|^q\right]^{\frac{1}{q}} \qquad q \in \{1,2,...\} \cup \{\infty\}$$

A weighted L_q-metric for the points **a** and **b** is computed as follows:

$$\|\mathbf{a}-\mathbf{b}\|_q^w = \left[\sum_{k=1}^{m}(w_k\left|a_k - b_k\right|)^q\right]^{\frac{1}{q}} \qquad q \in \{1,2,...\} \cup \{\infty\}$$

where $w \in \Re^m$ is a nonnegative vector of weights and $\sum_{k=1}^{m} w_k = 1$.

A solution z^i is said to be convex dominated by $z^1, z^2, ..., z^t$ if there exists $w \geq 0$, $\sum_{r \neq i} w_r = 1$ such that $\sum_{r \neq i} w_r . z^r \geq z^i$

For each supported efficient solution, there exists a w vector satisfying $w \geq 0$, $\sum_{k=1}^{m} w_k = 1$ that solves the below weighted sum problem.

$$\text{Maximize} \sum_{k=1}^{m} w_k . f_k\ (x)$$
$$\text{subject to} \quad x \in X$$

$$\text{for some } w \geq 0 \text{ and } \sum_{k=1}^{m} w_k = 1$$

It should be noted that a non-supported (convex dominated) efficient solution cannot be obtained by solving the weighted sum of criteria problem. Figure 2.2 demonstrates supported and unsupported efficient solutions as well as inefficient solutions. In this figure, x^1 to x^5 are supported efficient solutions, x^6 is an unsupported efficient solution and x^7 is an inefficient solution.

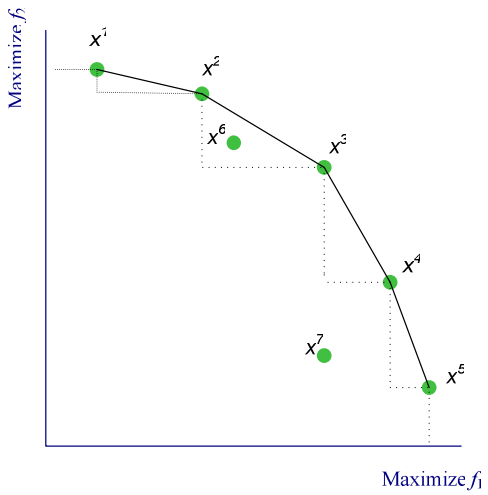

Figure 2.2 Illustration of supported, unsupported and inefficient solutions

Literature Review

Multi-objective Evolutionary Algorithms (MOEAs) and application of them on different problems have been an active research area with over 2600 references given at http://www.lania.mx/~ccoello/EMOO/EMOObib.html

In the review articles by Fonseca and Fleming (1995), Tamaki, Kita and Kobayashi (1996), Horn (1997), Veldhuizen and Lamont (2000), Deb (1999), Coello (1999), Jones et al. (2002) and Marler and Arora (2004), significant space is devoted to multiple criteria evolutionary algorithms. The survey papers of Multi-objective Combinatorial Optimization (MOCO) (Ulungu and Teghem, 1994; Ehrgott and Gandibleux, 2000; Jaszkiewicz, 2001) emphasize the increasing trend of evolutionary algorithms in the field of MOCO.

MOEAs have two purposes: (1) to converge to the efficient frontier and, (2) to achieve diverse and well spread set of solutions over the frontier. In the literature, for the first purpose, different fitness assignment strategies are applied. These fitness assignment strategies are generally categorized as non-dominated sorting concept based approaches and others (see Fonseca and Fleming, 1995; Coello, 1999). For the second purpose, crowding based algorithms (Dejong, 1975; Deb et al., 2000), sharing function based algorithms (Goldberg and Richardson, 1987) and clustering based algorithms (Zitzler et al., 2000) exist in the literature.

An important property of EA that we propose in this study is favorable weight mechanism. Therefore, we first review the literature considering the weight of criteria. Then, we categorize the other literature according to whether they include the non-dominated sorting concept or not.

Criteria Weights-Based Approaches

Schaffer (1984) proposes the first MOEA, which is called as Vector Evaluated Genetic Algorithm (VEGA). In VEGA, at every generation, GA population is divided into random subpopulations according to the number of objectives (say m).

Each objective function is assigned to a subpopulation, i.e. each subpopulation takes a weight vector of which one of the components is equal to 1.0. Then, each subpopulation is evaluated based on the corresponding objective function; however, it is not tested for the remaining *m*-1 objective functions.

Hajela and Lin (1992) introduce the weight based genetic algorithm in which the total fitness of each solution is calculated by summing each weighted objective function value (normalized). A different weight vector is assigned for each member. Therefore, multiple efficient solutions could be obtained in a single run. In order to preserve diversity, a sharing strategy is suggested in the weight vector space. Note that, if the distance between weight vectors of the neighboring solutions is small, sharing (as a function of distance) value will be high. Then, fitness value is reevaluated considering this sharing effect.

Hajela and Lin (1993) also propose the vector evaluated approach. Instead of assigning a different weight vector for each solution they set T different weight vectors. All N members of the population is evaluated for each weight vector $w^{(t)}$. Then the best N / T members according to each weight vector are copied to the parent subpopulation. Selection, crossover and mutation operations occur among the solutions of each subpopulation.

Knowles (2005) presents ParEGO, which is an extension of single objective EGO (efficient global optimization) algorithm in multi-objective framework. He converts the different objective values of a solution into a single value via a weighted augmented Tchebycheff function, where a different weight vector is chosen at each iteration. The performance of ParEGO is compared with NSGAII (Deb et al., 2000) on some test problems. According to reported results, ParEGO generally outperforms NSGAII especially when the worst-case performance is measured.

Köksalan and Phelps (2006) introduce an evolutionary metaheuristic, called EMAPS, for approximating preference-non-dominated solutions. EMAPS

combines proximity and diversity goals of MOEAs into a single fitness function by using a linear weighting function. A third goal has also been added to the other two. It aims to approximate only the relevant segments of the efficient frontier by using the partial information on DM preferences. This partial information is transformed into linear inequalities on criteria weights, which is used to constrain the set of weight vectors *W*. Since the favorable weights are the elements of *W*, the population of solutions evolves towards the considered regions of the objective function space *Z*.

In addition to EAs, Hansen (1997) proposes a tabu search based algorithm (MOTS). In this algorithm, the set of current solutions is optimized towards the Pareto frontier through manipulation of randomly chosen criteria weights. Ulungu et al. (1999) present a simulated annealing based approach (MOSA). The algorithm uses the criterion scalarization function, with linear or Tchebycheff distance functions. Jaskiewicz (2004) suggests the Pareto memetic algorithm, which does not use the Pareto ranking but applies a scalarizing functions-based selection mechanism. The main idea of the algorithm is to draw at random a weight vector in each iteration. The weight vector defines a scalarizing function. The efficiency of the algorithm is evaluated on the multi-objective knapsack problem.

Non-dominated Sorting Concept-Based Algorithms

Goldberg (1989) is the first who proposes the idea of using Pareto based fitness assignment strategy. The method categorizes the population according to frontiers. It assigns the non-dominated individuals to the first frontier and temporarily removes them from the population. It then finds a new set of non-dominated individuals, assigns to second frontier, then to third frontier etc. until all the population categorized.

The algorithms presented in this section generally use crowding measure, sharing function or clustering scheme to preserve diversity. Among them, sharing function

based algorithms are common. In these algorithms, fitness value of an individual *i* is reduced considering the close neighbors, *j,* of this individual. In order to do this, for each solution belonging to the same frontier, a sharing function value $Sh(d_{ij})$ is computed as follows (Goldberg and Richardson, 1987):

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\dfrac{d_{ij}}{\sigma_{share}} \right)^{\alpha}, & if \quad d_{ij} \le \sigma_{share}; \\ 0, & otherwise \end{cases}$$

where d_{ij} is the distance between *i* and *j*, and σ_{share} is the possible maximum distance for sharing. These sharing values are added together to obtain the niche count (nc_i) of a solution *i*. Density of solutions near individual *i* is estimated by the niche count.

Fonseca and Fleming (1993) introduce a multiobjective GA (MOGA) that uses the non-dominated classification of a GA population for the first time. First, a ranking scheme is applied. An individual's rank corresponds to the number of individuals dominating it. After determining the rank of the members, a raw-fitness is assigned to each member based on its rank. Then, starting from the best solution, a fitness score N (population size) through 1 is assigned. Thereafter, solutions of each rank are considered at a time and their fitness scores are averaged. In order to preserve diversity among non-dominated solutions, Fonseca and Fleming use the sharing approach on the objective function values among solutions of each rank.

Srinivas and Deb (1994) present the Non-dominated Sorting GA (NSGA) which uses the non-dominated sorting concept of Goldberg (1989). Since, the solutions on the first frontier are the closest ones to the Pareto frontier, highest raw-fitness value, which is proportional to the population size, is assigned to these solutions. Then, the fitness values of other frontier members are made worse progressively. To maintain the diversity of the population, they perform sharing on the decision variables instead of the objective function values among individuals of each class.

NSGA II (Deb et al., 2000) does not have much similarity with NSGA. After the offspring population is generated, parent and offspring populations are combined. Next, the combined population is categorized according to non-dominance. The members of the next generation are chosen from better ranked solutions. If all solutions with the same rank could not be included in the population due to limited population size, diverse (according to a crowding measure) set of solutions are selected.

Deb et al. (2005) propose a steady state ε-MOEA that incorporates ε-dominance concept. According to this concept, a solution is said to be efficient if there does not exist any other solution performing at least amount of ε well in terms of all objectives and better in terms of at least one objective. In the ε-MOEA, two populations (EA and archive) are evolved simultaneously and independently. Using one solution each from both populations, two offspring solutions are created. Each offspring is then used to update both parent and archive populations. The archive population is updated based on the ε-dominance concept, whereas a usual domination concept is used to update the parent population.

Other MOEAs

Allenson (1992) suggests Biobjective Gender Genetic Algorithm (BGGA). The algorithm solves biobjective problems. First, a gender is assigned to each solution randomly at birth. One of the objective functions is used to compute the fitness of male solutions and the other one is used to compute the fitness of female solutions. In the algorithm, only male-female crossovers are possible. Moreover, the chromosome of each solution includes a sexual attraction gene that gives information about percentile of the population that its mate comes from.

Horn et al. (1994) propose the Niched-Pareto Genetic Algorithm (NPGA) that combines tournament selection and the concept of Pareto dominance. Two competing individuals and a comparison set of other individuals are picked at random from the population. If one of the competing individuals is dominated by

any member of the set, then the other is chosen as winner of the tournament. If both individuals are dominated or non-dominated, the result is determined by sharing: the individual that is far away from the current offspring population wins.

Zitzler and Thiele (1999) introduce the Strength Pareto Evolutionary Algorithm (SPEA). The algorithm maintains a randomly generated population of size N and an external population (empty at the beginning) with a maximum capacity of $\overline{\mathrm{N}}$. At each generation *t*, the non-dominated solutions of the population are copied to the external population and then the dominated solutions in the external population are deleted. Genetic operations are performed on the external population of SPEA. For each member *i* of the external population, SPEA assigns a fitness (called strength), S_i, which shows the proportion of the current population members that an external population member *i* dominates:

$$S_i = \frac{n_i}{N+1}$$

where n_i is the number of current population members dominated by external population member *i.*

Thereafter, the fitness of a current population member *j* is assigned by adding strengths of the individuals that dominates *j* as follows;

$$F_j = 1 + \sum_{i>j} S_i$$

In order to reduce the size of the external population, close contenders are assumed to be clustered and just one solution is chosen from each cluster.

In SPEA II (Zitzler et al., 2001), fitness assignment strategy is updated by simply adding a density value to the fitness function as:

$$F(i) = R(i) + D(i)$$

where $R(i) = \sum_{j>i} S_j$ and $\mathrm{D}(i) = \frac{1}{\sigma_i^k + 2}$. Here S_j counts the number of solutions that member *j* dominates and σ_i^k measures the k^{th} largest distance to solution *i*.

Knowles and Corne (2000) suggest the Pareto-Archived Evolution Strategy (PAES). At each generation, PAES keeps an archive of the best solutions together with the parent and offspring populations. First, a random solution (parent) is chosen. It is then mutated. This new member is called as offspring. Then, the parent and offspring are compared for domination. If parent dominates offspring, a new offspring is generated. On the other hand, if offspring dominates parent, then this solution becomes the parent of the next generation and copied to the archive. However, there exist some strategies to include a member in the archive. First, it must not be dominated by any member of the archive. When it is included, the dominated members of the archive are deleted. If it is non-dominated by the other members of the archive, it is included if there is an empty place. If the archive is full, the density of solutions is computed with respect to *m* dimensional hypercube. The member located in a less dense hypercube is the winner.

Zitzler and Künzli (2004) propose an Indicator-Based Evolutionary Algorithm (IBEA). They integrate preference information of the decision maker into MOEA. They assume that the preferences of the decision maker are given in terms of a binary quality indicator, which gives the minimum distance such that one of the two approximations is weakly dominated. They sum up the indicator values with respect to the rest of the population and assign as a fitness value. They test the performance of algorithm on some test problems and compare the results with SPEA2 and NSGAII.

Beume et al. (2006) present a hypervolume measure based evolutionary algorithm (SMS-EMOA). The main idea of the algorithm is to maximize the dominated hypervolume within the optimization process. SMS-EMOA combines the archiving strategy by Knowles et al. (2003), non-dominated sorting concept of Goldberg (1989) and hypervolume metric of Zitzler et al.(2003). The performance of the algorithm is tested on some 2 and 3 objective continuous test problems and compare with the NSGAII, C-NSGAII, SPEA2 and ε-MOEA.

Nebro et al. (2005) suggest a scatter search based EA. They adapt the scatter search template of single objective optimization into multi-objective optimization by incorporating non-dominated sorting and crowding.

CHAPTER 3

AN EVOLUTIONARY ALGORITHM FOR MULTIPLE CRITERIA PROBLEMS

In this chapter, we present Favorable Weight-based Evolutionary Algorithm (FWEA) for multiple criteria problems. FWEA tries to both approximate the Pareto frontier (first goal) and evenly distribute the solutions over the frontier (second goal). These two goals are common for most MOEAs literature. The fitness function of FWEA allows each member to obtain advantage over the other members by the help of a favorable weight mechanism and each solution contributes the convergence to the Pareto frontier along its own favorite direction. Thus, multiple efficient solutions are obtained by maintaining different favorable weight vectors in the population. The algorithm also encourages even distribution over the frontier by including the nearest neighbor information to the fitness function. The other key issue is to obtain a well-spread frontier. Here, well-spread means that the solutions must cover the wide range of the objective functions. Finding close to best solutions in each criterion helps achieving this. For this purpose, a single criterion evolutionary algorithm (Koçkesen, 2004) is briefly run and the best resulting solution in each criterion is incorporated into the initial population.

FWEA can easily be adapted to approximate the Pareto frontier of different problems. These problems may arise in the field of MOCO such as MOKP, multi-objective location problems etc., or continuous test problems simulating the difficulties that an EA has to deal with while approximating the Pareto frontier.

In this chapter, since we experiment on minimization type continuous test problems, the problem environment can be defined as stated in equation (3.1): Let

x_j, j=1,2,…,n be the j^{th} decision variable, X be the set of feasible solutions and $f_k(x)$, k = 1,2, … ,m be the value of k^{th} objective for solution $x = (x_1, x_2,\ldots, x_n) \in X$.

$$\begin{aligned} &\text{"Min"}\{f_1(x), f_2(x),..., f_m(x)\} \\ &\text{subject to} \quad x \in X \end{aligned} \tag{3.1}$$

where we have m (≥ 2) objective functions, n decision variables and $f_k(x)$ is a function $\Re^n \longrightarrow \Re$.

The sections of this chapter are organized as follows: In Section 3.1, we present the details of FWEA and highlight the significance and differences of it. In Section 3.2, we give the results of experiments on continuous test problems and comparisons with NSGAII. We discuss our findings and future research directions in Section 3.3.

3.1 A NEW APPROACH

A critical and difficult aspect of MOEAs is finding a suitable fitness function. The fitness value of each member should indicate the capability of various features of this member to survive. Each member wants to choose such weights that provide itself an advantage over other solutions. While assigning the fitness, the members that are located at the currently underrepresented portions of the efficient frontier should also be favored. Such a favoring scheme distributes the population over the frontier well. Köksalan and Phelps (2006) introduce an evolutionary metaheuristic, called EMAPS, for approximating preference-non-dominated solutions. EMAPS combines these two goals of MOEAs into a single fitness function by using a linear weighting function. Let X be the set of all solutions in the population, X' be a representative set of solutions, f_k be the k^{th} objective and W be the set of feasible positive weight vectors. They solve the following LP to find favorable weights that maximize the fitness of the considered member $\hat{x}$:

LP:

$$\text{Max} \quad \frac{\sum_{x^i \in X'} \alpha.\Delta(x^i)}{|X'|} + (1-\alpha).\Delta_{\min}$$

subject to

$$\sum_{k=1}^{m} \hat{w}_k (f_k(\hat{x}) - f_k(x^i)) + \Delta_{\min} = 0 \qquad \forall \ x^i \in X' \tag{3.2}$$

$$\Delta_{\min} - \Delta(x^i) \le 0 \qquad \forall \ x^i \in X' \tag{3.3}$$

$$\sum_{k=1}^{m} \hat{w}_k = 1 \tag{3.4}$$

$$\hat{w} \in W \tag{3.5}$$

$$\Delta_{\min} \text{ and } \Delta(x^i) \text{ unrestricted in sign} \tag{3.6}$$

where $\Delta(x^i)$ represents the strength of $\hat{x}$ relative to $x^i \in X'$ using a linear function, *m* is the number of objectives and $\hat{w}$ is the favorable weight vector of $\hat{x}$ that should be decided. In the objective function, they take the average of strengths:

$$\overline{\Delta} = \frac{\sum_{x^i \in X'} \Delta(x^i)}{|X'|}$$

They also use the worst case measure $\Delta_{\min}$ to favor the $\hat{x}$'s that fall in an underrepresented portion of the efficient frontier:

$$\Delta_{\min} = \min_{x^i \in X'} \{\Delta(x^i)\}$$

Thus, the fitness score of $\hat{x}$ is a convex combination of $\Delta_{\min}$ and $\overline{\Delta}$: $fitness(\hat{x}) = \alpha.\overline{\Delta} + (1-\alpha).\Delta_{\min}$, which is equal to objective function of LP $\frac{\sum_{x^i \in X'} \alpha.\Delta(x^i)}{|X'|} + (1-\alpha).\Delta_{\min}$

The fitness score of $\hat{x}$ is calculated considering the existing population at the time of $\hat{x}$'s entry. However, fitness scores keep changing as the population evolves.

Therefore, the fitness values of all members are updated upon every change in the population.

The fitness function of EMAPS is designed to reach the goals of MOEAs without requiring the use of ranking, fitness sharing, or other traditional approaches. In the design of FWEA, we construct a similar fitness function structure with EMAPS. By the help of favorable weight assignment, a solution chooses the best search direction that suits its position. EMAPS is also designed for approximating the preferred regions of the MOCO problems. Therefore, adapting EMAPS for continuous test problems needs some changes both in main algorithm and in the seeding mechanism. It also needs calibration for input parameter settings. However, we modify it to approximate the multi-objective knapsack problem in chapter 5 and give the comparison results. Although the differences of FWEA from EMAPS are emphasized in each subsection, some of them can be summarized as follows: in FWEA, to compute the strengths ($\Delta(x^i)$), we consider Tchebycheff distance function rather than linear distance function. This is a more natural way of capturing unsupported efficient solutions. The fitness scores in FWEA are adjusted by the help of non-dominated sorting concept of Goldberg (1989). Crowding measure, which is presented in Section 3.1.6, helps to achieve the diversity purpose as different from EMAPS. The main aspects of FWEA are discussed below.

3.1.1 Fitness Function

The strengths constitute important part of our fitness mechanism. They represent a solution's capability to survive. In the proposed algorithm, we measure the strength $\Delta(x^i)$ of a member $\hat{x} \in X$ relative to each $x^i \in X$, i.e. the distance between x^i and $\hat{x}$, by using weighted Tchebycheff distance function as follows:

$\Delta(x^i) = \phi(\hat{w}, x^i) - \phi(\hat{w}, \hat{x})$,

where

$\phi(w, x) = \max_{k=1,\ldots,m} \{w_k (f_k(x) - z_k^*)\}$ and z_k^* is an ideal point in objective k.

Then, the raw fitness value is calculated as: $rawfitness(\hat{x}) = \alpha.\bar{\Delta} + (1-\alpha).\Delta_{\min}$

where $\bar{\Delta} = \frac{\sum_{x^i \in X} \Delta(x^i)}{|X|}$ is the average relative strength, $\Delta_{\min} = \min_{x^i \in X}\{\Delta(x^i)\}$ is the worst case measure and α is a weighting constant between 0 and 1.

Raw-fitness value of an individual not only indicates its average strength over the other members, but also incorporates a worst case measure. The worst case measure penalizes those solutions having similar neighbors and favors those solutions in the underrepresented portions of the solution space, hence helping to diversify. In other words, it is expected that an individual far from other members will have large average strength which is additionally rewarded with an amount of $\Delta_{\min}$ that indicates the distance with its closest contender. A negative $\Delta_{\min}$ value, on the other hand, shows that $\hat{x}$ is outperformed by some $x^i \in X$ by its own favorable weights, $\hat{w}$.

In the computation of raw-fitness values, we use the Tchebycheff distance function because of its property that for each efficient solution (supported and unsupported), there exists a unique set of weights that makes that efficient solution at minimum distance from an ideal point (see Steuer, 1986 pp.425). In EMAPS, on the other hand, different α values are used to capture unsupported efficient solutions. We assign these weights as favorable weights to each solution. The main idea behind assigning favorable weights is to make each member of the population superior over the other members of the population as much as possible within its chosen direction. Given a solution, finding its most favorable weights for a weighted Tchebycheff distance function has a closed form solution (see Steuer, 1986 pp.425):

$$w_k = \begin{cases} \dfrac{1}{(z_k - z_k^*)}\left[\displaystyle\sum_{j=1}^{m} \frac{1}{(z_j - z_j^*)}\right]^{-1} & \text{if} \quad z_j \neq z_j^* \quad \text{for all} \quad j \\ 1 & \text{if} \quad z_k = z_k^* \\ 0 & \text{if} \quad z_k \neq z_k^* \quad \textit{but} \quad \exists j \ni z_j = z_j^* \end{cases} \tag{3.7}$$

where z_k is the k^{th} objective function value of the solution and z_k^* is an ideal point for objective function k.

In our algorithm, we assign the favorable weights of each individual according to (3.7). With regard to favorable weight assignment, we have the shortest Tchebycheff distance between z^* and z. Each solution desires to be close to ideal point and this is fulfilled by the favorable weights. Here, determining the components of the ideal point may be difficult. It requires solving each criterion optimally. However, finding the exact ideal point is not critical. A lower bound for the ideal point would be sufficient. For practical problems natural bounds may exist for criteria and these may be used directly. Figure 3.1 shows the contours according to Tchebycheff distance function, diagonal direction and strengths for a solution $\hat{x}$.

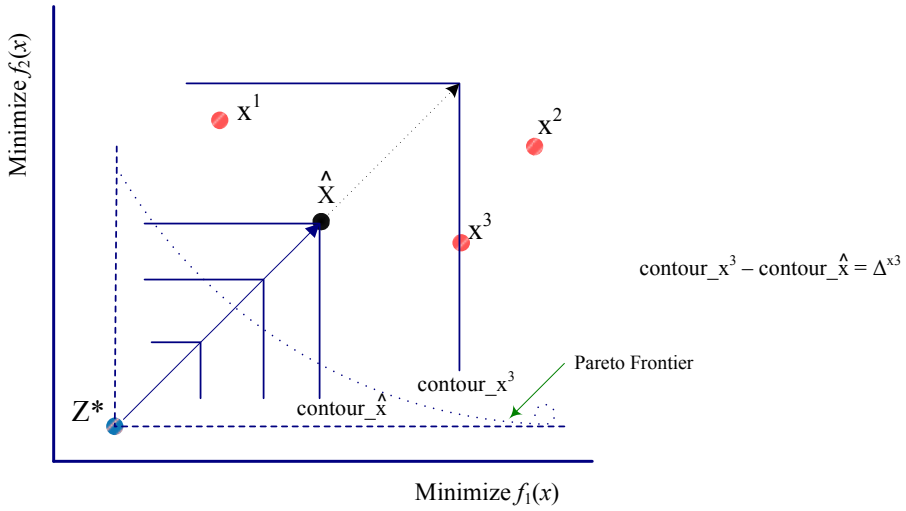

Figure 3.1 Contours and diagonal direction for $\hat{x}$

The non-dominated sorting based fitness assignment scheme proposed by Goldberg (1989) has been applied by some known MOEAs, such as NSGA I-II (Srinivas and Deb, 1995; Deb et al. 2002). According to this, non-dominated members of a population constitute the first frontier. After removing these members temporarily, remaining non-dominated solutions constitute the second frontier, and so on. We adapt this traditional fitness assignment scheme to adjust the raw-fitness values and obtain the final fitness scores as follows:

Step 1. Rank the population (Determine the frontiers)

Step 2. Let the fitness value of an individual on the first frontier be equal to its raw-fitness value. Starting with the individuals on the second frontier, adjust the raw-fitness scores so that they are less than the minimum fitness score of the previously ranked members by at least an amount of ε.

Let $X^{front-j}$ be the set of solutions on the j^{th} frontier. Consider $\hat{x} \in X^{front-j}$, $j = 2, \ldots, N$.

$$fitness(\hat{x}) = \min_{x \in X^{front-j-1}} \{fitness(x)\} - \max_{x \in X^{front-j}} \{rawfitness(x)\} + rawfitness(\hat{x}) - \varepsilon$$

where ε is a small positive constant.

This scheme guarantees the fitness values of all solutions on a better frontier to be at least an ε amount better than that of all solutions on a worse frontier.

3.1.2 Scaling

In many problems, the range of different criteria could be different. In this case, a scaling procedure must be used in order to eliminate the effects of different ranges in the evolutionary algorithms. Otherwise, the algorithm may be biased over the criteria having large ranges. In the literature, different scaling techniques exist. For instance, Steuer (1986 pp.313-314) proposes range equalization weight technique. Chang et al. (2003) propose another technique. They use the following sigmoid function (2.4.1) to equalize the range of criteria:

$$\Psi(y) = \left(\frac{1}{1 + e^{-\frac{y}{\lambda}}} - C \right).G \qquad (3.8)$$

where different C and G values lead to different shapes of the curve and λ controls the slope of the sigmoid function. y is the value that needs to be scaled and $\psi(y) \in [0,1]$ is the scaling range. In our study, we use a similar idea. We treat the values between the ideal point and the worst possible point among the efficient solutions (i.e. the nadir point) differently from the values beyond the nadir point. As can be seen in Figure 3.2, we use linear scaling for each criterion k up to the nadir point value (np_k) of that criterion. We use a sigmoid function to scale values beyond the nadir point. Linear scaling within the efficient range of each criterion maintains equal importance to those values. Beyond np_k, we squeeze the values into a small interval using a sigmoid function since these values are less important.

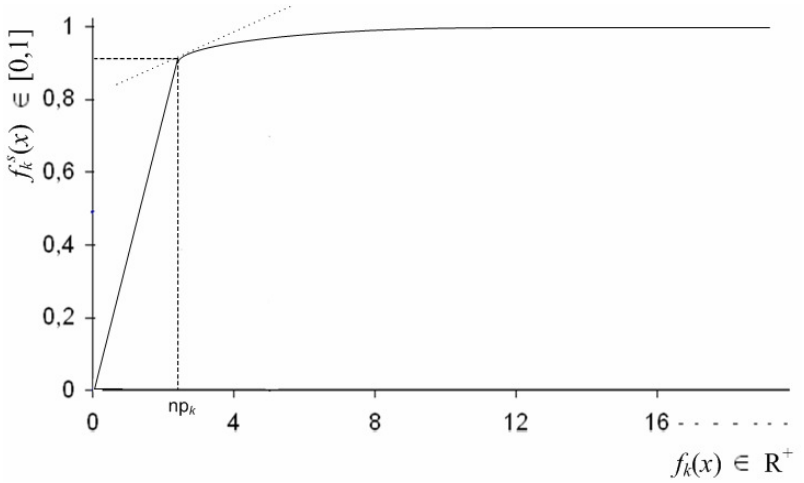

Figure 3.2 Linear and sigmoid function scaling

In order to allocate a large range for efficient values and a narrow range for inefficient values, we control the sigmoid function's parameters such that it has a very small slope (0.01) at np_k. Since it is difficult to find the exact nadir point and its exact value is not critical for our approach, we estimate it by using the payoff table. The payoff table is estimated by employing the single objective EA of Koçkesen (2004) for each objective separately and the maximum values of each objective are used to estimate nadir point. The main idea of this EA will be briefly explained in Section 3.1.6.

The steps of our scaling strategy are given below. Here, we consider a minimization problem. For a maximization problem, $f_k(x)$ values must be multiplied by "-1" before scaling. The following procedure is applied to all population members. Note that the nadir point is estimated at the beginning of the algorithm and the equation in Step2 can easily be handled by mathematical solvers or a trial and error method, the computational time of the following approach can be ignorable.

Step 1. Shift all $f_k(x)$ values to make the k^{th} component of ideal criterion vector "0".

Step 2. Set $C = \frac{1}{2}$ and G=2 (see Chang et al., 2003). Define a small slope value at point np_k and according to this slope determine λ as follows:

$$\left.\frac{d}{dy}\Psi(y)\right|_{y=np_k} = \frac{e^{-\frac{np_k}{\lambda}}}{\lambda.(1+e^{-\frac{np_k}{\lambda}})^2}.G = slope_{np_k}$$

Since there is no closed form solution to this equation, an approximate value for λ can be obtained by trial and error or mathematical solvers.

Step 3. Let $f_k^s(x)$ denote the scaled $f_k(x)$ values. If $f_k(x)$ values are before the np_k, then the values are scaled according to a linear function. Otherwise, they are scaled according to sigmoid function.

$$f_k^s(x) = \begin{cases} \frac{\Psi(np_k)}{np_k} f_k(x) & if \quad f_k(x) \le np_k \quad (linear \quad scaling) \\ \left(\frac{1}{1+e^{-\frac{f_k(x)}{\lambda}}} - C\right).G & otherwise \quad (sigmoid \quad function \quad scaling) \end{cases}$$

3.1.3 Insertion and Replacement Rules

Köksalan and Phelps (2006) define an upper limit on the cardinality of the population. Every member not being "stillborn" or "duplicate" could be introduced into the population if there exists an empty place. We use the same strategy in our algorithm. They refer to an offspring as "stillborn", if it has a fitness score worse than that of the worst member in the population. We refer to an offspring as "stillborn" if it is dominated by any member of the worst frontier. We also deny duplicate solutions in the decision variable space.

In this study, steady-state replacement strategy is used after the cardinality of the population reaches the upper limit. In every generation, two offspring are generated and if some replacement rules are satisfied, they replace two members of the current population.

Köksalan and Phelps (2006) state some replacement rules when the cardinality of the population reaches the upper limit. When the offspring (x^{off}) enters the population, it is compared to each of the existing members $x^i \in X$ with respect to their own favorable weights. If any x^i performs worse under its own favorable weights (w^i) than x^{off} , i.e. x^i is dominated by x^{off} and

$D(x^i) = \phi(w^i, x^{off}) - \phi(w^i, x^i) < 0$,

then $x^i \in X$ is a candidate for removal from the population. Among these candidates $x^i \in X$ with the most negative $D(x^i)$ is removed from the population. The same strategy is also applied in our algorithm. Such a replacement decision encourages replacing the similar individuals with better-performing ones. Therefore, this is consistent with both goals of MOEAs.

In the design of our algorithm, we include a crowding mechanism. If all $D(x^i) > 0$ (this situation occurs when x^{off} does not dominate any of the solutions) and the population size exceeds a preset upper limit, then elimination is performed by means of a crowding measure. The individual with the highest crowding measure is removed from the population. Our crowding measure is described below.

3.1.4 Crowding Measure

Crowding measure denotes the proximity of a solution to its neighbors. Deb et al. (2002) present a crowding measure. They construct a cuboid around each member (This operation requires the sorting of members at each objective separately) to find the nearest neighbors of a solution at both side of each objective and then average side length of the cuboid is taken as crowding measure. In our approach

we also try to find the nearest neighbor of a solution but in a different way from Deb et al. (2002).

Here, we apply the crowding operator to the worst (last) frontier's members (X^{last}) only. As time passes, most of the population members tend to belong to the first frontier which then is also the last frontier. Therefore, the crowding operator also helps to achieve diversity in the first frontier. For this purpose, each member (in the last frontier) determines its own nearest neighbor with its own favorable weights. Note that, to determine the nearest neighbor of a member we also search the previous frontier. That is, for each member $x^i \in X^{last}$, we first identify the "nearest neighbor" based on the Tcebycheff distance using the favorable weights of x^i, among $x^j \in X^{last}$ or X^{last-1}. That is, we calculate the weighted Tchebycheff distance of x^i to x^j with the weights w^i as follows:

$$\Delta(x^j) = \max_{k=1,\ldots,m}\{w_k^i(f_k(x^j) - z_k^*)\} - \max_{k=1,\ldots,m}\{w_k^i(f_k(x^i) - z_k^*)\}$$

Then x^j having the smallest $\Delta(x^j)$ value is marked as the nearest neighbor of x^i.

The crowding measure is computed between each member and its nearest neighbor. Here, we use the rectilinear distance. When we use the crowding measure as a basis for removing solutions from the population, we remove the member having the lowest crowding measure. If there is just a single element on the last frontier, we remove it. The steps of the crowding-based replacement procedure are provided below.

Step 1. Compute the rectilinear distance between $x^i \in X^{last}$ and its nearest neighbor x^j as follows:

$$crowdence\_measure(x^i) = \sum_{k=1}^{m} \left| f_k(x^i) - f_k(x^j) \right|$$

Step 2. Find the member x^i having the lowest crowding measure and check the fitness score of x^i and x^j. Replace the offspring with x^i or x^j, whichever has the lower fitness score.

3.1.5 Ranking and Fitness Updates

In our algorithm, the raw fitness value and rank of a solution are determined relative to the population at the time of x^{off}'s entry. After that, as a new member is introduced into the population, fitness values and ranks of the existing members must be updated considering this new member. Note that, x^{off}'s entry may cause the removal of an existing member x^{i}. Therefore, current fitness value of each member is updated by considering both entering and leaving solutions. For these updates, we must first determine the frontier of the new member x^{off}. This could be done by checking the dominance relation of x^{off} with the members of the frontiers. New solution x^{off} is assigned to the first frontier of which none of the members dominates x^{off}. However, if x^{off} dominates some members of the assigned frontier, all dominated members now belong to the next frontier. In such a case, non-dominated sorting operation is needed. After updating the rank of the members, raw-fitness scores are adjusted as given in section 3.1.1.

3.1.6 Generating the Initial Population

We generate most of the initial population members randomly. However, several solutions are seeded into the population. These solutions can be obtained by heuristic approaches or they could be the best solutions already known. For this purpose, in this study, we consider two types of seeding,

1. *Heuristic seeding*: These seed solutions are obtained by running the single objective EA of Koçkesen (2004) for a short period (that is, 500 generations and just one solution is created at each generation) for each criterion. For the special parameters, we use the values suggested by Koçkesen. Then these solutions are seeded into the population. The main features of the algorithm are briefly summarized as follows: He uses real valued representation (see Deb and Agrawal, 1995). Each decision variable is represented by a gene. Note that in our algorithm for continuous test problems, we also use real valued representation. He constructs

an offspring generation interval around each pair of genes, *j*, coming from the parent's chromosomes. The range of this interval depends on the parent range for this gene, *j*, and range in the population for the same gene. Then, genes of the offspring are generated from this interval. In his algorithm, Koçkesen does not use any mutation operator. Since we observe some improvement for our continuous test problems, we incorporate the polynomial mutation operator of Deb & Agrawal (1995). In this case, we estimate the ideal and nadir point as mentioned in section 3.1.2.

2. *Perfect seeding*: To demonstrate the performance of our algorithm with best seed solutions, we include the optimal solutions of each criterion into the population. In this case, we know the exact ideal points and estimated nadir points.

3.1.7 The FWEA Algorithm

Steps of the algorithm are presented below. The genetic operators and common parameter values we use are compatible with those suggested for NSGAII.

Parameters:

- Pop.Size1: size of the initial population,
- Pop.Size2: upper limit on the size of the population,
- *N*: size of the current population
- p_c: crossover probability, p_{mut}: mutation probability
- N_c: distribution index for SBX crossover (Deb and Agrawal, 1995), N_{mut}: distribution index for polynomial mutation (Deb and Agrawal, 1995)
- # gen.: maximum number of generations
- α - controls the balance between average and minimum strength in raw-fitness function

Step 0. Initialization

Generate an initial population and seed it with several members by using the approaches proposed in section 3.1.6.

Step 1. Evaluation of the initial population

1.1 Compute the objective values

1.2 Estimate the ideal and nadir criterion vectors (see section 3.1.2).

1.3 Scale $f_k(x)$ values for $k = 1,2,...,m$ and $\forall x \in X$ (see section 3.1.2).

1.4 Compute the favorable weights as presented in section 3.1.1.

1.5 Determine the frontiers. Compute the fitness scores by adjusting raw-fitnesses according to frontiers (see section 3.1.1).

Step 2. Selection

Select two parents according to a selection operator. In this study, binary tournament selection with replacement (i.e. a parent can be chosen again) is used.

Step 3. Crossover

Apply crossover with the crossover probability p_c to generate two offspring. In this study, SBX crossover (Deb and Agrawal, 1995) operator is applied.

Step 4. Mutation

Apply mutation with the mutation probability p_{mut}. In this study, polynomial mutation (Deb and Agrawal, 1995) is applied.

Step 5. Duplication check (see section 3.1.3)

If any offspring is a duplicate of any existing population member (in decision variable space), then discard that offspring. If both of them are duplicate, then discard both and goto Step 10.

Step 6. Evaluation of the offspring

Repeat Step 1.1-1.4.

Step 7. Stillborn check (see section 3.1.3)

If any offspring is dominated by any member of the worst frontier of current population (offspring is stillborn), then discard that offspring. If both of them are stillborn, then discard both and goto Step 10.

Step 8. Insertion and Replacement (see section 3.1.3 and 3.1.4).

8.1 Increment population cardinality and insert the offspring one by

one into the population.

8.2 If the offspring does not outperform any population member x^i with its own weights w^i (i.e. $D(x^i) > 0$ for all *i*), remove the member having lowest crowding measure if the population cardinality exceeds a preset upper limit.

8.3 If the offspring outperforms any existing population members at their own favorable weights (i.e. $D(x^i) < 0$ for some *i*)

8.3.1 Remove the weakest such member (minimum of $D(x^i)$) unless that member is on the first frontier

8.3.2 Remove the member with lowest crowding measure, if the weakest such member is on the first frontier.

Step 9. Update ranks and raw-fitnesses. Adjust raw fitness scores. (See section 3.1.5)

Step 10. Termination

If a stopping condition is not reached, goto Step 2.

The complexity of the above algorithm for a single generation is presented below.

- Step 1.5 – Determination of frontiers for initial population has $O(mN^2)$ complexity.
- Step 5 – Duplication checking has $O(mN)$ complexity
- Step 8.2 – Computation of crowding measure has $O(N^2)$ complexity
- Step 9 – Updating ranks has $O(mN^2)$ complexity

Therefore, Step 1.5 and Step 9 determine the overall complexity of the algorithm as $O(mN^2)$. The complexity of FWEA is equal to NSGAII for a single generation. However, FWEA needs larger number of generations and higher run time due to steady state replacement strategy.

3.2 COMPUTATIONAL RESULTS FOR CONTINUOUS OPTIMIZATION PROBLEMS

MOEA literature has been growing very fast. Many new EAs and test problems are continuously introduced. In this chapter, we experiment on some well-known test problems (ZDT and DTLZ) and we compare the performance of FWEA with one of the leading benchmark algorithms, NSGAII (Deb et al., 2002). The test problems used have some properties, such as multimodality, non-convexity, non-uniformity etc. that may cause difficulties in achieving the proximity and diversity goals of MOEAs (Deb, 1999). We carry out experiments on 2-objective (Deb, 1999), 3 and 5-objective (Deb et al., 2001) test problems given in Table 3.1.

Table 3. 1 Continuous test problems.

Test Problem	Decision Variables
ZDT1 Convex Pareto frontier	$x_i \in [0,1]$ for $i = 1,...,30$
ZDT2 Non-convex Pareto frontier	$x_i \in [0,1]$ for $i = 1,...,30$
ZDT3 Convex, Disconnected Pareto frontier	$x_i \in [0,1]$ for $i = 1,...,30$
ZDT4 Convex, Multimodal Pareto frontier	$x_1 \in [0,1]$ and $x_i \in [-5,5]$ for $i = 2,...,10$
ZDT6 Non-convex, Non-uniformly spaced Pareto frontier	$x_i \in [0,1]$ for $i = 1,...,10$
DTLZ1 3-objective Linear Pareto frontier 5-objective Linear Pareto frontier	$x_i \in [0,1]$ for $i = 1,...,7$ $x_i \in [0,1]$ for $i = 1,...,9$
DTLZ2 3-objective Spherical Pareto frontier 5-objective Spherical Pareto frontier	$x_i \in [0,1]$ for $i = 1,...,12$ $x_i \in [0,1]$ for $i = 1,...,14$

In this chapter, the representation scheme of both algorithms is real coding (i.e. each gene represents the actual value of a decision variable x_i, i=1,2,…,n) and the chromosome is a vector of decision variables. In FWEA, size of the initial

population (Pop.Size1) is set to 80 for 2 and 3-objective test problems and it is set to 200 for 5-objective test problems. We choose the upper limit on the cardinality of the population (Pop.Size2) equal to the population size of NSGA II, which is 100 for 2 and 3-objective test problems and 240 for 5-objective test problems. Crossover probability (p_c) is equal to 0.9 for 2-objective test problems and 1.0 for 3 and 5-objective test problems in both algorithms. Mutation probability (p_{mut}), N_c and N_{mut} parameters are taken as $\frac{1}{n}$, 20 and 20 respectively in both algorithms. We carry out 10 replications for 2 and 3-objective test problems and 5 replications for 5-objective test problems. Different from NSGA II, several seed solutions obtained by approaches in section 3.1.6 are introduced into the initial population of our algorithm. Our remaining members of initial population are same as NSGA II. Since we apply steady state replacement strategy, replacing two members at each generation, the stopping condition is set to 12,500 generations (i.e.maximum 25000 crossovers) that corresponds to 250 and 100 generations of NSGA II for 2, 3-objective test problems and 5-objective test problems, respectively. Note that, for the heuristic seeding case, to be able to make a fair comparison, we reduce the number of generations consumed in the initialization stage from the main algorithm. In the perfect seeding case, we assume that these seed solutions are already available.

We made preliminary experiments to decide on the value of α (controls the balance between average and minimum strength in raw-fitness function). Figures A.1 to A.9 in Appendix A illustrate the box plots of FWEA for *reverse proximity indicator* and different α values. Smaller α values performed better and we decided to use $\alpha = 0.001$ in the rest of the experiments. The reason for observing such a low α level is probably because every member is sufficiently better than others with own favorable weights especially towards the termination that it emphasizes the minimum strength portion of the raw-fitness score to take an advantage.

When applying Koçkesen's algorithm, in addition to the special parameter values suggested by him, we use p_c, p_{mut}, N_c and N_{mut} values mentioned above.

3.2.1 Performance Measures

In order to test our algorithm, we use two performance measures. Our first measure is *reverse proximity indicator* (Bosman and Thierens, 2003). In this measure, the Euclidean distance from each solution on efficient (Pareto) frontier (X^{pareto}) to the nearest solution found by algorithm is computed. This indicator measures closeness to the Pareto frontier and diversity at the same time. In order to represent the pareto frontier well, we filter 500, 1000 and 2500 solutions from 10,000 randomly chosen efficient solutions for 2, 3 and 5-objective test problems respectively. For filtering, we use "method of first point outside the neighborhoods" (Steuer 1986 pp.314-318). Reverse proximity indicator is given by equation (3.9)

$$D_{X^{Pareto} \to X^{Algorithm}} = \frac{1}{\left|X^{Pareto}\right|} \sum_{z^1 \in X^{Pareto}} \min_{z^2 \in X^{Algorithm}} \left\| z^1 - z^2 \right\|_2 \qquad (3.9)$$

where $|X|$ denotes the cardinality of set X and $\left\| z^1 - z^2 \right\|_2$ denotes the Euclidean distance between z^1 and z^2.

Our second measure is the "C measure" by Zitzler and Thiele (1999). This measure compares two sets of solutions and calculates the proportion of solutions in the second set for which there are solutions at least as good in every objective in the first set, i.e. proportion of solutions in Set 2 that are weakly dominated by some solutions in Set 1. Let S_1 and S_2 represent the set of solutions in Sets 1 and 2 respectively. Then the measure $C(S_1, S_2)$ is computed as

$$C(S_1, S_2) = \frac{\left|\{z^2 \in S_2 : \exists \;\; z^1 \in S_1 : z^2 \text{ is dominated by } z^1\}\right|}{|S_2|}$$

We report average, best and worst values of indicators across replications and we use Mann-Whitney rank-sum non-parametric statistical test to show whether the medians of the two algorithms' results are significantly different or not.

3.2.2 Simulation Results

In our experiments, we use NSGA II version 1.1 code downloaded from http://www.iitk.ac.in/kangal/codes.shtml. The performance of both algorithms can be observed visually in Figure 3.3 for a single run of test problem ZDT6, which has a non-convex and non-uniformly spaced Pareto frontier. As can be seen in the figure, FWEA heuristic seeding (FWEA_HS) is closer to efficient frontier and has a better spread than NSGA II. Some portions of the efficient frontier are not represented by NSGA II in this problem. Figure 3.4 shows the non-dominated solutions of both algorithms for single runs of test problems DTLZ1-3D and DTLZ2-3D. As seen in the figure, FWEA_HS yields a better distribution than NSGA II. Since the differences between the two algorithms are not clear visually for the other test problems, we do not provide those pictures. We evaluate them based on the reverse proximity indicator and the C-measure.

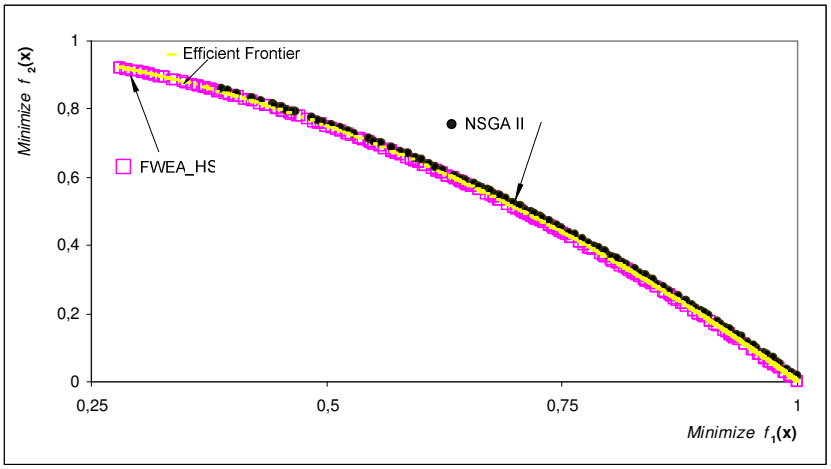

Figure 3.3 Non-dominated solutions of FWEA_HS and NSGA II for a single run of test problem ZDT6.

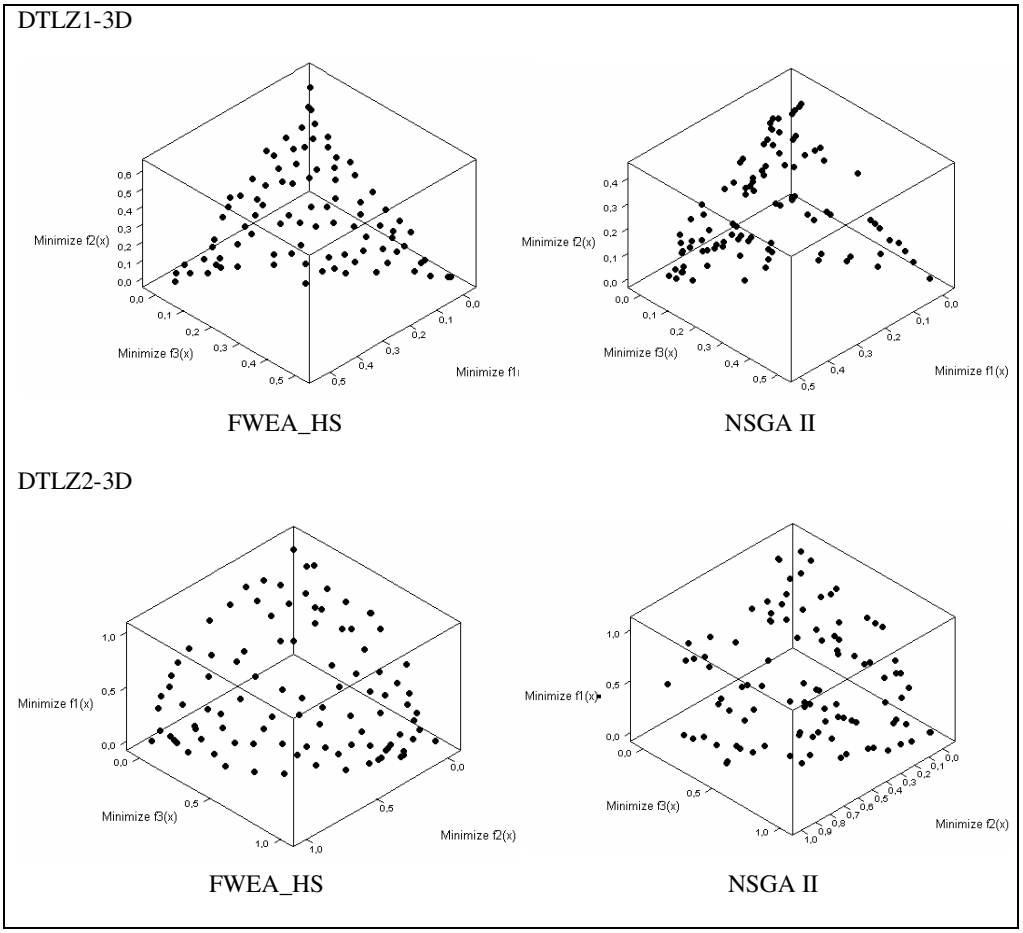

Figure 3.4 Non-dominated solutions of FWEA_HS and NSGA II for a single run of test problems DTLZ1-3D and DTLZ2-3D.

Average, best and worst values for the reverse proximity indicator and the C-measure values are summarized in Tables 3.2 and 3.3, respectively. The values marked with (*) indicate the better of FWEA_HS or NSGAII algorithms. In all but one of the test problems, FWEA_HS outperforms NSGA II with respect to both measures. Although NSGA II seems to slightly outperform FWEA_HS in problem (ZDT4) based on reverse proximity indicator, this is not statistically significant as discussed below. ZDT4 is a multi-model test problem, which includes multiple local frontiers parallel to each other and the last one is the Pareto frontier. We need to try different parameter settings and different mechanisms for FWEA_HS to have better convergence in this problem. The results for the perfect seeding case indicate that FWEA perfect seeding (FWEA_PS) outperforms NSGA II in both measures for all problem instances. In most of the problems especially in ZDT4 and DTLZ1-5D, we observe improvements in the performance of FWEA in perfect seeding case. A careful examination of performance measures for DTLZ problems also shows that as the number of criteria increases, so does the performance gap between the two algorithms in favor of FWEA. Table 3.2 also includes the significance level of Mann-Whitney non-parametric two-tailed statistical test results. In most of the problems, P value is less than 1% indicating the strong statistical evidence to reject H_0. However, in ZDT4 we cannot reject H_0. In 5D test problems, P value is somewhat higher. This is probably due to fewer replications performed.

Table 3.2 Average, best and worst values for the reverse proximity indicator.

	$D_{X^{Pareto} \to X^{FWEA\_HS}}$			$D_{X^{Pareto} \to X^{FWEA\_PS}}$			$D_{X^{Pareto} \to X^{NSGAII}}$			P-value $H_0: \eta_{FWEA\_HS} = \eta_{NSGAII}$ $H_1: \eta_{FWEA\_HS} \neq \eta_{NSGAII}$
	Avg.	**Best**	**Worst**	**Avg.**	**Best**	**Worst**	**Avg.**	**Best**	**Worst**	
ZDT 1	0.00446*	0.00419*	0.00474*	0.00460	0.00430	0.00523	0.00493	0.00459	0.00524	< 1%
ZDT 2	0.00445*	0.00431*	0.00454*	0.00438	0.00420	0.00473	0.00497	0.00472	0.00533	< 1%
ZDT 3	0.00503*	0.00475*	0.00531*	0.00508	0.00486	0.00548	0.00538	0.00506	0.00570	< 1%
ZDT 4	0.00921	0.00489	0.03246	0.00517	0.00489	0.00534	0.00711*	0.00478*	0.01389*	Fail to reject H_0
ZDT 6	0.08438*	0.08395*	0.08466*	0.06628	0.06623	0.06635	0.12894	0.12877	0.12909	< 1%
DTLZ 1-3D	0.02477*	0.02322*	0.02668*	0.02172	0.02109	0.02263	0.05957	0.02804	0.30263	< 1%
DTLZ 2-3D	0.05847*	0.05687*	0.05980*	0.05905	0.05721	0.06125	0.07156	0.06890	0.07759	< 1%
DTLZ 1-5D	5.06587*	2.77150*	10.15749*	0.05585	0.04933	0.06725	14.96275	5.71207	24.69812	< 5%
DTLZ 2-5D	0.16368*	0.16142*	0.16553*	0.16230	0.16050	0.16539	0.18921	0.18204	0.19852	< 5%

* Better of FWEA_HS or NSGAII algorithms

Table 3.3 Average, best and worst values for the C measure.

	C(FWEA_HS, NSGAII)			C(NSGAII, FWEA_HS)			C(FWEA_PS, NSGAII)			C(NSGAII, FWEA_PS)		
	Avg.	**Best**	**Worst**	**Avg.**	**Best**	**Worst**	**Avg.**	**Best**	**Worst**	**Avg.**	**Best**	**Worst**
ZDT 1	0.17*	0.26*	0.12*	0.00	0.00	0.00	0.17*	0.24*	0.10*	0.00	0.01	0.00
ZDT 2	0.21*	0.30*	0.10*	0.00	0.00	0.00	0.21*	0.31*	0.11*	0.00	0.00	0.00
ZDT 3	0.20*	0.26*	0.09*	0.00	0.02	0.00	0.19*	0.25*	0.11*	0.00	0.01	0.00
ZDT 4	0.17	0.62*	0.00	0.17	0.46	0.00	0.48*	0.78*	0.14*	0.00	0.00	0.00
ZDT 6	0.89*	0.96*	0.80*	0.00	0.00	0.00	1.00*	1.00*	1.00*	0.00	0.00	0.00
DTLZ 1-3D	0.20*	0.99*	0.02*	0.04	0.16	0.00	0.24*	0.99*	0.02*	0.00	0.02	0.00
DTLZ 2-3D	0.04*	0.06*	0.01*	0.01	0.02	0.00	0.04*	0.06*	0.01*	0.01	0.03	0.00
DTLZ 1-5D	0.69*	0.93*	0.25*	0.28	0.45	0.00	0.60*	1.00*	0.35*	0.00	0.01	0.00
DTLZ 2-5D	0.08*	0.09*	0.07*	0.00	0.01	0.00	0.11*	0.15*	0.07*	0.00	0.00	0.00

*Better of the two algorithms

Figure 3.5 shows convergence of both algorithms for DTLZ1-3D and DTLZ2-3D test problems. The trajectory of the reverse proximity indicator over generations is similar for both algorithms. Both decrease sharply at the beginning and stay steady after reaching a certain level. One can observe that FWEA converges faster than NSGA II and keeps closer to the efficient frontier through generations.

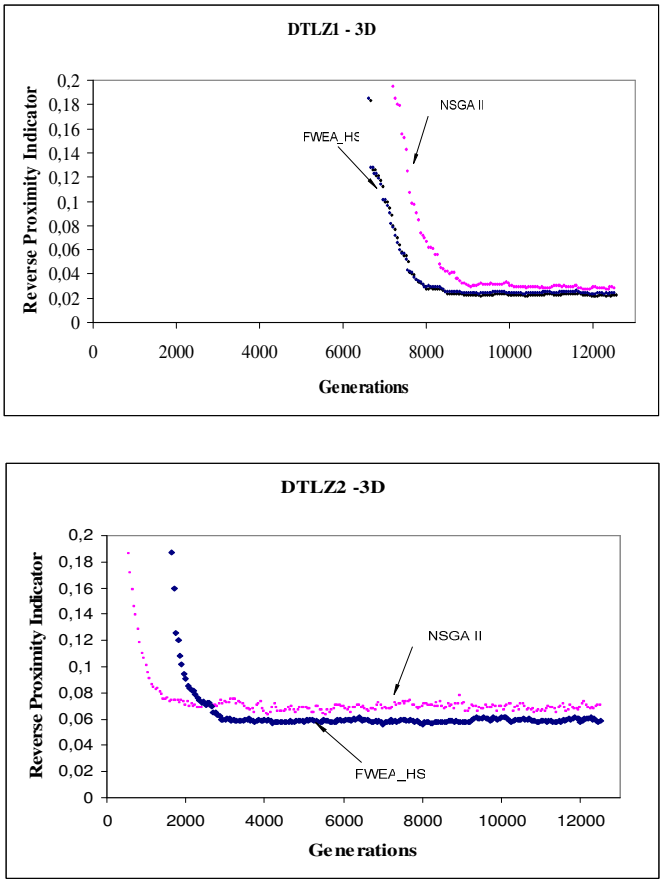

Figure 3.5 Convergence of FWEA_HS and NSGA II.

3.3 DISCUSSION

In this chapter, an evolutionary algorithm is developed for multiple criteria problems. Tchebycheff distance function is considered while computing the fitness scores. Performance of the algorithm is tested on 2, 3 and 5-objective continuous test problems taken from the literature. These test problems have special complexities that a MOEA has to deal with. We compare our algorithm with NSGA II, which is one of the well-known benchmarks in the literature. According to experiments, the results of FWEA are promising in heuristic seeding and perfect seeding cases. However, there is room for improvement especially in multimodal problems and higher number of objectives. Although the worst-case complexities of both algorithms are equal to each other as stated in section 3.1.7, the run time of FWEA is approximately 30 times higher than NSGAII. This difference may be the result of steady state replacement strategy, which needs to call the most time consuming step of the FWEA (Step 9) 50 times more than NSGA II for the same number of function evaluations.

An area for future research is to adapt our approach to multi-objective combinatorial optimization (MOCO) problems. Those problems are computationally hard and warrant the use of heuristic procedures. However, the efficient frontiers are not as complex as the test problems designed for continuous problems. There are problem specific issues that need to be addressed. In Chapter 4 and 5, we provide the application of FWEA on two of these MOCO problems.

CHAPTER 4

BICRITERIA p-HUB LOCATION PROBLEMS

In this chapter, we present two bicriteria uncapacitated multiple allocation p-hub location problem (Bp-HLP1 and Bp-HLP2). In the first problem, our first objective is to minimize the total transportation costs of uncapacitated multiple allocation p-hub median problem (UMAp-HMP) with a positive interhub transfer cost ($\alpha > 0$). Our second objective is to minimize the total traveling costs between hub points and origin-destination points. In this case, interhub transfer cost is ignorable ($\alpha=0$) and the problem turns into a well-known facility location problem, i.e. the p-median problem. In the second problem, we address the delays occurring while being served at each hub. We consider the trade off between our first objective and a new objective function, which is to minimize the maximum delay at each hub. We propose a bicriteria evolutionary algorithm to approximate the efficient frontiers of these problems. We test the performance of our algorithm on Turkish Postal System (PTT) data sets (TPDS), AP (Australian Post) and CAB (US Civil Aeronautics Board) data sets.

4.1 INTRODUCTION

In today's competitive environment, cost and time-efficient serving strategies between every origin and its destination provide the firms a competitive advantage. However, in a distribution network, direct shipments between every origin-destination pair are not cheap or practical. The cost efficiency of these networks can be improved by inserting some hub points (switching/sorting centers) onto the link connecting origins and destinations. Reducing direct links and concentrating overall flow between fully interconnected hub points give rise to economies of scale. Figure 4.1 provides a comparison between two network structures: without hubs and with hubs.

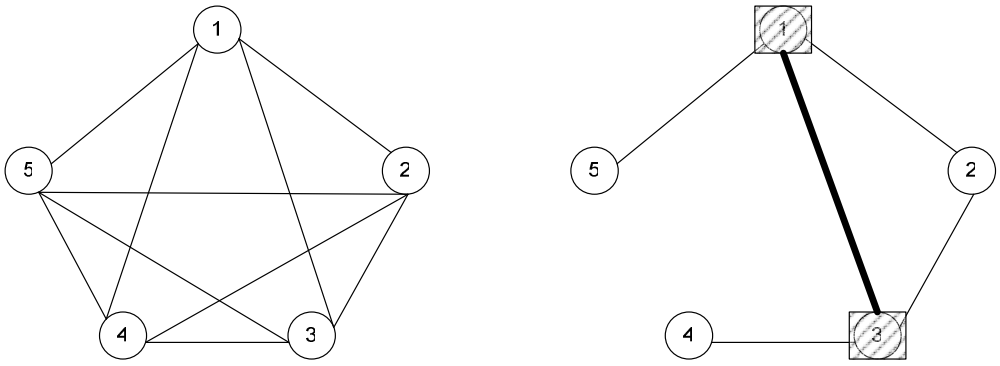

Figure 4.1 Hub network structure

Figure 4.1 (a) illustrates a network structure without hubs. Here, nodes 1-5 are fully connected. In such a network, if you have *n* origin/destination points, you will need $C_2^n = \frac{n!}{2!(n-2)!}$ direct connections. However, it is possible to provide service by decreasing the number of direct links as in Figure 4.1 (b). In this case, nodes 1 and 3 are chosen as hubs and the flow between them is intensified. All the nodes are connected to at least one or more hub points and the flow between every origin and destination is routed via these hubs. Here two main decisions are to determine the location of hubs and to allocate the non-hub nodes to these hubs.

Today, hub location problems have many applications such as postal services, passenger/cargo transportation services, telecommunication networks (Campbell, 1994; Kara, 1999). Similarly, Turkish PTT, which is a huge network covering 81 cities, needs an improvement due to high operating costs. Çetiner (2003) is the first who proposes alternative design strategies to achieve a more effective postal delivery system for Turkish PTT. Çamlar (2005) suggests a bi-criteria approach that considers p-hub median and p-hub center objectives, for the restructuring of Turkish PTT.

In this study, we consider uncapacitated multiple allocation p-hub median problem (Hereafter this problem is abbreviated as UMAp-HMP as given in Ernst and Khrishnamoorthy, 1998). Here, each city can be assigned to more than one hub and there are no capacity restrictions on hubs or links. In our first bicriteria problem Bp-HLP1, we decide to consider the two conflicting objectives: minimizing the total transportation (collection/transfer/distribution) cost of UMAp-HMP and minimizing the total cost (collection/distribution) of p-median problem ignoring interhub transfers. Collection, transfer and distribution operations in a distribution network are illustrated in Figure 4.2. While the minimization of the total transportation cost is important for a logistic firm, improving response to customer is another significant issue in the design of efficient logistic systems as stated in Nozick and Turnquist (2001). Here, the second objective implies the hubs to become close to the demand points. In our case, it needs to focus on the collection/distribution operations of mail from origins to hubs and hubs to destinations. Since the interhub transportation is generally more planned, better scheduled and less costly, reducing the distribution/collection costs provides the logistic firms with competitive advantage especially when the efficiency of interhub transportation is very high. Therefore, we consider the trade off between p-hub median and p-median objectives in our first bicriteria problem.

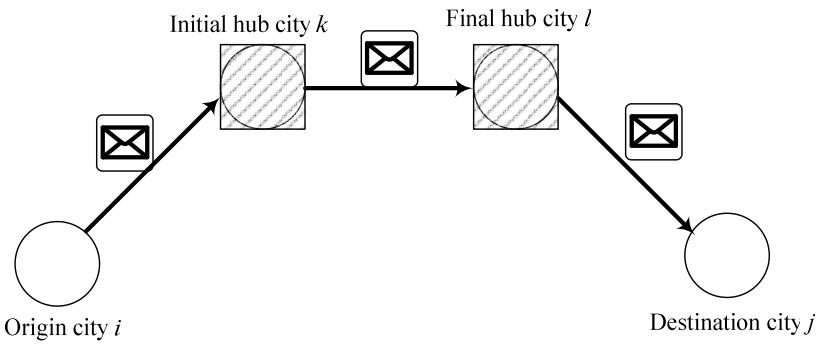

Figure 4.2 Collection, distribution and transfer operations

While designing a hub network, another important issue that needs to be addressed is the delays that occur while being served at each hub. In a network, delays in some hubs may be more than the other ones. A reasonable strategy in this situation is to shift the traffic from busiest hubs to idle ones. Therefore, in our second bicriteria problem Bp-HLP2, one of our objectives is related to the minimization of the maximum delay over all hubs. With regard to this objective, we aim to prevent the extreme delays by partitioning the traffic among the hubs. Here it should be noted that another objective might be to minimize the sum of the delays over all hubs; however, in this case delays at some hubs may be very high while others have too small delays.

Typically, delay and demand/capacity ratio are directly related. Demand/capacity ratio is a measure of congestion and as the former increases so does the latter. Increased congestion, on the other hand, results with higher delays in hubs. For example, in a hub airport, as the demand approaches the capacity, the delays of customers increase due to congestion (Janic, 2002).. Similarly, in postal systems, a hub office after reaching its daily capacity may delay and transfer the excess mail

for the next day. Therefore, we consider minimizing the maximum demand/capacity ratio objective and minimizing the total transportation (collection/transfer/distribution) cost of UMAp-HMP in Bp-HLP2. Here, the demand is regarded as the volume of traffic entering a hub via collection. Ebery et al. (2000) explain this by stating that in postal delivery once mail has been sorted after collection it does not need to be sorted again for distribution. In our model, we do not apply capacity restriction on traffic entering a hub. That means demand/capacity ratio may be greater than one. In this study, we use the capacity only to estimate the delays. If the capacities in real life can be increased, then the above arguments are not directly applicable.

Kara and Tansel (2001) consider the transient times spent at hubs due to loading, unloading and sorting operations; however, their problem structure and assumptions are very different from the structure we use.

In Section 4.2, we review the literature. In Section 4.3, we define the problem and properties of criteria space. In Sections 4.4 and 4.5, we give the details of the evolutionary algorithms for Bp-HLP1 and Bp-HLP2, respectively. We present the computational results for test problems in Section 4.6. We discuss the conclusion and direction for further research in Section 4.7.

4.2 LITERATURE REVIEW

p-hub median problems are generally categorized as capacitated versus uncapacitated or multiple versus single allocation. The capacity restriction might be on the hub itself or on the flow between nodes. Single allocation is a special case of multiple allocation. In this case, a city (node) can only be assigned to a single hub city. Due to more flexible allocation possibilities, multiple allocation is less costly than single allocation. Therefore, UMAp-HMP provides a lower bound on the single allocation problem (USAp-HMP).

O'Kelly (1987) gives the first mathematical formulation for the discrete hub location problem. His formulation is a quadratic integer programming model for the single allocation case. He also provides two heuristic approaches that are based on allocation of demand nodes to the nearest hub or the second nearest hub.

After this study, Campbell (1994) presents integer programming formulations for hub location problems. These formulations include UMAp-HMP, UMAp-HMP with flow thresholds and fixed costs on links, uncapacitated multiple allocation hub location problem (UMAHLP), p-hub center and hub covering problem. Campbell's formulation linearizes O'Kelly's formulation by using four indexed variables. Campbell (1994) also shows that for $\alpha = 0$ (no interhub transfer cost) UMAp-HMP, USAp-HMP and p-median problem are all identical.

Campbell (1994) proposes the first mathematical formulation for UMAp-HMP. Then, Skorin-Kapov et al. (1996) make some modifications on this formulation and reduce the number of constraints. Their formulations give promising results for solving the LP relaxation of this problem.

Campbell (1996) highlights the difference between USAp-HMP and UMAp-HMP, and presents two heuristics for solving USAp-HMP. An initial solution to UMAp-HMP is found by using a greedy-interchange heuristic. This solution is a starting point to find a solution for USAp-HMP. To develop a solution for USAp-HMP, all multiple allocations are replaced with single allocation according to two rules: "maxflo" and "allflo". According to the maxflo rule, an origin point i is assigned to hub $k \in H_i$, if the flow between i and k is maximum. Here note that hub k is an element of UMAp-HMP solution and H_i is the set of these k hubs. The second rule, allflo, allocates each demand point i to a hub in H_i so that the total transportation cost is minimized. Allflo considers all possible single allocation combinations.

The last efficient formulation for the UMAp-HMP is due to Ernst and Krishnamoorthy (1998). In this formulation, they use fewer decision variables and constraints. They also remove four indexed variables by treating the interhub

transfers as a multi-commodity flow problem. Each commodity represents the traffic flow originating from a particular node.

Ernst and Krishnamoorthy (1998) describe an exact solution approach for solving the multiple allocation case of the p-hub median problem and adapt this procedure to solve the single allocation case. Their procedure is based on a branch and bound algorithm and they solve shortest path problems to obtain lower bounds.

Campbell et al. (2002) review the hub location research. They discuss the basic hub location models in terms of complexity and difficulty to solve large problem instances. They address two issues. The first one is to solve larger problems to optimality and the second one is to extend these basic models to be more realistic by integrating hub location and network design decisions. In their survey study, they also categorize the hub location literature and review the applications of hub location problems.

Marin et al. (2006) review the integer linear formulations of the UMAHLP. In this problem, the model also decides on the optimal number of hubs to be opened. They also provide new and better performing formulations.

Sohn and Park (1998) consider the uncapacitated p-hub location problems, where only multiple and single allocation are considered. They assume that the hub locations are given; therefore, remaining decision is the allocation decision. They show that the allocation part of UMAp-HMP can be solved polynomially as it reduces to all-pair shortest path problem.

Aykin (1995) introduce a hub location and routing problem in which flow from origin to destination are either routed via hubs (one-hub-stop or two-hub-stop) or, if allowed, shipped directly with no hub-stop. Depending on the application, direct connections may be permitted for all, some or none of the routes served by the system.

Aykin (1994) presents the capacitated hub and spoke network design problem. In the paper, a greedy-interchange heuristic is provided. In the greedy step, the heuristic initially assumes a hub at every potential hub node and greedily removes hubs with the smallest (Lagrange based) lower bound.

In the literature, several researchers have studied the congestion and delay problem at hubs. Marianov and Serra (2003) propose models and solution methods for the hub airport location problem by modeling it as queuing systems. In a study, Chen and Choi (2001) try to reduce the web service delays by balancing the load of servers. They represent the load of a server as total access cost per http connections.

Several researchers present genetic algorithms (GAs) for the uncapacitated p-hub median problem. One of them is due to Perez et al. (1998). They discuss that the most natural representation of a solution to USAp-HMP is a two-dimensional array whose i, k^{th} element is "1" if city i is assigned to hub k, and "0" otherwise. They represent this two-dimensional array with the use of a hub array and assignment array. One point crossover operator is applied for each array separately. Mutation operator is also applied for each array separately. To mutate a hub array, a local search procedure and to mutate an assignment array, random allocation procedure is implemented. They compare this GA with the Skorin-Kapov and Skorin-Kapov's TABUHUB (1994), which is a tabu search based heuristic for hub location problem.

Topçuoğlu et al. (2005) present a GA for USAp-HMP. Similar to Perez et al. (1998), in their GA, each chromosome consists of two arrays: hub array and assign array. On the other hand, the length of these arrays is equal to the number of nodes (n) in the network. Hub array includes binaries where a "1" indicates that the node is a hub and "0" means it is not. The assign array represents the allocation of nodes to hubs. They use fitness proportional selection with roulette wheel sampling. They apply one point crossover and, if needed, repair by allocating a node to the nearest hub. They suggest two mutation operators just for allocation decisions.

They compare their GA with the tabu search algorithm by Abdinnour-Helm (1998).

Multiple criteria applications to hub location problems is a new research area. Several researchers have begun to address some multiple criteria hub location problems. One recent study is by Costa et al. (2006). They present a bicriteria single allocation hub location problem. Their first objective is to minimize total transportation cost for the capacitated single allocation hub location problem and their second objective is to minimize the time to process the flow entering the hubs. They propose an interactive decision aid method.

Çamlar (2005) considers the hub location problem of Turkish PTT for median and center objectives. He applies NSGA-II and SPEA2 algorithms to this problem. He uses a different version of Perez et al.'s (1998) chromosome representation. In his representation, the assignment array includes n positions and hub nodes are allocated to themselves. He generates the initial population randomly. Hub arrays and assignment arrays are crossovered and mutated separately. The performance of the algorithm is tested on AP and CAB data sets.

4.3 PROBLEM DEFINITION

In Turkey, an important part of transportation operations of postal items is managed by Turkish PTT. Today, Turkish PTT has central offices (principal directorates) in all 81 cities of Turkey. It has a total of 4471 central offices, branch offices and agencies (Detailed statistics are available at www.ptt.gov.tr). The central offices are equipped with required machinery for sorting operations of postal items according to districts. Postal items include ordinary letters, registered letters, registered and reply paid letters, postcards, packages and cargo. Hereafter, we refer to them as mail. In the current system, express mail service and international mail services are available as well. However, in this study we just consider intercity regular mail transportation. The mail flow within a city

(collection and distribution of mail from central offices to branches and mail boxes) is not considered.

We represent the problem on a network structure. Each city can be evaluated as (origin-destination) nodes of the network and there is mail flow between them. However, it is not practical to have a direct flow between all pairs of nodes of the network. It is reasonable to use hub nodes to achieve economies of scale within the interhub flow. So, the main decision is to define the nodes that will serve as hubs. Each city's central office can serve as a hub. Although these hub central offices may need some additional investments for personnel, equipment and machinery, these upgrading investments are approximately the same for all cities and we do not need to add extra hub opening costs into our objective function.

Turkish PTT provides the transportation of mail by mostly airplanes and trucks. We therefore, restrict the hubs to cities that have airports as stated in Çetiner (2003). In Turkey, there are 32 cities having an airport. We assume that there is no capacity restriction on hubs; however, after sending daily post, the incomings are delayed for the next day's post.

4.3.1 Mathematical Model of Bicriteria p-Hub Location Problems

In this section, we give the mathematical model of the considered bicriteria problems based on Skorin-Kapov et al.'s (1996) formulation and we use the notation by Ernst and Khrishnamoorthy (1998) partially.

Parameters

N : Set of nodes in the network, i.e. N = {1, 2, … , n}

p : Number of hubs

F_{ij} : Flow from origin i to destination j

d_{ij} : Distance between node i and node j

C^1_{ijkl} : For the p-hub median objective, cost per unit flow from origin i to destination j via hubs k and l

$$C^1_{ijkl} = \chi d_{ik} + \alpha d_{kl} + \delta d_{lj}$$

C^2_{ijkl} : For the p-median objective, cost per unit flow from origin i to destination j via hubs k and l

$$C^2_{ijkl} = \chi d_{ik} + \delta d_{lj}$$

where χ, α, δ are unit costs for collection, transfer and distribution operations respectively. Here α is generally less than χ and δ to reflect the economies of scale due to intensified traffic flow between hubs.

Decision Variables

X_{ijkl} : $\begin{cases} 1 & \text{if flow from origin } i \text{ to destination } j \text{ is routed via hubs } k \text{ and } l \\ 0 & \text{otherwise} \end{cases}$

H_k : $\begin{cases} 1 & \text{if node } k \text{ is a hub} \\ 0 & \text{otherwise} \end{cases}$

Bp-HLP1 is as follows:

$$\text{Min} \quad \sum_{i \in \mathrm{N}} \sum_{k \in \mathrm{N}} \sum_{l \in \mathrm{N}} \sum_{j \in \mathrm{N}} F_{ij} C^1_{ijkl} \mathrm{X}_{ijkl}$$

$$\text{Min} \quad \sum_{i \in \mathrm{N}} \sum_{k \in \mathrm{N}} \sum_{l \in \mathrm{N}} \sum_{j \in \mathrm{N}} F_{ij} C^2_{ijkl} \mathrm{X}_{ijkl}$$

Subject to

$$\sum_{k \in \mathrm{N}} \mathrm{H}_k = \mathrm{p} \qquad (4.1)$$

$$\sum_{k \in \mathrm{N}} \sum_{l \in \mathrm{N}} \mathrm{X}_{ijkl} = 1 \qquad \forall i, j \in \mathrm{N} \qquad (4.2)$$

$$\sum_{k \in \mathrm{N}} \mathrm{X}_{ijkl} \le \mathrm{H}_l \qquad \forall i, j, l \in \mathrm{N} \qquad (4.3)$$

$$\sum_{l \in \mathrm{N}} \mathrm{X}_{ijkl} \le \mathrm{H}_k \qquad \forall i, j, k \in \mathrm{N} \qquad (4.4)$$

$$\mathrm{H}_k \in \{0,1\} \qquad \forall k \in \mathrm{N} \qquad (4.5)$$

$$\mathrm{X}_{ijkl} \in \{0,1\} \qquad \forall i, j, k, l \in \mathrm{N} \qquad (4.6)$$

The first objective is to minimize overall transportation cost for UMAp-HMP and the second objective is to minimize the total costs (collection and distribution costs) of p-median problem. Constraint (4.1) ensures that exactly p hubs are opened. Constraint (4.2) guarantees that the flow between every origin/destination pair is routed. Constraint (4.3) and (4.4) state that a hub cannot be used unless it is available. Constraint (4.5) stipulates that H_k variables can only take values of zero or one. In Bp-HMP1, constraint (4.6) is different from Skorin-Kapov's formulation of UMAp-HMP in the sense that they do not need to restrict X_{ijkl} variables to zero or one. Considering UMAp-HMP, there is only one shortest path between each origin/destination pair. This makes the X_{ijkl} variables naturally zero or one. However, for the Bp-HMP, you can always find efficient solutions including fractional flow that implies a continuous efficient frontier.

P-median is an NP-hard problem. Optimal locations of p facilities can be found by evaluating $\mathrm{C}_{\mathrm{p}}^{n}$ possible solutions. Although small size instances may be solved in polynomial time, the solution space becomes very large for moderate size instances (Daskin, 1995). UMAp-HMP is known to be NP-hard as well. One can find the optimal solution via the complete enumeration with the complexity of $O(\mathrm{C}_{\mathrm{p}}^{n} n^2 \mathrm{p})$. The problem can be evaluated by taking into consideration location and allocation decisions, respectively. There are $\mathrm{C}_{\mathrm{p}}^{n}$ possible locations for a given p. For the allocation decision (for given hub locations), the remaining problem is all-pair shortest path problem which is totally unimodular. Because of the unimodularity property, LP relaxation of the problem gives the optimal that is integer (Wolsey, 1998). An efficient algorithm (i.e. all pair shortest path algorithm) also provides the optimal path for each origin-destination pair with the complexity of $O(n^2 \mathrm{p})$ (Ernst and Khrishnamoorthy, 1998; Sohn and Park, 1998). In the multiple criteria framework, this algorithm just presents the supported efficient solutions.

In Bp-HLP2, we use the same parameters and decision variables except the additional ones presented below. The mathematical model of Bp-HLP2 is as follows;

Parameters

Cap_k: Capacity of hub k

Decision Variables

Z_2: Maximum delay of hubs which is represented as $\max_k \left\{ \dfrac{\sum_i \sum_j F_{ij} \sum_l X_{ijkl}}{Cap_k} \right\}$

where $\sum_i \sum_j F_{ij} \sum_l X_{ijkl}$ is the total flow entering hub k via collection (demand).

Bp-HLP2:

$$\text{Min} \quad \sum_{i \in N} \sum_{k \in N} \sum_{l \in N} \sum_{j \in N} F_{ij} C^{1}_{ijkl} X_{ijkl}$$

$$\text{Min} \quad Z_2$$

Subject to

$$\frac{\sum_i \sum_j F_{ij} \sum_l X_{ijkl}}{Cap_k} \le Z_2 \qquad \forall k \in N \qquad (4.7)$$

(4.1) – (4.6)

The first objective is to minimize overall transportation cost of UMAp-HMP and the second objective is to minimize the maximum demand/capacity (D/*Cap*) ratio of hubs. In the second objective, we are actually interested in balancing D/*Cap* ratio of hubs. Constraint (4.7) identifies the maximum D/*Cap* ratio over all hubs. Here, in terms of second objective, the allocation problem corresponds to determine the initial hub k for all (i,j) pairs (Note that for UMAp-HMP objective allocation problem corresponds to determine both initial hub k and final hub l for

all (*i,j*) pairs). Since we only consider the flow entering a hub via collection, in the single objective case we may assign any one of the hubs as final hub *l* to each (*i,j*) pair.

Complexity of UMAp-HMP is presented above. In the literature, there has not been a study considering the minimization of maximum D/*Cap* objective within hub network structure. One related study is proposed by Chen and Choi (2001). They allocate the web documents among web servers in order to achieve load balancing. Since the web servers are known, they only solve the allocation problem and give the complexity results. Their objective is to minimize the maximum load over all servers. They represent the load of a server as total access cost per http connection. Our allocation problem to minimize the maximum D/*Cap* objective is similar to the problem proposed by Chen and Choi (2001); however, we represent the load of a hub as D/*Cap*. Next, we give the complexity results of our second objective.

First, we introduce the allocation decision problem related to our second objective as follows:

Given F_{ij}, H_k, Cap_k and an upper bound value (UB) on Z_2, does there exist a feasible allocation structure such that $Z_2 \leq UB$?

The corresponding allocation optimization problem is:

Given F_{ij}, H_k and Cap_k find a feasible allocation structure that minimizes Z_2, i.e. find initial hub *k* for each (*i,j*) pair that minimizes Z_2.

Lemma 1. Allocation decision problem is NP-complete in the strong sense.

Proof. Allocation decision problem is in class NP since it can be checked in polynomial time whether a specific allocation structure yields a 'yes' answer for allocation decision problem. The lemma is proved by reducing *Bin Packing* problem to this problem. The *Bin Packing* problem is stated in Garey and Johnson (1979) as follows:

Instance: Finite set U of items, size $s(u) \in Z^+$ for each $u \in U$, a positive integer bin capacity B and a positive integer p.

Question: Is there a partition of U into disjoint sets U_1, U_2, … ,U_p such that the sum of the sizes of the items in each U_k is ≤ B ?

Bin Packing problem is NP-complete in strong sense (Garey and Johnson, 1979).

Given a set U of (i,j) pairs, let UB be an upper bound and $\frac{F_{ij}}{Cap_k}$ be the size of (i,j) pair at hub k (=1,2,…,p). Then allocation decision problem is to partition the set U of (i,j) pairs into p disjoint subsets U_1, U_2, … , U_p such that the sum of the sizes of the pairs at each hub $k \le$ UB , i.e. $\sum_{i-j \in U_k} \frac{F_{ij}}{Cap_k} \le$ UB for each k, which is equivalent to $\frac{\sum_i \sum_j F_{ij} \sum_l X_{ijkl}}{Cap_k} \le$ UB for each k. Therefore, the allocation decision problem is NP-complete in strong sense. □

Lemma 2. Allocation optimization problem is NP-hard in the strong sense.

Proof. Let B^* denote the optimal objective value of the allocation optimization problem. By using the binary search algorithm, we can determine B^* by a sequence of polynomial number of calls of the allocation decision problem. That means the allocation optimization problem is at least as hard as the allocation decision problem. □

Proposition. Choosing the p nodes having highest capacities as hubs is an optimal solution to minimize Z_2 objective.

Proof. Let $Cap_{[1]} \ge ... \ge Cap_{[k]} \ge ... \ge Cap_{[p]} \ge Cap_{[p+1]} \ge \ldots \ge Cap_{[n]}$ denote the decreasing order of the capacity of nodes and B^* denote the optimal value of the minimize Z_2 objective. Assume that we choose the p nodes having highest capacities as hubs.

$$\sum_{i-j \in U_{[1]}} \frac{F_{ij}}{Cap_{[1]}} \le B^*, \ldots, \sum_{i-j \in U_{[k]}} \frac{F_{ij}}{Cap_{[k]}} \le B^*, \ldots, \sum_{i-j \in U_{[p]}} \frac{F_{ij}}{Cap_{[p]}} \le B^*$$

To prove the optimality of B^* we will show that replacing one of the hubs with another one having a smaller capacity, such as replacing $Cap_{[k]}$ with $Cap_{[p+t]}$ where $t \geq 1$, cannot reduce B^* as long as the total demand is constant. Let B^{new} denote the value of the minimize Z_2 objective after replacement.

Two cases should be considered:

Case 1. if $\sum_{i-j \in U_{[p+t]}} \frac{F_{ij}}{Cap_{[p+t]}} \leq B^*$ then $B^{new} = B^*$

Case 2. if $\sum_{i-j \in U_{[p+t]}} \frac{F_{ij}}{Cap_{[p+t]}} > B^*$

i. In this case, if sum of D/*Cap* ratios for all hubs (Note that $U_{[k]}$ has been removed) is greater than $p.B^*$, i.e. $\sum_{r=1}^{p} \sum_{i-j \in U_{[r]}} \frac{F_{ij}}{Cap_{[r]}} > p.B^*$, then $B^{new} > B^*$.

ii. Otherwise, some (*i,j*) flow is shifted from $hub_{[p+t]}$ to the other hubs for optimality and the new subsets are $U_{[1]}^{new}, U_{[2]}^{new}, ..., U_{[r]}^{new}$. Note that $U_{[k]}$ has been removed. In this case, $B^{new} \geq B^*$.

In both cases replacing one of the hubs with another one having smaller capacity cannot reduce B^*. This proves our proposition. □

4.3.2 Computation of Input Parameter F_{ij} for TPDS

In this study, the flow matrix for Turkish PTT data set (TPDS) is constructed as proposed in Çetiner (2003). He assumes that, every body writes a letter every day, in other words each city produces an amount of letters equal to its population daily. Considering the bulk mails, formal mails and large size envelopes, this assumption may be reasonable. The amount of mail sent from city *i* to city *j* is assumed to be proportionate to the relative population of city *j* and it is estimated as follows:

$$F_{ij} = P_i \frac{P_j}{\left[\sum_{k=1}^{n} P_k\right] - P_i} \qquad \text{for } i \neq j \text{ and } i, j = 1, 2, \ldots, n \qquad (4.8)$$

$$F_{ii} = 0 \qquad \text{for } i = 1, 2, \ldots, n \qquad (4.9)$$

where P_i is the population of i^{th} city and F_{ij} is the amount of mail flow from city i to city j. The population figures are out of official Turkish Census 2000 statistics. The list of cities according to decreasing order of population is given in Appendix B.1.

4.3.3 Criteria Space

Criteria space of Bp-HLPs has some interesting properties. For each hub set, different allocation structures generate different solutions. Allocation decision for each possible hub set can be evaluated as a separate bicriteria allocation problem. So, each sub problem has its own criteria space and set of efficient solutions. The entire criteria space may be visualized as in Figure 4.3 for a small hypothetical example of Bp-HLP's for n=9 and p=3.

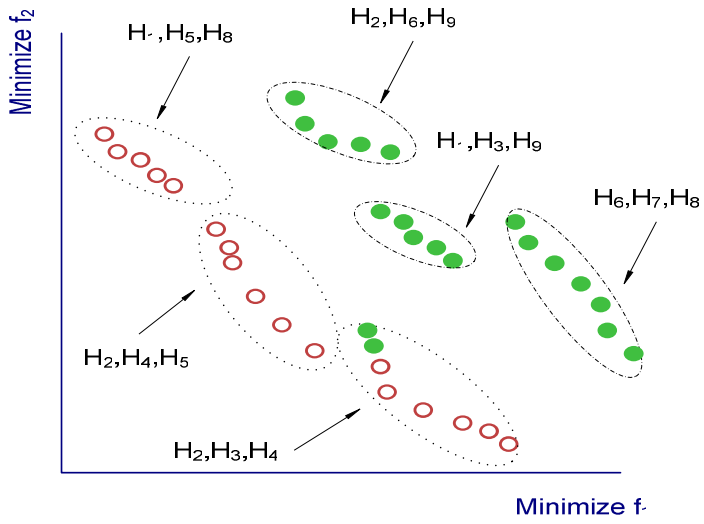

Figure 4.3 Hypothetical example solutions for Bp-HLPs.

As it can be observed from Figure 4.3, each hub set has its own subpopulation (say family). Note that in this figure, only efficient allocation structures of each family are represented. H_1,H_5,H_8 family, H_2,H_4,H_5 family and H_2,H_3,H_4 family are efficient hub families of Bp-HLPs and the empty circles are the efficient allocation structures of these efficient hub families. In this setting, the main decision is to choose the locations of hubs. If Figure 4.3 is carefully examined, one notices that there is no need to find the entire allocation structures of all families. If the efficient hub families are known or approximated by an algorithm, another algorithm may approximate the corresponding set of efficient allocation structures of each family. This is the strategy that we apply in our study. We first approximate the efficient hub families of Bp-HLPs. For this purpose, we present an evolutionary algorithm FWEA_loc, which is explained in section 4.4. FWEA_loc uses a different approach, which is fitting an Lq distance function (see section 4.4.1), to represent the allocations of each hub family. Then we approximate the efficient allocation structures of each efficient family by using FWEA_alloc, which is presented in section 4.5. Note that FWEA_alloc algorithm is applied to each efficient family separately.

4.4 AN EVOLUTIONARY ALGORITHM FWEA_LOC

In the design of our EA, each family is represented by some points taken on the fitted Lq distance function. These are hypothetical solutions that are assumed to represent actual solutions. Consequently, parent selection, insertion and replacement decisions are given based on these representative points.

A critical and difficult point while designing EAs is finding a suitable fitness function. The fitness value of each member should indicate the capability of this member to survive. Each member wants to choose such weights that provide greatest advantage to itself. As more advantage is gained, evolution will be towards the efficient frontier. While assigning the fitness, the members that are

located at the currently underrepresented portions of the efficient frontier should also be favored. Such a favoring structure distributes the population over the frontier. In chapter 3, we introduce an evolutionary algorithm, called FWEA, for approximating the efficient set of solutions. The algorithm tries to evolve the population of solutions towards the efficient frontier and distribute it over the efficient frontier in order to maintain a diverse and well spread representation. The raw-fitness score of each member is a convex combination of average strength of that member relative to the entire population and its strength compared to closest contender along a search direction. The search direction is obtained by allowing each member to choose its own weights for Tchebycheff distance function. Then, the fitness scores are computed by progressively decreasing the raw-fitness scores according to the non-dominated frontiers. Thanks to such a fitness function, each solution tries to converge to the efficient frontier along its own favorite direction.

In the design of our algorithm for the p-hub location problem, we construct a similar fitness function structure. The main aspects of our algorithm are discussed below:

4.4.1 Fitting an Lq Distance Function to Hub Families

In this study, a novel approach is proposed to compute the fitness scores of each family. The major question that needs to be answered is: how can we represent the overall efficient solutions (allocation structures) of the family? The answer that we suggest is to fit an Lq distance function for each family. Köksalan (1999) presents such an approach for a bicriteria scheduling problem. He observes that in some combinatorial problems, the efficient frontier can be roughly represented by a smooth Lq distance function. He defines some known Lq distance functions and takes a sample of points on each of them. Then he finds non-dominated points close to these sample points. An alternative approach is also suggested in the paper. First an efficient (or approximately efficient) solution between extreme efficient solutions, say (a, b), is found. Then the Lq distance function passing through extreme efficient solutions and (a, b) is determined. Since we can easily

find the extreme efficient solutions (or approximately efficient) $(x = (f_1^l, f_2^u), z = (f_1^u, f_2^l))$ of a family they can be used as our bounds. Then, normalized values can be defined as $\left(a' = \frac{(a - f_1^l)}{(f_1^u - f_1^l)}, b' = \frac{(b - f_2^l)}{(f_2^u - f_2^l)} \right)$. In our study, we use the approach suggested by Köksalan (1999). The idea is represented in Figure 4.4.

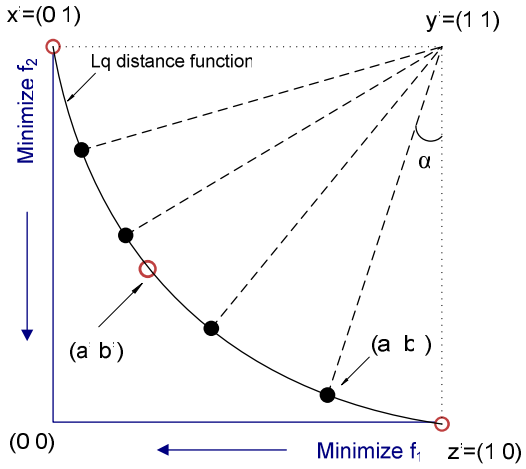

Figure 4.4 Lq distance function passing through three efficient points of a family

In Bp-HLP1, for each family, first we try to find extreme efficient solutions and a supported efficient solution (a, b) by solving bicriteria all-pair shortest path problem with single criterion and equal weighted linear aggregation of criteria respectively. For this purpose, Floyd-Warshall all-pair shortest path algorithm is applied to resulting single criterion problem (see Ernst and Khrishnamoorthy, 1998a; Sohn and Park, 1998). If we cannot encounter any supported efficient solution except extreme efficient solutions, we try the Tchebycheff algorithm with equal weights. However, since the allocation problem of minimizing the maximum D/*Cap* criterion is NP-hard, finding these points is not so easy for Bp-HLP2.

Therefore, we try the LP relaxed Tchebycheff algorithm for the allocation problem to approximate the points x, z and (a, b). After normalizing these points, an Lq distance function that passes through them is found as follows:

$$(1-a')^q + (1-b')^q = 1 \qquad (4.10)$$

Since there is no closed form solution of equation (4.10) for q, an approximation algorithm, such as binary search, can be used. After finding q, some equally spaced points on the Lq distance function are taken as in Köksalan (1999). To find k+1 equally spaced points, the 90^o angle $\hat{x'y'z'}$ is divided by k. In this case, $\alpha_i = \frac{90i}{k}$ for $i = 0, 1, \ldots, k$, then a point (a'^i, b'^i) is computed as follows:

$$a'^i = 1 - r_i \sin\alpha_i \quad \text{and} \quad b'^i = 1 - r_i \cos\alpha_i,$$

where r_i is the Euclidean distance from (1,1) to the point (a'^i, b'^i). The corresponding points on the criteria space are as follows:

$$a^i = f_1^l + (f_1^u - f_1^l)\, a'^i \quad \text{and} \quad b^i = f_2^l + (f_2^u - f_2^l)\, b'^i$$

Figure 4.5 denotes some solutions for AP data set for *n*=10, p=3 and Bp-HLP1 (colored lines), and the fitted Lq distance functions (dark lines). According to figure some points on the H_3,H_4,H_7 family, H_1,H_4,H_7 family and H_3,H_7,H_8 family belong to the efficient frontier. While these three families are the efficient families of criteria space, H_2,H_3,H_7 family and H_1,H_7,H_8 family are inefficient.

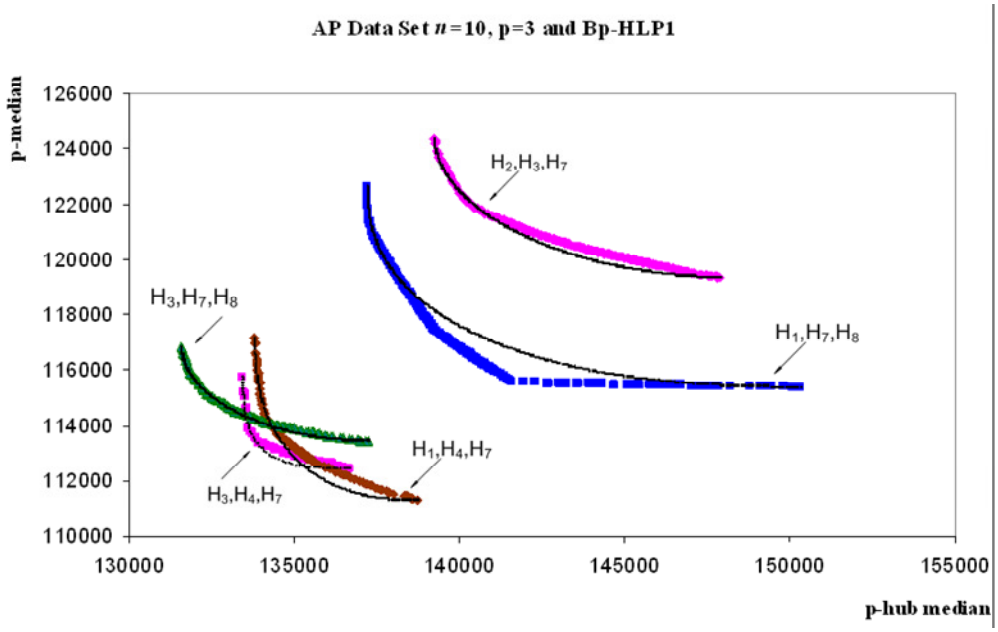

Figure 4.5 Example solutions and fitted Lq distance functions for AP data set, *n*=10, p=3 and Bp-HLP1.

Figure 4.6 illustrates some solutions for AP data set for *n*=10, p=2, Loose capacities case and Bp-HLP2 (colored lines), and the fitted Lq distance functions (dark lines). AP data set includes two types of capacities as Tight and Loose. Here we figure out Loose capacity case, however we have performed experiments on both cases. According to figure H_3,H_7 family, H_4,H_7 family and H_7,H_8 family are the efficient families of criteria space.

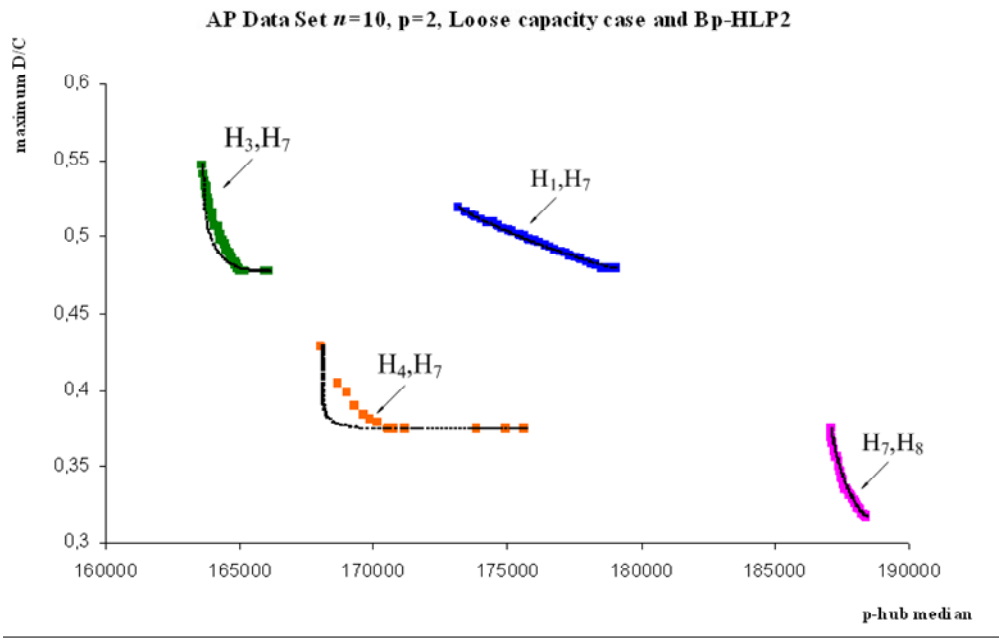

Figure 4.6 Example solutions and fitted Lq distance functions for AP data set, *n*=10, p=2, Loose capacities and Bp-HLP2.

As it can be observed in the figures, if appropriate points are chosen from each family, Lq distance function may represent the entire family well. To demonstrate the closeness of each family with its Lq distance function an experiment is performed. For this purpose, some equally spaced discrete points are taken on the Lq distance function and the *proximity indicator* (Bosman and Thierens, 2003) is computed as stated in equation (4.11).

$$D_{X^{family} \to X^{Lq}} = \frac{1}{\left|X^{Lq}\right|} \sum_{x^1 \in X^{Lq}} \min_{x^2 \in X^{family}} \left\| x^1 - x^2 \right\|_2 \qquad (4.11)$$

where $|X|$ denotes the cardinality of set X and $\left\| x^1 - x^2 \right\|_2$ denotes the Euclidean distance between solutions x^1 and x^2. $D_{X^{family} \to X^{Lq}}$ gives the average Euclidean distance between each point belonging to family and its nearest neighbor on the corresponding Lq distance function.

Table 4.1 shows the *proximity indicator* results for the families given in Figures 4.5 and 4.6. Before computing the *proximity indicator*, each point in a family is normalized based on the range of the family. To be able to make a comparison, we also compute the *proximity indicator* for a rectilinear distance function (i.e. L_1). It can be seen that the values for Lq distance function are very small compared with the values for L_1 distance function showing the closeness of families and fitted Lq distance functions.

Table 4.1 *Proximity indicator* results for the families given in Figures 4.5 and 4.6

Problem	**Family**	$D_{X^{family} \to X^{Lq}}$	$D_{X^{Lq} \to X^{family}}$	$D_{X^{family} \to X^{L_1}}$	$D_{X^{L_1} \to X^{family}}$
Bp-HLP1	H_3,H_4,H_7	0.04085	0.03877	0.23415	0.21321
	H_1,H_4,H_7	0.05862	0.05231	0.21237	0.19299
	H_3,H_7,H_8	0.02133	0.01960	0.19093	0.17627
	H_2,H_3,H_7	0.04073	0.03213	0.17276	0.16196
	H_1,H_7,H_8	0.03227	0.03002	0.30379	0.24922
Bp-HLP2	H_3,H_7	0.03070	0.05881	0.31819	0.26417
	H_4,H_7	0.05836	0.15194	0.29859	0.29550
	H_7,H_8	0.01370	0.02173	0.11563	0.12391
	H_1,H_7	0.01924	0.02593	0.03702	0.04108

4.4.2 Fitness Function

The strengths constitute an important part of our fitness mechanism. They represent a solution's power of endurance in the current population. In the proposed algorithm, the fitness score of each representative point ($\hat{x}$) on the fitted Lq distance functions is considered. We measure the strength $\Delta(x^i)$ of a member $\hat{x} \in X^{family\_s}$ relative to some $x^i \in X^{family\_t}$, i.e. the distance between x^i and $\hat{x}$, by using weighted Tchebycheff distance function as follows:

$$\Delta(x^i) = \phi(\hat{w}, x^i) - \phi(\hat{w}, \hat{x}),$$

where

$\phi(w,x) = \max\{w_1(f_1(x)-z_1^*), w_2(f_2(x)-z_2^*)\}$, and $\hat{w}$ is the favorable weight vector of $\hat{x}$.

Then, the raw fitness value includes convex combination of the average of the relative strengths $\overline{\Delta} = \left(\frac{\sum_{x^i \in X^{family\_t}} \Delta(x^i)}{|X| - |X^{family\_s}|} \right)$ and worst-case measure $\Delta_{\min} = \left(\min_{x^i \in X^{family\_t}} \{\Delta(x^i)\} \right)$.

$$raw\_fitness(\hat{x}) = \beta.\overline{\Delta} + (1-\beta).\Delta_{\min}$$

where β controls the balance between average and minimum strength and takes values between 0 and 1.

Raw-fitness value of a member not only indicates average strength over other members, but also incorporates its diversity measure. It is expected that an individual far from the other members will have large average strength. This member is also rewarded with a premium $\Delta_{\min}$ that indicates the distance with the closest contender. A negative $\Delta_{\min}$ value shows that $\hat{x}$ is outperformed by some $x^i \in X^{family\_t}$ even at its favorable weights.

The non-dominated sorting based fitness assignment structure proposed by Goldberg (1989) has been applied by some known MOEAs, such as NSGA I-II. According to this, non-dominated members of a population constitute the first frontier. After removing these members temporarily, remaining non-dominated solutions constitute the second frontier, and so on. We adapt this traditional fitness assignment structure to adjust the raw-fitness values and finally to obtain fitness scores as follows:

Step 1. Find the frontier each member belongs to.

Step 2. Let the fitness value of an individual on the first frontier be equal to its raw-fitness value. Starting with the individuals on the second frontier, adjust the raw-fitness scores so that they are less than the minimum

fitness score of the member in the previous frontier by an amount ε (=0.0001).

Let X^{rank_j} be the set of solutions on frontier j. Consider $\hat{x} \in X^{rank\_j}$, $j = 2,\ldots,N$.

$$fitness(\hat{x}) = \min_{x \in X^{rank\_j-1}} \{fitness(x)\} - \max_{x \in X^{rank\_j}} \{raw\_fitness(x)\} + raw\_fitness(\hat{x}) - \varepsilon$$

Due to such a fitness assignment structure, relative distance among individuals' raw-fitness scores (on the same frontier) is preserved and the fitness value of a member on a worse frontier could not be greater than the fitness value of those on a better frontier. Therefore, the members on better frontiers will have more chance to survive and reproduce.

4.4.3 Favorable Weights

In our algorithm, favorable weights of each representative point is computed by using the same method presented in FWEA algorithm.

4.4.4 Representation

In FWEA_loc, each chromosome includes p genes that represent the location of p hubs. Given a network with *n* nodes, each gene can take a value between 1 and *n*. The following example demonstrates our representation structure for *n*=9 and p=4, where nodes 2,5,8 and 9 correspond to hubs.

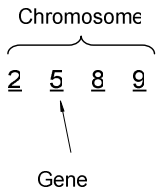

Figure 4.7 Chromosome representation structure for hub location problem.

4.4.5 Generating Initial Population

While generating our initial population, the main principle is to represent every city in the initial population with the same frequency. If a city is not seen in the initial population, it may never be seen and the algorithm may never find the efficient hub set(s). In this study, we apply the procedure suggested by Alp et al. (2003) to generate a population with these properties. Smallest number of members required for the population to represent each gene is $d=\left\lceil \frac{n}{p} \right\rceil$. Selecting the population size as kd for some constant k, each city can be replicated k times in the initial population. For the first set of $\frac{n}{p}$ genes, they assign the cities to the genes with an increment of 1 and for the k^{th} set, they assign the cities to the genes with an increment of k. For instance, for (n, p, k) = (9, 3, 2), the sets and members are generated as follows:

$$\underbrace{(1\ 2\ 3), (4\ 5\ 6), (7\ 8\ 9)}_{k=1}, \underbrace{(1\ 3\ 5), (7\ 9\ 2), (4\ 6\ 8)}_{k=2}$$

Figure 4.8 Example for generating the initial population

If $\frac{n}{p}$ is not an integer, the remaining empty slots are filled randomly and in this case some cities will appear more than k times in the initial population.

4.4.6 Scaling

In most of the problems, the range of each criterion is different. In this case, a scaling procedure must be used to evaluate each criterion on the same scale in the evolutionary algorithms. Otherwise, the algorithm may be biased over the criteria

having large ranges. In this study, we scale the range of each criterion into [0, 1] interval as follows:

$$f_1^{'} = \frac{f_1 - f_1^l}{f_1^u - f_1^l} \quad , \qquad f_2^{'} = \frac{f_2 - f_2^l}{f_2^u - f_2^l}$$

where (f_i^u) defines an upper bound and (f_i^l) defines a lower bound for each criterion *i*. The lower bounds may be found by using LP relaxation and the upper bounds may be the worst criterion values in the initial population.

4.4.7 Crossover

In this study, we apply the single-point crossover operator with probability p_c. Parent chromosomes are split into two parts from a randomly chosen point and tails of each parent chromosome are exchanged to generate two new offspring. Before the cutting operation, as stated in Perez et al. (1998) common hubs are marked and shifted through the end of the chromosome. This is to preserve building blocks of parents. The following figure illustrates the crossover scheme.

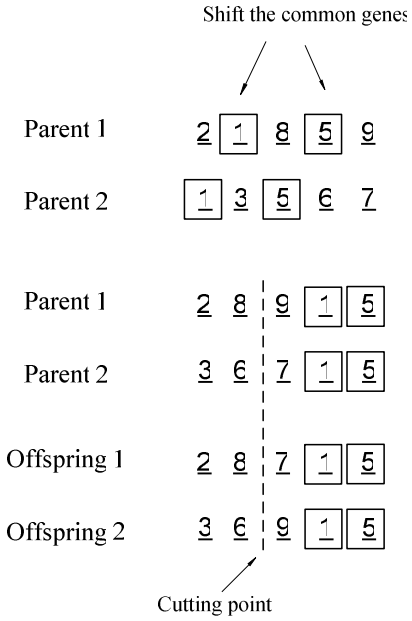

Figure 4.9 Crossover scheme

4.4.8 Mutation

Mutation operator helps to achieve diversity and to escape from local optimal. According to our mutation scheme, a randomly chosen gene is replaced with another randomly chosen node, which is not seen on this chromosome, with probability p_m.

4.4.9 Insertion and Replacement Rules

Our EA works with a linearly increasing population size until a preset upper limit as have been stated in chapter 3. Every family not being "stillborn" or "duplicate" can be introduced into the population if there exists an empty place. We refer to an offspring as "stillborn family", if all the representative elements on it are dominated by any other representative element on the worst frontier. We also deny duplicate solutions in terms of hub sets.

We use steady-state replacement strategy after the cardinality of the population reaches the upper limit. In every generation, two offspring are generated and replacements are realized according to best fitness score in the family. The family having the weakest of best fitness scores (according to representative points) is replaced. The main reason behind this strategy is to preserve the best members of the families. Most of the members of a family may be located on worse frontiers, however, just a few of them may be on pareto frontier so we do not want to loose them.

4.4.10 Ranking and Fitness Updates

In our algorithm, the raw fitness value and frontier of a member are determined relative to the population at the time of offspring family's entry. After that, as new families are introduced into the population, fitness values and frontiers of the existing members must be updated considering this new family. Note that, x^{off}'s entry may cause the removal of an existing family. For these updates, we must first

determine the frontier of the representative members in the new family. After updating the frontiers of the members, raw fitness scores are adjusted as given in section 4.4.2.

4.4.11 The FWEA_loc Algorithm

Steps of the algorithm are presented below.

Input parameters:

- Pop.Size1-size of the initial population,
- Pop.Size2-upper limit on the size of the population,
- *n*-number of nodes in the network,
- p-number of hubs
- p_c- crossover probability, p_m- mutation probability
- # gen. –Maximum number of generations
- β- controls the balance between average and minimum strength in raw_fitness function

Step 0. Initialization

Generate an initial population by using the approach proposed in section 4.4.5.

Step 1. Evaluation of the initial population

For each family;

1.1 Solve the allocation problem for each criterion and find extreme efficient solutions of this family. Also find another efficient solution (a, b) belonging to this family (see section 4.4.1).

1.2 Estimate lower and upper bounds for the scaling issues and scale the criteria values (see section 4.4.6).

1.3 Find an Lq distance function passing from the points x', z' and (a', b') of the family (see section 4.4.1).

1.4 Take k equally spaced representative points on this fitted Lq distance function (see section 4.4.1)

For each point on the Lq distance function;

1.5 Compute the favorable weights (see section 4.4.3)

1.6 Determine the frontiers (see section 4.4.2)

1.7 Compute the fitness scores by adjusting raw-fitnesses according to frontiers (see section 4.4.2).

Step 2. Selection

Select two parents according to a selection operator. Take the hub family of chosen points as parents. In this study, binary tournament selection with replacement (i.e. the same parent can be chosen twice) is used.

Step 3. Crossover

Apply crossover with crossover probability p_c to generate two offspring (see section 4.4.7).

Step 4. Mutation

Apply mutation with mutation probability p_m (see section 4.4.8).

Step 5. Duplication check

If any offspring is a duplicate of any existing family (according to hubs), then discard it. If both of them are duplicate, then discard both and goto Step 10.

Step 6. Evaluation of the offspring

Repeat step 1.1-1.6.

Step 7. Stillborn check (see section 4.4.9).

If all the representative points of the family are dominated by other points on the worst frontier (offspring is stillborn), then discard this family. If both of the offspring are stillborn, then discard both and goto Step 10.

Step 8. Insertion and replacement

Increment population cardinality and insert the offspring one by one into the population. If the population cardinality exceeds the preset upper limit (POPSIZE2), remove the weakest family based on best fitness score of its representative points (see section 4.4.10).

Step 9. Update ranks and raw-fitnesses. Adjust raw fitness scores (see section 4.4.2).

Step 10. Termination

If a stopping condition is not reached, go to Step 2.

4.5 AN EVOLUTIONARY ALGORITHM FWEA_ALLOC

After approximating the efficient hub families of Bp-HLP's by using FWEA_loc, we can approximate the efficient allocation structures with another evolutionary algorithm FWEA_alloc. We modify the FWEA_loc presented in section 4.4 in such a way that the algorithm tries to approximate the efficient allocation structures of each family. Since we use the same strategy for scaling and computing the favorable weights, we do not discuss these parts in this section. We also do not discuss the insertion and replacement rules, crowding measure and ranking and fitness updates parts since they are the same with FWEA. The modified parts of our EA for the allocation problem are described below.

4.5.1 The Fitness Function

The Fitness function of FWEA_alloc is similar to the fitness function of FWEA_loc. Since we consider just one of the efficient families (say X^{family}), we measure the strength $\Delta(x^i)$ of a member (an allocation structure) $\hat{x} \in X^{family}$ relative to each $x^i \in X^{family}$ i.e. the distance between x^i and $\hat{x}$, by using a weighted Tchebycheff distance function as follows:

$\Delta(x^i) = \phi(\hat{w}, x^i) - \phi(\hat{w}, \hat{x})$,

where

$\phi(w, x) = \max\{w_1(f_1(x) - z_1^*), w_2(f_2(x) - z_2^*)\}$ and $\hat{w}$ is the favorable weight vector of $\hat{x}$.

Then, the raw fitness value is calculated as: $raw\_fitness(\hat{x}) = \beta.\overline{\Delta} + (1-\beta).\Delta_{\min}$

where $\overline{\Delta} = \left(\frac{\sum_{x^i \in X^{family}} \Delta(x^i)}{|X^{family}| - 1} \right)$ is the average relative strength, $\Delta_{\min} = \left(\min_{x^i \in X^{family}} \{\Delta(x^i)\} \right)$ is the worst case measure and β controls the balance between average and minimum strength.

4.5.2 Representation

In FWEA_alloc, given a network with n nodes and p hubs, each chromosome includes n^2 genes that represent the origin-destination (i,j) pairs. Since the traffic is routed between any (i,j) pair via hubs k and l, the corresponding value of each gene is one of the p^2 (k,l) hub pairs. The following example illustrates the representation scheme for n=9 and p=4. In this case, the chromosome includes 81 genes (if $F_{ii} = 0$, the number of genes reduces to $n(n-1)$) and one of the sixteen hub pairs is assigned to each of them. Figure 4.10 shows the chromosome representation.

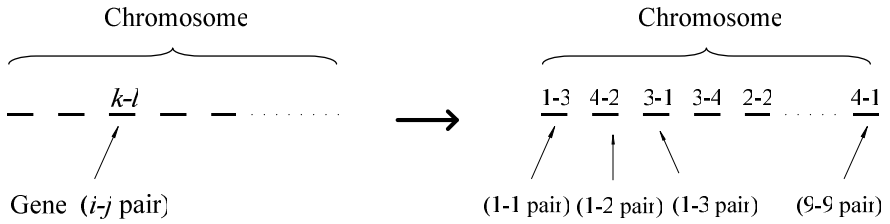

Figure 4.10 Chromosome representation for allocation problem

Generating Initial Population

In FWEA_alloc, initial population is generated randomly, i.e. a randomly chosen (k,l) hub pair is allocated to each gene. Since we know the three efficient allocations of the Bp-HLP1, we also seed the initial population with these

solutions in this problem. In Bp-HLP2, we know the three solutions obtained from LP relaxed Tchebycheff algorithm in order to approximate the point x, z and (a, b). We repair these solutions and seed into the initial population of the allocation problem. In LP relaxation, only infeasibility occurs due to constraint 4.2, which guaranties that the flow between each (*i,j*) pair is routed by one hub (*k,l*) pair. In LP relaxation, flow between each (*i,j*) pair may be fractional and routed by different (*k,l*) pairs. We repair such infeasibility by assigning (*i,j*) pair to such a (*k,l*) pair having maximum partition.

4.5.3 Criteria Space Restriction for the Allocation Problem

As stated in section 4.5.2, in a network with *n* nodes and p hubs, there are n^2 origin-destination (*i,j*) pairs (if $F_{ii} = 0$, it reduces to *n*(*n*-1)) and p^2 initial-final (*k,l*) hub pairs. Since one of the (*k,l*) hub pairs has to be allocated to each (*i,j*) pair (note that this does not mean single allocation), the size of the criteria space for the allocation structure of a given hub family is $(p^2)^{n^2}$. However, there is no need to explore the entire criteria space for efficient allocation structures. For this purpose, at the beginning of the algorithm we may reduce the search space by defining non-dominated hub pairs for each (*i,j*) pair and by only allowing the allocation of those hub pairs to (*i,j*) pair. For instance, let us consider the H_3,H_7,H_8 family for AP data set *n*=10, p=3 and Bp-HLP1. One of the following (*k,l*) hub pairs given in Table 4.2 has to be allocated to each (*i,j*) pair in the network. In this table, we show all possible allocation structures and contributions to the objective functions (partial criteria values) due to (6,2) node pair. Here the partial criteria values are computed by using the input parameters of AP data set for *n*=10 and p=3.

Table 4.2 Possible allocation structures of (6,2) node pair of H_3,H_7,H_8 family in Bp-HLP1.

Initial hub *k*	Final hub *l*	Partial $f_1(x)$ $F_{62}.C^1_{62kl}$	Partial $f_2(x)$ $F_{62}.C^2_{62kl}$	Non-dominated
3	3	1777.4	1777.4	X
3	7	2354.4	2134.6	X
3	8	2223.6	1970.7	X
7	7	1972.9	1972.9	X
7	3	1835.5	1615.7	X
7	8	1930.3	1808.9	X
8	8	1377.7	1377.5	√
8	7	1662.6	1541.2	X
8	3	1437.1	1184.1	√

Since the other allocation structures are dominated, only (8,8) hub pair or (8,3) hub pair can be allocated to (6,2) node pair. By constructing such a list, we can specify for each (*i,j*) pair, the set of (*k,l*) hub pairs that can be allocated to each (*i,j*) pair.

When we consider the Bp-HLP2, the criteria space restriction is different from Bp-HLP1. In the second criterion of Bp-HLP2 we cannot construct a linear relationship between partial criteria values (i.e. $\frac{F_{ij}}{Cap_k}$) and Z_2. That means although a partial allocation structure dominates the other partial allocations, it may not be chosen. A similar case can be observed in Table 4.3. According to table, if (6,2) node pair is assigned to (8,8) hub pair, it has minimum partial criteria values and dominates the other allocations; however, in terms of second criteria this allocation does not guarantee the best objective value due to non-linearity. In this case, we may reduce the search space deciding on the final hub *l* for each initial hub *k* in terms of first criteria. For instance, in Table 4.3 (3,8), (7,3) and (8,8) hub pairs can only be allocated to (6,2) node pair.

Table 4.3 Possible allocation structures of (6,2) node pair of H_3,H_7,H_8 family in Bp-HLP2.

Initial hub k	Final hub l	Partial $f_1(x)$ $F_{62}.C^1_{62kl}$	Partial $f_2(x)$ $\frac{F_{62}}{Cap_k}$	Non-dominated
3	3	1777.4	0.02865	√
3	7	2354.4	0.02865	X
3	8	2223.6	0.02865	X
7	7	1972.9	0.05945	X
7	3	1835.5	0.05945	√
7	8	1930.3	0.05945	X
8	8	1377.7	0.01371	√
8	7	1662.6	0.01371	X
8	3	1437.1	0.01371	X

4.5.4 Crossover

As in FWEA_loc, we apply single point crossover operator with probability p_c in FWEA_alloc. Parent chromosomes are split into two parts from a randomly chosen point and tails of each parent chromosome are exchanged to generate two new offsprings.

4.5.5 Mutation

According to our mutation scheme, each gene is replaced with probability p_{mut}. That means another randomly chosen (*k,l*) hub pair from reduced set is assigned to this gene.

4.5.6 The FWEA_alloc Algorithm

Steps of the algorithm is presented below.

Input parameters:

- Pop.Size1-size of the initial population,
- Pop.Size2-upper limit on the size of the population,
- *n*-number of nodes in the network,
- p-number of hubs

- p_c- crossover probability, p_{mut}- mutation probability
- # gen. –Maximum number of generations
- β-weight for raw_fitness function

Step 0. Initialization

Generate an initial population randomly and seed it (see section 4.5.3)

Step 1. Evaluate the initial population

1.1 Compute the criteria values

1.2 Estimate lower and upper bounds for the scaling issues and scale the criteria values (see section 4.4.6).

1.3 Compute the favorable weights (see section 4.4.3).

1.4 Determine the frontiers (see sections 4.5.1 and 4.4.2)

1.5 Compute the fitness scores by adjusting raw-fitnesses according to frontiers (see section 4.4.2).

Step 2. Selection

Select two parents according to a selection operator. In this study binary tournament selection with replacement (i.e. the same parent can be chosen twice) is used.

Step 3. Crossover

Apply crossover with the crossover probability p_c to generate an offspring (see section 4.5.5).

Step 4. Mutation

Apply mutation with the mutation probability p_m (see section 4.5.6).

Step 5. Duplication check (see related part of FWEA).

If any offspring is a duplicate of any existing population members, then discard it. If both of them are duplicate, then discard both and goto Step 10.

Step 6. Evaluation of the offspring

Repeat step 1.1-1.4.

Step 7. Stillborn check (see related part of FWEA).

If any offspring is dominated by any members of the worst frontier of current population (offspring is stillborn), then discard it. If both of them are stillborn, then discard both and goto step 10.

Step 8. Insertion and Replacement (see related part of FWEA).

8.1 Increment population cardinality and insert the offspring one by one into the population.

8.2 If the offspring does not outperform any population member x^i with its own weights w^i (i.e. $D(x^i) > 0$ for all i), remove the member having lowest crowding measure if the population cardinality exceeds a preset upper limit.

8.3 If the offspring outperforms any existing population members at their own favorable weights (i.e. $D(x^i) < 0$ for some i)

8.3.1 Remove the weakest such member unless that member is on the first frontier

8.3.2 Remove the member with lowest crowding measure, if the weakest such member is on the first frontier.

Step 9. Update ranks and raw-fitnesses. Adjust raw fitness scores (see related part of FWEA).

Step 10. Termination

If a stopping condition is not reached, goto Step 2.

4.6 COMPUTATIONAL RESULTS

Performance of our algorithm was tested on TPDS, AP and CAB data sets. Algorithms were coded in C and run on a Pentium IV 1.6 GHz PC. To find the efficient solutions on Pareto frontier CPLEX 8.1 Mixed Integer Problems solver was used.

TPDS data set was compiled by Çetiner (2003). It includes the flow and distance matrices of 81 cities of Turkey. The distance matrix d is symmetric whereas the flow matrix F is not symmetric. Since the mail traffic within a city is not considered, F_{ii} equals to 0 for all i. The collection and distribution per unit costs

are $\chi = 1$ and $\delta = 1$, respectively. In our study, we have experimented with 25 major cities having population over 700,000 according to Turkish Cencus 2000 statistics. These cities are illustrated in Figure 4.11 and they represent 74.78% of Turkish population. Since we want the network with *n* nodes to represent the overall mail traffic of Turkey, we took the *n* most crowded of the 81 cities in Turkey. We assumed the remaining cities' mail is sent to the closest of the chosen *n* cities.

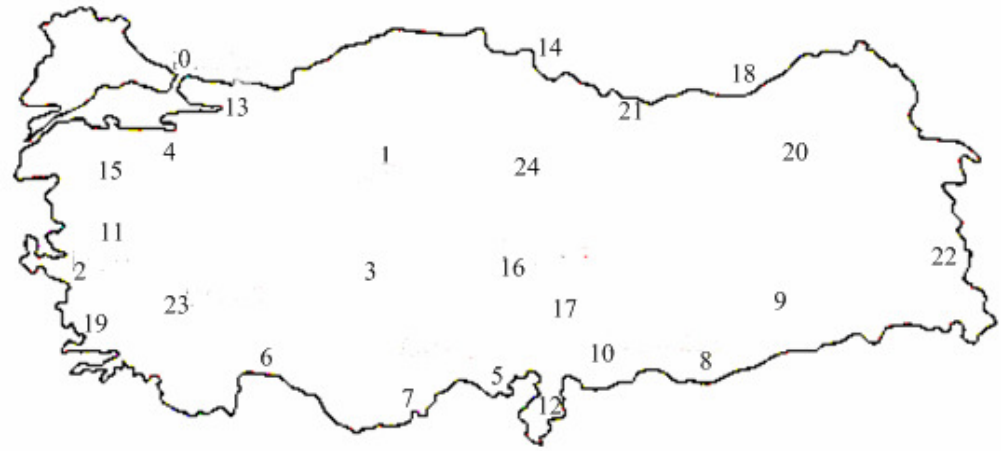

Figure 4.11 Most crowded 25 cities of Turkey

AP data set was introduced by Ernst and Krishnamoorthy. It is based on the mail flow of an Australian city. The mail flow matrix is not symmetric and $F_{ii} \neq 0$. The coordinates of the districts are given and the Euclidean distance is computed for each pair of districts. The collection, transfer and distribution per unit costs are $\chi = 3,\ \ \alpha = 0.75$ and $\delta = 2$ respectively. The maximum values of *n* and p are equal to 200 and 8, respectively. In the literature, single criterion UMAp-HMP optimal solutions are available up to *n*=50 and p=5 (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/phub2.txt). We also compare FWEA_loc results with OR-Lib optimal hub sets just based on UMAp-HMP

criterion. The performance of our algorithm is tested for *n*=25, p=5 in Bp-HLP1 and for *n*=10, p=5, tight (T) and loose (L) capacity cases in Bp-HLP2.

CAB data set was introduced by O'Kelly (1987). It includes airline passenger flow data of 25 US cities. The flow matrix is symmetric, $F_{ii} = 0$, and $\chi = 1$, $\delta = 1$.

In TPDS and CAB data sets, we test the performance of our algorithm for *n*=25, α=0.2, 0.5, 0.8, p=5 in Bp-HLP1 and *n*=10, α=0.2, 0.5, 0.8, p=5 in Bp-HLP2. However, TPDS and CAB data sets do not include capacities while AP data set does. Ebery et al. (2000) suggest the following equation to generate capacities for each node if the (*x, y*) coordinates and the volumes of flows F_{ij} are provided.

$$Cap_k = \left(\frac{n}{\mathrm{p}} + \frac{3d_k O_k}{5d_m O_m} \right) O_k \qquad 4.12$$

where d_k is the distance of node *k* from the center of mass of the system, O_k is the total outflow from node k, d_m and O_m are the maximum values of d_i and O_i respectively, and p is the number of hubs. In this formulation, nodes having higher outflows and nodes that are farther from the center of mass are assigned relatively larger capacities. Here the single unknown is the center of mass of the system. In TPDS data set, it is reasonable to choose the center of network as node 1 (Ankara), which is both the capital city of Turkey and located nearly in the center of Turkey. Since we experiment on a network with 10 nodes in Bp-HLP2, Cincinnati seems very close to the center of the network among the first 10 cities in the CAB data set. Then we compute capacities as stated in equation 4.12.

In order to obtain efficient hub sets and efficient allocation structures, we used the ε-constraint approach (Haimes et al., 1971). We treat one of the criteria as a constraint and we restrict some parts of the criteria space with the help of this constraint. We also augment the optimized criterion with a small coefficient, ρ, times the value of the other criterion to exclude the weakly efficient but inefficient solutions. For Bp-HLP1, the resulting single criterion problem is as follows:

ε – Constraint Model:

Minimize $f_1(\mathrm{X}) + \rho.f_2(\mathrm{X})$

Subject to

(4.1) – (4.6)

$$f_2(\mathrm{X}) \le \varepsilon_t \qquad (4.13)$$

$$f_1(\mathrm{X}) = \sum_{i\in \mathrm{N}} \sum_{k\in \mathrm{N}} \sum_{l\in \mathrm{N}} \sum_{j\in \mathrm{N}} F_{ij} C^1_{ijkl} \mathrm{X}_{ijkl} \qquad (4.14)$$

$$f_2(\mathrm{X}) = \sum_{i\in \mathrm{N}} \sum_{k\in \mathrm{N}} \sum_{l\in \mathrm{N}} \sum_{j\in \mathrm{N}} F_{ij} C^2_{ijkl} \mathrm{X}_{ijkl} \qquad (4.15)$$

where ε_t is an upper bound for the second criterion and $\rho > 0$ is sufficiently small. The following theorem defines the ranges of ρ. The reader may refer to Steuer (1986, pp.440-443) for a similar results for the Tchebycheff distance case.

Theorem. Let X be the set of solutions in criteria space and $x^{\mathrm{a}}, x^b \in \mathrm{X}$ be any two solutions such that $x^{\mathrm{a}} \neq x^{\mathrm{b}}$. Assume without loss of generality that $f_1(x^a) < f_1(x^b)$. Then $f_1(x^{\mathrm{a}}) + \rho.f_2(x^{\mathrm{a}}) < f_1(x^{\mathrm{b}}) + \rho.f_2(x^{\mathrm{b}})$ (4.16),

when $0 < \rho < \min_{\substack{x^1, x^2 \in \mathrm{X} \\ f_1(x^1) < f_1(x^2)}} \left\{ \dfrac{f_1(x^2) - f_1(x^1)}{f_2(x^1) - f_2(x^2)} \right\}$

Proof. We may transform (4.16) to the following inequality: $\rho < \dfrac{f_1(x^{\mathrm{b}}) - f_1(x^{\mathrm{a}})}{f_2(x^{\mathrm{a}}) - f_2(x^{\mathrm{b}})}$.

We know that $x^{\mathrm{a}}, x^b \in \mathrm{X}$ and $f_1(x^a) < f_1(x^b)$. Two cases should be considered:

Case 1. $f_2(x^{\mathrm{b}}) \ge f_2(x^{\mathrm{a}})$. This case is straight forward, since x^a dominates x^b.

Case 2. $f_2(x^{\mathrm{b}}) < f_2(x^{\mathrm{a}})$. In this case, we need $0 < \rho < \dfrac{f_1(x^{\mathrm{b}}) - f_1(x^{\mathrm{a}})}{f_2(x^{\mathrm{a}}) - f_2(x^{\mathrm{b}})}$.

Choosing the minimum value of $\{f_1(x^2) - f_1(x^1)\}$ $\left(\textit{i.e.} \min_{\substack{x^1, x^2 \in \mathrm{X} \\ f_1(x^1) < f_1(x^2)}} \{f_1(x^2) - f_1(x^1)\} \right)$

and the maximum value of $\{f_2(x^1) - f_2(x^2)\}$ $\left(\textit{i.e.} \max_{\substack{x^1, x^2 \in \mathrm{X} \\ f_1(x^1) < f_1(x^2)}} \{f_2(x^1) - f_2(x^2)\} \right)$

guarantees that (4.16) is satisfied. □

Constraint 4.13 guarantees that the total transportation cost of the p-median problem is less than or equal to a predefined level of ε_t. Since the number of solutions on the efficient frontier is exponential for both of the problems, we define some levels for ε_t and solve the ε-constraint problem for these levels. For this purpose, first we solve the single objective problems separately and define upper and lower bounds. Then, the range of the second criterion is divided into equal subranges and the upper bound of each subrange is set as ε. Figure 4.12 illustrates the proposed idea. Note that, this method does not find all the Pareto solutions and in order not to obtain the same Pareto solution we skip some ε_t values (e.g. ε_4) as they are greater than the second criterion value of the last found Pareto solution. In this study, we divide the range of second criterion into 100 subranges.

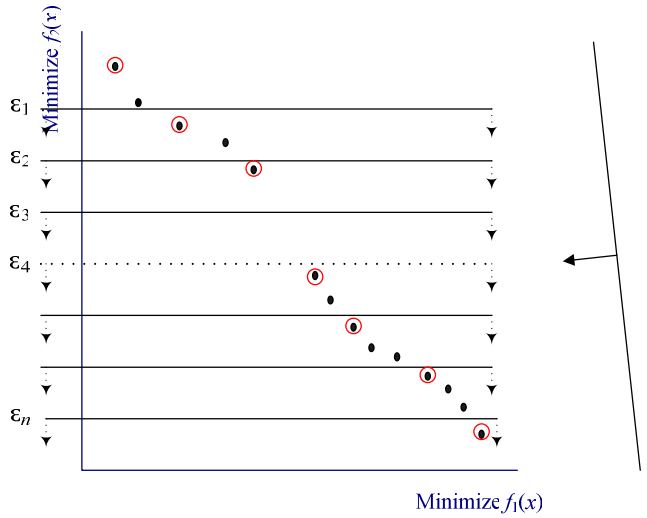

Figure 4.12 ε-constraint approach

In the computational experiments with our EA, we set the stopping condition to $100n$ and $1000n$ generations for FWEA_loc and FWEA_alloc algorithms respectively. We also performed a long run of FWEA_loc for Bp-HLP1. Crossover

and mutation probabilities are taken as 0.90 and 0.05, respectively in both of the algorithms. We performed some experiments to define the level of β in the raw-fitness function. For this purpose, we compute the *proximity indicator* (Bosman and Thierens, 2003) as stated in equation 4.11 for some instances of Bp-HLP1 and Bp-HLP2. Our findings are presented in Table B.2-Table B.5 and Figure B.1-Figure B.6 in Appendix B. Based on these results we set the level of β to 0.8 for Bp-HLP1 and Bp-HLP2 in FWEA_loc and we set the level of β to 0.01 for Bp-HLP1 and Bp-HLP2 in FWEA_alloc. Although the high levels of β in location problem give better results, the low levels of it are better in allocation problem. Actually, this is reasonable when we consider the criteria space and population size of the both algorithms. In location problem, criteria space includes the all families and each family is represented with some dummy points distant to each other. Depending on the distribution of the families on the criteria space, dummy points of a family may also distant to dummy points of other families. In such a case, average relative strength portion of the fitness function takes importance and encourage choosing a high level of β. In FWEA_loc, each family is represented by 5 equally spaced dummy points on Lq distance function. We take the initial population size as 20 families and the upper limit as 25 families. In other words, we start with 100 (=20x5) individuals and finally population reaches to 125 (=25x5) individuals. We perform 5 replications for each problem instances by using different seeds. In FWEA_alloc, the initial population size is set to 80 and the upper limit is set to 100. FWEA_loc takes the resulting efficient hub families of FWEA_loc as an input.

4.6.1 Evaluation of Results

Table 4.4 shows the hub sets found by a single run of FWEA_loc, efficient hub sets of Bp-HLP1 found by ε-constarint approach, OR-Lib single criterion UMAp-HMP optimal hub locations (*) and single criterion p-median problem optimal locations (**) for AP data set. Table 4.5 and Table 4.6 illustrate the results for different α levels of TPDS data set and CAB data set respectively. According to these results, our EA is capable of finding the efficient hub sets in nearly all of the

problems. Due to computational complexity, we could not find the efficient hubs for $n > 25$ problems. However, from OR-Lib we know one efficient hub set, which corresponds to the optimal solution of UMAp-HMP for AP data set. Additionally, we solve the single criterion p-median problem to find optimal locations. So, we perform further experiments on our EA to find these two efficient hub sets. For this purpose, we run our algorithm with the parameter settings mentioned before for n = 40, 50 and p = 4, 5. According to results, for this moderate size instances we do find the OR-Lib UMAp-HMP optimal hub set and p-median optimal locations. They are marked with (*) and (**) in Table 4.4. The complete results for different levels of n, p and all data sets are presented in Table B.6 – Table B.8 in Appendix B.

Table 4.4 AP data set results for Bp-HLP1

n	**p**	**Bp-HLP1 Efficient Hub Sets**	**FWEA_loc Hub Sets**
25	5	2, 8, 17, 18, 20 * 2, 8, 15, 17, 18 2, 8, 15, 16, 18 **	2, 8, 17, 18, 20 * 2, 8, 15, 17, 18 2, 8, 15, 16, 18 **
40	4	-	12, 23, 26, 28 * 3, 15, 26, 28 3, 15, 18, 28 **
40	5	-	3, 13, 23, 26, 28 *, **
50	4	-	14, 28, 32, 35 * 6, 28, 32, 35 6, 29, 32, 35 **
50	5	-	4, 14, 28, 32, 35 * 4, 15, 29, 33, 35 4, 14, 28, 33, 35 4, 15, 29, 32, 35 **

* OR-Lib UMAp-HMP optimal hub sets

** p-median problem optimal locations

\- Efficient hub sets could not be found by ε-constraint approach due to complexity of the problem

Table 4.5 TPDS data set results for *n*=25, p=5 and Bp-HLP1

Bp-HLP1 Efficient Hub Sets			**FWEA_loc Hub Sets**		
α = 0.2	**α = 0.5**	**α = 0.8**	**α = 0.2**	**α = 0.5**	**α = 0.8**
1,8,12,17,18	0,1,7,12,17	0,1,7,12,17	1,8,12,17,18	0,1,7,12,17	-
1,8,9,12,17	0,1,8,12,17	0,1,8,12,17	1,8,9,12,17	0,1,8,12,17	0,1,8,12,17
1,9,12,17,19	1,8,12,17,18	1,8,12,17,18	1,9,12,17,19	1,8,12,17,18	1,8,12,17,18
1, 9,12,13,19	1,9,12,17,19	1,9,12,17,19	1, 9,12,13,19	1,9,12,17,19	1,9,12,17,19
1, 8, 9,12,13	1,8,9,12,17	1,8,9,12,17	1, 8, 9,12,13	1,8,9,12,17	1,8,9,12,17
1, 8,12,13, 18	1,9,12,13,19	1,9,12,13,19	1, 8,12,13, 18	1,9,12,13,19	1,9,12,13,19
	1,8,9,12,13	1,8,9,12,13		1,8,9,12,13	1,8,9,12,13
	1,8,12,13,18	1,8,12,13,18		1,8,12,13,18	1,8,12,13,18
				0,1,8,12,13	1,12,17,18,19
					0,1,8,12,13
					0,1,7,12,13
					0,1,7,15,17
					1,13,15,18,19

- Efficient hub sets FWEA_loc could not find

Table 4.6 CAB data set results for *n*=25, p=5 and Bp-HLP1

Bp-HLP1 Efficient Hub Sets			**FWEA_loc Hub Sets**		
α = 0.2	**α = 0.5**	**α = 0.8**	**α = 0.2**	**α = 0.5**	**α = 0.8**
4,7,12,14,17	4,7,12,14,17	4,7,12,17,24	4,7,12,14,17	4,7,12,14,17	4,7,12,14,17
		4,7,12,14,17		4,7,12,17,24	4,7,12,17,24
					4,7,12,18,24

A close examination of Table 4.5 and 4.6 indicates that as α increases, different efficient hub sets appear. This is because, as the economies of scale between hubs decreases, the tradeoff between the two criteria increases.

When we examine the results for TPDS data set (Table 4.5), we observe higher number of hub sets at each replication as compared with the AP and CAB data sets. This may be the result of the distribution of cities over Turkey. When we check the map of Turkey, we see that the crowded cities of Turkey are distributed rather than being concentrated in a region. This makes the tradeoff clear between objectives.

Also examining the different efficient hub sets of problem instances for all data sets enables us to observe the fact that at least one hub is common, which means

that both our criteria prefer this district as hub. The main reason is possibly that this district is the center of a dense region. For instance, in TPDS data set node 1 and node 12 (Ankara and Hatay), which are located in the center of crowded regions of Turkey, appear in all efficient hub sets.

Table 4.7 shows the hub sets found by a single run of FWEA_loc, efficient hub sets of Bp-HLP2, OR-Lib single criterion UMAp-HMP optimal hub locations and optimal hub locations of min. max. D/*Cap* problem for AP data set. Tables 4.8 and 4.9 illustrate the results for different α levels of TPDS data set and CAB data set respectively. According to these results, our EA is also capable of finding the efficient hub sets in nearly all of the problems. Due to computational complexity, we could not find the efficient hubs for $n > 10$ problems. The complete results for different levels of n, p and all data sets are presented in Table B.9 – Table B.11 in Appendix B.

Table 4.7 AP data set results for n=10, p=5 and Bp-HLP2

capacity	Bp-HLP2 Efficient Hub Sets	FWEA_loc Hub Sets
T	1, 2, 3, 7, 8 *	1, 2, 3, 7, 8 *
	1, 2, 5, 7, 8	1, 2, 5, 7, 8
	1, 4, 5, 7, 8	1, 4, 5, 7, 8
	1, 2, 5, 8, 10	1, 2, 5, 8, 10
	1, 4, 5, 8, 10	1, 4, 5, 8, 10
	1, 4, 5, 9, 10	1, 4, 5, 9, 10
	4, 5, 6, 9, 10 **	4, 5, 6, 9, 10 **
		1, 4, 5, 7, 10
		1, 3, 4, 7, 8
		1, 2, 5, 7, 10
		1, 2, 3, 7, 10
L	1, 2, 3, 7, 8 *	1, 2, 3, 7, 8 *
	1, 3, 4, 7, 8	1, 3, 4, 7, 8
	1, 2, 5, 7, 8	-
	1, 4, 5, 7, 8	1, 4, 5, 7, 8
	2, 6, 7, 8, 10 **	2, 6, 7, 8, 10 **
		2, 5, 7, 8, 10
		2, 7, 8, 9, 10
		1, 4, 7, 8, 10
		1, 5, 6, 7, 8
		2, 4, 7, 8, 10

* OR-Lib UMAp-HMP optimal hub locations
** Optimal hub locations for min. max. D/*Cap* problem
- Efficient hub sets FWEA_loc could not find

Table 4.8 TPDS data set results for *n*=10, p=5 and Bp-HLP2

Bp-HLP2 Efficient Hub Sets			**FWEA_loc Hub Sets**		
α = 0.2	**α = 0.5**	**α = 0.8**	**α = 0.2**	**α = 0.5**	**α = 0.8**
0,1,4,6,7	0,1,4,6,7	0,1,4,6,7	0,1,4,6,7	0,1,4,6,7	0,1,4,6,7
1,3,4,6,7	1,3,4,6,7	1,3,4,6,7	1,3,4,6,7	1,3,4,6,7	1,3,4,6,7
			1,4,6,7,8		

Table 4.9 CAB data set results for *n*=10, p=5 and Bp-HLP2

Bp-HLP2 Efficient Hub Sets			**FWEA_loc Hub Sets**		
α = 0.2	**α = 0.5**	**α = 0.8**	**α = 0.2**	**α = 0.5**	**α = 0.8**
3,4,6,7,9	3,4,6,7,9	3,4,6,7,9	3,4,6,7,9	3,4,6,7,9	3,4,6,7,9
1,3,4,7,9	1,3,4,7,9	1,3,4,7,9	1,3,4,7,9	1,3,4,7,9	1,3,4,7,9
3,4,7,8,9	3,4,7,8,9	1,3,4,6,7	3,4,7,8,9	3,4,7,8,9	1,3,4,6,7
3,4,6,7,8	3,4,6,7,8		3,4,6,7,8	3,4,6,7,8	
			3,4,7,9,10	3,4,7,9,10	

A close examination of AP and CAB data sets in Bp-HLP1 and Bp-HLP2 (Tables 4.4 - 4.7 and Tables 4.6 – 4.9) indicates that more efficient hub sets appear in Bp-HLP2 than in Bp-HLP1 implying the higher tradeoff between criteria. However, the converse is true for TPDS data set. The number of efficient hub sets for TPDS in Bp-HLP2 (Table 4.8) is extremely less as compared to Bp-HLP1 (Table 4.5). The reason for this may be the fact that the cities with highest capacities (nodes 1,3,4,6 and 7) are closed to each other (for instance, nodes 1 and 3, nodes 6 and 7, nodes 3 and 7) in TPDS data set and both criteria prefer these nodes as hub.

We also observe that some additional hub sets are found by FWEA_loc in addition to the efficient ones. These additional hub sets may be the ones that are missed by the ε-constraint approach. Moreover, the number of additional hub sets is also higher in Bp-HLP2 than in Bp-HLP1. This may be the result of approximation with LP relaxation to the points x, z and (a, b) to fit an Lq distance function in Bp-HLP2. In this case, we allow the continuous allocations and the fitted Lq distance function for each family actually belongs to the continuous version of the Bp-

HLP2. Therefore, these additional efficient hub sets may appear if we allow continuous allocations in Bp-HLP2.

Tables 4.10-4.12 show the computational results of FWEA_alloc in terms of *proximity indicator* (P.I.) (Bosnan and Thierens, 2003), *hypervolume indicator* (H.I.) (Zitzler et al. 2003) and CPU time (seconds). Before computing the *proximity indicator* and *hypervolume indicator*, each point is normalized based on the range of the Pareto frontier. *Hypervolume indicator* computes the region or hypervolume weakly dominated by a set of points and thus it gives information about both the proximity of solutions to the Pareto frontier and the diversity of them over the Pareto frontier. The closed hypervolume is bounded by the nadir point. Note that, higher value of this indicator is better and after normalization, at most one unit square area (hypervolume) can be dominated by a set of points. CPU time of FWEA_alloc for each hub set is computed separately and if there are more than one hub sets, they are summed. We experimented with the resulting hub sets of FWEA_loc algorithm and Tables 4.10-4.12 denotes the average best and worst values of the performance measures for n=25, p=5 and all data sets. Tables also show the hub sets corresponding to the replications of best and worst values of the *proximity indicator*. In the performance evaluation of FWEA_alloc, we also compute the proximity indicator only for no-show hub sets which could not be reached by FWEA_alloc. For this purpose, we only consider the efficient points belonging to no-show hub sets in Pareto frontier and compute the proximity indicator with FWEA_alloc results. Here our aim is to show how close FWEA_alloc resulting solutions with these no-shows and whether we can suggest alternative close solutions to decision maker, who is interested in these no-shows, or not.

According to Tables 4.10-4.12, our algorithm generally produces close solutions to Pareto frontier; however, especially in AP and CAB data sets some undesirable results are observed due to extreme run results that occur in some of the runs of FWEA_loc algorithm and they directly affect the performance of FWEA_alloc algorithm. For instance, in Table 4.10, although the hub set found in one of the

replications has similar building blocks with the efficient hub sets, its allocation structures are not so close to the efficient ones. This is also true for the results in Table 4.12. Nevertheless, if FWEA_loc algorithm finds the efficient hub sets, then FWEA_alloc algorithm gives better results in terms of P.I. and H.I. For this purpose, we decide to perform a long replication of FWEA_loc algorithm by increasing the number of generations from 100*n* to 1000*n*. The results presented in column *long* indicate that FWEA_loc algorithm is capable of finding efficient hub sets in a long replication. Considering the importance of hub location decision, performing long runs are actually reasonable. Generally, in our problem instances we observe all efficient hub sets in at least one of the replications. That means a no-show does not appear at least one of the runs, but once it appears, P.I. (no-show) measure is high indicating the long distance between our results and no-show parts of the Pareto frontier. According to H.I., the unit square area (hypervolume) dominated by Pareto frontier and FWEA_alloc is very close to each other in most cases, which demonstrates the well distribution of solutions and proximity to the Pareto frontier.

Tables 4.10-4.12 also include the performance evaluation of the union of the replications. For this operation, we first combine the last populations of all replications and then evaluate the non-dominated solutions of the combined population. As might be expected, the performance of the union operation is better than the best performing run.

Table 4.10 Computational results for AP data set, *n*=25, p=5 and Bp-HLP1

	Avg.	**Best**	**Worst**	**Union**	**Long**
Hub Sets		2,8,15,17,18 2,8,15,16,18 2,8,17,18,20	2,7,14,17,18		2,8,15,17,18 2,8,15,16,18 2,8,17,18,20
P.I.	0.13153	0.01629	0.57013	0.01092	0.01654
P.I. (no-shows)	0.39070	-	0.57013	-	-
H.I. (Pareto)	0.86749	0.86749	0.86749	0.86749	0.86749
H.I. (FWEA_alloc)	0.73332	0.86043	0.23634	0.86433	0.85752
CPU (FWEA_loc)	737.0	688.7	831.9	2257.6	5140.9
CPU (FWEA_alloc)	1031.5	419.0	1417.0	2867.5	1421.5

Table 4.11 Computational results for TPDS data set, n=25, p=5 and Bp-HLP1

	α = 0.2				
	Avg.	**Best**	**Worst**	**Union**	**Long**
Hub Sets		1,8,12,17,18 1,8,9,12,17 1,9,12,17,19 1,9,12,13,19 1,8,9,12,13 1,8,12,13,18	1,8,12,17,18 1,8,9,12,17 1,9,12,17,19		1,8,12,17,18 1,8,9,12,17 1,9,12,17,19 1,9,12,13,19 1,8,9,12,13 1,8,12,13,18
P.I.	0.04809	0.00907	0.08252	0.00645	0.03385
P.I. (no-shows)	0.13463	-	0.21974	-	-
H.I. (Pareto)	0.74482	0.74482	0.74482	0.74482	0.74482
H.I. (FWEA_alloc)	0.72041	0.74212	0.68912	0.74460	0.72280
CPU (FWEA_loc)	763.6	743.0	813.5	3818.0	4864.1
CPU (FWEA_alloc)	2113.3	1026.3	3047.7	10566.5	3037.6
	α = 0.5				
	Avg.	**Best**	**Worst**	**Union**	**Long**
Hub Sets		0,1,8,12,17 0,1,7,12,17 1,8,12,17,18 1,8,9,12,17 1,9,12,17,19	0,1,8,12,17		0,1,7,12,17 0,1,8,12,17 1,8,12,17,18 1,9,12,17,19 1,8,9,12,17 1,9,12,13,19 1,8,9,12,13 1,8,12,13,18 0,1,8,12,13
P.I.	0.03510	0.02147	0.08423	0.01865	0.02091
P.I. (no-shows)	0.05857	0.02194	0.12605	-	-
H.I. (Pareto)	0.82159	0.82159	0.82159	0.82159	0.82159
H.I. (FWEA_alloc)	0.78749	0.79525	0.76344	0.80034	0.79419
CPU (FWEA_loc)	723.4	660.1	787.4	3617.0	3899.7
CPU (FWEA_alloc)	2820.2	602.2	4922.5	14101.0	5518.6
	α = 0.8				
	Avg.	**Best**	**Worst**	**Union**	**Long**
Hub Sets		1,8,12,13,18 0,1,7,12,13 0,1,7,13,15 0,1,8,12,17 0,1,7,12,17 0,1,8,12,13 0,1,8,15,17 1,8,9,12,17 1,8,12,17,18 1,8,9,12,13	1,15,17,19,23 1,13,15,18,21 1,13,15,18,19 0,1,15,17,21 1,7,13,15,18 1,15,17,18,21 1,15,17,18,19 1,9,15,17,19 1,9,15,17,21 9,12,16,17,19		- 0,1,8,12,17 1,8,12,17,18 1,9,12,17,19 1,8,9,12,17 1,9,12,13,19 1,8,9,12,13 1,8,12,13,18 1,12,17,18,19 0,1,8,12,13 0,1,7,15,17 1,13,15,18,19
P.I.	0.05864	0.02525	0.16908	0.02197	0.02491
P.I. (no-shows)	0.06499	0.02506	0.16908	0.04674	0.05596
H.I. (Pareto)	0.81511	0.81511	0.81511	0.81511	0.81511
H.I. (FWEA_alloc)	0.72248	0.77115	0.57887	0.77570	0.76566
CPU (FWEA_loc)	681.3	634.1	703.8	3406.5	4682.5
CPU (FWEA_alloc)	6176.0	3551.7	7730.2	30880.0	641.9

Table 4.12 Computational results for CAB data set, n=25, p=5 and Bp-HLP1

	α = 0.2				
	Avg.	**Best**	**Worst**	**Union**	**Long**
Hub Sets		4,7,12,14,17	6,12,14,17,21		4,7,12,14,17
P.I.	1.70369	0.01712	5.79046	0.01367	0.02926
P.I. (no-shows)	2.82664	-	5.79046	-	-
H.I. (Pareto)	0.78995	0.78995	0.78995	0.78995	0.78995
H.I. (FWEA_alloc)	0.31014	0.77713	0.00000	0.78025	0.76730
CPU (FWEA_loc)	678.3	635.2	709.6	3391.5	4212.7
CPU (FWEA_alloc)	801.5	589.0	1516.5	4007.5	625.5
	α = 0.5				
	Avg.	**Best**	**Worst**	**Union**	**Long**
Hub Sets		4,7,12,14,17 4,7,12,17,24	4,10,12,14,17 4,10,12,17,24		4,7,12,14,17 4,7,12,17,24
P.I.	0.085948	0.02184	0.18090	0.01737	0.02223
P.I. (no-shows)	0.18069	-	0.18090	-	-
H.I. (Pareto)	0.84384	0.84384	0.84384	0.84384	0.84384
H.I. (FWEA_alloc)	0.72123	0.81681	0.57742	0.82305	0.81122
CPU (FWEA_loc)	667.5	584.3	737.9	3337.5	4158.9
CPU (FWEA_alloc)	1500.7	1417.9	1756.9	7503.5	1414.8
	α = 0.8				
	Avg.	**Best**	**Worst**	**Union**	**Long**
Hub Sets		4,7,12,17,24 4,7,12,14,17	6,12,14,17,21 6,12,17,21,24		4,7,12,17,24 4,7,12,14,17 4,7,12,18,24
P.I.	0.14360	0.03231	0.30922	0.02985	0.03209
P.I. (no-shows)	0.21754	-	0.30922	-	-
H.I. (Pareto)	0.82445	0.82445	0.82445	0.82445	0.82445
H.I. (FWEA_alloc)	0.61575	0.76881	0.41975	0.77276	0.77019
CPU (FWEA_loc)	3208.6	614.9	693.1	16043	4844.1
CPU (FWEA_alloc)	2795.0	1708.6	4608.9	13975	2559.7

Tables 4.13 to 4.15 present the results of Bp-HLP2 for AP, TPDS, CAB data sets respectively. According to tables, our algorithm generally produces close solutions to the Pareto frontier. In AP data set and tight capacity case (Table 4.13), at each replication FWEA_loc algorithm finds all efficient hub sets and average value of the P.I. is 3.88 % of the range of the Pareto frontier, which indicates the close approximation. According to H.I., set of solutions on the Pareto frontier dominates 0.80931 unit square area and FWEA_alloc dominates 0.75201 unit square area on the average. That means FWEA_alloc algorithm dominates 93 % ($=\frac{0.75201}{0.80931}$) of the area that should be dominated. When we look at the loose capacity case, we

obtain similar results in terms of H.I.; however, the value of P.I. is slightly higher than the tight capacity case. In TPDS data set and α = 0.2 case, although P.I. demonstrates the closeness of solutions to the Pareto frontier, both H.I.(Pareto) and H.I.(FWEA_alloc) is very small. This may be related with the shape of the Pareto frontier, which includes big gaps between efficient hub sets.

Table 4.13 Computational results for AP data set, *n*=10, p=5 and Bp-HLP2

	Tight Capacity Case			
	Avg.	**Best**	**Worst**	**Union**
Hub Sets		1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10 1,5,6,9,10	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,5,6,9,10 1,4,5,7,10 1,3,4,7,8	
P.I.	0.03886	0.03595	0.04169	0.02719
P.I. (no-shows)	-	-	-	-
H.I. (Pareto)	0.80931	0.80931	0.80931	0.80931
H.I. (FWEA_alloc)	0.75201	0.75717	0.74982	0.76968
CPU (FWEA_loc)	3610.6	3368.0	3884.2	18053.0
CPU (FWEA_alloc)	2102.9	1820.2	2359.8	10514.5
	Loose Capacity Case			
	Avg.	**Best**	**Worst**	**Union**
Hub Sets		1,5,6,7,10 1,4,5,7,8 2,7,8,9,10 1,2,3,7,8 1,4,7,8,10 2,6,7,8,10 2,4,7,8,10 1,5,6,7,8 1,3,4,7,8 1,2,5,7,8 2,5,7,8,10	1,4,5,7,8 2,7,8,9,10 1,2,3,7,8 1,5,6,7,8 1,3,4,7,8 2,6,7,8,10 2,4,7,8,10 3,4,7,8,10 4,5,6,7,10 1,2,5,7,8	
P.I.	0.05661	0.05152	0.06099	0.04721
P.I. (no-shows)	0.04603	-	0.05068	-
H.I. (Pareto)	0.69449	0.69449	0.69449	0.69449
H.I. (FWEA_alloc)	0.65950	0.68117	0.63506	0.68825
CPU (FWEA_loc)	3536.4	3058.9	3994.0	17682.0
CPU (FWEA_alloc)	1799.8	1648.5	2014.8	8999.0

Table 4.14 Computational results for TPDS data set, *n*=10, p=5 and Bp-HLP2

	α = 0.2			
	Avg.	**Best**	**Worst**	**Union**
Hub Sets		0,1,4,6,7 1,3,4,6,7 1,4,6,7,8	0,1,4,6,7 1,3,4,6,7 1,4,6,7,8	
P.I.	0.06617	0.04657	0.07630	0.03467
P.I. (no-shows)	-	-	-	-
H.I. (Pareto)	0.43539	0.43539	0.43539	0.43539
H.I. (FWEA_alloc)	0.26878	0.33767	0.17200	0.36650
CPU (FWEA_loc)	7271.7	5298.6	8891.2	36358.5
CPU (FWEA_alloc)	627.0	624.0	630.8	3135.0
	α = 0.5			
	Avg.	**Best**	**Worst**	**Union**
Hub Sets		0,1,4,6,7 1,3,4,6,7	0,1,4,6,7 1,3,4,6,7	
P.I.	0.06104	0.04815	0.06643	0.04612
P.I. (no-shows)	-	-	-	-
H.I. (Pareto)	0.89597	0.89597	0.89597	0.89597
H.I. (FWEA_alloc)	0.84688	0.85777	0.83747	0.87473
CPU (FWEA_loc)	6402.3	5218.1	6801.9	32011.5
CPU (FWEA_alloc)	416.4	412.9	419.8	2082.0
	α = 0.8			
	Avg.	**Best**	**Worst**	**Union**
Hub Sets		0,1,4,6,7 1,3,4,6,7	0,1,4,6,7 1,3,4,6,7	
P.I.	0.06897	0.06174	0.07670	0.05286
P.I. (no-shows)	-	-	-	-
H.I. (Pareto)	0.87140	0.87140	0.87140	0.87140
H.I. (FWEA_alloc)	0.74769	0.77616	0.67936	0.79703
CPU (FWEA_loc)	4937.0	4384.3	5585.5	24685.0
CPU (FWEA_alloc)	403.4	400.5	405.2	2017.0

Table 4.15 Computational results for CAB data set, *n*=10, p=5 and Bp-HLP2

	α = 0.2			
	Avg.	**Min.**	**Max.**	**Union**
Hub Sets		3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	
P.I.	0.04708	0.03683	0.05345	0.03179
P.I. (no-shows)	-	-	-	-
H.I. (Pareto)	0.90290	0.90290	0.90290	0.90290
H.I. (FWEA_alloc)	0.85637	0.86557	0.84360	0.87272
CPU (FWEA_loc)	2764.2	2096.7	3555.0	13821.0
CPU (FWEA_alloc)	1027.5	1025.0	1029.6	5137.5
	α = 0.5			
	Avg.	**Best**	**Worst**	**Union**
Hub Sets		3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	
P.I.	0.04353	0.04127	0.04889	0.03125
P.I. (no-shows)	-	-	-	-
H.I. (Pareto)	0.85938	0.85938	0.85938	0.85938
H.I. (FWEA_alloc)	0.80017	0.81351	0.78146	0.82197
CPU (FWEA_loc)	2744.1	2645.7	2877.4	13720.5
CPU (FWEA_alloc)	1084.5	1039.9	1250.7	5422.5
	α = 0.8			
	Avg.	**Best**	**Worst**	**Union**
Hub Sets		3,4,6,7,9 1,3,4,7,9 1,3,4,6,7	3,4,6,7,9 1,3,4,7,9 1,3,4,6,7	
P.I.	0.07335	0.06177	0.08973	0.04800
P.I. (no-shows)	-	-	-	-
H.I. (Pareto)	0.86972	0.86972	0.86972	0.86972
H.I. (FWEA_alloc)	0.75553	0.78588	0.72076	0.75528
CPU (FWEA_loc)	2527.7	1975.4	2973.1	12638.5
CPU (FWEA_alloc)	623.9	622.9	625.9	3119.5

Recall that we suggest criteria space restriction procedure for Bp-HLP1 and Bp-HLP2 separately in section 4.5.4 and we use these procedures in all experiments up to this point. To show the effects of criteria space restriction procedure on the performance of our FWEA_alloc algorithm, now we replicate the runs for AP data set (Table 4.10 and 4.13) without criteria space restriction by using the same settings and seeds of with restriction case. Tables 4.16 and 4.17 present our results.

When we compare Table 4.16 with Table 4.10 for Bp-HLP1 and Table 4.17 with Table 4.13 for Bp-HLP2, we see that P.I. and H.I. in without restriction case are worse than in with restriction case. This indicates that our FWEA_alloc algorithm obtains higher benefit from criteria space restriction in Bp-HLP1 and Bp-HLP2. The complete run results are given in Tables B.18 and B.19 in Appendix B.

Table 4. 16 Computational results for AP data set, *n*=25, p=5 and Bp-HLP1

	Avg.	**Best**	**Worst**
Hub Sets		2,8,15,17,18 2,8,15,16,18 2,8,17,18,20	2,7,14,17,18
P.I.	0.22143	0.10561	0.59956
P.I. (no-shows)	0.4005	-	0.59956
H.I. (Pareto)	0.86749	0.86749	0.86749
H.I. (FWEA_alloc)	0.62729	0.73496	0.20598
CPU (FWEA_loc)	737.0	688.7	831.9
CPU (FWEA_alloc)	918.4	374.6	1260.6

* FWEA_alloc is run without restriction of criteria space presented in Section 4.5.4.

Table 4.17 Computational results for AP data set, *n*=10, p=5 and Bp-HLP2

	Tight capacities		
	Avg.	**Best**	**Worst**
Hub Sets		1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10 1,5,6,9,10	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10 1,5,6,9,10 1,5,6,7,10
P.I.	0.07009	0.06264	0.07415
P.I. (no-shows)	-	-	-
H.I. (Pareto)	0.80931	0.80931	0.80931
H.I. (FWEA_alloc)	0.70298	0.71487	0.69641
CPU (FWEA_loc)	3610.6	3368.0	3884.2
CPU (FWEA_alloc)	2132.2	1828.1	2394.2

4.7 DISCUSSION

In this study, we present two bicriteria uncapacitated multiple allocation p-hub location problems (Bp-HLP1 and Bp-HLP2). In both of the problems, our aim is to define the location of the hubs and then allocate the nodes to these hubs. We propose two evolutionary algorithms for location and allocation decisions separately. FWEA_loc has a novel property that we fit an Lq distance function passing through some efficient (or approximately) efficient points of a family and the performance of FWEA_alloc algorithm is enhanced by restricting the criteria space.

Although our EAs are very successful at finding efficient hub sets and approximating the efficient allocation structures, some drawbacks may occur. One of them is related with the fact that the performance of the FWEA_loc algorithm directly affects the performance of the FWEA_alloc algorithm. According to our experiments, FWEA_loc algorithm achieves the best performance in the long run. Considering the importance of location decision, it is reasonable to perform long replications. The other drawback may be related with the CPU times of the EAs. Especially in Bp-HLP2, LP relaxation used to approximate the extreme efficient solutions of each family increases the computational time of the FWEA_loc due to large size of the model. It should be replaced with a faster and better performing approach.

As a further study, our intention is to approximate the efficient families that are preferred by decision maker instead of all efficient ones. This may be possible by interacting with the decision maker during the FWEA_loc algorithm. Also in the current case, FWEA_alloc algorithm spends effort on the efficient allocation structures of a family that are not located on the actual Pareto frontier. We intend to consider FWEA_alloc to focus on all efficient families simultaneously instead of separately and integrate the location and allocation algorithms. Additionally, in our models, the flow between origin i and destination j is confined to be routed through one path (constraint 4.2). Another future research direction may be

allowing the partition of flow into more than one path (i.e. relax constraint 4.6 as $0 \leq X_{ijkl} \leq 1$). In this case, if different paths are desirable for different criteria, then we will have a continuous efficient frontier. When we consider the higher number of criteria, to exclude this restriction may be reasonable.

CHAPTER 5

MULTIPLE CRITERIA KNAPSACK PROBLEM

In this chapter, we adapt our evolutionary algorithm (FWEA) to approximate the efficient frontier (the set of efficient solutions) in Multi-objective Knapsack Problem (MOKP). Our EA for MOKP, which is called as FWEA_KP, is mainly modified from original FWEA and similarly favorable weight based fitness mechanism is used in order to maintain diverse and well spread population of solutions. We carry out experiments on test data for MOKP given in the literature and compare the performance of FWEA_KP with EMAPS, NSGAII and SPEA2, which are well performing EAs.

5.1 INTRODUCTION

MOKP is one of the well-known Multi-objective Combinatorial Optimization (MOCO) problems with applications in both industry and real life. EAs have successfully been applied to MOCO problems in the last decades. This is because they maintain a population of solutions instead of a single one at each iteration and thus they can obtain multiple efficient solutions in a single run. We refer to Gandibleux and Ehrgott (2005) for a review of recent developments in metaheuristics and hybrid approaches for MOCO problems. Here we mention some of the related literature. SPEA (Zitzler and Thiele, 1999) and SPEA2 (Zitzler et al., 2001) are among the first elitist Multi-objective Evolutionary Algorithms (MOEAs) that are applied to MOKP and yield promising results. NSGA (Srinivas and Deb, 1994) and NSGAII (Deb et al., 2002) are also well-performing MOEAs, using the non-dominated sorting concept of Goldberg (1989). Köksalan and Pamuk (2006) introduce an evolutionary metaheuristic, called EMAPS, for approximating the preference-non-dominated solutions. They apply EMAPS to multi-objective spanning tree problem and MOKP (single constrained).

Recently, we have seen successful applications of genetic local search, in other words memetic algorithms, in the literature as well. Knowles and Corne (2000) perform one of the first studies. They present M-PAES, which incorporates both local search and evolutionary algorithms in multiple criteria framework. Jaskiewicz (2002) proposes a genetic local search based evolutionary algorithm (MOGLS) and carries out experiments on MOKP. Guo et al. (2006) present another memetic algorithm, which performs the local search operation combined with the simulated annealing. For MOKP, Vianna and Aroyo (2004) propose a greedy randomized search procedure (GRASP), which first constructs a feasible solution by using a greedy algorithm and then improves it with a local search procedure. Da Silva et al. (2004) suggest a Scatter Search algorithm, performing together with surrogate relaxation, for bi-criteria knapsack problem. In addition to EAs, Hansen (1997) proposes a tabu search based algorithm and Ulungu et al. (1999) presents a simulated annealing algorithm for MOCO problems.

In this chapter, we aim to adapt our EA proposed in chapter 3 for MOKP. Hereafter we call new version of our EA as FWEA_KP. The fitness function of FWEA_KP is designed to both approximate the Pareto frontier and have a good distribution over the Pareto frontier. For this purpose, each member chooses its own favorable weight according to a Tchebycheff distance function. Some seed solutions at initial population and a crowding mechanism also help to achieve the mentioned objectives.

In Section 5.2, we define the MOKP. In Section 5.3, we give the details of the evolutionary algorithm for MOKP. We present the computational results for test problems in Section 5.4. We discuss the conclusion and direction for further research in Section 5.5.

5.2 PROBLEM DEFINITION

Both single criteria and multiple criteria knapsack problem are among the best-known combinatorial optimization problems. The single criterion knapsack problem is known to be NP-hard and is defined as follows: Given a set of *J* items, each having a profit contribution p_j, weight w_j and a knapsack with capacity *C*, the problem is to select a subset of the items whose total weights do not exceed knapsack capacity *C* and whose total profit contribution is maximized. The corresponding of multiple criteria version of this problem may be obtained by considering multiple knapsacks. Given a set of *J* items and a set of *m* knapsacks with capacities C_k, k=1,2,…,*m,* the multiple criteria knapsack problem is formulated as (Zitzler and Thiele, 1999):

$$"Maximize" \sum_{j=1}^{J} p_{k,j}.x_j, \qquad k = 1,...,m$$

$$Subject \quad to$$

$$\sum_{j=1}^{J} w_{k,j}.x_j \le C_k \qquad \forall k$$

$$x_j \in \{0,1\} \qquad \forall j$$

Where $p_{k,j}$ is the profit obtained by placing item *j* in knapsack *k*, $w_{k,j}$ is the weight of item *j* in knapsack *k*, and x_j takes on the value 1 if the j^{th} item is selected and 0 otherwise. Here we should note that the selected items are included in all knapsacks; however, the weight of the items and the contribution of them to the objective differ from knapsack to knapsack.

Next, we mention some issues regarding the representation and repair operator of our algorithm.

In our study, we use a binary chromosome representation of length *J* as in the study of Zitzler et al. (1999). Each gene of the chromosome represents the value of one decision variable x_j. Since such a coding may lead to infeasible solutions, the

following repair algorithm is applied to the infeasible solutions as in Zitzler et al. (1999):

Step 1. Compute the maximum profit / weight ratio (q_j) for each item *j*.

$$q_j = \max_{k=1,\ldots,m} \left\{ \frac{p_{k,j}}{w_{k,j}} \right\}$$

Step 2. Remove the items with the lowest q_j values until all the capacity constraints are satisfied.

However, this repairing approach is optimistic and it may not perform well in such cases where the solution is better in just one of the criteria while it is not so good in the others.

5.3 AN EVOLUTIONARY ALGORITHM (FWEA_KP)

The main concern while designing EAs is to assign each member a suitable fitness value that will indicate the capability of this member to survive. Therefore, each member needs to have a high fitness score. In our algorithm, this is satisfied by allowing each member to select its own favorite direction according to Tchebycheff distance function. In EMAPS, these favorable weights are assigned based on a linear distance function. In this case, it may be possible to observe some difficulties in finding the unsupported efficient solutions. However, using Tchebycheff distance function enables us to find out such solutions. The main aspects of the algorithm are similar to FWEA proposed in chapter 3. Regarding these similarities, we should remark the following issues: Fitness values and favorable weights are calculated in the same way. Since the profit values, i.e. $p_{k,j}$, for different criterion *k* are from the same distribution with the same range, we do not need to scale the criteria. We also use the same insertion and replacement rules, ranking, fitness update strategies, and crowding measure. Only difference from FWEA is that seed solutions in the initial population are found by using an LP relaxation-based simple approach, which is discussed in the next section.

5.3.1 Generating the Initial Population

Several high quality solutions are tried to be introduced into the initial population of our algorithm. These seed solutions are obtained from the LP relaxation of the single criterion problems, where the single criterion problem is formulated by linearly aggregating the criteria using different weight vectors. In this study, we solve the LP relaxation of each criterion separately and additionally solve one equally weighted sum of the criteria, i.e. we solve m+1 LP relaxed single criterion problems. Then non-integer valued decision variables are rounded to their upper bounds, which means if a decision variable corresponding to one of the items has a value greater than 0, it is assumed to be included in the knapsacks. Afterwards, if the solution violates any of the knapsack constraints, it is repaired by applying the repairing procedure given in section 5.2. Remaining members of the population are generated randomly. Since we solve LP relaxations of the problems, it does not need a considerable computational effort. In some problems where LP relaxation is not possible, a single criterion genetic algorithm may be used as stated in chapter 3.

5.4 COMPUTATIONAL RESULTS

In this section, we compare the performance of our algorithm with EMAPS, SPEA2 and NSGAII that are well-performing algorithms in the literature. We present the computational results in terms of *proximity indicator* (Bosnan and Thierens, 2003) and *hypervolume indicator* (Zitzler et al. 2003). Before computing the *proximity indicator* and *hypervolume indicator*, each point is normalized based on the range of the Pareto frontier as follows:

$$f_i^N(x) = \frac{f_i(x) - f_i^l}{f_i^u - f_i^l}$$

where $f_i^N(x)$ is the normalized value of i^{th} criterion for solution x, f_i^u and f_i^l are lower (estimated by using payoff table approach) and upper bounds (estimated from LP relaxation) of the Pareto frontier for i^{th} criterion respectively. By doing so, we may exclude the problems resulting from criteria ranges and the indicators

may become more meaningful. Here we need to provide some remarks regarding the use of indicators as follows: *Proximity indicator* is generally computed according to Pareto frontier. However, in MOKP, we do not know the Pareto frontier of the considered instance of the problem for more than two criteria. In order to overcome this problem, we construct a Non-Dominated Union Set (NDUS), which includes the non-dominated members of all algorithms, and thus we compute the proximity indicator according to NDUS. We also compute the percent contribution of each algorithm to NDUS. That means we count the number of non-dominated solutions belonging to each algorithm in NDUS and divide by the cardinality of NDUS. *Hypervolume indicator* computes the region or hypervolume dominated by a set of points and thus it gives information about both the proximity of solutions to the Pareto frontier and the diversity of them over the Pareto frontier. Note that, higher value of this indicator is better and after normalization, at most one unit hypervolume can be dominated by a set of points.

Next, we give the details regarding the experimental conditions and parameter settings.

We consider m=2, 3 and 4 knapsacks with J=750 items and perform 10 independent runs for each problem in our experiments. The parameters $p_{k,j}$, $w_{k,j}$ are uniform random integers in the interval [10,100] and the values of these parameters are taken from the site http://www.tik.ee.ethz.ch/~zitzler/testdata.html/ by Zitzler. The knapsack capacities are set to half of total weight of all the items considered for the k^{th} knapsack:

$$c_k = 0.5\sum_{j=1}^{J} w_{k,j}$$

In our study, we use a binary chromosome of length J, single point crossover operator with probability p_c and bit wise mutation with probability p_{mut} as in the study of Zitzler et al. (1999). In FWEA_KP and EMAPS, the following weights given in Table 5.1 are used for the LP relaxation. Note that, the upper bounds

obtained from LP relaxation of each criterion constitute the components of ideal point.

Table 5.1 Weights to seed the initial population of FWEA_KP and EMAPS

Weights		
m=2	*M*=3	*m*=4
[1.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0, 0.0]
[0.0, 1.0]	[0.0, 1.0, 0.0]	[0.0, 1.0, 0.0, 0.0]
[0.5, 0.5]	[0.0, 0.0, 1.0]	[0.0, 0.0, 1.0, 0.0]
	[1/3, 1/3, 1/3]	[0.0, 0.0, 0.0, 1.0]
		[0.25, 0.25, 0.25, 0.25]

Table 5.2 shows the parameter settings of FWEA_KP and EMAPS for MOKP, which are in accordance with the study of Zitzler et al. (1999). We named the problem instances as *m*KP_*J* for convenience, where *m* denotes the number of knapsacks and *J* denotes the number of items.

Table 5.2 Parameter settings of FWEA_KP and EMAPS for MOKP

Parameters	**2KP_750**	**3KP_750**	**4KP_750**
POPSIZE 1	200	240	280
POPSIZE 2	250	300	350
# of function evaluations	125000	150000	175000

* p_c – crossover probability (one point crossover), 0.80
* p_m – mutation probability (bit wise mutation), 0.01

The following remarks should be noted regarding the implementation of EMAPS to the multiple constraint problem: We use the genetic operators and parameter settings as have been stated in Köksalan and Phelps (2006); however, repair operator is replaced with the one proposed in section 5.2. Although EMAPS is designed for generating part of the efficient frontier, we do not restrict the weight space for fair comparison purpose. As a reminder, in EMAPS, 2/3 of the initial population is seeded by applying the greedy approach on the single criteria problem, which is formed by linear aggregation of criteria with different weights.

We extended this heuristic in multiple knapsacks case as well and we perform both experiments with the greedy based seeding mechanism of EMAPS and the seeding mechanism that we suggest above separately. As have been suggested in EMAPS, the algorithm is run for different levels of α parameter that controls the balance between average and minimum strength in the raw-fitness function (see section 3.2), and the resulting populations are combined into a union set. For this purpose, we use 5 levels of α, which are given as 0.0, 0.25, 0.50, 0.75 and 1.0 in Köksalan and Pamuk (2006) in our experiments. For a fair comparison, number of function evaluations performed is also partitioned into the different α levels, i.e. termination condition of EMAPS is set to $\frac{\#\text{ of function evaluation}}{5}$.

We performed some preliminary experiments that lead us to choose the best α level of FWEA_KP for the rest of the study. The box plots illustrated in Figure 5.1 for 2KP_750 problem reveal that although there is not a substantial difference, α level of 0.1 is best in *proximity indicator*. The reason for observing such a low α level is probably because every member is sufficiently better than others with own favorable weights especially towards the termination that it emphasizes the minimum strength portion of the raw-fitness score to take an advantage.

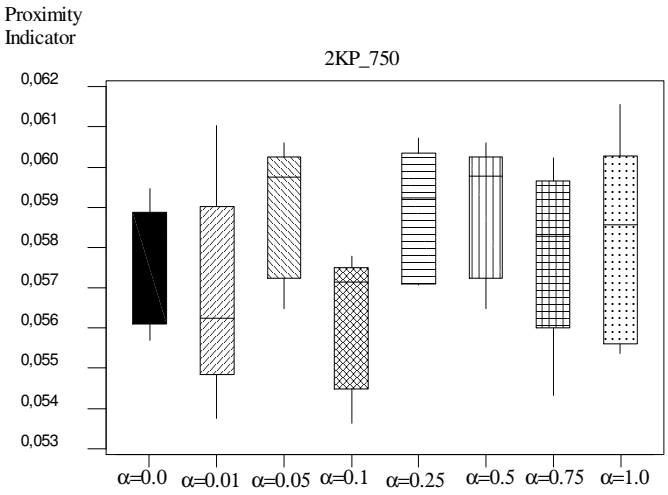

Figure 5.1 Box plots of FWEA_KP results for different α levels

Average, best and worst *Proximity indicator* results of algorithms are provided in Table 5.3. We can understand from the table that FWEA_KP is closer to the Pareto frontier than EMAPS, SPEA2 and NSGAII. This may also be observed in Figure 5.2. It seems that performances of SPEA2 and EMAPS are not so different from each other and better than NSGAII. Compared with original greedy seeding mechanism of EMAPS, *m*+1 seeded EMAPS performs somewhat better. This may be due to the fact that the seed solutions obtained from LP relaxation is superior to the ones from greedy approach and closely located to the critical portions of the Pareto frontier. Our inference is that although EMAPS performs well, since it is designed for single constrained MOKP, it needs more calibration. Hereafter, we continue experiments on FWEA_KP, SPEA2 and NSGAII. We also performed more experiments on 2KP_750 problem with FWEA_KP by using more seed solutions. For this purpose, we use 11 equally spaced seed vectors of which each component changes with 0.1 increments such as [0.0, 1.0], [0.1, 0.9], …, [1.0, 0.0]. The results reveal that as the number of seeds increases, the performance of the FWEA_KP improves further.

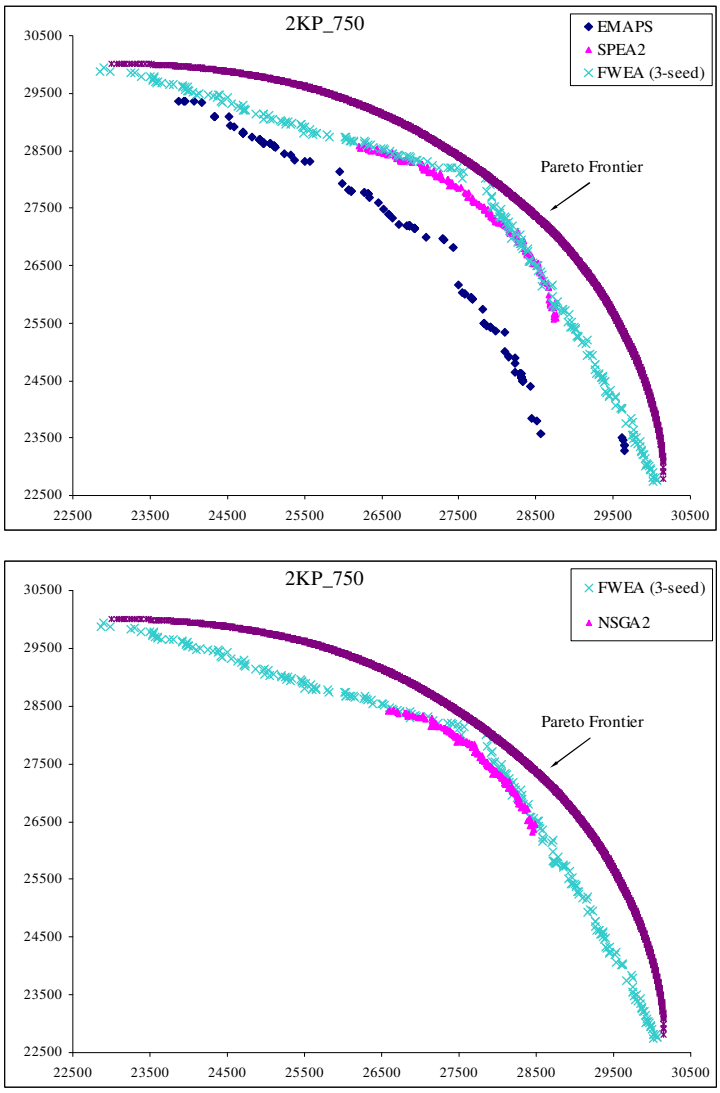

Figure 5.2 Efficient solutions of algorithms for a single run

Table 5.3 *Proximity indicator* results of FWEA_KP, EMAPS, SPEA2 and NSGAII

Problem		**2KP_750 Proximity to Pareto Frontier**	**3KP_750 Proximity to NDUS**	**4KP_750 Proximity to NDUS**
SPEA2	Avg.	0.11265	0.17270	0.20422
	Best	0.10001	0.13259	0.19246
	Worst	0.12104	0.18883	0.24616
NSGAII	Avg.	0.14552	0.18338	0.21890
	Best	0.12489	0.15083	0.19835
	Worst	0.15887	0.19982	0.24284
FWEA_KP (*m*+1-seed)	Avg.	0.05684	0.01972	0.01017
	Best	0.05363	0.01389	0.00086
	Worst	0.06120	0.02456	0.05053
EMAPS (*m*+1-seed)	Avg.	0.10548		
	Best	0.09629	-	-
	Worst	0.12450		
FWEA_KP (11-seed)	Avg.	0.02099		
	Best	0.01852	-	-
	Worst	0.02180		
EMAPS	Avg.	0.11949		
	Best	0.10795	-	-
	Worst	0.14627		

* Lower value is better according to this performance measure

Regarding *hypervolume indicator* presented in Table 5.4, our inferences are parallel to the ones proposed for proximity indicator except EMAPS, which has the lowest hypervolume according to 2KP_750. Another observation is that as the number of criteria increases, the difference among the performances of algorithms becomes clearer in favour of FWEA_KP. We also realize an interesting fact that as the number of criteria increases, hypervolume dominated by each algorithm drastically decreases. In this case, since we do not know the Pareto frontier, we could not find the hypervolume dominated by it and make a comparison.

The results regarding the percent contribution of algorithms to NDUS are given in Table 5.5. According to this table, FWEA_KP makes the highest contribution to NDUS by far followed by SPEA2. Moreover, as the number of criteria increases, contribution of the FWEA_KP increases while the contributions of other algorithms decrease, which implies the higher performance of FWEA_KP as the number of criteria increases.

Table 5.4 *Hypervolume indicator* results of FWEA_KP, EMAPS, SPEA2 and NSGAII

Problem		2KP_750	3KP_750	4KP_750
SPEA2	Avg.	0.60083	0.33185	0.03435
	Best	0.60686	0.34111	0.03887
	Worst	0.58695	0.32192	0.03117
NSGAII	Avg.	0.57647	0.31747	0.01941
	Best	0.58789	0.32113	0.01998
	Worst	0.56223	0.31257	0.01458
FWEA_KP (*m*+1-seed)	Avg.	0.69189	0.40911	0.05915
	Best	0.69418	0.41463	0.06042
	Worst	0.68444	0.40419	0.05603
EMAPS (*m*+1-seed)	Avg.	0.59370		
	Best	0.61686	-	-
	Worst	0.55501		
FWEA_KP (11-seed)	Avg.	0.75173		
	Best	0.75503	-	-
	Worst	0.74829		
EMAPS	Avg.	0.54036		
	Best	0.54463	-	-
	Worst	0.53655		

* Note that hypervolume of the efficient frontier is 0.78316 for 2KP_750 problem. Higher value is better according to this performance measure.

Table 5.5 Percent contribution of FWEA_KP, SPEA2 and NSGAII to the NDUS.

Problem		3KP_750	4KP_750
SPEA2	Avg.	24.6	6.7
	Best	32.9	12.6
	Worst	11.9	1.1
NSGAII	Avg.	10.9	2.42
	Best	16.0	10.7
	Worst	5.2	0.0
FWEA_KP (*m*+1-seed)	Avg.	64.6	90.9
	Best	72.4	98.6
	Worst	54.4	76.9

Up to this point, although the performance of FWEA is satisfactory, we perform some further experiments in higher number of function evaluations, i.e. 480000. Our *proximity* and *hypervolume* indicator results are provided in Table 5.6 and 5.7 respectively for 2KP_750. According to these tables, as the number of function evaluations increases, improvements occur in the performances of algorithms. It should also be noted that the performance measures of FWEA_KP recorded for

125,000 function evaluations have still been better than the ones of SPEA2 and NSGAII for 480,000 function evaluations.

Table 5.6 *Proximity indicator* results of FWEA_KP, SPEA2 and NSGAII for 2KP_750 and 480,000 function evaluations.

Problem		**2KP_750**
SPEA2	Avg.	0.08739
	Best	0.07738
	Worst	0.09487
NSGAII	Avg.	0.11326
	Best	0.09943
	Worst	0.11929
FWEA_KP (*m*+1-seed)	Avg.	0.04755
	Best	0.04423
	Worst	0.05506

Table 5.7 *Hypervolume indicator* results of FWEA_KP, SPEA2 and NSGAII for 2KP_750 and 480000 function evaluations.

Problem		**2KP_750**
SPEA2	Avg.	0.65479
	Best	0.66478
	Worst	0.64953
NSGAII	Avg.	0.62999
	Best	0.63930
	Worst	0.61636
FWEA_KP (*m*+1-seed)	Avg.	0.70729
	Best	0.71236
	Worst	0.69455

5.5 DISCUSSION

In this chapter, we adapted our FWEA to approximate the Pareto frontier of MOKP. In addition to distinctive aspects of the algorithm, such as favorable weight mechanism, fitness function etc, we also seed the initial population with some high quality solutions in order to enhance the performance. According to our experimental results, FWEA_KP outperforms EMAPS, SPEA2 and NSGAII. Here we should mention that seeding mechanism might also have an impact on the

performance of SPEA2 and NSGAII. We only make comparisons considering the given results in the literature.

Another future research direction regarding the performance measure should be noted as well. During the performance evaluation of our algorithm, we had to deal with some drawbacks due to unknown Pareto frontier of the test problems. We may be able to find ways to better approximate the efficient frontier or generate it exactly for larger size problems in order to evaluate the performance of the algorithms better.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this thesis, we presented a new multi-objective evolutionary algorithm (FWEA) and tested its performance of algorithm on some test problems given in the literature. Fitness function, where Tchebycheff distance function is considered, and seeding mechanism are distinctive properties of FWEA. Non-dominated sorting concept and crowding measure also contribute to the performance of FWEA.

The performance of the algorithm has been tested on 2, 3 and 5 objective continuous test problems. These test problems have special complexities that a multi-objective evolutionary algorithm has to deal with. We compare the results of our algorithm with NSGAII, which has been shown to perform well in the literature. According to experiments, our algorithm performs quite well.

We also applied FWEA to some multi criteria combinatorial optimization problems. One of these combinatorial problems is bi-criteria p-hub location problem. We considered two bicriteria uncapacitated multiple allocation p-hub location problems (Bp-HLP1 and Bp-HLP2). In both of the problems, our aim is to define the location of the hubs and then allocate the nodes to these hubs. We propose two evolutionary algorithms adapted from FWEA for location and allocation decisions separately. FWEA_loc has a novel property that we fit an Lq distance function passing through some efficient (or approximately efficient) allocations of a set of p hubs and the performance of FWEA_alloc algorithm is enhanced by restricting the criteria space.

The other multiple criteria combinatorial optimization problem that we consider in this thesis is multi-objective knapsack problem (MOKP). Our EA for MOKP, which is called as FWEA_KP, is adapted from original FWEA. We carried out experiments on test data given in the literature and compared the results with EMAPS, SPEA2 and NSGAII. According to our experiments, FWEA_KP yields promising results.

The main contribution of this thesis is in the field of developing a multi-objective evolutionary algorithm and applying it to some multi-objective continuous and combinatorial optimization problems. Application of our algorithm to other combinatorial problems is one of the future research directions.

In this study, we observed the contribution of few seed solutions on the performance of our algorithm. We can find several good solutions for most problems and introduce into the initial population. Investigating effect of seeding on different problems and algorithms may be another future research direction.

A future research direction regarding the performance measure should also be noted as follows. In many cases, since we do not know the Pareto frontier of the problems, we may not be able to estimate whether our solutions are close to Pareto frontier or not. There is a need to find ways to better approximate the Pareto frontier or generate it exactly for reasonably large size problems in order to evaluate the performance of the algorithms better. Additionally, there are difficulties in finding appropriate performance measures to evaluate the algorithms in terms of both proximity and diversity goals. Developing new performance measures should be considered as a future research direction.

Another future research direction is related with approximating only those portions of the Pareto frontier preferred by decision maker. This may be possible by interacting with the decision maker during the algorithm or at the beginning. He/she may provide some information that will restrict the weight space and thus will guide the algorithm to the preferred regions of the Pareto frontier.

During the process of our algorithms, some levels of parameter settings and some type of strategies may be more effective at some stage of the algorithms. To take into consideration these issues as a future research, we may use dynamic or adaptive parameter values and strategies.

REFERENCES

Abdinnour-Helm, S. (1998), "A Hybrid Heuristic for the Uncapacitated Hub Location Problem," *European Journal of Operational Research*, Vol.106, pp.489-499.

Allenson, R. (1992), "Genetic Algorithms with gender for Multifunction Optimization," *Technical Report*, No.92-01, Edinburgh Parallel Computing Centre, Edinburgh, Scotland.

Alp, O., E. Erkut and Z. Drezner (2003), "An Efficient Genetic Algorithm for the p-median Problem," *Annals of Operations Research*, Vol.122, pp.21-42.

Aykin, T. (1994), "Lagrangian Relaxation Based Approaches to Capacitated Hub-And-Spoke Network Design Problem," *European Journal of Operational Research*, Vol.79, No.3, pp.501-523.

Aykin, T. (1995), "The Hub Location and Routing Problem," *European Journal of Operational Research*, Vol.83, No.1, pp.200-219.

Beume, N., B. Naujoks and M. Emmerich (2006), "SMS-EMOA: Multiobjective Selection Based on Dominated Hypervolume," to appear in *European Journal of Operational Research*.

Bosman, P.A.N and Thierens, D. (2003), "The Balance Between Proximity and Diversity in Multiobjective Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, Vol.7, No.2, pp.174-188.

Campbell, J.F. (1994), "Integer Programming Formulations of Discrete Hub Location Problems," *European Journal of Operational Research*, Vol.72, No.2, pp.387-405.

Campbell, J.F. (1996), "Hub Location and p-Hub Median Problem," *Operations Research*, Vol.44, pp.923-935.

Campbell, J.F., A.T. Ernst and M. Khrishnamoorthy (2002), "Facility Location, Applications and Theory," In Drezner, Z. and Hamacher, H.W., editors, Chapter 12: Hub Location Problems, pp.373-407, Springer-Verlag, Heidelberg.

Chang, W. C., Sutcliffe, A. G. and R., Neville (2003), "A distance function-based multi-objective evolutionary algorithm," in J. Foster (Ed.), *Proceedings: Genetic*

and Evolutionary Computation 2003 Conference (GECCO-2003), Chicago, pp.47-53.

Chen, L and H. Choi (2001), "Approximation Algorithms for Data Distribution with Load Balancing of Web Servers," In Proceedings of the 2001 IEEE International Conference on Cluster Computing (CLUSTER'01), pp.274-281.

Coello, C.A.C. (1999), "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques," *Knowledge and Information Systems. An International Journal*, Vol.1, No.3, pp.269-308.

Costa, M.G., E.M. Captivo and J. Climaco (2005), "Capacitated Single Allocation Hub Location Problems: A Bicriteria Approach," Working paper No:7/2005, Centro de Investigação Operacional da Universidade de Lisboa.

Çamlar, O. (2005), *Solution to Multi-Objective Hub Location Problem Using Evolutionary Algorithm – An Application to PTT Network*, MSc. Thesis, Department of Industrial Engineering, Middle East Technical University, Ankara, Turkey.

Çetiner, S. (2003), *An Iterative Hub Location and Routing Problem for Postal Delivery Systems*, MSc. Thesis, Department of Industrial Engineering, Middle East Technical University, Ankara, Turkey.

Da Silva, C.G., J. Climaco and J. Figueira (2004), "A Scatter Search Method for the Bi-criteria Multi-Dimensional {0-1} Knapsack Problem using Surrogate Relaxation," *Journal of Mathematical Modeling and Algorithms*, Vol.3, pp.183-208.

Deb, K. (1999), "Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design," in *Proceedings of Evolutionary algorithms in Engineering and Computer Science (EUROGEN'' 99).*

Deb, K. and R.B. Agrawal (1995), "Simulated Binary Crossover for Continuous Search Space," *Complex Syst.*, Vol.9, pp.115-148.

Deb, K., L. Thiele, M. Laumanns and E. Zitzler (2001), "Scalable test problems for evolutionary multi-objective optimization," KanGAL Report 2001001, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology, Kanpur, India.

Deb, K.**,** P**.** Amrit, S. Agarwal and T. Meyarivan (2002), "A Fast and Elitist Multi-Objective Genetic Algorithm-NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No.2, pp.182-197.

DeJong, K.A. (1975), An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D.Thesis, *Dissertation Abstracts International*, Vol.36, University of Michigan, Ann Arbor.

Ebery, J., M. Khrishnamoorthy, A. Ernst and N. Boland (2000), "The Capacitated Multiple Allocation Hub Location Problem: Formulations and Algorithms," *European Journal of Operational Research*, Vol.120, pp.614-631.

Ehrgott, M. and X. Gandibleux (2000), "An Annotated Bibliography of Multiobjective Combinatorial Optimization," *OR Spectrum*, Vol.22, pp.425-460.

Ernst, A.T and M. Khrishnamoorthy (1998a), "An Exact Solution Approach Based on Shortest-Paths for p-Hub Median Problems," *INFORMS Journal on Computing*, Vol.10, No.2, pp.149-162.

Ernst, A.T and M. Khrishnamoorthy (1998b), "Exact and Heuristic Algorithms for the Uncapacitated Multiple Allocation p-Hub Median Problem," *European Journal of Operational Research*, Vol.104, No.1, pp.100-112.

Fonseca, C.M and P.J. Fleming (1993), "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion, and Generalization," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp.416-423.

Fonseca, C.M and P.J. Fleming (1995), "An Overview of Evolutionary Algorithms in Multiobjective Optimization," *Evolutionary Computation*, Vol.3, No.1, pp.1-16.

Gandibleux, X. and M. Ehrgott (2005), "1984-2004 – 20 Years of Multiobjective Metaheuristics. But What About the Solution of Combinatorial Problems with Multiple Objectives?," *Lecture Notes in Computer Science*, Vol.3410, pp.33-46.

Garey, M.R. and D.S. Johnson (1979), Computers and Intractability – A guide to the Theory of NP-Completeness. Freeman, San Francisco.

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

Goldberg, D.E. and J. Richardson (1987), "Genetic Algorithms with Sharing for Multimodal Function Optimization," in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pp.41-49.

Guo, X., G. Yang and Z. Wu (2006), "A Hybrid-Fine Tuned Multi-Objective Memetic Algorithm," *IEICE Trans. Fundamentals*, Vol.E89-A, No.3, pp.790-797.

Gurney, K. (1997), An Introduction to Neural Networks, Univ. College of London Press.

Haimes, Y.Y., L.S. Lasdon and D.A. Wismer (1971), "On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization," *IEEE Transactions on Systems, Man and Cybernetics*, Vol.1, No.3, pp.296-297.

Hajela, P. and C.Y. Lin (1992), "Genetic Search Strategies in Multicriterion Optimal Design," *Structural Optimization*, Vol.4, pp. 99-107.

Hajela, P. and C.Y. Lin (1993), "Genetic Algorithms in structural Topology Optimization," in *Proceedings of the NATO Advanced Research Workshop on Topology Design of structures*, pp.117-133.

Hansen, M.P (1997), "Tabu Search for Multiobjective Optimization," *In Proceedings of the 13th International Conference on Multiple Criteria Decision Making ({MCDM}'97), Cape Town, South Africa.*

Hansen, P.M. and A. Jaszkiewicz (1998), "Evaluating the Quality of Approximations to the Non-Dominated Set," IMM Technical Report, 1998-7.

Haykin, S.S. (1998), Neural Networks: A Comprehensive Foundation, 2nd edition, Prentice Hall.

Horn, J. (1997), Handbook of Evolutionary Computation, *chapter Multicriterion Decision Making*, F1.9:1-F1.9:15, Back et al., Vol.1, No.19.

Horn, J., N. Nafploitis, and D. Goldberg (1994), "A Niched Pareto Genetic Algorithm for Multi-Objective Optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp.82-87.

Janic, M. (2002), The Problem of Airport Capacity: Case of London Heathrow Airport, OTB Research Institute for Housing, Urban and Mobility Studies, Delft University of Technology.

Jaskiewicz, A. (2001), Multiple Objective Metaheuristic Algorithms for Combinatorial Optimization, Politechnika Poznanska, Poznan.

Jaskiewicz, A. (2004), "On the Computational Efficiency of Multiple Objective Metaheuristics. The knapsack problem case study," *European Journal of Operational Research*, Vol.158, pp.418-433.

Jones, D.F., S.K. Mirrazavi, and M. Tamiz (2002), "Multi-Objective Meta-Heuristics: An Overview the Current State-of-the-Art," *European Journal of Operational Research*, Vol.137, pp.1-9.

Kara, B. (1999), *Modeling and Analysis Issues in Hub Location Problem*, PhD. Thesis, Department of Industrial Engineering, Bilkent University, Ankara, Turkey.

Knowles, J. (2005), "ParEGO: A Hybrid Algorithm with On-Line Landscape Approximation for Expensive Multi-objective Optimization Problems," *IEEE Transactions on Evolutionary Computation*, Vol.10, No.1, pp.50-66.

Knowles, J.D and D.W. Corne (2000), "Approximating the Non-dominated front using the Pareto Archived Evolution Strategy," *Evolutionary Computation*, Vol.8, No.2, pp. 149-172.

Knowles, J.D and D.W. Corne (2003), "Properties of an Adaptive Algorithm for Storing Non-dominated Vectors," *IEEE Transactions on Evolutionary Computation*, Vol.7, No.2, pp.100-116.

Koçkesen, K. (2004), *Evolutionary Algorithms for Deterministic and Stochastic Unconstrained Function Optimization*, MSc. Thesis, Middle East Technical University, Ankara, Turkey.

Köksalan, M. (1999), "A Heuristic Approach to Bicriteria Scheduling," *Naval Research Logistics*, Vol.46, pp.777-789.

Köksalan M. and S. (Pamuk) Phelps (2006), "An Evolutionary Metaheuristic for Approximating Preference-Non-dominated Solutions," *INFORMS Journal on Computing*, forthcoming.

Marin, A., L. Canovas and M. Landete (2006), "New Formulations for the Uncapacitated Multiple Allocation Hub Location Problem," *European Journal of Operational Research*, Vol.172, No.1, pp.274-292.

Marler, R.T. and J.S. Arora (2004), "Survey of Multi-Objective Optimization Methods for Engineering," *Struct Multidisc Optim*, Vol.26, pp.369-395.

Miettinen, K. (1999), Nonlinear Multiobjective Optimization, Kluwer Academic Publishers.

Newell, G.F. (1979), "Airport Capacity and Delays," *Transportation Science*, Vol.13, No.3, pp.201-241.

Nozick, L.K. and M.A. Turnquist (2001), "Inventory, Transportation, Service Quality and the Location of Distribution Centers," *European Journal of Operational Research*, Vol.129, No.2, pp.362-371.

O'Kelly, M.E. (1987), "A Quadratic Integer Program for the Location of Interacting Hub Facilities," *European Journal of Operational Research*, Vol.32, No.3, pp.393-404.

Perez, M., F. Almeida and J.M. Moreno-Vega (1998), "Genetic Algorithm with Multistart Search for the p-Hub Median Problem," *In Proceedings of 24th IEEE Euromicro Conference*, pp. 702-707.

Phelps (Pamuk) S. and M. Köksalan (2003), "An Interactive Evolutionary Metaheuristic for Multiobjective Combinatorial Optimization," *Management Science*, Vol.49, No.2, pp.1726-1738.

Schaffer, J.D. (1984), *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*, Ph.D. Thesis, Vanderbilt University, Nashville.

Skorin-Kapov, D. and J. Skorin-Kapov (1994), "On Tabu Search for the Location of Interacting Hub Facilities," *European Journal of Operational Research*, Vol.73, No.3, pp.502-509.

Skorin-Kapov, D., J. Skorin-Kapov and M.E. O'Kelly (1996), "Tight Linear Programming Relaxations of Uncapacitated p-Hub Median Problems," *European Journal of Operational Research*, Vol.94, No.3, pp.582-593.

Sohn, J. and S. Park (1998), "Efficient Solution Procedure and Reduced Size Formulations for p-Hub Location Problems," *European Journal of Operational Research*, Vol.108, No.1, pp.118-126.

Srinivas, N. and K. Deb (1994), "Multiobjective Function Optimization Using Non-dominated Sorting Genetic Algorithm," *Evolutionary Computation Journal*, Vol.2, No.3, pp. 221-248.

Steuer, R.E. (1986), Multiple Criteria Optimization: Theory, Computation, and Application, John Wiley & Sons, Inc., New York.

Tamaki, H., H. Kita, and S. Kobayashi (1996), "Multiobjective Optimization by Genetic Algorithms: A Review," In *Proceedings of 1996 IEEE International Conference on Evolutionary Computation (ICEC'96),* IEEE*,* Piscataway, NJ, pp.517-522.

Topçuoğlu, H., F. Corut, M. Ermiş and G. Yılmaz (2005), "Solving the Capacitated Hub Location Problem Using Genetic Algorithms," Computers and Operations Research, Vol.32, pp.967-984.

Ulungu, E.L. and J. Teghem (1994), "Multi-objective Combinatorial Optimization Problems: A Survey," *Journal of Multi-Criteria Decision Analysis*, Vol.3, pp.83-104.

Ulungu, E.L., P.H. Teghem and D. Tuyttens (1999), "MOSA Method: A Tool for Solving Multiobjective Combinatorial Optimization Problems," *Journal of Multi-Criteria Decision Analysis*, Vol.8, pp.221-236.

Vianna, D.S. and J.E.C. Arroyo (2004), "A GRASP Algorithm for the Multi-Objective Knapsack Problem," *In Proceedings of the XXIV International Conference of the Chilean Computer Science Society (SCCC'04).*

Veldhuizen, D.A.V. and G.B. Lamont (2000), "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art," *Evolutionary Computation*, Vol.8, No.2, pp.125-147.

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

Zitzler, E, K. Deb, and L. Thiele (2000), "Comparison of Multiobjective Evolutionary Algorithms," *Evolutionary Computation*, Vol.8, No.2, pp.173-195.

Zitzler, E., M. Laumanns, and L. Thiele (2002), "SPEA2: Improving the Strength Pareto Evolutionary Algorithm, " *TIK-Report*, No.103, Swiss Federal Institute of Technology, Switzerland.

Zitzler, E., M. Laumanns, L. Thiele, Fonseca, C.M, and V.G., Fonseca (2003), "Performance Assessment of Multiobjective Optimizers: An Analysis and Review," *IEEE Transactions on Evolutionary Computation*, Vol.7, No.2, pp.117-132.

Zitzler, E. and S. Künzli (2004), "Indicator-based Selection in Multiobjective Search," in Xin Yao et al. (editors), Parallel Problem Solving from Nature - PPSN VIII, Springer-Verlag, Lecture Notes in Computer Science, Vol. 3242, pp. 832-842, Birmingham, UK.

APPENDIX A

RESULTS FOR CONTINUOUS TEST PROBLEMS

Table A.1 *Reverse Proximity Indicator* and C measure results for 10 runs

Run no	$D_{X^{Pareto} \to X^{EMAPSII}}$	C(EMAPSII,NSGAII)	$D_{X^{Pareto} \to X^{NSGAII}}$	C(NSGAII,EMAPSII)
Test Problem ZDT 1 (Heuristic Seeding)				
1	0.00465	0.17	0.00499	0.00
2	0.00436	0.26	0.00497	0.00
3	0.00431	0.17	0.00515	0.00
4	0.00435	0.19	0.00468	0.00
5	0.00474	0.17	0.00508	0.00
6	0.00445	0.19	0.00524	0.00
7	0.00434	0.14	0.00488	0.00
8	0.00469	0.12	0.00464	0.00
9	0.00419	0.16	0.00459	0.00
10	0.00456	0.13	0.00505	0.00
Test Problem ZDT 1 (Perfect Seeding)				
1	0.004549	0.22	0.00499	0.00
2	0.004296	0.18	0.00497	0.00
3	0.004469	0.10	0.00515	0.01
4	0.004592	0.14	0.00468	0.00
5	0.004396	0.15	0.00508	0.00
6	0.004602	0.18	0.00524	0.00
7	0.005228	0.20	0.00488	0.00
8	0.004454	0.10	0.00464	0.00
9	0.004358	0.17	0.00459	0.00
10	0.005068	0.24	0.00505	0.00
Test Problem ZDT 2 (Heuristic Seeding)				
1	0.00452	0.30	0.00523	0.00
2	0.00447	0.28	0.00495	0.00
3	0.00434	0.23	0.00517	0.00
4	0.00431	0.24	0.00533	0.00
5	0.00447	0.10	0.00472	0.00
6	0.00447	0.15	0.00492	0.00
7	0.00454	0.21	0.00473	0.00
8	0.00438	0.25	0.00497	0.00
9	0.00452	0.20	0.00497	0.00
10	0.00450	0.16	0.00475	0.00
Test Problem ZDT 2 (Perfect Seeding)				
1	0.00473	0.26	0.00523	0.00
2	0.00438	0.31	0.00495	0.00
3	0.00447	0.21	0.00517	0.00
4	0.00430	0.26	0.00533	0.00
5	0.00438	0.19	0.00472	0.00
6	0.00446	0.11	0.00492	0.00
7	0.00434	0.19	0.00473	0.00
8	0.00420	0.21	0.00497	0.00
9	0.00427	0.21	0.00497	0.00
10	0.00431	0.11	0.00475	0.00

Table A.1 (Continued) *Reverse Proximity Indicator* and C measure results

Run no	$D_{X^{Pareto} \to X^{EMAPSII}}$	C(EMAPSII,NSGAII)	$D_{X^{Pareto} \to X^{NSGAII}}$	C(NSGAII,EMAPSII)
		Test Problem ZDT 3 (Heuristic Seeding)		
1	0.00499	0.18	0.00570	0.01
2	0.00505	0.26	0.00536	0.00
3	0.00475	0.19	0.00535	0.00
4	0.00498	0.26	0.00528	0.00
5	0.00514	0.17	0.00535	0.02
6	0.00521	0.09	0.00517	0.00
7	0.00480	0.21	0.00531	0.00
8	0.00515	0.23	0.00570	0.00
9	0.00531	0.20	0.00554	0.00
10	0.00497	0.17	0.00506	0.00
		Test Problem ZDT 3 (Perfect Seeding)		
1	0.00508	0.14	0.00570	0.00
2	0.00495	0.21	0.00536	0.01
3	0.00498	0.20	0.00535	0.00
4	0.00548	0.25	0.00528	0.00
5	0.00497	0.17	0.00535	0.01
6	0.00518	0.11	0.00517	0.00
7	0.00486	0.25	0.00531	0.00
8	0.00494	0.21	0.00570	0.00
9	0.00528	0.21	0.00554	0.01
10	0.00512	0.15	0.00506	0.01
		Test Problem ZDT 4 (Heuristic Seeding)		
1	0.00685	0.01	0.00578	0.22
2	0.00643	0.15	0.00617	0.00
3	0.00605	0.03	0.00511	0.11
4	0.00489	0.51	0.01389	0.00
5	0.03246	0.00	0.00478	0.38
6	0.00889	0.00	0.00615	0.40
7	0.00512	0.62	0.00793	0.00
8	0.00765	0.00	0.00693	0.46
9	0.00613	0.33	0.00762	0.00
10	0.00767	0.00	0.00675	0.14
		Test Problem ZDT 4 (Perfect Seeding)		
1	0.00517	0.35	0.00578	0.00
2	0.00529	0.54	0.00617	0.00
3	0.00494	0.24	0.00511	0.00
4	0.00512	0.57	0.01389	0.00
5	0.00520	0.14	0.00478	0.00
6	0.00489	0.62	0.00615	0.00
7	0.00514	0.75	0.00793	0.00
8	0.00530	0.26	0.00693	0.00
9	0.00527	0.78	0.00762	0.00
10	0.00534	0.57	0.00675	0.00

Table A.1 (Continued) *Reverse Proximity Indicator* and C measure results

Run no	$D_{X^{Pareto} \to X^{EMAPSII}}$	C(EMAPSII,NSGAII)	$D_{X^{Pareto} \to X^{NSGAII}}$	C(NSGAII,EMAPSII)
Test Problem ZDT 6 (Heuristic Seeding)				
1	0.08460	0.96	0.12895	0.00
2	0.08466	0.90	0.12909	0.00
3	0.08422	0.89	0.12877	0.00
4	0.08443	0.92	0.12886	0.00
5	0.08421	0.93	0.12900	0.00
6	0.08465	0.89	0.12884	0.00
7	0.08463	0.85	0.12899	0.00
8	0.08414	0.91	0.12892	0.00
9	0.08395	0.80	0.12903	0.00
10	0.08430	0.82	0.12894	0.00
Test Problem ZDT 6 (Perfect Seeding)				
1	0.06628	1.00	0.12895	0.00
2	0.06624	1.00	0.12909	0.00
3	0.06626	1.00	0.12877	0.00
4	0.06633	1.00	0.12886	0.00
5	0.06635	1.00	0.12900	0.00
6	0.06631	1.00	0.12884	0.00
7	0.06626	1.00	0.12899	0.00
8	0.06631	1.00	0.12892	0.00
9	0.06623	1.00	0.12903	0.00
10	0.06630	1.00	0.12894	0.00
Test Problem DTLZ 1 - 3D (Heuristic Seeding)				
1	0.02449	0.08	0.02844	0.11
2	0.02483	0.05	0.02960	0.01
3	0.02421	0.99	0.30263	0.02
4	0.02331	0.46	0.05078	0.02
5	0.02481	0.18	0.03374	0.00
6	0.02507	0.09	0.02804	0.00
7	0.02628	0.00	0.03109	0.06
8	0.02322	0.14	0.03202	0.02
9	0.02668	0.01	0.03007	0.16
10	0.02477	0.02	0.02925	0.02
Test Problem DTLZ 1 – 3D (Perfect Seeding)				
1	0.02202	0.16	0.02844	0.00
2	0.02109	0.09	0.02960	0.00
3	0.02183	0.99	0.30263	0.00
4	0.02263	0.61	0.05078	0.00
5	0.02250	0.26	0.03374	0.00
6	0.02137	0.09	0.02804	0.00
7	0.02144	0.02	0.03109	0.00
8	0.02155	0.15	0.03202	0.00
9	0.02162	0.03	0.03007	0.02
10	0.02119	0.04	0.02925	0.01

Table A.1 (Continued) *Reverse Proximity Indicator* and C measure results

Run no	$D_{X^{Pareto} \to X^{EMAPSII}}$	C(EMAPSII,NSGAII)	$D_{X^{Pareto} \to X^{NSGAII}}$	C(NSGAII,EMAPSII)
Test Problem DTLZ 2 – 3D (Heuristic Seeding)				
1	0.05827	0.02	0.07109	0.01
2	0.05971	0.05	0.07002	0.02
3	0.05958	0.06	0.06935	0.01
4	0.05760	0.06	0.07759	0.00
5	0.05936	0.03	0.07025	0.00
6	0.05741	0.04	0.07062	0.02
7	0.05898	0.05	0.06923	0.01
8	0.05980	0.04	0.06890	0.01
9	0.05687	0.01	0.07225	0.00
10	0.05709	0.04	0.07627	0.00
Test Problem DTLZ 2 – 3D (Perfect Seeding)				
1	0.05964	0.02	0.07109	0.00
2	0.05721	0.03	0.07002	0.01
3	0.05937	0.05	0.06935	0.02
4	0.05779	0.03	0.07759	0.01
5	0.06013	0.04	0.07025	0.00
6	0.05955	0.02	0.07062	0.01
7	0.05755	0.01	0.06923	0.00
8	0.05892	0.06	0.06890	0.03
9	0.06125	0.04	0.07225	0.00
10	0.05904	0.05	0.07627	0.01
Test Problem DTLZ 1 - 5D (Heuristic Seeding)				
1	3.20153	0.71	24.69812	0.22
2	4.89244	0.81	10.71172	0.43
3	4.30637	0.25	13.34435	0.28
4	10.15749	0.75	5.71207	0.45
5	2.77150	0.93	20.34749	0.00
Test Problem DTLZ 1 – 5D (Perfect Seeding)				
1	0.06725	0.48	24.69812	0.00
2	0.04933	0.81	10.71172	0.00
3	0.05871	0.36	13.34435	0.00
4	0.05091	1.00	5.71207	0.00
5	0.05304	0.35	20.34749	0.00
Test Problem DTLZ 2 - 5D (Heuristic Seeding)				
1	0.16312	0.08	0.18777	0.00
2	0.16553	0.07	0.18204	0.00
3	0.16142	0.09	0.19852	0.00
4	0.16549	0.07	0.18527	0.01
5	0.16284	0.09	0.19247	0.00
Test Problem DTLZ 2 – 5D (Perfect Seeding)				
1	0.16349	0.15	0.18777	0.00
2	0.16050	0.07	0.18204	0.00
3	0.16119	0.12	0.19852	0.00
4	0.16093	0.07	0.18527	0.00
5	0.16539	0.13	0.19247	0.00

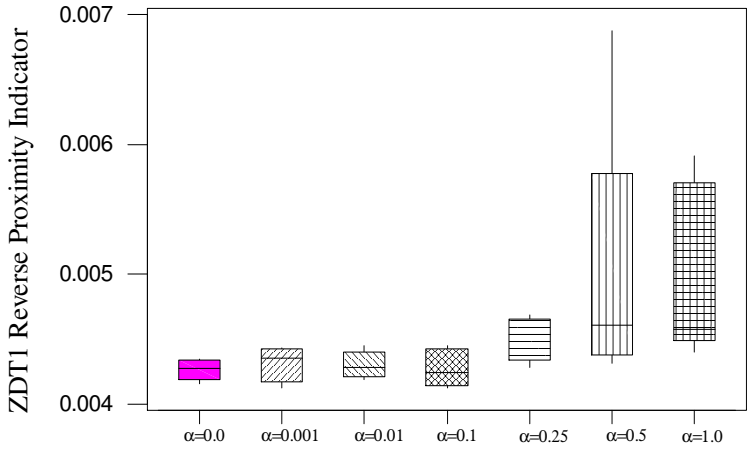

Figure A. 1 Box plots of reverse proximity indicator for different α parameters of ZDT1 problem.

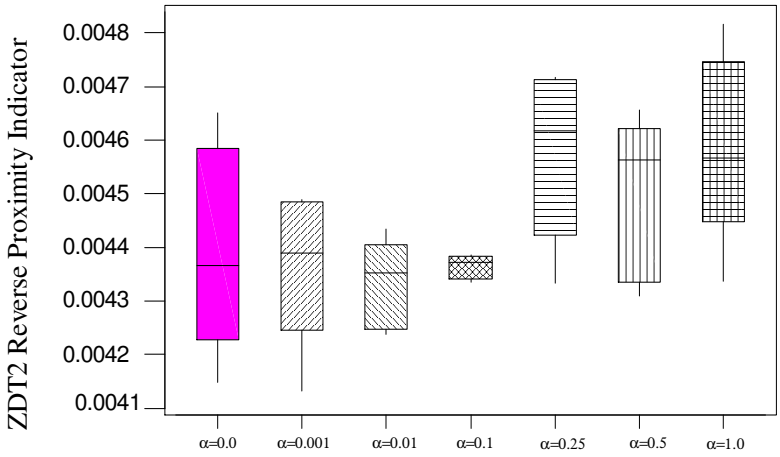

Figure A. 2 Box plots of reverse proximity indicator for different α parameters of ZDT2 problem.

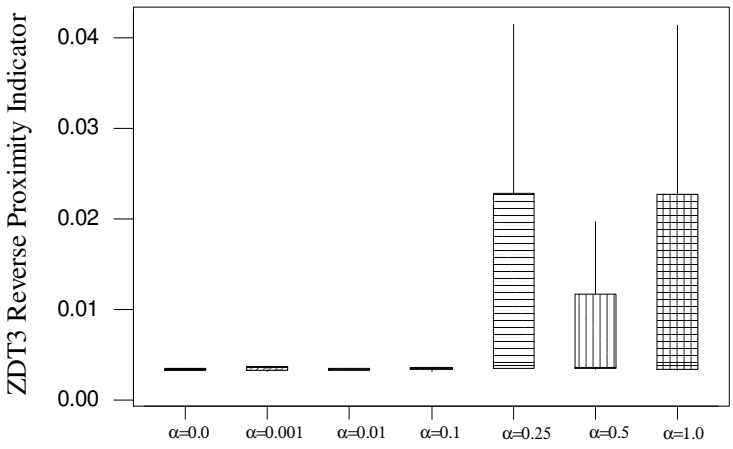

Figure A.3 Box plots of reverse proximity indicator for different α parameters of ZDT3 problem

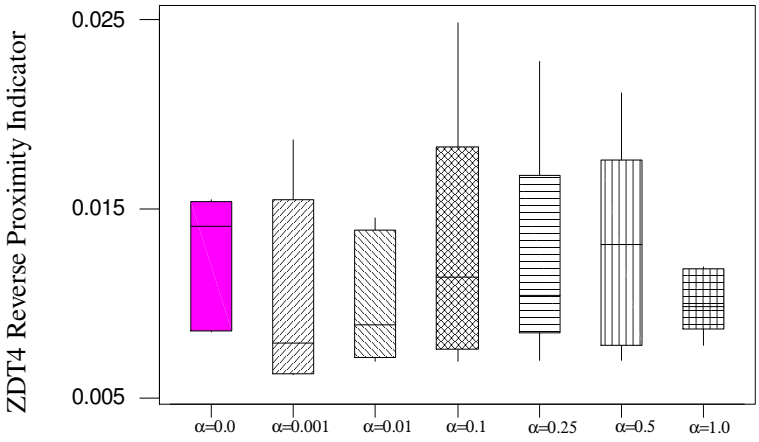

Figure A.4 Box plots of reverse proximity indicator for different α parameters of ZDT4 problem

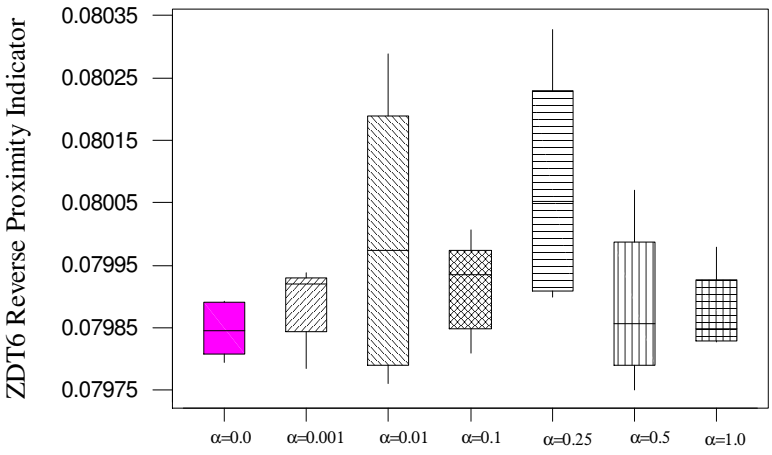

Figure A.5 Box plots of reverse proximity indicator for different α parameters of ZDT6 problem

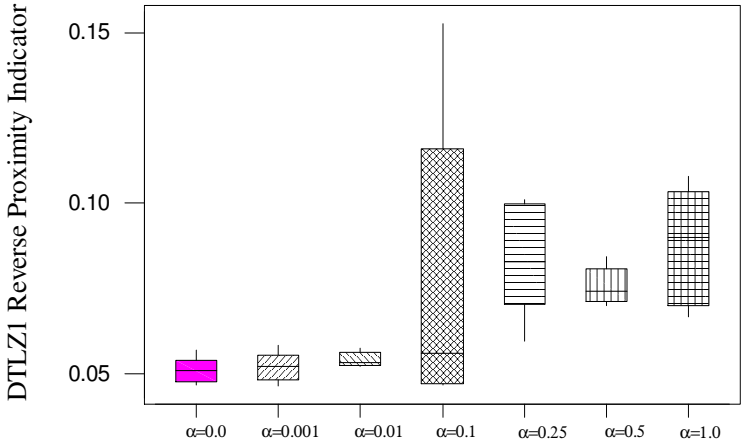

Figure A.6 Box plots of reverse proximity indicator for different α parameters of DTLZ1_3D problem

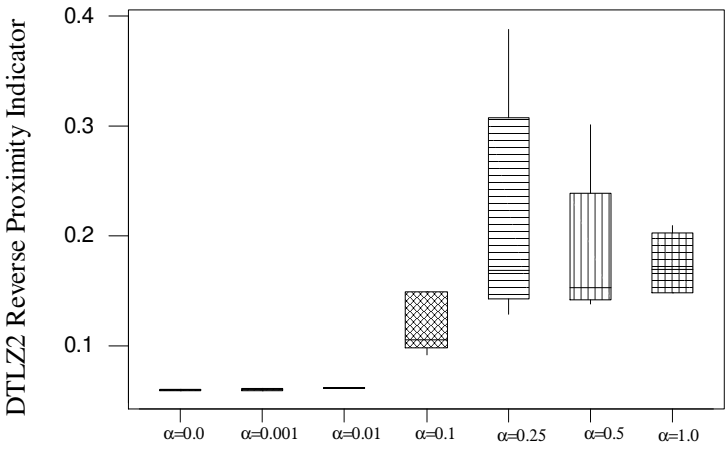

Figure A.7 Box plots of reverse proximity indicator for different α parameters of DTLZ2_3D problem

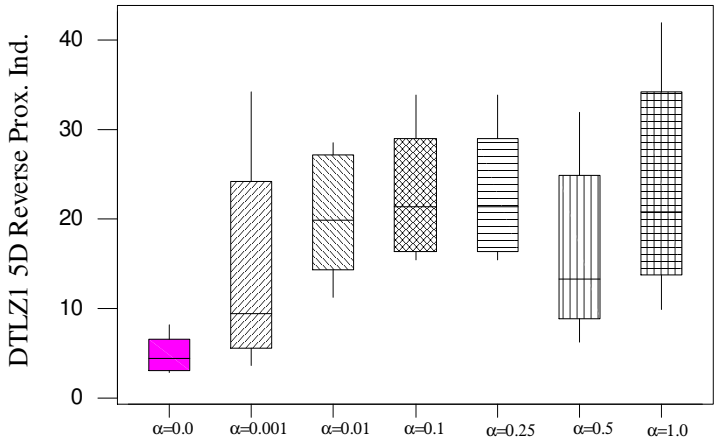

Figure A.8 Box plots of reverse proximity indicator for different α parameters of DTLZ1_5D problem

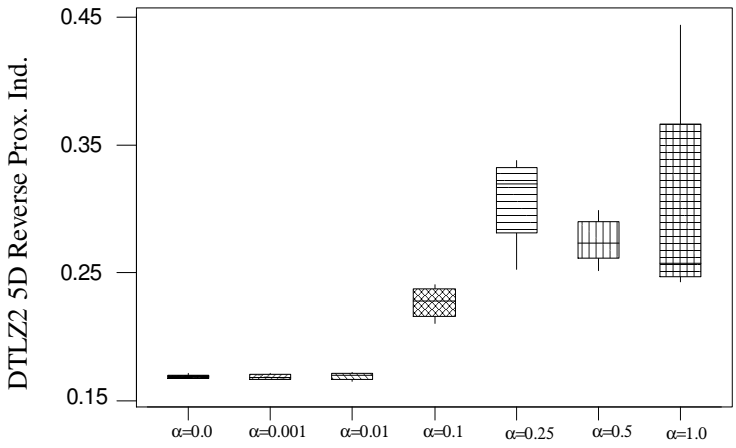

Figure A.9 Box plots of reverse proximity indicator for different α parameters of DTLZ2_5D problem

APPENDIX B

RESULTS FOR BI-CRITERIA HUB LOCATION PROBLEMS

Table B.1 List of most crowded 25 cities of Turkey according to decreasing order of population

Rank	City
0	Istanbul
1	Ankara
2	İzmir
3	Konya
4	Bursa
5	Adana
6	Antalya
7	Mersin
8	Şanlıurfa
9	Diyarbakir
10	Gaziantep
11	Manisa
12	Hatay
13	Kocaeli
14	Samsun
15	Balıkesir
16	Kayseri
17	Kahramanmaraş
18	Trabzon
19	Aydın
20	Erzurum
21	Ordu
22	Van
23	Denizli
24	Tokat

Table B.2 Beta analysis of hub location problem for AP data set set and Bp-HLP1

			Generation Number That All Efficient Hub Sets are Found				
n	**p**	**Beta**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
25	3	0.0	5000 / 1 *	5000 / 1	1000	5000 / 1	5000 / 1
		0.01	4500	1500	1000	4500	5000 / 1
		0.1	5000 / 1	500	1000	3000	2000
		0.3	2500	2500	1500	1000	4000
		0.5	500	1000	2000	2000	3000
		0.8	2500	1500	3000	1000	2000
		1.0	3000	4500	2500	1500	4000
25	5	0.0	5000 / 1	5000 / 0	5000 / 0	1500	5000 / 1
		0.01	5000 / 0	5000 / 1	5000 / 0	1500	5000 / 2
		0.1	5000 / 0	1500	5000 / 1	2500	1500
		0.3	5000 / 0	3500	5000 / 0	2500	4000
		0.5	5000 / 2	2500	3500	4500	5000 / 1
		0.8	5000 / 2	2000	3000	3000	2000
		1.0	2500	3500	5000 / 2	5000 / 2	5000

* Number of efficient hub sets found by FWEA_loc over 5000 generations

Table B.3 Beta analysis of hub location problem for AP data set and Bp-HLP2

n	**p**	**Beta**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
25	3	0.2	2216 *	895	3070	1038	2161
		0.5	4038	6235	6904	5123	558
		0.8	1358	847	1121	1101	1682
25	5	0.2	3832	10807	5030	18532	737
		0.5	26166	9971	7786	15887	2597
		0.8	4401	2849	1701	6369	2760

* Generation number that all efficient hub sets are found by FWEA_loc over 30000 generations

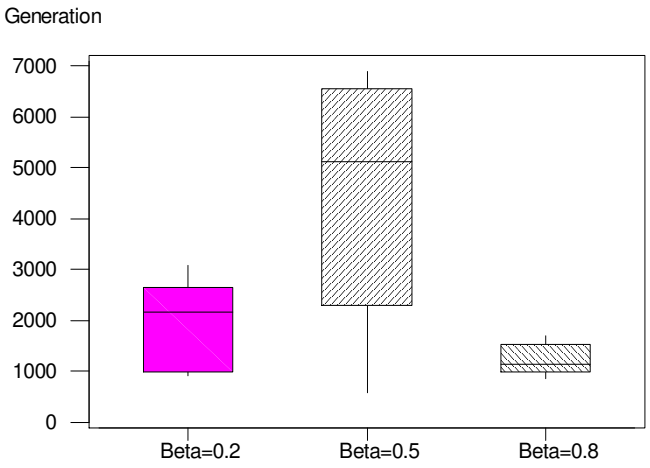

Figure B.1 Box plots of Table B.3 for n=25 and p=3.

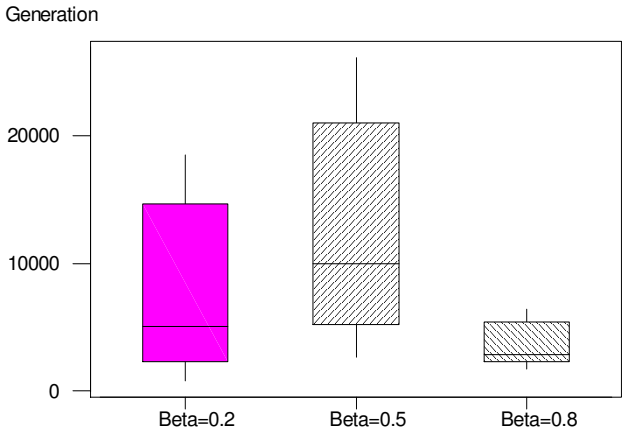

Figure B.2 Box plots of Table B.3 for n=25 and p=5.

Table B.4 Beta analysis of allocation problem for AP data set and Bp-HLP1

			Proximity Indicator $D_{X^{Pareto} \to X^{FWEA\_alloc}}$				
n	**p**	**Beta**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
25	3	0.0	0.012939	0.015931	0.016053	0.014696	0.017286
		0.01	0.016661	0.016862	0.017738	0.016303	0.013877
		0.1	0.019294	0.017788	0.019506	0.019251	0.014560
		0.3	0.020394	0.019141	0.017858	0.018263	0.015659
		0.5	0.017553	0.019562	0.014133	0.017362	0.018250
		0.8	0.019678	0.019367	0.022156	0.016751	0.016086
		1.0	0.017050	0.016235	0.017632	0.019013	0.018134
25	5	0.0	0.016750	0.019950	0.019519	0.017212	0.022722
		0.01	0.015518	0.015853	0.017776	0.016375	0.018625
		0.1	0.018970	0.020018	0.019953	0.018540	0.021283
		0.3	0.021945	0.016654	0.018798	0.020238	0.016785
		0.5	0.023337	0.018519	0.020374	0.019069	0.019135
		0.8	0.018082	0.022375	0.022295	0.018955	0.019607
		1.0	0.019517	0.019625	0.018832	0.020641	0.018318

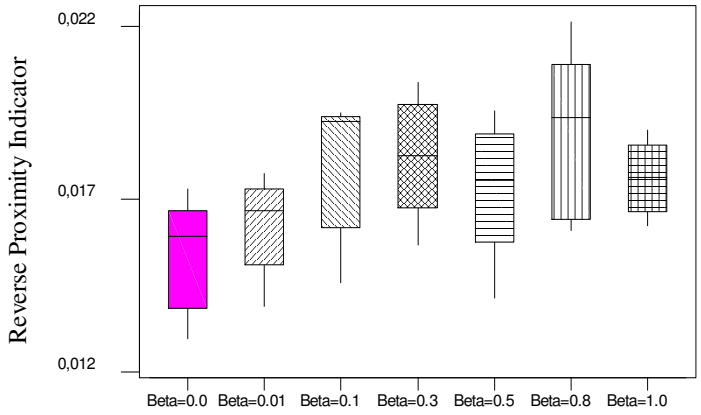

Figure B.3 Box plots of Table B.4 for n=25 and p=3.

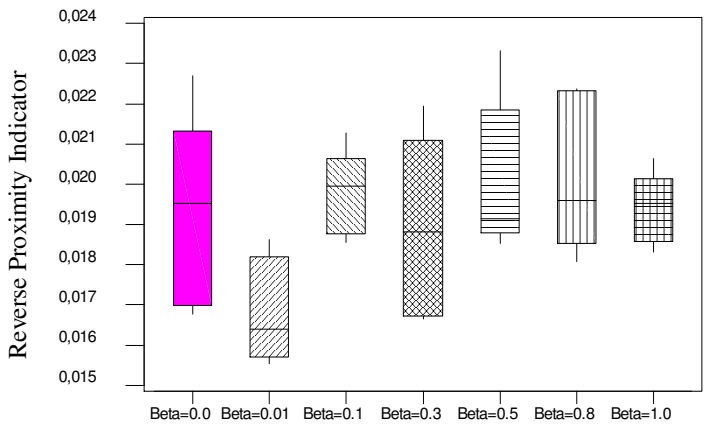

Figure B.4 Box plots of Table B.4 for n=25 and p=5.

Table B.5 Beta analysis of hub location problem for AP data set and Bp-HLP2

n	p	Capacity	Alfa	Run 1	Run 2	Run 3	Run 4	Run 5
10	3	T	0.2	55 *	59	143	234	46
			0.5	85	86	462	277	200
			0.8	199	39	196	225	198
10	5	T	0.2	70	86	76	159	148
			0.5	401	127	72	515	68
			0.8	X	114	38	85	144

* Generation number that all efficient hub sets are found by FWEA_loc over 1000 generations

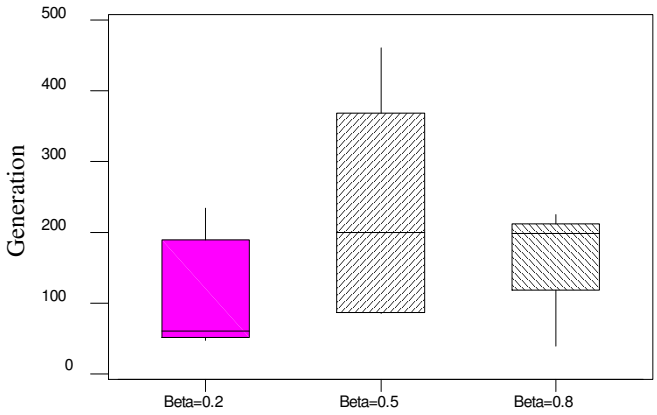

Figure B.5 Box plots of Table B.5 for n=10 and p=3.

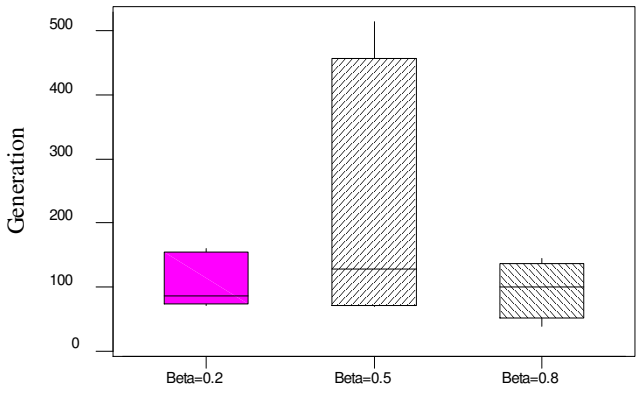

Figure B.6 Box plots of Table B.5 for n=10 and p=5.

Table B.6 AP data set results for Bp-HLP1

n	p	Bp-HLP1 Efficient Hub Sets	FWEA_loc Hub Sets
10	2	3,7 *, **	3,7 *, **
10	3	3, 7, 8 *	3, 7, 8 *
		3, 4, 7	3, 4, 7
		1, 4, 7 **	1, 4, 7 **
10	4	2, 3, 7, 8 *, **	2, 3, 7, 8 *, **
10	5	1, 2, 3, 7, 8 *, **	1, 2, 3, 7, 8 *, **
20	2	6, 14 *, **	6, 14 *, **
20	3	6, 12, 14 *, **	6, 12, 14 *, **
20	4	2, 6, 12, 14 *, **	2, 6, 12, 14 *, **
20	5	2, 6, 12, 13, 14 *, **	2, 6, 12, 13, 14 *, **
25	2	8, 18 *	8, 18 *
		7, 18 **	7, 18 **
25	3	2, 8, 18 *	2, 8, 18 *
		7, 15, 18 **	2, 9, 18
			7, 15, 18 **
25	4	2, 8, 17, 18 *	2, 8, 17, 18 *
		2, 8, 15, 18	2, 8, 15, 18
		2, 7, 15, 18 **	2, 7, 15, 18 **
			2, 8, 18, 20
25	5	2, 8, 17, 18, 20 *	2, 8, 17, 18, 20 *
		2, 8, 15, 17, 18	2, 8, 15, 17, 18
		2, 8, 15, 16, 18 **	2, 8, 15, 16, 18 **
40	4	-	12, 23, 26, 28 *
			3, 15, 26, 28
			3, 15, 18, 28 **
40	5	-	3, 13, 23, 26, 28 *, **
50	4	-	14, 28, 32, 35 *
			6, 28, 32, 35
			6, 29, 32, 35 **
50	5	-	4, 14, 28, 32, 35 *
			4, 15, 29, 33, 35
			4, 14, 28, 33, 35
			4, 15, 29, 32, 35 **

* OR-Lib UMAp-HMP optimal hubs

** p-median problem optimal locations

- Efficient hub sets that could not be found by ε-constraint approach in a reasonable time

Table B.7 TPDS Data set results for Bp-HLP1

n	p	Bp-HLP1 Efficient Hub Sets		
		α = 0.2	**α = 0.5**	**α = 0.8**
10	2	0, 3	0, 3	1, 6
		0, 6	0, 6	0, 3
		3, 9	3, 9	0, 6
				3, 9
10	3	1, 4, 6	1, 4, 6	0, 1, 6
		4, 6, 8	4, 6, 8	1, 6, 9
				1, 4, 6
				4, 6, 8
10	4	1, 4, 6, 7	1, 4, 6, 7	1, 6, 7, 9
		1, 6, 7, 9	1, 6, 7, 9	1, 4, 6, 7
10	5	0, 1, 4, 6, 7	0, 1, 4, 6, 7	0, 1, 4, 6, 7
20	2	13, 16	12, 13	12, 13
			13, 16	13, 16
20	3	1, 7, 13	1, 13, 16	1, 13, 16
		10, 11, 12	1, 7, 13	1, 7, 13
			10, 11, 12	10, 11, 12
20	4	1, 7, 10, 11	1, 10, 15, 16	1,10, 15, 16
			1, 10, 11, 16	1,10, 11, 16
			1, 7, 10, 15	1, 7, 10, 15
			1, 7, 10, 11	1, 7, 10, 11
20	5	1,7, 10, 11, 17	1,10,15,16, 17	0,1,6,10, 15
		1,7, 10, 11, 18	1,10,11, 16, 17	1,10,15,16,17
			1,7,10, 15, 17	1,10,11,16,17
			1, 7, 10, 11, 17	1,7,10,15,17
			1, 7, 10, 11, 18	1,7,10,11,17
				1,7,10,11,18
25	2	14, 15	14, 15	14, 15
		15, 18	15, 18	15, 18
25	3	12, 14, 17	1, 15, 18	1, 15, 18
		12, 13, 14	12, 14, 17	12, 14, 17
			12, 13, 14	12, 13, 14
25	4	1, 9, 12, 17	1, 12, 17, 18	1, 12, 17, 18
		1, 9, 12, 13	1, 9, 12, 17	1, 9, 12, 17
		1, 12, 13, 23	1, 9, 12, 13	1, 9, 12, 13
			1, 12, 13, 23	1, 12, 13, 23
25	5	1,8,12,17,18	0,1,7,12,17	0,1,7,12,17
		1,8,9,12,17	0,1,8,12,17	0,1,8,12,17
		1,9,12,17,19	1,8,12,17,18	1,8,12,17,18
		1, 9,12,13,19	1,9,12,17,19	1,9,12,17,19
		1, 8, 9,12,13	1,8,9,12,17	1,8,9,12,17
		1, 8,12,13, 18	1,9,12,13,19	1,9,12,13,19
			1,8,9,12,13	1,8,9,12,13
			1,8,12,13,18	1,8,12,13,18

Table B.8 CAB Data set results for Bp-HLP1

n	p	Bp-HLP1 Efficient Hub Sets		
		α = 0.2	**α = 0.5**	**α = 0.8**
10	2	7, 9	7, 9	7, 9
10	3	4, 6, 7	4, 6, 7	4, 6, 7
		3, 4, 7	3, 4, 7	3, 4, 7
10	4	3, 4, 6, 7	3, 4, 6, 7	3, 4, 6, 7
10	5	3, 4, 6, 7, 8	3, 4, 6, 7, 8	1, 3, 4, 6, 7
				3, 4, 6, 7, 8
20	2	4, 17	4, 17	4, 17
		11, 17	11, 17	11, 17
20	3	4, 12, 17	4, 12, 17	4, 7, 17
				4, 12, 17
20	4	4, 12, 16, 17	4, 7, 12, 17	4, 7, 12, 17
			1, 4, 12, 17	1, 4, 12, 17
			4, 12, 14, 17	4,12, 14, 17
			4, 12, 16, 17	4,12,16,17
20	5	4,7,12,14,17	4,7,12,14,17	4,7,12,14,17
25	2	12, 20	12, 20	12, 20
25	3	12, 17, 21	4, 12, 18	4, 12, 17
		4, 12, 17	4, 12, 17	4, 12, 18
			12, 17, 21	12, 17, 21
25	4	4, 12, 17, 24	1, 4, 12, 17	1, 4, 12, 17
			4, 12, 17, 24	4, 12, 17, 24
25	5	4,7, 12, 14, 17	4, 7, 12, 14, 17	4,7,12,17,24
				4,7,12,14,17

Table B.9 AP data set results for Bp-HLP2

n	p	capacity	**Bp-HLP2 Efficient Hub Sets**	**FWEA_loc Hub Sets**
10	2	T	3, 7 *	3, 7 *
			4, 7	4, 7
			3, 8	3, 8
			3, 10	3, 10
			5, 8	5, 8
			4, 10	4, 10
			5, 10 **	5, 10 **
10	3	T	3, 7, 8 *	3, 7, 8 *
			3, 4, 10	3, 4, 10
			1, 4, 10	1, 4, 10
			4, 5, 10	4, 5, 10
			5, 9, 10 **	5, 9, 10 **
				1, 5, 8
				5, 6, 10

* OR-Lib UMAp-HMP optimal hub locations

** Optimal hub locations for min. max. D/*Cap* problem

\- Efficient hub sets FWEA_loc could not find

Table B.9 (continued) AP data set results for Bp-HLP2

n	p	capacity	Bp-HLP2 Efficient Hub Sets	FWEA_loc Hub Sets
10	4	T	2, 3, 7, 8 * 3, 4, 7, 8 1, 5, 7, 8 1, 2, 5, 8 1, 4, 5, 10 1, 5, 6, 10 5, 6, 9, 10 **	2, 3, 7, 8 * 3, 4, 7, 8 1, 5, 7, 8 1, 2, 5, 8 1, 4, 5, 10 1, 5, 6, 10 5, 6, 9, 10 ** 3, 4, 7, 10 4, 5, 9, 10
10	5	T	1, 2, 3, 7, 8 * 1, 2, 5, 7, 8 1, 4, 5, 7, 8 1, 2, 5, 8, 10 1, 4, 5, 8, 10 1, 4, 5, 9, 10 4, 5, 6, 9, 10 **	1, 2, 3, 7, 8 * 1, 2, 5, 7, 8 1, 4, 5, 7, 8 1, 2, 5, 8, 10 1, 4, 5, 8, 10 1, 4, 5, 9, 10 4, 5, 6, 9, 10 ** 1, 4, 5, 7, 10 1, 3, 4, 7, 8 1, 2, 5, 7, 10 1, 2, 3, 7, 10
10	2	L	3, 7 * 4, 7 7, 8 8, 10 **	3, 7 * 4, 7 7, 8 8, 10 ** 5, 7
10	3	L	3, 7, 8 * 3, 4, 7 1, 7, 8 4, 7, 8 2, 7, 8 7, 8, 10 **	3, 7, 8 * 3, 4, 7 1, 7, 8 4, 7, 8 2, 7, 8 7, 8, 10 ** 1, 4, 7 5, 7, 8 4, 7, 10
10	4	L	2, 3, 7, 8 * 1, 4, 7, 8 2, 5, 7, 8 4, 5, 7, 8 2, 7, 8, 10 **	2, 3, 7, 8 * 1, 4, 7, 8 2, 5, 7, 8 4, 5, 7, 8 2, 7, 8, 10 ** 3, 4, 7, 8 1, 3, 7, 8
10	5	L	1, 2, 3, 7, 8 * 1, 3, 4, 7, 8 1, 2, 5, 7, 8 1, 4, 5, 7, 8 2, 6, 7, 8, 10 **	1, 2, 3, 7, 8 * 1, 3, 4, 7, 8 - 1, 4, 5, 7, 8 2, 6, 7, 8, 10 ** 2, 5, 7, 8, 10 2, 7, 8, 9, 10 1, 4, 7, 8, 10 1, 5, 6, 7, 8 2, 4, 7, 8, 10

Table B.10 TPDS Data set results for Bp-HLP2

n	p	Bp-HLP2 Efficient Hub Sets		
		α = 0.2	**α = 0.5**	**α = 0.8**
10	2	1,4	1,4	1,4
		1,6	1,6	1,6
		0,6	0,3	
		0,3		
10	3	1,4,6	1,4,6	1,4,6
				0,1,6
10	4	1,4,6,7	1,4,6,7	1,4,6,7
				0,1,6,7
10	5	0,1,4,6,7	0,1,4,6,7	0,1,4,6,7

Table B.11 CAB Data set results for Bp-HLP2

n	p	Bp-HLP2 Efficient Hub Sets		
		α = 0.2	**α = 0.5**	**α = 0.8**
10	2	4,9	4,9	4,9
		4,7	4,7	4,7
		7,9	7,9	7,9
10	3	3,4,9	3,4,9	3,4,9
		3,4,7	4,7,9	4,7,9
		4,6,7	3,4,7	4,6,7
			4,6,7	
10	4	3,4,7,9	3,4,7,9	3,4,7,9
		3,4,6,7	3,4,6,7	3,4,6,7
10	5	3,4,6,7,9	3,4,6,7,9	3,4,6,7,9
		1,3,4,7,9	1,3,4,7,9	1,3,4,7,9
		3,4,7,8,9	3,4,7,8,9	1,3,4,6,7
		3,4,6,7,8	3,4,6,7,8	

Table B.12 Computational results for TPDS data set, *n*=25, p=5 and Bp-HLP1.

	α = 0.2				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	1,8,12,17,18 1,8,9,12,17 1,9,12,17,19	1,8,12,17,18 1,8,9,12,17	1,8,9,12,17 1,9,12,17,19 1,9,12,13,19 1,8,9,12,13	1,8,12,17,18 1,8,9,12,17 1,9,12,17,19 1,9,12,13,19 1,8,9,12,13 1,8,12,13,18	1,8,12,17,18 1,8,9,12,17 1,9,12,17,19 1,9,12,13,19 1,8,9,12,13 1,8,12,13,18
P.I.	0.08252	0.05574	0.08203	0.01107	0.00907
P.I. (no-shows)	0.11164	0.07251	0.21974	-	-
H.I. (Pareto)	0.74482	0.74482	0.74482	0.74482	0.74482
H.I. (FWEA_alloc)	0.71475	0.68912	0.71530	0.74077	0.74212
CPU (FWEA_loc)	813.5	747.8	743.0	743.2	770.7
CPU (FWEA_alloc)	1517.6	1026.3	1952.1	3022.6	3047.7

	α = 0.5				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	0,1,8,12,17 1,9,12,17,19 1,8,9,12,17 1,8,12,17,18	0,1,8,12,17 1,8,12,13,18 1,9,12,17,19 0,1,8,12,13 1,9,12,13,19 1,8,9,12,17 1,8,9,12,13 1,8,12,17,18	0,1,8,12,17 0,1,7,12,17 1,8,12,17,18 1,8,9,12,17 1,9,12,17,19	0,1,8,12,17	1,9,12,13,19 0,1,8,12,13 1,8,12,13,18 1,9,12,17,19 0,1,8,12,17
P.I.	0.02227	0.02247	0.02147	0.08423	0.02506
P.I. (no-shows)	0.02299	0.12605	0.02194	0.09625	0.02560
H.I. (Pareto)	0.82159	0.82159	0.82159	0.82159	0.82159
H.I. (FWEA_alloc)	0.79525	0.79252	0.79427	0.76344	0.79197
CPU (FWEA_loc)	682.4	660.1	750.8	736.4	787.4
CPU (FWEA_alloc)	2450.9	4922.5	3072.6	602.2	3052.6

	α = 0.8				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	1,12,17,18,21 1,9,12,17,21 1,8,9,12,17 0,1,7,15,17 0,1,8,15,17 0,1,8,12,17 1,9,12,17,19 1,8,12,17,18	1,9,12,17,19 1,8,12,13,18 0,1,8,12,17 1,15,17,18,21 1,8,12,17,18 1,8,9,12,13 1,9,12,13,19 1,12,17,18,19 1,12,17,18,21 1,8,15,17,18 1,12,13,18,19	1,15,17,19,23 1,13,15,18,21 1,13,15,18,19 0,1,15,17,21 1,7,13,15,18 1,15,17,18,21 1,15,17,18,19 1,9,15,17,19 1,9,15,17,21 9,12,16,17,19	1,8,12,13,18 0,1,7,12,13 0,1,7,13,15 0,1,8,12,17 0,1,7,12,17 0,1,8,12,13 0,1,8,15,17 1,8,9,12,17 1,8,12,17,18 1,8,9,12,13	1,8,15,17,18 1,12,17,18,19 0,1,7,15,17 1,8,12,17,18 1,9,12,17,19
P.I.	0.02759	0.02856	0.16908	0.02525	0.04272
P.I. (no-shows)	0.03448	0.04514	0.16908	0.02506	0.05122
H.I. (Pareto)	0.81511	0.81511	0.81511	0.81511	0.81511
H.I. (FWEA_alloc)	0.76522	0.76257	0.57887	0.77115	0.73461
CPU (FWEA_loc)	700.7	675.0	634.1	692.8	703.8
CPU (FWEA_alloc)	5633.3	7730.2	6897.7	7067.1	3551.7

- Efficient hub sets FWEA_loc could not find

Table B.13 Computational results for CAB data set, *n*=25, p=5 and Bp-HLP1.

	α = 0.2				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	4,10,12,14,17	6,12,14,17,21	4,7,12,14,17	4,7,12,14,17	4,12,14,16,17 4,12,13,14,17
P.I.	1.01939	5.79046	0.01712	0.02142	1.67007
P.I. (no-shows)	1.01939	5.79046	-	-	1.67007
H.I. (Pareto)	0.78995	0.78995	0.78995	0.78995	0.78995
H.I. (FWEA_alloc)	0.00000	0.00000	0.77713	0.77359	0.00000
CPU (FWEA_loc)	635.2	679.9	709.6	645.3	703.4
CPU (FWEA_alloc)	653.6	589.0	623.2	625.3	1516.5

	α = 0.5				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	4,7,12,14,17 4,7,12,17,24	4,10,12,14,17 4,10,12,17,24	4,7,12,17,24 4,7,12,14,17	4,7,12,14,17 4,7,12,17,24	4,10,12,14,17 4,10,12,17,24
P.I.	0.02393	0.18090	0.02259	0.02184	0.18048
P.I. (no-shows)	-	0.18090	-	-	0.18048
H.I. (Pareto)	0.84384	0.84384	0.84384	0.84384	0.84384
H.I. (FWEA_alloc)	0.81681	0.57966	0.81553	0.81674	0.57742
CPU (FWEA_loc)	737.9	633.6	688.4	584.3	693.2
CPU (FWEA_alloc)	1417.9	1456.2	1756.9	1421.0	1451.5

	α = 0.8				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	6,12,17,21,24 6,12,14,17,21 4,11,12,14,17 4,11,12,17,24 4,12,14,17,21	4,7,12,17,24 1,4,7,12,17 4,7,12,14,17	6,12,14,17,21 6,12,17,21,24	4,11,12,17,24 4,12,14,17,21 4,10,12,14,17 4,10,12,17,24	4,7,12,17,24 4,7,12,14,17
P.I.	0.24290	0.03305	0.30922	0.10050	0.03231
P.I. (no-shows)	0.24290	-	0.30922	0.10050	-
H.I. (Pareto)	0.82445	0.82445	0.82445	0.82445	0.82445
H.I. (FWEA_alloc)	0.48059	0.76803	0.41975	0.64156	0.76881
CPU (FWEA_loc)	625.5	693.1	655.5	619.6	614.9
CPU (FWEA_alloc)	4608.9	2568.7	1646.4	3442.3	1708.6

- Efficient hub sets FWEA_loc could not find

Table B.14 Computational results for AP data set, *n*=25, p=5 and Bp-HLP1.

	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	2,8,15,17,18 2,8,15,16,18	2,8,15,17,18 2,8,15,16,18 2,8,17,18,20	2,7,14,17,18	2,8,15,17,18 2,8,15,16,18	2,8,15,16,18 2,8,15,17,18 2,8,17,18,20
P.I.	0.02355	0.01629	0.57013	0.02643	0.02123
P.I. (no-shows)	0.30097	-	0.57013	0.30097	-
H.I. (Pareto)	0.86749	0.86749	0.86749	0.86749	0.86749
H.I. (FWEA_alloc)	0.85735	0.86043	0.23634	0.85507	0.85739
CPU (FWEA_loc)	695.4	690.7	778.2	831.9	688.7
CPU (FWEA_alloc)	954.6	1417.0	419.0	953.4	1413.6

- Efficient hub sets FWEA_loc could not find

Table B.15 Computational results for TPDS data set, *n*=10, p=5 and Bp-HLP2.

	α = 0.2				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	0,1,4,6,7 1,3,4,6,7 1,4,6,7,8	0,1,4,6,7 1,3,4,6,7 1,4,6,7,8	0,1,4,6,7 1,3,4,6,7 1,4,6,7,8	0,1,4,6,7 1,3,4,6,7 1,4,6,7,8	0,1,4,6,7 1,3,4,6,7 1,4,6,7,8
P.I.	0.07630	0.06807	0.04657	0.07366	0.06623
P.I. (no-shows)	-	-	-	-	-
H.I. (Pareto)	0.43539	0.43539	0.43539	0.43539	0.43539
H.I. (FWEA_alloc)	0.27241	0.23352	0.32829	0.17200	0.33767
CPU (FWEA_loc)	5298.6	7052.6	7551.5	8891.2	7564.4
CPU (FWEA_alloc)	624.3	624.0	630.0	630.8	625.9

	α = 0.5				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	0,1,4,6,7 1,3,4,6,7	0,1,4,6,7 1,3,4,6,7	0,1,4,6,7 1,3,4,6,7	0,1,4,6,7 1,3,4,6,7	0,1,4,6,7 1,3,4,6,7
P.I.	0.06562	0.06643	0.06193	0.04815	0.06308
P.I. (no-shows)	-	-	-	-	-
H.I. (Pareto)	0.89597	0.89597	0.89597	0.89597	0.89597
H.I. (FWEA_alloc)	0.84389	0.85215	0.84313	0.85777	0.83747
CPU (FWEA_loc)	6776.4	6801.9	6703.2	6511.8	5218.1
CPU (FWEA_alloc)	419.8	419.2	412.9	415.5	414.6

	α = 0.8				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	0,1,4,6,7 1,3,4,6,7	0,1,4,6,7 1,3,4,6,7	0,1,4,6,7 1,3,4,6,7	0,1,4,6,7 1,3,4,6,7	0,1,4,6,7 1,3,4,6,7
P.I.	0.06174	0.07034	0.07039	0.07670	0.06567
P.I. (no-shows)	-	-	-	-	-
H.I. (Pareto)	0.87140	0.87140	0.87140	0.87140	0.87140
H.I. (FWEA_alloc)	0.77616	0.76806	0.75973	0.75514	0.67936
CPU (FWEA_loc)	4384.3	5423.9	4666.1	4625.3	5585.5
CPU (FWEA_alloc)	403.5	400.5	405.1	402.6	405.2

- Efficient hub sets FWEA_loc could not find

Table B.16 Computational results for CAB data set, *n*=10, p=5 and Bp-HLP2.

	α = 0.2				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10
P.I.	0.04844	0.04467	0.05200	0.05345	0.03683
P.I. (no-shows)	-	-	-	-	-
H.I. (Pareto)	0.90290	0.90290	0.90290	0.90290	0.90290
H.I. (FWEA_alloc)	0.84360	0.86205	0.86303	0.84762	0.86557
CPU (FWEA_loc)	2616.5	2991.4	3555.0	2096.7	2561.2
CPU (FWEA_alloc)	1025.0	1029.6	1027.7	1028.7	1026.6

	α = 0.5				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10 1,3,4,6,7	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10	3,4,6,7,8 3,4,7,8,9 1,3,4,7,9 3,4,6,7,9 3,4,7,9,10
P.I.	0.04334	0.04278	0.04139	0.04889	0.04127
P.I. (no-shows)	-	-	-	-	-
H.I. (Pareto)	0.85938	0.85938	0.85938	0.85938	0.85938
H.I. (FWEA_alloc)	0.81193	0.79235	0.78146	0.80158	0.81351
CPU (FWEA_loc)	2678.7	2680.3	2877.4	2645.7	2838.3
CPU (FWEA_alloc)	1044.9	1044.0	1250.7	1039.9	1042.8

	α = 0.8				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	3,4,6,7,9 1,3,4,7,9 1,3,4,6,7	3,4,6,7,9 1,3,4,7,9 1,3,4,6,7	3,4,6,7,9 1,3,4,7,9 1,3,4,6,7	3,4,6,7,9 1,3,4,7,9 1,3,4,6,7	3,4,6,7,9 1,3,4,7,9 1,3,4,6,7
P.I.	0.07135	0.08973	0.07104	0.07284	0.06177
P.I. (no-shows)	-	-	-	-	-
H.I. (Pareto)	0.86972	0.86972	0.86972	0.86972	0.86972
H.I. (FWEA_alloc)	0.77673	0.75328	0.74098	0.72076	0.78588
CPU (FWEA_loc)	2344.0	2496.0	2849.8	1975.4	2973.1
CPU (FWEA_alloc)	625.0	622.9	625.9	623.1	622.9

- Efficient hub sets FWEA_loc could not find

Table B.17 Computational results for AP data set, *n*=10, p=5 and Bp-HLP2.

	Tight capacities				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10 1,5,6,9,10	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10 1,5,6,9,10 1,5,6,7,10	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10 1,5,6,9,10	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,5,6,9,10 1,4,5,7,10 1,3,4,7,8
P.I.	0.03729	0.03595	0.03924	0.04014	0.04169
P.I. (no-shows)	-	-	-	-	-
H.I. (Pareto)	0.80931	0.80931	0.80931	0.80931	0.80931
H.I. (FWEA_alloc)	0.75219	0.75717	0.75091	0.74996	0.74982
CPU (FWEA_loc)	3368.0	3732.6	3884.2	3600.4	3467.8
CPU (FWEA_alloc)	1987.3	2172.3	2359.8	2174.8	1820.2

	Loose capacities				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	1,4,5,7,8 2,5,7,8,10 2,7,8,9,10 1,2,3,7,8 1,4,7,8,10 1,5,6,7,8 1,3,4,7,8 2,6,7,8,10 2,4,7,8,10	1,4,5,7,8 2,5,7,8,10 2,7,8,9,10 1,2,3,7,8 1,4,7,8,10 1,5,6,7,8 1,3,4,7,8 2,6,7,8,10 2,4,7,8,10 1,5,6,7,10	1,4,5,7,8 2,7,8,9,10 1,2,3,7,8 1,5,6,7,8 1,3,4,7,8 2,6,7,8,10 2,4,7,8,10 3,4,7,8,10 4,5,6,7,10 1,2,5,7,8	1,5,6,7,10 1,4,5,7,8 2,7,8,9,10 1,2,3,7,8 1,4,7,8,10 2,6,7,8,10 2,4,7,8,10 1,5,6,7,8 1,3,4,7,8 1,2,5,7,8 2,5,7,8,10	1,4,5,7,8 3,4,7,8,10 5,6,7,8,9 1,2,3,7,8 4,5,6,7,8 2,3,7,8,10 1,3,4,7,8 1,5,6,7,8 2,6,7,8,10
P.I.	0.05428	0.05673	0.06099	0.05152	0.05954
P.I. (no-shows)	0.04681	0.04061	-	-	0.05068
H.I. (Pareto)	0.69449	0.69449	0.69449	0.69449	0.69449
H.I. (FWEA_alloc)	0.68117	0.67275	0.63962	0.66891	0.63506
CPU (FWEA_loc)	3058.9	3129.5	3994.0	3773.5	3726.2
CPU (FWEA_alloc)	1648.5	1839.7	1832.2	2014.8	1663.8

- Efficient hub sets FWEA_loc could not find

Table B.18 Computational results for AP data set, *n*=25, p=5 and Bp-HLP1.

	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	2,8,15,17,18 2,8,15,16,18	2,8,15,17,18 2,8,15,16,18 2,8,17,18,20	2,7,14,17,18	2,8,15,17,18 2,8,15,16,18	2,8,15,16,18 2,8,15,17,18 2,8,17,18,20
P.I.	0.11846	0.10561	0.59956	0.17793	0.10561
P.I. (no-shows)	0.30097	-	0.59956	0.30097	-
H.I. (Pareto)	0.86749	0.86749	0.86749	0.86749	0.86749
H.I. (FWEA_alloc)	0.73028	0.73496	0.20598	0.73028	0.73496
CPU (FWEA_loc)	695.4	690.7	778.2	831.9	688.7
CPU (FWEA_alloc)	849.8	1258.9	374.6	848.3	1260.6

* FWEA_alloc is run without restriction of criteria space presented in Section 4.5.4.
- Efficient hub sets FWEA_loc could not find

Table B.19 Computational results for AP data set, *n*=10, p=5 and Bp-HLP2.

	Tight capacities				
	Run1	**Run2**	**Run3**	**Run4**	**Run5**
Hub Sets	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10 1,5,6,9,10	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10 1,5,6,9,10 1,5,6,7,10	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,4,5,7,10 1,3,4,7,8 1,2,5,7,10 1,2,3,7,10 1,5,6,9,10	1,2,3,7,8 1,2,5,7,8 1,4,5,7,8 1,2,5,8,10 1,4,5,8,10 1,4,5,9,10 4,5,6,9,10 1,5,6,9,10 1,4,5,7,10 1,3,4,7,8
P.I.	0.07273	0.06850	0.07415	0.06264	0.07242
P.I. (no-shows)	-	-	-	-	-
H.I. (Pareto)	0.80931	0.80931	0.80931	0.80931	0.80931
H.I. (FWEA_alloc)	0.69641	0.70502	0.70010	0.71487	0.69851
CPU (FWEA_loc)	3368.0	3732.6	3884.2	3600.4	3467.8
CPU (FWEA_alloc)	2018.5	2201.5	2394.2	2218.6	1828.1

* FWEA_alloc is run without restriction of criteria space presented in section 4.5.4.
- Efficient hub sets FWEA_loc could not find

APPENDIX C

RESULTS FOR MULTIPLE CRITERIA KNAPSACK PROBLEM

Table C. 1 Proximity indicator results of FWEA_KP for α=0.0, 0.25, 0.5, 0.75, 1.0 and 2KP_750 problem.

Run	α =0.0	α =0.01	α =0.05	α =0.1	α =0.25	α =0.50	α =0.75	α =1.0
1	0.05829	0.05703	0.05800	0.05534	0.06074	0.05800	0.06024	0.05858
2	0.05948	0.05624	0.05647	0.05363	0.05998	0.05647	0.05432	0.05898
3	0.05569	0.05374	0.05976	0.05779	0.05707	0.05979	0.05828	0.06158
4	0.05826	0.05592	0.06063	0.05715	0.05710	0.06063	0.05910	0.05584
5	0.05652	0.06104	0.05989	0.05722	0.05925	0.05989	0.05770	0.05536

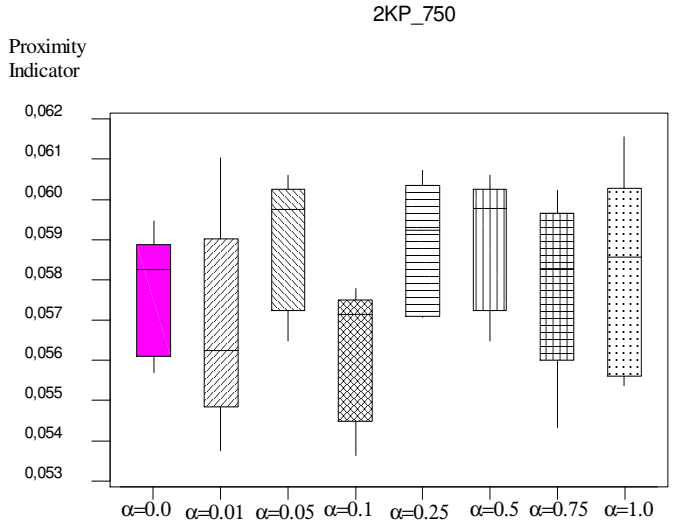

Figure C.1 Box plots of FWEA_KP results for different α levels

Table C. 2 *Proximity indicator* results of FWEA_KP, EMAPS, SPEA2 and NSGAII for 2KP-750

Run #	FWEA_KP (3-seed)	FWEA_KP (11-seed)	EMAPS	EMAPS (3-seed)	SPEA2	NSGAII
1	0.05534	0.02139	0.11472	0.11319	0.11670	0.14689
2	0.05363	0.02188	0.12024	0.10197	0.10259	0.15812
3	0.05779	0.02335	0.12144	0.10386	0.10001	0.15535
4	0.05715	0.02019	0.14627	0.09631	0.10400	0.13703
5	0.05722	0.02111	0.10820	0.10191	0.11646	0.15066
6	0.05643	0.02180	0.10795	0.09697	0.11946	0.13718
7	0.06120	0.01852	0.11849	0.09629	0.11770	0.13878
8	0.05651	0.02047	0.12589	0.11769	0.11081	0.14743
9	0.05721	0.02120	0.11884	0.10212	0.11773	0.12489
10	0.05591	0.02002	0.11289	0.12450	0.12104	0.15887

Table C.3 *Hypervolume indicator* results of FWEA_KP, EMAPS, SPEA2 and NSGAII for 2KP-750

Run #	FWEA_KP (3-seed)	FWEA_KP (11-seed)	EMAPS	EMAPS (3-seed)	SPEA2	NSGAII
1	0.69352	0.75096	0.54335	0.58784	0.60204	0.57365
2	0.69542	0.75109	0.54020	0.59165	0.59125	0.57261
3	0.69052	0.74829	0.53752	0.59681	0.60351	0.57531
4	0.69206	0.75268	0.53859	0.57007	0.60470	0.58167
5	0.69173	0.75136	0.54140	0.61561	0.60431	0.56810
6	0.69209	0.75054	0.53901	0.61686	0.60613	0.58561
7	0.68444	0.75503	0.53981	0.59757	0.60573	0.58229
8	0.69418	0.75242	0.53655	0.59172	0.60686	0.57531
9	0.69269	0.75188	0.54463	0.61385	0.59682	0.58789
10	0.69227	0.75307	0.54256	0.55501	0.58695	0.56223

* Note that hypervolume of the efficient frontier is 0.78316.

Table C.4 *Proximity indicator* results of FWEA_KP, EMAPS, SPEA2 and NSGAII to NDUS for 3KP-750

Run #	FWEA_KP (4-seed)	SPEA2	NSGAII
1	0.02158	0.18942	0.17385
2	0.01737	0.20094	0.19439
3	0.01812	0.19775	0.20446
4	0.02156	0.13259	0.15083
5	0.01775	0.14350	0.15727
6	0.01506	0.14790	0.19980
7	0.01389	0.17226	0.18434
8	0.02357	0.17879	0.17062
9	0.02375	0.18883	0.19982
10	0.02456	0.17504	0.19841

Table C.5 *Hypervolume indicator* results of FWEA_KP, EMAPS, SPEA2 and NSGAII for 3KP-750

Run #	FWEA_KP (4-seed)	SPEA2	NSGAII
1	0.40867	0.32951	0.33069
2	0.40981	0.33059	0.31908
3	0.40443	0.33246	0.31415
4	0.40756	0.34111	0.31859
5	0.40961	0.33410	0.31431
6	0.41463	0.33909	0.31257
7	0.41042	0.32857	0.31801
8	0.40931	0.33044	0.32113
9	0.41244	0.32192	0.31324
10	0.40419	0.33066	0.31292

Table C.6 Percent contribution of each algorithm to NDUS for 3KP-750

Run #	FWEA_KP (4-seed) %	SPEA2 %	NSGAII %
1	66.6	25.4	8.0
2	68.0	26.8	5.2
3	68.7	26.1	5.2
4	54.4	32.9	12.7
5	55.0	29.2	15.8
6	65.4	28.7	5.9
7	72.4	13.2	14.4
8	60.5	26.6	12.9
9	72.1	11.9	16.0
10	63.0	24.7	13.3

Table C.7 *Proximity indicator results* of FWEA_KP, EMAPS, SPEA2 and NSGAII to NDUS for 4KP-750

Run #	FWEA_KP (5-seed)	SPEA2	NSGAII
1	0.00755	0.17452	0.21020
2	0.00820	0.17528	0.20017
3	0.00265	0.24616	0.24284
4	0.00253	0.21750	0.23533
5	0.01912	0.20516	0.19835
6	0.00153	0.20651	0.23306
7	0.00574	0.20312	0.20639
8	0.05053	0.19246	0.20496
9	0.00086	0.21067	0.23202
10	0.00301	0.21079	0.22564

Table C.8 *Hypervolume indicator* results of FWEA_KP, EMAPS, SPEA2 and NSGAII for 4KP-750

Run #	FWEA_KP (5-seed)	SPEA2	NSGAII
1	0.05897	0.02336	0.03506
2	0.05984	0.04184	0.01746
3	0.05828	0.03473	0.01878
4	0.05946	0.03420	0.01710
5	0.05603	0.03701	0.01906
6	0.05981	0.03208	0.01998
7	0.05858	0.03533	0.01656
8	0.06015	0.03491	0.01995
9	0.06042	0.03117	0.01458
10	0.05998	0.03887	0.01561

Table C.9 Percent contribution of each algorithm to the union non-dominated set of points for 4KP-750

Run #	FWEA_KP (5-seed) %	SPEA2 %	NSGAII %
1	87.7	10.3	2.0
2	88.7	11.3	0.0
3	97.2	2.2	0.6
4	96.4	3.6	0.0
5	78.0	12.6	9.4
6	98.0	1.4	0.6
7	92.5	7.5	0.0
8	76.9	12.4	10.7
9	98.6	1.1	0.3
10	94.8	4.6	0.6

Table C.10 *Proximity indicator* results of FWEA_KP, SPEA2 and NSGAII for 2KP-750 and 480000 function evaluations

Run #	FWEA_KP (3-seed)	SPEA2	NSGAII
1	0.04702	0.09248	0.12887
2	0.04383	0.09169	0.12090
3	0.04783	0.09322	0.10903
4	0.04579	0.08645	0.09943
5	0.04677	0.08250	0.10962
6	0.05192	0.07970	0.11635
7	0.04423	0.09213	0.11070
8	0.04667	0.07738	0.11477
9	0.05506	0.08346	0.10368
10	0.04640	0.09487	0.11929

Table C.11 *Hypervolume indicator* results of FWEA_KP, SPEA2 and NSGAII for 2KP-750 and 480000 function evaluations

Run #	**FWEA_KP (3-seed)**	**SPEA2**	**NSGAII**
1	0.70879	0.65223	0.61914
2	0.71310	0.64802	0.62524
3	0.70689	0.64317	0.63600
4	0.70951	0.65720	0.63930
5	0.70837	0.65389	0.63541
6	0.70169	0.66156	0.63204
7	0.71236	0.64953	0.63103
8	0.70935	0.66478	0.62864
9	0.69455	0.66006	0.63672
10	0.70832	0.65744	0.61636

CURRICULUM VITAE

PERSONAL

Date of Birth : October 3rd, 1978
Place of Birth : Ankara-TURKEY

EDUCATIONAL BACKGROUND

- M.Sc. in Industrial Engineering, METU
  Years : 1999 - 2002
  Thesis Title: A Mixed Model Flow Line Sequencing Problem with Makespan Minimization
  Supervisor: Prof. Dr. Ömer Kırca
  Co-Supervisor: Prof. Dr. Meral Azizoğlu

- B.Sc. in Industrial Engineering, Erciyes University
  Years : 1994 - 1998

POSITIONS

Research Assistant : Industrial Engineering Department, METU
March 1999 - Present

Courses Assisted
IE 333 Work Systems Analysis and Design : Fall 2006
IE 372 Simulation : Spring 2006
IE 361 Operations Research III : Fall 2005
IE 322 Production Planning : Spring 2005
IE 266 Engineering Statistics II : Fall 2004
IE 265 Engineering Statistics I : Spring 2004
IE 454 Network Flows and Project Management : Fall 2003
IE 453 Topics in Optimization : Spring 2003
ES 303 Statistical Methods for Engineers : Spring 2002
IE 424 Scheduling : Spring 2001
IE 404 Management for Engineers: Fall 2003, Fall 2002, Fall 2001, Fall 2000, Fall 1999
IE 102 Industrial Engineering Orientation : Spring 2002, Spring 2000
IE 428 Systems Design : Spring 2000, Spring 2001

Editorial Assistant: Transactions on Operational Research (Journal of the Operational Research Society of Turkey)
2002-2004

PUBLICATIONS

Refereed Journal Articles

Soylu, B., Ö. Kırca and M. Azizoğlu (2006), "Flow Shop Sequencing with Synchronous Transfers and Makespan Minimization," to appear in *International Journal of Production Research*

Soylu, B. and M. Köksalan (2006), "Favorable Weight Based Evolutionary Algorithm for Multiple Criteria Problems," submitted to *IEEE Transactions on Evolutionary Computation*

CONFERENCE PRESENTATIONS

Soylu, B. and M. Köksalan, "A Multi-Criteria Evolutionary Algorithm for Generating the Efficient Frontier," *INFORMS 2006 International Conference*, Hong Kong, June 2006.

Soylu, B. and M. Köksalan, "Evolutionary Search for Efficient Solutions in Multiple Criteria Combinatorial Optimization," *18th International Conference on Multiple Criteria Decision Making*, Chania, Greece, June 2006.

Soylu, B. and M. Köksalan, "A Favorable Weight Based Evolutionary Algorithm for Multiobjective Optimization Problems," *INFORMS Annual Meeting*, San Francisco, USA, November 2005.

Soylu, B. and M. Köksalan, "An Evolutionary Algorithm for Multiobjective Combinatorial Optimization", *XI ELAVIO Latin American Summer Workshop on Operations Research*, Bogota, Colombia, July 2005.

Soylu, B. and M. Köksalan, "An Evolutionary Algorithm for Multiobjective Combinatorial Problems", *25th National Meeting on Operational Research and Industrial Engineering*, Koç University, Istanbul, Turkey, July 2005 (in Turkish).

Soylu, B. and M. Köksalan, "An Evolutionary Algorithm for Multiobjective Combinatorial Optimization Problems", *35th International Conference of Computers & Industrial Engineering*, Istanbul, Turkey, June 2005.

Soylu, B. "A Genetic Local Search Algorithm for Sequencing Problem on a Synchronous Flow Line ", *24th National Meeting on Operational Research and Industrial Engineering*, Gaziantep - Adana, Turkey, June 2004 (in Turkish).

Soylu, B, "Application of a Genetic Local Search Algorithm for Sequencing Problem on a Synchronous Flow Line ", *Euro XX 20th European Conference on Operational Research*, 4-7 July 2004, Rhodes-Greece.

Soylu, B., Ö. Kırca and M. Azizoğlu, "A Mixed Model Line Sequencing Problem with Makespan Minimization", *EURO/INFORMS International Meeting*, Istanbul-Turkey, July 2003.

Soylu, B., Ö. Kırca and M. Azizoğlu, "A Mixed Model Line Sequencing Problem with Makespan Minimization", *23th Operations Research and Industrial Engineering National Conference*, Istanbul- Turkey, July 2002.

SEMINARS AND PANELS

January 2007 *An Evolutionary Algorithm for Multiple Criteria Problems*, Industrial Engineering Department Ph.D. Defense Seminar, Middle East Technical University, Ankara, Turkey.
April 2006 *An Evolutionary Algorithm for Multiple Criteria Problems*, Industrial Engineering Department Seminar, Middle East Technical University, Ankara, Turkey.
October 2000 *Place of women in Economic Life*, Panel organized by Erciyes University and Turk Telekom A.Ş., Kayseri, Turkey.

OTHER RESEARCH PROJECTS

BOTAŞ A.Ş. Norm Kadro Oluşturma Projesi (Determination of descriptive list of permanent positions and required number of personals), June 2003-September 2003. Project Team: Prof.Dr. Ö. Saatçioğlu, Assoc.Prof.Dr. T. Şen, F. Ökmen, T. Altekin, B. Balçık, B. Soylu, S. Sözer, D. Türsel Eliiyi, N. Zeylan.

ADMINISTRATIVE WORK

2003 – to date	Member of *Engineering Management Certificate Program* Committee, Industrial Engineering Department, METU, Ankara
2002 – 2004	Member of *TOR Journal* Committee (Editorial Assistant of Transactions on Operational Research Journal), Industrial Engineering Department, METU, Ankara
2001 – 2003	Member of *IE 428 Systems Design* Committee, Industrial Engineering Department, METU, Ankara
1999 - 2003	Member of *ORST (Operational Research Society, Turkey)* Committee, Industrial Engineering Department, METU, Ankara

AWARDS & HONORS

Awarded research assistantships during graduate study.

2007 TÜBİTAK Post Doctoral Fellowship to study in the field of Multi-Objective Optimization at Arizona State University

1998 High Honor Student in the graduates list of Engineering Faculty, Erciyes University

1998 Ranked 1st in the graduates list of Industrial Engineering Department, Erciyes University
1994 Ranked 1st in the graduates list of Şeker High School, Kayseri

RESEARCH INTERESTS

Multiple Criteria Problems
Combinatorial Optimization
Evolutionary Algorithms
Metaheuristics
Scheduling

MEMBERSHIPS

2005 - to date	INFORMS, Institute for Operations Research and the Management Sciences
1999 – to date	ORST, Operational Research Society, Turkey