APPROACHES FOR MULTIOBJECTIVE COMBINATORIAL OPTIMIZATION PROBLEMS

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

BY

NAİL ÖZGÜR ÖZPEYNİRCİ

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

JANUARY 2008

Approval of the thesis:

APPROACHES FOR
MULTIOBJECTIVE COMBINATORIAL OPTIMIZATION PROBLEMS

submitted by **NAİL ÖZGÜR ÖZPEYNİRCİ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Industrial Engineering Department, Middle East Technical University** by,

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

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

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

Examining Committee Members:

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

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

Prof. Dr. Sencer Yeralan
Agricaltural and Biological Engineering Dept.,
University of Florida _____

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

Assoc. Prof. Osman Oğuz
Industrial Engineering Dept., Bilkent University _____

Date: _____

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: Nail Özgür Özpeynirci

Signature:

ABSTRACT

APPROACHES FOR
MULTIOBJECTIVE COMBINATORIAL OPTIMIZATION PROBLEMS

Özpeynirci, Nail Özgür
Ph.D., Department of Industrial Engineering
Supervisor: Prof. Dr. Murat Köksalan

January 2008, 132 pages

In this thesis, we consider multiobjective combinatorial optimization problems. We address two main topics. We first address the polynomially solvable cases of the Traveling Salesperson Problem and the Bottleneck Traveling Salesperson Problem. We consider multiobjective versions of these problems with different combinations of objective functions, analyze their computational complexities and develop exact algorithms where possible.

We next consider generating extreme supported nondominated points of multiobjective integer programming problems for any number of objective functions. We develop two algorithms for this purpose. The first one is an exact algorithm and finds all such points. The second algorithm finds only a subset of extreme supported nondominated points providing a worst case approximation for the remaining points.

Keywords: Multiobjective Combinatorial Optimization, Traveling Salesperson Problem, Bottleneck Traveling Salesperson Problem, Computational Complexity, Extreme Points, Approximation Algorithm.

ÖZ

ÇOK AMAÇLI KOMBİNATORYAL OPTİMİZASYON PROBLEMLERİ İÇİN YAKLAŞIMLAR

Özpeynirci, Nail Özgür
Doktora, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Murat Köksalan

Ocak 2008, 132 sayfa

Bu tezde, çok amaçlı kombinatoryal optimizasyon problemleri üzerinde çalıştık. Çalışmamızı iki ana başlıkta gruplayabiliriz. İlk başlık, gezgin satıcı probleminin ve darboğaz gezgin satıcı problemlerinin polinom çözülebilen durumlarıyla ilgilidir. Biz bu problemlerin, farklı amaç fonksiyonlarının birleşkeleri olan çok amaçlı türevlerini ele aldık, hesaplama karmaşıklıklarını analiz ettik ve mümkün olan durumlarda kesin yordamlar geliştirdik.

İkinci başlığımız, herhangi sayıda amaç fonksiyonu olan çok amaçlı tam sayılı programlama problemlerinin destekli uç etkin noktalarını bulmakla ilgidir. Bu başlık altında iki yordam geliştirdik. İlki bu noktaların hepsini bulan bir kesin yordamdır. İkinci yordam ise bu noktaların bir alt kümesini bulmakta ancak kalan noktalar için bir en kötü durum bilgisi sunmaktadır.

Anahtar Kelimeler: Çok Amaçlı Kombinatoryal Optimizasyon, Gezgin Satıcı Problemi, Darboğaz Gezgin Satıcı Problemi, Hesaplama Karmaşıklığı, Uç Noktalar, Yaklaşıklama Yordamı.

To my love and my parents

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Murat Köksalan for his guidance and advice in academic issues and beyond. I am extremely fortunate to have such a great supervisor as him. I hope we will keep working together in the future.

Thanks Assoc. Prof Haldun Sural, Prof. Dr. Sencer Yeralan, Assoc. Prof. Esra Karasal, Assoc. Prof. Canan Sepil and Assoc. Prof. Osman Oğuz for their valuable comments throughout the study.

My lovely wife and my best friend Selin Özpeynirci was my endless support. We lived the difficulties of being a graduate student together and we tried to understand, help, and encourage each other.

I strongly felt the support and faith of my parents Suzan and Ali Rıza Özpeynirci in every stage of my life. I deem myself blessed to have their presence and love. Also I would like to thank my sister Dilek Kırbıyık.

I am grateful to all my colleagues, friends and managers in the Scientific and Technological Research Council of Turkey (TÜBİTAK) and especially the ones in my department, KGO. It was a great pleasure for me to work in TÜBİTAK. I would like to express my gratitude to TÜBİTAK for the scholarship provided during some part of my graduate study.

I would like to thank Bora Kat, Oğuz Solyalı and Deniz Türsel Eliiyi for being kind supportive and cheerful friends. I am grateful to all my professors and friends at METU IE for their kindness.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Combinatorial optimization is a field of mathematical programming that has been attracting researchers for many years. It has various potential applications in real life problems. Some of these applications are radiation therapy, crew and resource scheduling in airline operations, internet traffic routing, vehicle routing, and portfolio optimization.

Classical combinatorial optimization problems deal with a single objective, whereas many real life problems have several conflicting objectives. Hence, *multiobjective combinatorial optimization* (*MOCO*) is a field of great interest due to its ability to represent real life problems well. Combinatorial optimization problems are generally difficult to solve, even with a single objective. Dealing with multiple objectives further complicates these problems, since one has to consider the trade-offs and conflicts between these objectives where there may be many solutions of interest.

In single objective optimization, there is a single optimal objective function value. There might be alternative optimal solutions giving the same objective function value. On the other hand, in multiobjective optimization, there are typically many "good" solutions called *nondominated solutions.* They represent the trade-offs and conflicts between the objectives. *A decision maker* (*DM*), or a group of DMs, who are the

owner(s) of the problem should evaluate these solutions and select the best one according to their preferences.

The *Traveling salesperson problem* (*TSP*) is one of the most widely studied combinatorial optimization problems in the literature. TSP aims to find the shortest tour that visits each node exactly once and returns to the starting node on a given graph. A variant of TSP is the *Bottleneck TSP* (*BTSP*), where the aim is to find the tour whose longest edge is as short as possible. Both TSP and BTSP are difficult problems in general. However, there are some special cases that are easy to solve.

The research on multiobjective TSP is limited compared to single objective TSP. The main reason is the complexity of TSP even with a single objective. The literature on multiobjective TSP mainly focuses on heuristic approaches for biobjective TSPs. There are few studies dealing with BTSP for multiobjective problems.

Some researchers classify the special cases of TSP and BTSP into two groups. The first class deals with problems having special distance matrices. The problems with special graph structures are in the second class. The special cases of TSP and BTSP studied so far are all single objective problems and, to the best of our knowledge, there is no study on their multiple objective versions.

We study two special cases, one from each class. The first one has a distance matrix such that there is a set of constraints defined on the distances between cities. These constraints ensure that the optimal tour has a special structure, i.e., it looks like a pyramid when the numbers of the cities are plotted in the order they are visited by the optimal tour. There is an exact algorithm that finds the optimal pyramidal tour quite easily. We define the multiobjective versions of these problems, develop some properties of nondominated solutions and propose an exact algorithm.

The second type has a special graph structure such in which only some roads between cities are available. These graphs are called Halin graphs. There are exact

algorithms using the special structure of Halin graphs efficiently, and they find the optimal tours of TSP and BTSP on Halin graphs easily. We define several combinations of TSP and BTSP with multiple objectives on Halin graphs, develop some properties of nondominated solutions, analyze the complexity of the problems, and propose exact algorithms.

A nondominated solution is an *extreme nondominated* solution if it is not possible to represent it as a convex combination of other nondominated solutions. Experimental studies on the *multiobjective knapsack problem* (*MOKP*) showed that number of nondominated solutions increases exponentially as the problem size increases. Interestingly, the number of extreme nondominated solutions increases linearly for the same problem. Hence finding only the set of extreme nondominated solutions may be useful because these solutions also provide valuable information about the trade-offs between the objectives. Finding the extreme nondominated solutions does not require more effort to find all nondominated points.

There is an approach to find all extreme nondominated solutions for biobjective problems. This approach systematically changes the weights of the objective functions and solves single objective problems with a weighted sum objective function. This approach is only applicable to biobjective problems due to their special structure. We develop an exact algorithm that finds all extreme nondominated solutions of a problem for any number of objectives, and apply it to TSP and two other well-known combinatorial optimization problems, the *Assignment Problem* (*AP*) and the *Knapsack Problem* (*KP*).

Although we develop an algorithm to find all extreme nondominated solutions, it may still be a difficult task to generate them because the underlying single objective problem may be difficult or the number of extreme nondominated solutions may be large. In this case, we can try to find a subset of solutions that is a good representation of all extreme nondominated solutions. For this purpose we define a measure and develop an approximation algorithm. This algorithm finds a subset of

extreme nondominated solutions that represents the whole set at a desired quality. We apply our approximation algorithm on a set of assignment problems.

This thesis consists of seven chapters. In Chapter 2, we review the literature and give necessary definitions. In Chapters 3 and 4, we discuss the solvable special cases of TSP and multiple objectives. In Chapter 3, we focus on multiobjective TSP and pyramidal tours. We address TSP and BTSP on Halin graphs and extend it to multiple objectives in Chapter 4. In Chapter 5, we explain our exact algorithm that finds all the extreme nondominated solutions for a multiobjective problem for any number of objectives. We develop an approximation algorithm that finds a subset of extreme nondominated solutions in Chapter 6. We discuss further research directions and conclude the thesis in Chapter 7.

CHAPTER 2

DEFINITIONS AND LITERATURE REVIEW

In this chapter, we give definitions related to combinatorial problems in general and to the Traveling Salesperson Problem in particular. We introduce multiobjective optimization and multiobjective combinatorial optimization problems with a review of literature.

2.1 Combinatorial Optimization

Combinatorial optimization is a field of mathematical programming, which has been attracting researchers for many years. It has various potential applications in real life problems. Some of these applications are radiation therapy, crew and resource scheduling in airline operations, internet traffic routing, vehicle routing, and portfolio optimization. We refer to Ehrgott and Gandibleux (2002) for a discussion on such real life applications.

Combinatorial optimization deals with *combinatorial problems*. The feasible set of a combinatorial problem has a finite number of elements. Let E be the finite set $E = \{e_1, \ldots, e_m\}$ and $w : E \to \mathbb{R}$ be a function assigning weights to the elements of E. We assume that w is a vector of rational numbers. The feasible set of a

combinatorial problem is given by $X \subset 2^E$ as a power set of *E*. An objective function *f*, which is to be minimized, is defined to for a feasible solution $x \in X$. We can write the combinatorial optimization problem as:

$$\min_{x \in X} f(x)$$

In general, there are two types of objective functions considered in combinatorial optimization problems:

$$f(x) = \sum_{e \in x} w(e) \text{ , and}$$

$$f(x) = \max_{e \in x} w(e).$$

The problem with the first type of objective function;

$$\min_{x \in X} \sum_{e \in x} w(e)$$

is called as the *sum problem*. The problem with the second type of objective function;

$$\min_{x \in X} \max_{e \in x} w(e)$$

is called as the *bottleneck problem*.

Combinatorial problems can also be formulated using binary variables. Let $x \in \{0,1\}^m$ and

$$x_i = \begin{cases} 1 \text{ if } e_i \in x \\ 0 \text{ otherwise} \end{cases}$$

where *x* is a feasible solution. Using binary variables, problems can be defined as

$$\min_{x \in X} \sum_{i=1}^{m} w_i x_i \text{ , and}$$

$$\min_{x \in X} \max_{i=1}^{m} w_i x_i$$

where $w_i = w(e_i)$.

The assignment, knapsack, minimum spanning tree, shortest path, traveling salesperson and set covering problems are well-known combinatorial optimization

problems. We refer to Nemhauser and Wolsey (1988) for the theory of combinatorial optimization. Korte and Vygen (2002) review the theory and algorithms on combinatorial optimization problems.

2.2 The Traveling Salesperson Problem

Let $G=(N,E)$ be a graph with the given set of nodes, $N=\{1,\ldots,n\}$ and the set of edges E. The node set may stand for the cities and the edge set for the roads directly connecting the cities. For each $e\in E$, a weight $w(e)$ is given. This weight may correspond to different objectives, such as duration, cost, distance, risk, etc. associated with traversing the edge. A traveling salesperson starts a tour from a city, visits all cities exactly once and returns to the city where the tour is started. Such a tour is called a *Hamiltonian tour*. The problem is to find the Hamiltonian tour with the minimum total weight.

Let φ be a Hamiltonian tour on G and let F denote the set of all Hamiltonian tours. Using these definitions, TSP can be stated as:

$$TSP:\min\left\{ f(\varphi)=\sum_{e\in\varphi} w(e)\right\} \text{ subject to } \varphi\in F .$$

We call above objective function as *TSP-type* objective function. An alternative representation is as follows:

Let $\varphi(i)$ represent the node succeeding node i in tour φ. A tour can be represented by $\varphi=(1,i_1,i_2,\ldots,i_{n-1})$ where $\varphi(1)=i_1$, $\varphi(i_1)=i_2,\ldots,$ $\varphi(i_{n-1})=1$. Let d be the distance matrix and $d[i, j]$ denote the distance between nodes i and j. Then the length of tour φ is $d(\varphi)=\sum_{i=1}^{n} d[i,\varphi(i)]$. TSP can be defined as:

$$\min_{\varphi\in F}\left\{ d(\varphi)=\sum_{i=1}^{n} d[i,\varphi(i)]\right\} .$$

To formulate a mathematical model of TSP, let $e=(i,j)$ be the edge between nodes i and j, $w(e)=w_{ij}$ and introduce the binary decision variable x_{ij} where

$$x_{ij} = \begin{cases} 1 \text{ if edge}(i, j) \text{ is used} \\ 0 \text{ otherwise} \end{cases}$$

Hence, the decision variable space is $x \in X = \{0,1\}^{|E|}$. If edge (i,j) is used in a tour, then the traveling salesperson visits node j immediately after node i in that tour. The integer programming formulation of TSP is as follows:

The aim is to minimize the total weight of the selected edges.

$$TSP\text{: } \min \sum_{(i,j)\in E} w_{ij} x_{ij} .$$

The first constraint set ensures that the tour enters node j exactly once, for each node $j \in N$.

$$\sum_{i:(i,j)\in E} x_{ij} = 1 \text{ for } j \in N .$$

The second constraint set ensures that the tour leaves node i exactly once, for each node $i \in N$.

$$\sum_{j:(i,j)\in E} x_{ij} = 1 \text{ for } i \in N .$$

The third constraint set eliminates all possible subtours. These constraints are called *subtour elimination constraints.* Let $U \subseteq N$. We define the third constraint set as:

$$\sum_{\substack{(i,j)\in E \\ i\in U \\ j\in N\setminus U}} x_{ij} \geq 1 \text{ for } 2 \leq |U| \leq |N| - 2 .$$

The last constraint set defines decision variables as binary.

$$x_{ij} \in \{0,1\} \text{ for } (i, j) \in E .$$

It is possible to represent the subtour elimination constraints and TSP itself in alternative ways. We refer to Punnen (2002a) for a discussion on the alternative formulations.

In a variant of TSP, we are not interested in the total distance traveled by the salesperson but in the maximum distance traveled between any two succeeding cities. This problem is called as *Bottleneck TSP* (*BTSP*). We refer to Kabadi and Punnen (2002) for a review of BTSP.

BTSP can be stated as:

$$BTSP: \min\left\{ f(\varphi) = \max_{e \in \varphi}\{w(e)\}\right\} \text{ subject to } \varphi \in F$$

We call above objective function the *BTSP-type* objective function. The BTSP-type objective function can be handled in the mathematical formulation of BTSP by defining the following constraint.

$$w_{ij}x_{ij} \le \beta \text{ for } (i,j) \in E$$

BTSP can be formulated as:

$$BTSP: \min \beta$$

Subject to

$$w_{ij}x_{ij} \le \beta \text{ for } (i,j) \in E$$

$$\sum_{i:(i,j)\in E} x_{ij} = 1 \text{ for } j \in N.$$

$$\sum_{j:(i,j)\in E} x_{ij} = 1 \text{ for } i \in N.$$

$$\sum_{\substack{(i,j)\in E \\ i\in U \\ j\in N\setminus U}} x_{ij} \ge 1 \text{ for } 2 \le |U| \le |N| - 2$$

$$x_{ij} \in \{0,1\} \text{ for } (i,j) \in E.$$

TSP and BTSP are both *NP*-Hard problems. For an overview of the complexity results for TSP, we refer to Punnen (2002b). Kabadi and Punnen (2002) discusses the complexity results for BTSP.

Although TSP and BTSP are *NP*-Hard in general, there are special cases of TSP and BTSP that are solvable in polynomial time. These are not trivial cases and special algorithms are developed to solve them optimally. Deineko and Woeginger (2000) discuss the combinatorial nature of the solution spaces of several such TSPs. We refer the reader to the surveys of Kabadi (2002), Burkard *et al*. (1998), and Gilmore, Lawler, and Shmoys (1985) for further information.

Polynomially solvable cases of TSP and BTSP can be classified under two main categories:

(i) those having a special distance matrix, and

(ii) those that have a special graph structure.

In the first category, the graphs are complete graphs and a set of restrictions is defined over the edge weights. Whereas in the second one, there are restrictions on the graph structure but no restrictions are imposed on edge weights.

The studies on special cases mainly focus on TSP rather than on BTSP. However, there are a number of papers on special cases of BTSP. See for example Phillips, Punnen and Kabadi (1998), Van der Veen (1993), and Burkard and Sandholzer (1991). Vairaktarakis (2003) considers a polynomially solvable TSP and shows that the corresponding BTSP is *NP*-Hard.

We refer to the books of Gutin and Punnen (2002) and Lawler *et al*. (1985) for further information on TSP and BTSP.

2.3 Multiobjective Optimization

In classical optimization problems, there is a single objective function and the aim is to find a solution that optimizes the objective function value. However, many real

life problems have several objectives and decisions should be made by considering these objective functions simultaneously.

Typically, different objectives are conflicting with each other and a solution that performs well in one objective will not perform as well in other objectives. There are many solutions that do not outperform each other in all objectives. It is not clear which of these solutions are better until the decision maker (DM) or a group of DMs evaluates them.

A *multiobjective problem* (*MOP*) can be written as

$$\text{"}min\text{"}\ Cx = \left(f_1(x), f_2(x), \ldots, f_p(x) \right)$$
$$s.t. \quad x \in X$$

where $x \in \mathbb{R}^n$ is a feasible solution and X is the set of all feasible solutions. In this problem, there are p objective functions to be minimized and C is a $p \times n$ matrix. The q^{th} row of C corresponds to the q^{th} objective function, $f_q(x)$. We use the quotation marks since vector minimization is not a well-defined mathematical operation.

The point $y = \left(y_1, \ldots, y_p \right)^T \in \mathbb{R}^p$ such that $y = Cx$ is the outcome of the solution $x \in X$. The sets X and $Y = \left\{ y \in \mathbb{R}^p : y = Cx, x \in X \right\}$ are called the *decision space* and the *objective* (*criterion*) *space*, respectively. All vectors in objective space are column vectors of dimension $p \times 1$. For the sake of simplicity, we drop the transpose figure in our notation.

We assume that there exists no point $y \in Y$ that minimizes all objective functions simultaneously to avoid a trivial case. Hence we are interested with a set of "good" points instead of a single optimal solution. We use the *dominance* concept to define "good" points. We can consider the dominance concept as the multiobjective counter part of the optimality concept.

Point *y* is said to *dominate* point y' if and only if $y_q \le y'_q$ for all *q* and $y_q < y'_q$ for at least one *q*. If $y_q < y'_q$ for all *q* then *y* is said to *strictly dominate* y'. If there exists no $y' \in Y$ such that y' dominates *y*, then *y* is said to be *nondominated*. A point *y* is said to be *weakly nondominated* if and only if there exists no point $y' \in Y$ such that $y_q > y'_q$ for all *q*. The set of weakly nondominated points includes all nondominated points and some special dominated points.

Let Y_{ND} denote the set of nondominated points. The point $y^{ideal} = \left(y_1^{ideal}, \ldots, y_p^{ideal}\right)$ is said to be the *ideal point* where $y_q^{ideal} = \min_{y \in Y_{ND}} \left\{y_q\right\}$. Similarly, the *nadir point* is defined as $y^{nadir} = \left(y_1^{nadir}, \ldots, y_p^{nadir}\right)$ where $y_q^{nadir} = \max_{y \in Y_{ND}} \left\{y_q\right\}$.

Let $y \in Y_{ND}$ and y^{conv} be a convex combination of the nondominated points except *y*. That is;

$$y^{conv} = \sum_{y^k \in Y_{ND} \setminus \{y\}} w^k y^k \,, \quad \sum_{y^k \in Y_{ND} \setminus \{y\}} w^k = 1 \text{ and } w^k \ge 0 \text{ for } y^k \in Y_{ND} \setminus \left\{y\right\}.$$

Using these definitions, we define three types of nondominated points. A point $y \in Y_{ND}$ is said to be

- an *extreme supported nondominated* point if and only if there exists no y^{conv} such that $y^{conv} \le y$,
- a *nonextreme supported nondominated* point if and only if there exists a y^{conv} such that $y^{conv} = y$,
- an *unsupported nondominated* point if and only if there exists a y^{conv} such that $y^{conv} < y$.

The terms dominance and *efficiency* are counterparts of each other in the objective and decision spaces, respectively. A solution $x \in X$ is said to be *efficient* if and only if $y = Cx$ is nondominated and solution $x \in X$ is *inefficient* if and only if $y = Cx$ is

dominated. A solution $x \in X$ is *weakly efficient* if and only if $y = Cx$ is weakly nondominated. Similarly, we can define *extreme supported efficient*, *nonextreme supported efficient* and *unsupported efficient* solutions. We refer Steuer (1986) for an overview of the multiple criteria optimization theory, methodology and applications.

In Figure 2.1, y^1, y^2, y^6, and y^7 are extreme supported nondominated points, y^3 is nonextreme supported nondominated point, and y^4 is unsupported nondominated point. Points y^0 and y^8 are weakly nondominated but dominated. Point y^5 is strictly dominated.

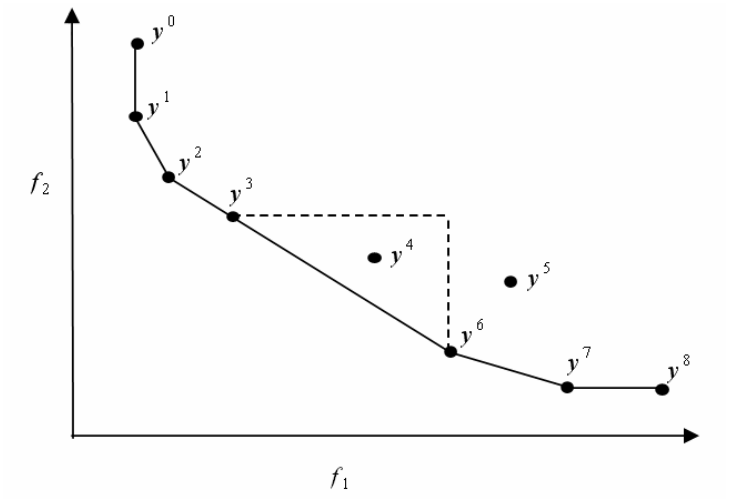


Figure 2.1 Different types of points in objective space

Throughout the thesis, we discuss our results mainly in the objective space and use Y_E for the set of extreme supported nondominated points. We should note that more than one efficient solution may correspond to the same nondominated point. In such cases, it is sufficient for our purposes to find only one of those efficient solutions.

A *multiobjective integer programming problem* (*MOIP*) with p objective functions can be written as:

$$\text{"}min\text{"}\ Cx = \left(f_1(x), f_2(x), \ldots, f_p(x) \right)$$
$$s.t. \quad x \in X$$

where $X = \{ Ax \le b, x \ge 0, x \in \mathbb{Z} \}$. A is a $m \times n$ matrix and $b \in \mathbb{R}^m$. The solution of the problem, $x \in \mathbb{Z}^n$ is the integer decision variable vector. Without loss of generality, we assume that $f_q(x) > 0$ and $f_q(x) \in \mathbb{Z}$ for $q = 1, \ldots, p$ and for all $x \in X$. Suppose $f_q(x) < 0$ for some q and $x \in X$, then we can shift the objective function value by adding a positive constant, c_q^{shift} which satisfies $\left(f_q(x) + c_q^{shift} \right) > 0$ for $q = 1, \ldots, p$ and for all $x \in X$. Similarly, suppose $f_q(x) \notin \mathbb{Z}$, then we can multiply the objective function value by a positive constant, $c_q^{multiply}$, so that the condition $\left(c_q^{multiply} f_q(x) \right) \in \mathbb{Z}$ is satisfied.

In a *multiobjective combinatorial optimization* (*MOCO*) problem, p weights are associated with each element of E. The weight of element e in objective q (q=1,…,p) is denoted by $w_q(e)$. The value of a solution x in objective q is $f_q(x)$. A MOCO problem can be defined as:

$$\text{"}\min_{x \in X}\text{"}\ f(x) = \left(f_1(x), f_2(x), \ldots, f_p(x) \right)$$

where $f_q(x)$ is a sum or a bottleneck objective. Ehrgott and Gandibleux (2000, 2002) review the MOCO theory, methodology and applications.

Multiobjective TSP and BTSP are examples of MOCO problems. In these problems, each edge is represented by several weights. These weights may correspond to different objectives such as cost, distance, risk, etc. associated with traversing an edge. Using the classification scheme of Ehrgott and Gandibleux (2002), a MOCO problem with p objectives can be denoted as *p-Σ TSP* if all objectives are TSP-type and *p-max TSP* if all objectives are BTSP-type. The notation p_1-*Σ* p_2-*max TSP* stands for a MOCO problem with p_1 TSP-type and p_2 BTSP-type objectives.

There are some recent studies on multiobjective TSP and BTSP. Some of these are heuristic approaches, some are local search methods, and some are exact algorithms. Ehrgott and Gandibleux (2002) review some of these approaches.

The number of efficient solutions is another important issue in MOCO problems. Finding all efficient solutions of a problem is said to be *intractable* if the number of efficient solutions may (potentially) increase exponentially as the size of the problem increases. It is known that *p-Σ TSP* is intractable for $p \geq 2$ (see Ehrgott, 2000).

In single objective optimization, enumerative algorithms, such as branch and bound or dynamic programming, use lower and upper bounds during their search. Using tighter bounds may decrease the size of the search space and the time required to find the optimal solution. Similarly we can use bounds in enumerative algorithms for multiobjective optimization. The ideal and nadir points as defined earlier in this section can be used as lower and upper bounds for the nondominated point set, respectively. However, these bounds may not be very useful in reducing the search space because the ideal and nadir points may be far away from the nondominated points set. Due to this, using a set of points instead of a single point may be more useful in reducing the search space. These sets are called bound sets. We refer to Ehrgott and Gandibleux (2007) for a discussion on the bound sets.

2.4 Approximation Algorithms for Multiobjective Problems

Many multiobjective approaches attempt to find all nondominated points (or efficient solutions). However this can be a difficult task if finding such points is time consuming, or if the number of such points is large, i.e., the problem may be intractable. It may be reasonable to generate a set of points that represents the nondominated points well. This set provides useful information to the DM, although not as complete as the whole nondominated set. The points in this set may be nondominated points (found by an exact algorithm) or approximations (found by a heuristic approach) of the nondominated points.

Cohon (1978) proposed an exact approximation algorithm for MOIPs with two objectives. Solanki, Appino and Cohon (1993) proposed an exact approximation algorithm for MOLPs with three or more objectives. We refer to Ruzika and Wiecek (2005) and Ehrgott and Gandibleux (2004) for exact and heuristic approximation algorithms for multiobjective optimization problems, respectively.

CHAPTER 3

PYRAMIDAL TOURS AND MULTIPLE OBJECTIVES

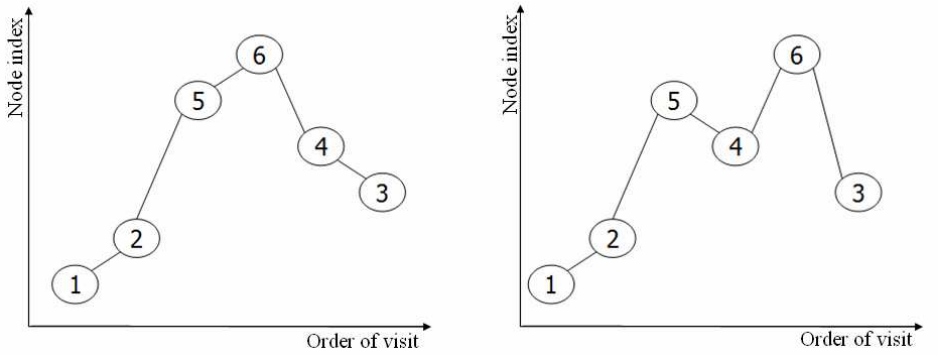
3.1 Introduction

In this chapter, we work on TSP and BTSP that have special matrix structures and lead to polynomially solvable cases. We extend the problems to multiple objectives and investigate the properties of nondominated points. We develop a pseudo-polynomial time algorithm to find a nondominated point for any number of objectives. Finally, we propose an approach to generate all nondominated points for the biobjective case. To the best of our knowledge, there exists no other study that addresses the polynomially solvable special cases of the multiobjective TSP.

3.2 Pyramidal Tours

A tour ρ is pyramidal if starting from node 1, a set of nodes are visited in ascending order up to node n and the remaining nodes are visited in descending order. Formally, tour ρ is called pyramidal if $\rho = (1, i_1, i_2, \ldots, i_k, n, j_1, j_2, \ldots, j_m)$ such that $1 < i_1 < i_2 < \ldots < i_k < n$ and $n > j_1 > j_2 > \ldots > j_m > 1$.

Consider two tours, $\rho = (1,2,5,6,4,3)$ and $\varphi = (1,2,5,4,6,3)$. In Figure 3.1, we plot the node numbers in the order they are visited. The plot of tour ρ (Figure 3.1a) looks like a pyramid and it has only one peak. Tour ρ is a pyramidal tour. On the other hand, the plot of tour φ has two peaks and φ is a non-pyramidal tour.



a) A pyramidal tour b) A non-pyramidal tour

Figure 3.1 Plots of the tours

In this chapter, we use the notation $d[i, j]$ instead of w_{ij} in order to differentiate between node indices and the objective function index. A complete graph has an edge directly connecting each pair of nodes. If the cost of traversing an edge is independent of the direction of the traverse for all edges (i.e., $d[i, j] = d[j, i]$ for all (i,j) pairs) then the graph is said to be undirected (symmetric). If $d[i, j] \neq d[j, i]$ for some (i,j) pair then the graph is said to be directed (asymmetric). We mainly use the term *edge* for the undirected graphs and *arc* for the directed graphs. Gutin, Yeo and Zverovitch (2002) prove that the number of pyramidal tours is 2^{n-3} in an undirected complete graph and 2^{n-2} in a directed graph. In both cases, the number of pyramidal tours is an exponential function of the number of nodes, n. Let F_{PYR} be the set of all

pyramidal tours for a given graph. By definition, $F_{PYR} \in F$ where F is the set of all tours.

Although the number of pyramidal tours is exponential in n, finding the shortest pyramidal tour for any distance matrix has a complexity of $O(n^2)$ using the dynamic program given in Gilmore, Lawler and Shmoys (1985). Let $\mathbf{D}_{PYR}$ denote the family of distance matrices for which a pyramidal tour is optimal. Then, for any matrix in $\mathbf{D}_{PYR}$, TSP is polynomially solvable.

Tour improvement (*TI*) technique is a proof technique developed by Van der Veen (1994). TI is used to prove that for a class of matrices in $\mathbf{D}_{PYR}$, the optimal tour is pyramidal. TI starts with an initial tour and iterates by exchanging a set of arcs with others to obtain a new tour and generates a sequence of tours without increasing the tour length. The new tour's length must be at most as large as that of the previous tour in order for this exchange to be a feasible transformation. TI is a framework of feasible transformations that needs to be developed for each class of matrices in $\mathbf{D}_{PYR}$. A feasible transformation for a class may not be feasible for another.

There are different classes of distance matrices in $\mathbf{D}_{PYR}$ that have been defined in the literature. There are symmetric and asymmetric matrices in $\mathbf{D}_{PYR}$. Burkard *et al.* (1998) applied TI technique to Monge, Supnick, Demidenko, Kalmanson, Van der Veen matrices and generalized distribution matrices. Let $\mathbf{D}_{TI} \in \mathbf{D}_{PYR}$ denote the set of matrix classes for which TI technique can be applied.

Note that, for a matrix in $\mathbf{D}_{TI}$, a non-pyramidal tour φ may also be optimal, giving the same length as the optimal pyramidal tour. A trivial case is a TSP where all arc lengths are equal. TI technique implies that there exists at least one pyramidal tour ρ, that can be obtained from φ by applying a sequence of feasible transformations. If φ is optimal, ρ should also be optimal. This is possible if all feasible transformations used to obtain ρ from φ keep the tour length unchanged, i.e. none

of the feasible transformations improve the tour length. By definition, feasible transformations cannot increase the tour length. On the other hand, if all possible feasible transformations strictly decrease the tour length, then a non-pyramidal tour cannot be optimal, since for every non-pyramidal tour, there exists at least one pyramidal tour that has a strictly shorter length.

A matrix that is in $\mathbf{D}_{PYR}$ may not be readily recognizable and may require a renumbering of the nodes to be recognized. There are polynomial time algorithms for recognizing some of these matrices in $\mathbf{D}_{PYR}$ (see, for example, Burkard and Deineko, 2004, and Burkard, Klinz and Rudolf, 1996).

3.3 The Multiobjective TSP

In multiobjective problems, nondominated points are important. The ability to find nondominated points is an important challenge in multiobjective combinatorial problems, many of which are *NP*-Hard. We first present some properties of the nondominated points for multiobjective TSPs having distance matrices in $\mathbf{D}_{TI}$. We then address finding nondominated points for these problems.

Let us define d_q as the q^{th} distance matrix, $d_q[i, j]$ as the length of arc (i,j) in q^{th} objective function, and $d_q(\varphi)$ as the length of tour φ in q^{th} objective function. The point $d(\varphi) = \left(d_1(\varphi), \ldots, d_p(\varphi)\right)$ is in the objective space and corresponds to tour φ.

Theorem 3.1. If all distance matrices are in the same class of $\mathbf{D}_{TI}$, then for each non-pyramidal tour there exists at least one pyramidal tour that is at least as good in every objective and possibly better in some objectives.

Proof: Since we assume that all distance matrices are in the same class of $\mathbf{D}_{TI}$, any feasible transformation does not increase the tour length in any of the objectives. In the worst case, TI results with a pyramidal tour having equal lengths in all objectives

to those of the initial tour. If any of the feasible transformations used in any of the objectives is positive, then the resulting pyramidal tour dominates the initial tour. □

Corollary 3.1. If all distance matrices are in the same class of $\mathbf{D}_{TI}$, there exists a pyramidal tour corresponding to each nondominated point.

Proof: Follows directly from Theorem 3.1. □

Corollary 3.2. If all distance matrices are in the same class of $\mathbf{D}_{TI}$, and all feasible transformations in all distance matrices are improving, then each non-pyramidal tour is strictly dominated by at least one pyramidal tour.

Proof: Follows directly from Theorem 3.1. □

Example. The following Van der Veen matrix, $\mathbf{D}_{VDV} \in \mathbf{D}_{TI}$ shows that there may be pyramidal tours strictly better than any other tour. The pyramidal tour ρ=(1,2,3,4) is shorter than any other tour for the following matrix. Therefore, no tour can be as good as ρ for this objective.

$$d = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 4 & 7 \\ 2 & 4 & 0 & 3 \\ 3 & 7 & 3 & 0 \end{bmatrix} \in \mathbf{D}_{VDV} \in \mathbf{D}_{TI}$$

Remark. If distance matrices belong to different classes in $\mathbf{D}_{TI}$, then a non-pyramidal tour may correspond to a unique nondominated point.

Consider the following $\mathbf{D}_{TI}$ matrices (Van der Veen, 1994) where $d_1 \in \mathbf{D}_{VDV}$ and $d_2 \in \mathbf{D}_{DEMI}$ (Demidenko matrix).

$$d_1 = \begin{bmatrix} 0 & 4 & 2 & 4 \\ 4 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 \end{bmatrix} \in \mathbf{D}_{VDV} \text{ and } d_2 = \begin{bmatrix} 0 & 4 & 4 & 2 \\ 4 & 0 & 1 & 0 \\ 4 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix} \in \mathbf{D}_{DEMI}$$

$\rho^1 = (1,2,3,4)$, $\rho^2 = (1,2,4,3)$ and $\varphi = (1,4,2,3)$ are all possible tours where the set of pyramidal tours is $F_{PYR} = \{\rho^1, \rho^2\}$. The tour lengths are $d(\rho^1)$=(8,7), $d(\rho^2)$=(7,8) and $d(\varphi)$=(7,7), respectively. $d(\varphi)$ dominates both $d(\rho^1)$ and $d(\rho^2)$, hence it is the only nondominated point for these two matrices.

Let $d(\rho^q) = (d_1(\rho^q), \ldots, d_p(\rho^q))$ be the point corresponding to shortest pyramidal tour ρ^q with respect to q^{th} distance matrix, i.e, $d_q(\rho^q) \leq d_q(\rho)$ for any $\rho \in F_{PYR}$.

Theorem 3.2. If distance matrices belong to different classes in $\mathbf{D}_{TI}$, then $d(\rho^q)$ is weakly nondominated.

Proof: Since ρ^q is the shortest tour in objective *q*, $d(\rho^q)$ is weakly nondominated. However, $d(\rho^q)$ may be dominated as demonstrated in Remark. □

For *p-Σ TSP*, finding nondominated points is an important problem. We may wish to find all nondominated points or a subset that could be of interest to the DM.

Let us define a convex combination of *p* matrices that are in the same class of $\mathbf{D}_{TI}$ as:

$$d_\mu = \sum_{q=1}^{p} \mu_q . d_q, \sum_{q=1}^{p} \mu_q = 1 \text{ and } \mu_q \geq 0 \; q = 1, \ldots, p \; .$$

In the following theorem we prove that this matrix is also in the same class of $\mathbf{D}_{TI}$.

Theorem 3.3. If all distance matrices are in the same class of $\mathbf{D}_{TI}$, then any convex combination of these matrices is also in the same class of $\mathbf{D}_{TI}$.

Proof: Currently existing classes of $\mathbf{D}_{TI}$ are defined with inequalities like

$d_q[i_1,i_2]+d_q[i_3,i_4]\le d_q[i_5,i_6]+d_q[i_7,i_8]$ for some q and $i_1,\ldots,i_8$ or,

$d_q[i_1,i_2]+d_q[i_3,i_4]+d_q[i_5,i_6]\le d_q[i_7,i_8]+d_q[i_9,i_{10}]+d_q[i_{11},i_{12}]$ for some q and $i_1,\ldots,i_{12}$

Since all convex combinations of these types of inequalities hold for all possible $i_1,\ldots,i_{12}$ used in the class definition, we conclude that all convex combinations are also in the same class. □

By convex combination of matrices, *p-Σ TSP* becomes a single objective TSP. For a given μ vector, an extreme supported nondominated point can be found using the DP given by Gilmore, Lawler and Shmoys (1985). We refer to this DP as $DP_{GLS}(d)$ where d is distance matrix. All extreme supported nondominated points can be found by choosing suitable μ vectors. In the next section, we show a method for determining suitable μ for *2-Σ TSP*. In Chapter 5, we develop a method for determining suitable μ vectors for MOIPs with any number of objective functions.

Let us define the following two problems, (P_k) and (P_k^ε) for k^{th} objective function. Let $d_k^\varepsilon = d_k + \varepsilon\sum_{\substack{q=1\\q\neq k}}^{p} d_q$ be a distance matrix. Given a set of upper bounds by B_q $q\neq k$, (P_k) finds the solution with shortest possible tour length in objective k. (P_k^ε), on the other hand, finds a tour corresponding to a nondominated point and satisfying constraints (1) and (2).

$$(P_k)\ min \sum_{i=1}^{n} d_k[i,\varphi(i)]$$

$$st \quad d_q(\varphi)=\sum_{i=1}^{n} d_q[i,\varphi(i)]\le B_q \ \forall q\neq k \qquad (1)$$

$$\varphi\in F \qquad (2)$$

where F is the set of all Hamiltonian tours and B_q is an upper bound for objective q. Let $B=\left(B_1,...,B_{k-1},B_{k+1},...,B_p\right)$ be the vector of upper bounds, B_q for criteria $q \neq p$.

$$\left(P_k^{\varepsilon}\right) \; min \sum_{i=1}^{n} d_k^{\varepsilon}[i,\varphi(i)]$$
$$st \quad (1) \text{ and } (2)$$

Let y_k^* be the optimal solution of problem $\left(P_k\right)$ and $y^*=\left(y_1^*,...,y_p^*\right)$ be the vector of tour lengths for the optimal tour of problem $\left(P_k\right)$. Similarly, for problem $\left(P_k^{\varepsilon}\right)$, let $y_k^{*\varepsilon}$ and $y^{*\varepsilon}=\left(y_1^{*\varepsilon},...,y_p^{*\varepsilon}\right)$ be the optimal solution and optimal tour length vector, respectively. Note that, different solutions can be obtained by changing ε in $\left(P_k^{\varepsilon}\right)$. The ε value should be positive to avoid dominated points but small enough to ensure $y_k^{*\varepsilon}=y_k^*$. Steuer (1986) showed an appropriate interval for the ε value for the augmented weighted Tchebycheff program. For our case, the appropriate value of ε can be determined through Theorem 3.4.

Theorem 3.4. Let $y^*=\left(y_1^*,...,y_p^*\right)$, $y^{*\varepsilon}=\left(y_1^{*\varepsilon},...,y_p^{*\varepsilon}\right)$ be the optimal solutions to $\left(P_k\right)$ and $\left(P_k^{\varepsilon}\right)$, respectively, and $y=\left(y_1,...,y_p\right)$ be any nondominated point satisfying (1) and (2). Then for any $\varepsilon \in \left(0,\left(\max_{y\in S}\sum_{q\neq k}\left[y_q^{*\varepsilon}-y_q\right]\right)^{-1}\right)$ we have $y_k^{*\varepsilon}=y_k^*$ where $S=\left\{(1)\cap(2)\cap\left(y_k>y_k^{*\varepsilon}\right)\right\}$.

Proof: It is known that $y^{*\varepsilon}$ is a nondominated point for any $\varepsilon>0$.

As ε value increases, the relative importance of k^{th} objective decreases. Increasing the ε value does not improve $y_k^{*\varepsilon}$. Then, ε should be small enough to ensure $y_k^{*\varepsilon}=y_k^*$. This implies,

$$y_k^{*\varepsilon}+\varepsilon\sum_{q\neq k}y_q^{*\varepsilon} < y_k+\varepsilon\sum_{q\neq k}y_q \tag{3}$$

for all nondominated points satisfying $y_k>y_k^*$, (1) and (2).

Since (3) needs to hold for all nondominated points satisfying $y_k > y_k^*$, $y_k^{*\varepsilon}$ cannot be greater than y_k^*, thus $y_k^* \geq y_k^{*\varepsilon}$. It is impossible to have $y_k^* > y_k^{*\varepsilon}$. Hence, $y_k^{*\varepsilon} = y_k^*$. If we can find suitable a ε satisfying inequality (3), we can rewrite it as

$$\varepsilon \sum_{q \neq k} \left[y_q^{*\varepsilon} - y_q \right] < y_k - y_k^{*\varepsilon}$$

We have two cases:

(i) If $\sum_{q \neq k} \left[y_q^{*\varepsilon} - y_q \right] > 0$ then $\varepsilon < \dfrac{y_k - y_k^{*\varepsilon}}{\sum_{q \neq k} \left[y_q^{*\varepsilon} - y_q \right]}$.

Since the minimum possible value for $y_k - y_k^{*\varepsilon} = 1$ assuming (without loss of generality) that all edge lengths are integers, we need

$$\varepsilon < \frac{1}{\sum_{q \neq k} \left[y_q^{*\varepsilon} - y_q \right]}.$$

A general bound over all nondominated points is then

$$\varepsilon < \frac{1}{\max\limits_{y \in S} \left\{ \sum_{q \neq k} \left[y_q^{*\varepsilon} - y_q \right] \right\}}$$

where

$$S = (1) \cap (2) \cap \left(y_k > y_k^{*\varepsilon} \right) \cap \left(\sum_{q \neq k} \left[y_q^{*\varepsilon} - y_q \right] > 0 \right)$$

(ii) If $\sum_{q \neq k} \left[y_q^{*\varepsilon} - y_q \right] \leq 0$ then setting $\varepsilon \geq 0$ is sufficient. Since $y_k^{*\varepsilon} < y_k$ and $\sum_{q \neq k} y_q^{*\varepsilon} \leq \sum_{q \neq k} y_q$, we have

$$y_k^{*\varepsilon} + \varepsilon \sum_{q \neq k} y_q^{*\varepsilon} < y_k + \varepsilon \sum_{q \neq k} y_q \text{ for any } \varepsilon \geq 0.$$

The range is defined as:

$$\varepsilon \in \left(0, \left(\max_{y \in S} \sum\nolimits_{q \neq k} \left[y_q^{*\varepsilon} - y_q \right] \right)^{-1} \right) \text{ where } S = \left\{ (1) \cap (2) \cap \left(y_k > y_k^{*\varepsilon} \right) \right\} \qquad \square$$

The above theorem gives the upper bound $\gamma_k = \left(\max_{y \in S} \sum_{q \neq k} \left[y_q^{*\varepsilon} - y_q \right] \right)^{-1}$ for ε for objective *k*. By taking the minimum γ_k, we generalize the upper bound for all objectives as follows.

Corollary 3.3. Replacing the range of ε with $\varepsilon \in \left(0, \min_k \gamma_k\right)$ in Theorem 3.4, the theorem is generalized for any number of objective functions.

To determine the above range, we need to know the set of nondominated points. This set may not be readily available, but this is not a problem in practice. A trivial upper bound for ε can be obtained by finding the total length of the longest *n* arcs, say UB_q, and substituting y_q for y_q^q, the shortest tour length in objective *q*, using $\sum_q \left[UB_q - y_q^q \right]$. Then the range $\varepsilon \in \left(0, \left(\sum_q \left[UB_q - y_q^q \right] \right)^{-1} \right)$ is a practical and valid range.

We develop a dynamic program to find the optimal pyramidal tour that solves $\left(P_k^{\varepsilon}\right)$. We define a state variable vector $R = \left(R_1, ..., R_{k-1}, R_{k+1}, ..., R_p\right)$. R_q corresponds to the remainder or the unconsumed portion of bound B_q by the partial tour constructed so far. Initially $R = B$ and as DP moves to inner stages, R decreases.

The DP we developed is quite different than DP_{GLS}. Given a distance matrix, DP_{GLS} finds the shortest pyramidal tour considering a single objective function. On the other hand, our DP considers multiple objective functions by imposing upper bounds to all but one objective. The DP either finds the shortest pyramidal tour that does not violate the upper bounds or reports that no such tour exists.

Let $C(i, j, R)$ be the length of the shortest Hamiltonian path with respect to the k^{th} distance matrix from *i* to *j* on cities 1,2,…,*max*{*i*,*j*} that visits a subset of these nodes in a descending order from *i* to 1 and the remaining nodes in ascending order from 1

to *j* without violating the bounds, R. $C(i, j, R)$ finds the shortest pyramidal path in criterion *k* from *i* to *j* while the bounds R_q for $q \neq k$, are not violated. Let *M* be a sufficiently large number, i.e., $M > n.\max_{i,j}\{d_k[i, j]\}$. At state $C(i, j, R)$, there are five possible cases. If any of the upper bounds is violated then the corresponding component of *R* vector is negative. In this case (Case 1), DP returns *M* value for the current state. If |*i*-*j*|>1 and the bounds are not violated then we consider Cases 2 or 4. In both cases, the selection of the next state is straight forward in order to keep the path pyramidal. In Case 2, arc (*j*-1,*j*) is added to the path. In Case 4, arc (*i*,*i*-1) is added to the path. In Cases 3 and 5, |*i*-*j*|=1 and the bounds are not violated. In these cases, the selection of the next state is not straight forward. The minimum valued state is selected among the possible states. In all cases except for the first one, the remaining bounds are updated according to the selection of the next state.

$$C(i, j, R) = \begin{cases} 1)\ M & \text{if } R_q < 0 \text{ for any } q \\ 2)\ C(i, j-1, R - d[j-1, j]) + d_k^{\varepsilon}[j-1, j] & \text{for } i < j\text{-}1 \text{ and } R_q \geq 0\ \forall q \neq k \\ 3)\ \min_{l<i}\{C(i, l, R - d[l, j]) + d_k^{\varepsilon}[l, j]\} & \text{for } i = j\text{-}1 \text{ and } R_q \geq 0\ \forall q \neq k \\ 4)\ C(i-1, j, R - d[i, i-1]) + d_k^{\varepsilon}[i, i-1] & \text{for } i > j+1 \text{ and } R_q \geq 0\ \forall q \neq k \\ 5)\ \min_{l<j}\{C(l, j, R - d[i, l]) + d_k^{\varepsilon}[i, l]\} & \text{for } i = j+1 \text{ and } R_q \geq 0\ \forall q \neq k \end{cases}$$

Number of states in this DP is $O\left(n^2 \prod_{\substack{q=1 \\ q \neq k}}^{Q} B_q\right)$. The number of states is a function of the magnitudes of the upper bounds. Hence, this DP has pseudo-polynomial complexity. The optimal objective function value to $\left(P_k^{\varepsilon}\right)$ is given by

$$DP\left(d_k^{\varepsilon}, B, d\right) = \min\{C(n-1, n, B - d[n, n-1]) + d_k^{\varepsilon}[n, n-1], \\ C(n, n-1, B - d[n-1, n]) + d_k^{\varepsilon}[n-1, n]\}.$$

Note that in $DP\left(d_k^{\varepsilon}, B, d\right)$, d_k^{ε} is a distance matrix, *B* is a vector of upper bounds, *d* is a vector of distance matrices, and *d*[*i*,*j*] is the vector of arc lengths. If there is no

feasible solution for the given bounds, then the DP will return an objective function value of at least *M*. This DP finds the shortest pyramidal tour for any distance matrix.

The ranges developed for ε value in Theorem 3.4 and Corollary 3.3 are valid in general. If all distance matrices are in the same class of $\mathbf{D}_{TI}$, as stated in Theorem 3.1, then the optimal tour to $\left(P_k^{\varepsilon}\right)$ is obtained.

All nondominated points can be found by this DP by changing the B_q values. In the next section, we propose an approach for finding all nondominated points for the biobjective TSP.

The above DP can be used for both symmetric and asymmetric matrices. If all distance matrices are symmetric then DP can be simplified as follows:

$$C(i,j,R)=\begin{cases} M & \text{if } R_q<0 \text{ for any } q \\ C\left(i',j'-1,R-d\left[j'-1,j'\right]\right)+d_k^{\varepsilon}\left[j'-1,j'\right] & \text{for } i'<j'\text{-1 and } R_q\geq 0\ \forall q\neq k \\ \min\limits_{l<i'}\left\{C\left(i',l,R-d\left[l,j'\right]\right)+d_k^{\varepsilon}\left[l,j'\right]\right\} & \text{for } i'=j'\text{-1 and } R_q\geq 0\ \forall q\neq k \end{cases}$$

where $i'=\min(i,j)$ and $j'=\max(i,j)$. In this case, the optimal objective function value also simplifies to

$$DP\left(d_{\mathrm{k}}^{\varepsilon},B,d\right)=C\left(n-1,n,B-d[n,n-1]\right)+d_k^{\varepsilon}\left[n,n-1\right].$$

3.4 The Biobjective TSP

We develop an approach to generate all nondominated points for the biobjective TSP. We first find the extreme supported nondominated points by using the weighting scheme proposed by Aneja and Nair (1979). Then we search for the nonextreme supported nondominated and unsupported nondominated points between each adjacent pair of extreme supported nondominated points.

We define nonextreme supported nondominated and unsupported nondominated points as *nonextreme nondominated* points, because we do not need to differentiate

between these two types of points in our method. Let Y_E and Y_{NE} be the sets of extreme supported nondominated points and nonextreme nondominated points, respectively.

Consider the optimal objective function values of the single objective TSPs, $y_q^q = \min_{\varphi \in F} d_q(\varphi)$, and let φ^q be the corresponding optimal tours for q=1, 2. Let $y_2^1 = d_2(\varphi^1)$, $y_1^2 = d_1(\varphi^2)$, $y^1 = (y_1^1, y_2^1)$ and $y^2 = (y_1^2, y_2^2)$. Without loss of generality, assume that $y_1^1 < y_1^2$ and $y_2^1 > y_2^2$. If $y^1 = y^2$ or $y^1 \leq y^2$ or $y^2 \leq y^1$ then there is a unique nondominated point and the problem is trivial.

Theorem 3.5. Using ε in the range $\left(0, \min\left\{\frac{1}{y_2^1 - y_2^2}, \frac{1}{y_1^2 - y_1^1}\right\}\right)$ is sufficient to avoid nondominated points in problem $\left(P_k^{\varepsilon}\right)$.

Proof: Follows directly from Theorem 3.4. □

Note that both points y^1 and y^2 can be weakly nondominated. The nondominated points, y^{q-eff} having $y_q^q = y_q^{q-eff}$ q=1,2 can be determined by the DP we developed. Using these points a larger upper bound for ε can be obtained.

Corollary 3.4. The upper bound for ε can be replaced by the following term: $\min\left\{\frac{1}{y_2^{1-eff} - y_2^2}, \frac{1}{y_1^{2-eff} - y_1^1}\right\}$.

Proof: The points y^1 and y^2 may be weakly nondominated. An overestimated range (for nondominated points) is obtained by the denominator term using y^1 and y^2. If nondominated points are used in the denominator, the range (for nondominated points) may decrease and the upper bound value for ε may increase. □

We define two algorithms to find all points in Y_E; *Recursive*$\left(y^a, y^b\right)$ and *A1*. *Recursive*$\left(y^a, y^b\right)$ finds all extreme supported nondominated points between two given extreme supported nondominated points y^a and y^b. *A1* solves $\left(P_1^\varepsilon\right)$ and $\left(P_2^\varepsilon\right)$. If two different solutions are obtained in *A1* then *Recursive*$\left(y^a, y^b\right)$ is called. *DP*(d_q) finds the shortest pyramidal tour with respect to distance matrix d_q and returns the point $y = \left(y_1, y_2\right)$

We can obtain the extreme supported nondominated points in $O(n^2)$ using distance matrices d_1^ε and d_2^ε. The extreme supported nondominated points can be determined by changing the weight μ of matrix $d^\mu = \mu d_1 + (1-\mu) d_2$, $\mu \in (0,1)$ and applying $DP_{GLS}\left(d^\mu\right)$ for the resulting single objective problem. For each weight set, a solution is obtained in $O(n^2)$.

A1

Initialization: Set $Y_E = \varnothing$.

Step 1. Solve $DP_{GLS}\left(d_1^\varepsilon\right)$, let the optimal point be y^1.

Step 2. Solve $DP_{GLS}\left(d_2^\varepsilon\right)$, let the optimal point be y^2.

Step 3. If $y_1^1 < y_1^2$ and $y_2^2 < y_2^1$ then go to Step 5 else go to Step 4.

Step 4. If $y^1 = y^2$ then single optimal solution is y^1, $Y_E = \left\{y^1\right\}$, go to Step 7.

Step 5. $Y_E = \left\{y^1, y^2\right\}$.

Step 6. Call *Recursive* $\left(y^1, y^2\right)$.

Step 7. Terminate the algorithm.

Recursive $\left(y^a, y^b\right)$

Step 1. Set $\mu = \dfrac{y_2^a - y_2^b}{\left(y_1^a - y_2^a\right) - \left(y_1^b - y_2^b\right)}$.

Step 2. Set $d^{\mu} = \mu d_1 + (1-\mu) d_2$.

Step 3. Solve $DP_{GLS}\left(d^{\mu}\right)$, let the solution be y^{new}.

Step 4. If $Y_E \cap \left\{y^{new}\right\} = \left\{y^{new}\right\}$ then go to Step 5

else

$Y_E = Y_E \cup \left\{y^{new}\right\}$,

Call *Recursive* $\left(y^a, y^{new}\right)$,

Call *Recursive* $\left(y^{new}, y^b\right)$.

Step 5. Terminate the algorithm.

If $\left|Y_E\right| \geq 2$ then we search for the nonextreme nondominated points using the DP we developed. Since there are only two objectives, we use the state variable $C(i, j, R)$ where R is a scalar. For each consecutive extreme supported nondominated point pair in Y_E, nonextreme nondominated points should be searched between them. A point is obtained in $O(n^2B)$ where B is the upper bound on one of the objectives. Without loss of generality, we select to use the second objective as a bound and minimize the augmented version of the first objective. Algorithm *A2* is used to find all nonextreme nondominated points in Y_{NE}. We refer to our DP as $DP\left(d_1^{\varepsilon}, B_2, d_2\right)$ where d_1^{ε} and d_2 are distance matrices and B_2 is a scalar.

A2

Initialization: Set $Y_{NE} = \varnothing$.

Sort elements of Y_E, such that $y_1^{[1]} < y_1^{[2]} < \ldots < y_1^{[|Y_E|]}$.

Set r=1.

Step 1. Solve $DP2\left(d_1^{\varepsilon}, y_2^{[r]} - 1, d_2\right)$, let the resulting point be y.

Step 2. If $y = y^{[r+1]}$ then r=r+1.

If $r = \left|Y_E\right|$ then go to Step 6 else go to Step 1.

Step 3. $Y_{NE} = Y_{NE} \cup \{y\}$.

Step 4. If $y_2 - 1 = y_2^{[r]} z_2 - 1 = z_2^{[r]}$ then $r=r+1$, go to Step 1.

Step 5. Solve $\left(d_1^{\varepsilon}, y_2^{[r]} - 1, d_2\right)$, let the resulting point be y, go to Step 2.

Step 6. Terminate the algorithm.

The set of nondominated points is $Y_{ND} = Y_E \cup Y_{NE}$. We can find a nondominated point satisfying the given bound with a pseudo-polynomial DP. However, the complexity of identifying all nondominated points is still an open problem.

3.5 The Bottleneck TSP

The optimal pyramidal tour for the Bottleneck TSP can be found in O(n^2) with a small modification in DP. Burkard and Sandholzer (1991) studied the polynomially solvable special cases of the Bottleneck TSP. They presented several conditions for pyramidally solvable Bottleneck TSPs.

One may be curious to know whether some results on pyramidal tours are applicable to the bottleneck-type objectives. For some classes in $\mathbf{D}_{PYR}$, using the "maximum" operator instead of the "sum" operator in the distance matrix definition results in a class which is also in $\mathbf{D}_{PYR}$. The class of Monge matrices is such an example (Burkard and Sandholzer, 1991). In a similar way, we can define the bottleneck version of the Van der Veen matrix as follows:

$$\mathbf{D}_{BVDV} = \left\{ d[i, j] \middle| \begin{array}{l} d[i, j] = d[j, i] \text{ for all i and j} \\ \max\{d[i, j], d[j+1, k]\} \le \max\{d[i, k], d[j, j+1]\} \text{ for all i<j<j+1<k} \end{array} \right\}.$$

Theorem 3.6. $\mathbf{D}_{BVDV} \not\in \mathbf{D}_{PYR}$.

Proof: We provide a counter example. $d \in \mathbf{D}_{BVDV}$ for y≥1. For the distance matrix given below, the length of tour ρ=(1,4,2,6,3,5) is 1. However, all pyramidal tours have tour lengths of y.

$$d = \begin{bmatrix} - & y & 0 & 0 & 1 & y \\ y & - & y & 0 & 0 & 1 \\ 0 & y & - & y & 1 & 1 \\ 0 & 0 & y & - & y & 0 \\ 1 & 0 & 1 & y & - & 0 \\ y & 1 & 1 & 0 & 0 & - \end{bmatrix} \in \mathbf{D}_{BVDV} \notin \mathbf{D}_{PYR}$$

Increasing the value of y in $d \in \mathbf{D}_{BVDV}$ the lengths of pyramidal tours can be increased arbitrarily. □

We next define the bottleneck version of the Demidenko matrix as follows:

$$\mathbf{D}_{BDEMI} = \left\{ d[i, j] \middle| \begin{array}{l} d[i, j] = d[j, i] \text{ for all i and j} \\ \max\{d[i, j], d[j+1, k]\} \le \max\{d[i, j+1], d[j, k]\} \text{ for all i<j<j+1<k} \end{array} \right\}.$$

Theorem 3.7. $\mathbf{D}_{BDEMI} \notin \mathbf{D}_{PYR}$.

Proof: We provide a counter example. $d \in \mathbf{D}_{BDEMI}$ for y≥0. For the distance matrix given below, the length of tour ρ=(1,5,3,4,2,6) is 0. However, all pyramidal tours have tour lengths of y.

$$d = \begin{bmatrix} - & y & y & y & 0 & 0 \\ y & - & 0 & 0 & 0 & 0 \\ y & 0 & - & 0 & 0 & y \\ y & 0 & 0 & - & 0 & y \\ 0 & 0 & 0 & 0 & - & 0 \\ 0 & 0 & y & y & 0 & - \end{bmatrix} \in \mathbf{D}_{BDEMI} \notin \mathbf{D}_{PYR}$$

Increasing the value of y in $d \in \mathbf{D}_{BDEMI}$ the lengths of pyramidal tours can be increased arbitrarily. □

Since $\mathbf{D}_{BVDV}$ and $\mathbf{D}_{BDEMI}$ do not guarantee that the optimal tour is pyramidal, we consider the bottleneck type objectives no further.

3.6 Discussions

In this chapter, we studied the multiobjective TSP on $\mathbf{D}_{TI} \in \mathbf{D}_{PYR}$ and showed some properties of nondominated points. We developed a pseudo-polynomial DP to find a nondominated point to the problem when all distance matrices are in the same class of $\mathbf{D}_{TI}$. For the biobjective case, we developed an approach to find all nondominated points. We also demonstrated that bottleneck types of Van der Veen matrices and Demidenko matrices are not in $\mathbf{D}_{PYR}$, and hence the developments are not applicable to these cases.

CHAPTER 4

HALIN GRAPHS AND MULTIPLE OBJECTIVES

4.1 Introduction

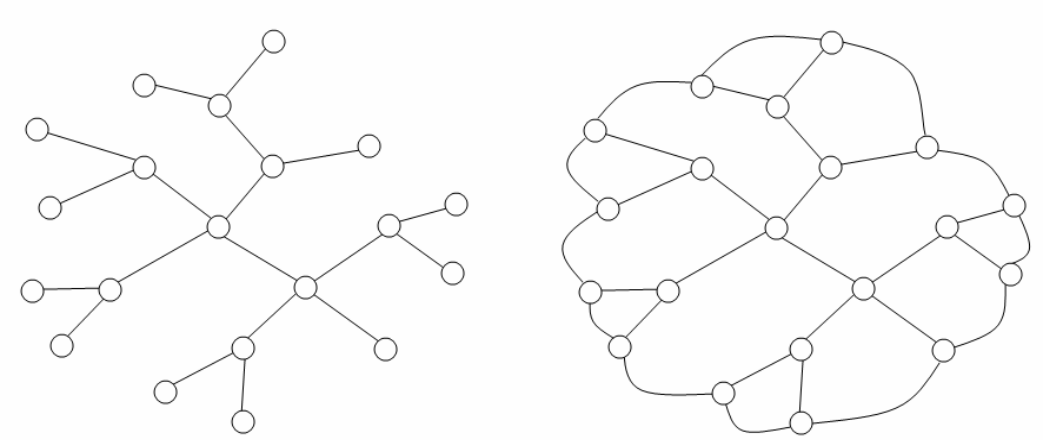
In this chapter, we study TSP and BTSP on special graphs called Halin graphs. Although both problems are *NP*-Hard on general graphs, they are polynomially solvable on Halin graphs. We address the multiobjective versions of these problems. We show computational complexities of finding a single nondominated point as well as finding all nondominated points for different objective function combinations. We develop algorithms for the polynomially solvable combinations.

4.2 Definitions and Background

Some definitions on Halin graphs are provided in this section. We review TSP and BTSP on Halin graphs and discuss the polynomial algorithms to solve these problems.

In a graph, the number of edges incident to a node gives the *degree* of that node. An undirected planar graph is called a *Halin graph* if it is a combination of a tree with no nodes of degree two and a cycle passing through the leaf nodes of the tree (see for example Kabadi, 2002). An example of such a tree and a Halin graph constructed

using this tree is given in Figure 4.1. The leaf nodes of the tree are called the *outer nodes* of the Halin graph and an edge set that connects the outer nodes is called a *cycle*. The remaining nodes of the Halin graph are called the *internal nodes*. If a Halin graph has only one internal node, it is called a *wheel* (see Figure 4.2b). Let t be an internal node adjacent to exactly one other internal node. Let $L(t)$ be the set of outer (leaf) nodes adjacent to node t. Then the subgraph of H induced by the set $\{t\} \cup L(t)$ is called a *fan*, and t is the center of the fan. In Figure 4.2a, we demonstrate a fan where $t = 1$ and $L(1) = \{1a, 1b, ..., 1x\}$.



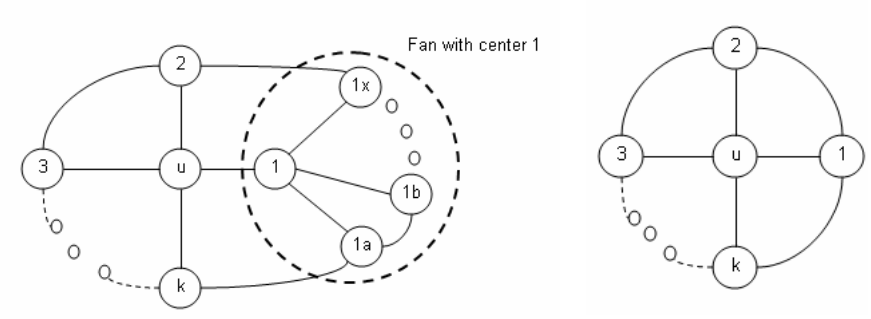
a) A tree having no nodes with degree two b) A Halin graph

Figure 4.1 Constructing a Halin graph

Theorem 4.1. (Cournejols, Naddef and Pulleyblank, 1983). A Halin graph H which is not a wheel has at least two fans.

Let H be a Halin graph that has at least two fans. Let $H(t)$ denote the graph obtained by shrinking the fan centered at t into a single node t. In Figure 4.2a, we show a graph H and a fan centered at node 1. In Figure 4.2b, we show graph $H(1)$ obtained

after shrinking this fan into node 1. Note that, $H(1)$ is a wheel. The shrinking operation can be used as a part of an algorithm as we discuss later.



a) A Halin graph, H b) A wheel, $H(1)$

Figure 4.2 Shrinking H to $H(1)$

Theorem 4.2. (Cournejols, Naddef and Pulleyblank, 1983). If $\{t\} \cup L(t)$ is a fan in a Halin graph, H, then $H(t)$ is also a Halin graph.

Cournejols, Naddef and Pulleyblank (1983) showed that TSP on Halin graphs can be solved in O(n). Coullard *et al*. (1993) developed an algorithm that solves the 2-Connected Steiner Subgraph Problem on Halin graphs in O(n). Since TSP is a special case of this problem, the algorithm developed by Coullard *et al*. (1993) also solves TSP on Halin graphs in O(n). Phillips, Punnen and Kabadi (1998) developed an O(n) algorithm for BTSP. Throughout this paper, we will refer to the algorithm of Cournejols, Naddef and Pulleyblank (1983) as *CNP* and to the algorithm of Phillips, Punnen and Kabadi (1998) as *PPK*.

In a Halin graph, each fan is connected to the rest of the graph with exactly three edges. For example, the fan centered at node 1 in Figure 4.2a is connected to the rest of the graph with edges (k,1a), (u,1) and (2,1x). In a tour, exactly two of these three

edges are used. If edge (u,1) is used then there are two possibilities to construct a tour: either edge (k,1a) or edge (2,1x) is included in the tour. If edge (k,1a) is included in the tour then nodes in $L(1)$ are visited by path 1a,1b,…,1x,1. If, on the other hand, edge (2,1x) is included in the tour then nodes in $L(1)$ are visited by path 1,1a,1b,…,1x. If edge (u,1) is not used in the tour then both edges (2,1x) and (k,1a) must be included in the tour. The nodes in $L(1)$ can be visited in $|L(1)|-1$ different ways with path 1a, 1b, …,1j, 1, 1(j+1),…, 1x for some node 1j. The decisions on which of the above edges are selected depend on the respective objective function values in *CNP* and *PPK*.

In each iteration of *CNP*, first a fan is selected. For example, for the fan centered at node 1 in Figure 4.2a, *CNP* selects the best node 1j for the pair (k,1a) and (2,1x). Then lengths of the paths corresponding to the three pairs, (u,1) and (k,1a), (u,1) and (2,1x), and (k,1a) and (2,1x), are calculated. In order to eliminate the fan and shrink the graph, new weights of edges (k,1), (u,1) and (2,1) in H(1) are determined by solving a system of three linear equations. *CNP* keeps shrinking the Halin graph until obtaining a wheel. The TSP is solved on the wheel and the optimal tour is obtained.

Although *CNP* solves TSP, it is not directly applicable to BTSP. *PPK* also uses the approach of shrinking the graph. The basic idea behind *PPK* is in the updating scheme of the weights. *PPK* defines penalties for the pairs of edges in addition to the edge weights. It keeps track of the longest edge, the second longest edge, and the pair of edges with highest, second highest, and third highest penalties and updates this information after each shrinking.

Both algorithms are straightforward if edge (u,1) of H is used since there are only two alternative paths to construct the tour. However, if (u,1) is not used, then the other two edges have to be used, and the algorithm needs to make the optimal selection for node 1j based on the objective function used. In *CNP*, the updated edge weights (those obtained after previous shrinking operations) are used during this

selection. However, in *PPK*, the selection is done using both the edge weights and the penalties of edge pairs. In some cases, there may not be a single best selection for node 1*j* and alternative optimal selections may exist. In these cases, both *CNP* and *PPK* break ties arbitrarily. However, if there are multiple objectives in the problem then all of the objectives must be considered to break ties.

4.3 Multiple Objective TSP and BTSP on Halin Graphs

In this section, we first work on biobjective cases and then extend the results to more than two objectives. For all problems considered in this section, we assume that a Halin graph, with multiple weights assigned to each of its edges, is given. We first consider the biobjective cases: *2-Σ TSP*, *1-Σ 1-max TSP* and *2-max TSP*. We then generalize the results to multiobjective cases. For each case, we define two problems: finding a single nondominated point and finding all nondominated points. In the remainder of the section, we refer to the Halin graph given in Figure 4.3 as *H* and use it to demonstrate our proofs.

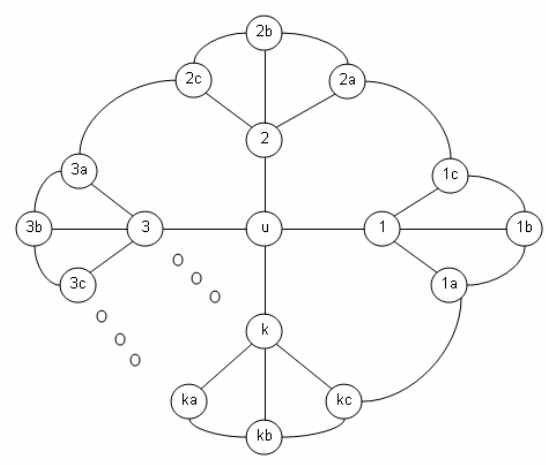


Figure 4.3 A Halin graph, *H*

4.3.1 *2-Σ TSP*

Finding all nondominated points

There are $4k+1$ nodes in H given in Figure 4.3. Cournejols, Naddef and Pulleyblank (1983) state that there are $k\times 2^{k-2}$ different tours in H without a proof. We next prove this result .

Theorem 4.3. Let $H=(N,E)$ and $|N|=4k+1$. There are $k\times 2^{k-2}$ different tours in H.

Proof: Define a set of edges $D=\{(1c,2a),(2c,3a),...,((k-1)c,ka),(kc,1a)\}$, and observe that $|D|=k$. Each tour on H contains k-1 of the k edges of D. Assume that edge ($1c$,$2a$) is not in tour φ. Then the edges (u, 1), (1, $1c$), ($1c$, $1b$), ($1b$, $1a$), ($1a$, kc), (ka,(k-1)c), ((k-1)a,(k-2)c), …, (3a, 2c), (2c, 2b), (2b, 2a), (2a, 2) and (2, u) must be in tour φ. Consider nodes 3, 3a, 3b and 3c. Tour φ uses the edge (2c, 3a) to reach these nodes and leaves these nodes using the edge (3c, 4a). These four nodes in the tour φ can be visited either by path (3a, 3, 3b, 3c) or by path (3a, 3b, 3, 3c). Similarly, for each node quartet (i, ia, ib, ic) i=3,…, k-1, there are two possible ways of visiting them all. For a fixed element of D, say (1c, 2a), there are 2^{k-2} tours. Since $|D|=k$, we can fix k different edges and obtain 2^{k-2} tours for each fixed edge. Hence, there are $k\times 2^{k-2}$ different tours in H. □

Theorem 4.4. The number of nondominated points in H is exponential in k.

Proof: Let

$w_1(e)=2^j$ and $w_2(e)=0$ for edge (ja, jb) for j=1,…,k,

$w_1(e)=0$ and $w_2(e)=2^j$ for edge (jb, jc) for j=1,…,k,

$w_1(e)=w_2(e)=M$ for edge (1c, 2a) where M is such a big number (i.e. $M>2^k$) that edge (1c, 2a) cannot be used in any efficient tour, and

$w_1(e)=w_2(e)=0$ for all other edges in H.

The tour length of any tour with respect to each objective is at least 2^1+2^2 since edges (1c, 1b), (1b, 1a), (2c, 2b) and (2b, 2a) are used in every efficient tour. If the length

of a tour in the first objective is represented in binary digits then the j^{th} digit of the number corresponding to the tour length is set to 1 if edge (*ja*, *jb*) is used and to 0 if edge (*jb, jc*) is used in the tour. Similarly, for the second objective, the j^{th} digit is set to 0 if edge (*ja, jb*) is used and 1 if edge (*jb, jc*) is used. Since each digit represents the selection of an edge and there are *k*-2 such edges, totally 2^{k-2} different numbers can be written in binary digits. The same number of distinct tour lengths can be also obtained for the second objective. Since the sum of the two objective function values is constant (and is equal to $\sum_{j=1}^{k} 2^j = 2^{k+1} - 2$), each of the 2^{k-2} tours is efficient in *H*. Since these tours have different objective function values, each of them corresponds to a different nondominated point in the objective space. □

Since the number of the nondominated points is exponential, the problem of identifying all of them in *H* is intractable.

Finding a nondominated point

CNP solves the single objective TSP for any given Halin graph. The point found by *CNP* may be dominated when we consider multiple objectives. The following perturbation in the weights guarantees to obtain a nondominated point using *CNP*:

$$w_1^{\varepsilon}(e) = w_1(e) + \varepsilon w_2(e) \quad \forall e \in H$$

where ε is a sufficiently small positive constant to avoid weakly nondominated but dominated points. In Theorem 3.5, we developed an appropriate range of ε value for *2-Σ TSP*.

Let *w*(*e*) be a convex combination of two weights:

$$w(e) = \lambda w_1(e) + (1-\lambda) w_2(e) \quad \forall e \in H, \ \lambda \in [\varepsilon, 1-\varepsilon].$$

CNP can be used with this weight set to find an extreme supported nondominated point. Aneja and Nair (1979) develop a method to find all extreme supported nondominated points by systematically varying the λ value for the biobjective transportation problem. In Section 3.4, we implemented this approach on *2-Σ TSP*.

All extreme supported nondominated points can be found using the above approach. In order to find any nondominated point, we may use a variation of the ε-constraint approach (see for example Steuer, 1986, pp 202-206). We impose an upper bound U on the first objective and minimize the second objective, breaking ties in favor of the first objective. This corresponds to problem *P1*:

$$P1: \min\ f_2(\varphi) = \sum_{e\in\varphi}\left[w_2(e) + \varepsilon w_1(e)\right]$$

st.

$$f_1(\varphi) = \sum_{e\in\varphi} w_1(e) \le U$$

$$\varphi \in F$$

Consider graph H given in Figure 4.3. Let H' be a special case of H such that $w_1(e) = w_2(e) = M$ for all edges (u, j) j=2,…,k-1, where M is such a big number that these edges cannot be in any efficient tour, $w_1(e) = w_2(e) = 0$ for edges (u, 1) and (u,k), $w_1(e) = w_2(e) = 0$ for all edges (jc, (j+1)a) j=1,…,k-1 and for (kc,1a), $w_1(e) = w_2(e) = 0$ for all edges (j, ja), (j, jb) and (j, jc) j=1,…,k. In Theorem 4.5, we prove that solving *P1* on H' is *NP*-Hard in the ordinary sense.

Theorem 4.5. Problem *P1* on H' is *NP*-Hard in ordinary sense.

Proof: Every efficient tour starts with path (u, 1, 1a, 1b, 1c, 2a) and ends with path ((*k*-1)c, ka, kb, kc, k, u). Tour lengths are determined by the selections of the paths to visit the inner nodes at fans centered at nodes 2,3,…,*k*-1. As in the proof of Theorem 4.3, node j can be visited in two different ways, (ja, j, jb, jc) or (ja, jb, j, jc). Let us define a binary decision variable X_j, j=1,…,k, such that

$$X_j = \begin{cases} 1 \text{ if path } (ja, jb, j, jc) \text{ is selected} \\ 0 \text{ if path } (ja, j, jb, jc) \text{ is selected} \end{cases}$$

Define penalties

$$p_1(j) = w_1(ja, jb) - w_1(jb, jc) \text{ and}$$

$$p_2(j) = \left[w_2(ja, jb) - w_2(jb, jc)\right] + \varepsilon p_1(j) \text{ for each } j,\ j=1,\ldots,k.$$

Problem *P1* can be redefined as

$$P2: min \sum_{j=1}^{k} \left[p_2(j) X_j + w_2(jb, jc) + \varepsilon w_1(jb, jc) \right]$$

$$st.$$

$$\sum_{j=1}^{k} \left[p_1(j) X_j + w_1(jb, jc) \right] \le U$$

$$X_j \in \{0,1\} \quad j = 1, ..., k.$$

Since $w_1(jb, jc)$, $w_2(jb, jc)$ and ε are constants, $w_2(jb, jc) + \varepsilon w_1(jb, jc)$ can be dropped from the objective function. Let $p_2'(j) = -p_2(j)\ \forall j$ and $U' = U - \sum_{j \in J} w_1(jb, jc)$. Then *P2* is transformed to *P3*.

$$P3: max \sum_{j=1}^{k} p_2'(j) X_j$$

$$st.$$

$$\sum_{j=1}^{k} p_1(j) X_j \le U'$$

$$X_j \in \{0,1\} \quad j = 1, ..., k.$$

The transformations from *P1* to *P2* and from *P2* to *P3* are both polynomial. *P3* is a 0-1 Knapsack problem, which is *NP*-Hard in the ordinary sense. Since the 0-1 Knapsack problem is a special case of *P1* on H', *P1* on H' is also *NP*-Hard in the ordinary sense. □

For H', we can use the pseudo polynomial dynamic program developed for knapsack problems (Martello and Toth, 1990) to solve *2-Σ TSP*. However, the graph structure of H' is a special case of *H*, and hence solving *P*1 on a general Halin graph may be more difficult than solving it on H'. Further analyzing the computational complexity of *P*1 and developing an algorithm for solving *2-Σ TSP* on general Halin graphs are future research topics.

For *2-Σ TSP*, we showed that each extreme supported nondominated point can be found in O(*n*) by using convex combinations of two weight sets. However, finding other nondominated points (both nonextreme supported and unsupported) is *NP*-Hard

in the ordinary sense for a special case of Halin graphs. We also showed that finding all nondominated points is intractable. We next consider *1-Σ 1-max TSP*.

4.3.2 *1-Σ 1-max TSP*

Finding a nondominated point

Let $f_1(\varphi)=\sum_{e\in\varphi} w_1(e)$ and $f_2(\varphi)=\max_{e\in\varphi}\{w_2(e)\}$. We develop an algorithm that finds a nondominated point for this problem. Our algorithm uses *CNP* and *PPK* simultaneously. The inputs of the algorithm are the weights of the edges and an upper bound *U* on the *1-max* objective. The output is the minimum *1-Σ* objective value satisfying the upper bound *U* on the *1-max* objective.

Max_Algorithm

Input: $w_1(e)$ and $w_2(e)$ for all $e\in E$, and *U*.

Output: A tour φ with minimum $f_1(\varphi)$ where $f_2(\varphi)\le U$

Steps of the algorithm

Step 0. Using a sufficiently large number *M*, update the edge weights as follows:

$$w_1'(e)=\begin{cases} w_1(e) \text{ if } w_2(e)\le U \\ M \text{ if } w_2(e)>U \end{cases} \qquad e\in E$$

Step 1. Run *CNP* with $w_1'(e)$ weight set and calculate the pair penalties of the BTSP-type objective function with the selections of *CNP* (not with the selections of *PPK*). Break the ties of *CNP* using the penalties of *PPK*. Let the optimal tour be φ^* and the objective function value be $f_1(\varphi^*)$.

Step 2. If $f_1(\varphi^*)<M$ then φ^* is optimal; stop. Otherwise, there is no solution satisfying the given upper bound.

In Step 0 of *Max_Algorithm*, $w_2(e)$ values are compared with *U*. If the weight of an edge in the second objective is larger than *U*, then its weight in the first objective is

set to *M*. This comparison and updating is done in O(|*E*|). By Euler's formula, $|E| \le 3n-6$ for $|E| \ge 3$ for planar graphs (see Bondy and Murthy, 1979). Since Halin graphs are planar graphs, $O(|E|) = O(n)$. Step 1 uses *CNP* and *PPK*, both algorithms run in O(*n*). Step 2 runs in O(1). Overall complexity of *Max_Algorithm* is O(*n*).

In Step 1, we use *CNP* and *PPK* simultaneously. In the original *PPK* algorithm, best selections are made using the BTSP-type objective. We force the *PPK* algorithm to use the selections that are best for the *CNP* algorithm. The selections of the *PPK* algorithm are important only if there is a tie in the *CNP* algorithm's selections

Theorem 4.6. Given a Halin graph and an upper bound *U* on the *1-max* objective, a nondominated point to *1-Σ 1-max TSP* can be found in O(*n*).

Proof: The *Max_Algorithm* finds a nondominated point in O(*n*). □

Finding all nondominated points

In Theorem 4.3, we proved that the number of tours grows exponentially. Hence the objective function value for *1-Σ* may take an exponential number of distinct values. However, the number of distinct values for *1-max* objective is bounded by the number of distinct edge weights and that is bounded by the number of edges. The number of edges is bounded by O(*n*). Therefore, the number of nondominated points for *1-Σ 1-max TSP* is bounded by O(*n*) due to the *1-max* objective.

The complexity of finding a nondominated point is O(*n*) and the number of nondominated points is bounded by O(*n*). All nondominated points can be determined in O(n^2) using the *Max_Algorithm* by systematically varying the upper bound *U*. We develop the *Iterative_Algorithm* to find all nondominated points.

Iterative_Algorithm

Input: $w_1(e)$ and $w_2(e)$ for all $e \in E$.

Output: Set of nondominated points

Steps of the algorithm

Step 0.a. Sort $w_2(e)$ values in nonincreasing order and let

$\Psi = \{\psi^{[1]}, \psi^{[2]}, ..., \psi^{[r]}\}$ be the set of distinct edge weights, such that $\psi^{[i]} > \psi^{[i+1]}$,

Set $S=\varnothing$.

Step 0.b. Call *Max_Algorithm* on $w_1(e)$ with $U=\psi^{[1]}$ and let the optimal tour be φ^{CNP}.

$f^{CNP} = \left(f_1(\varphi^{CNP}), f_2(\varphi^{CNP}) \right)$.

Step 0.c. Call *PPK* on $w_2(e)$ and let the optimal tour be φ'.

Call *Max_Algorithm* on $w_1(e)$ with $U= f_2(\varphi')$ and let the optimal tour be φ^{PPK}.

$f^{PPK} = \left(f_1(\varphi^{PPK}), f_2(\varphi^{PPK}) \right)$,

If $f^{CNP} = f^{PPK}$ then set $S = S \cup \{\varphi^{PPK}\}$ and stop since there is a single optimal solution, else go Step *0.d.*

Step 0.d. Set c=0, $\varphi^c = \varphi^{CNP}$ and $f_2(\varphi^c) = f_2(\varphi^{CNP})$.

Step 1. Let $j = \underset{i=1,..,r}{argmax}\{\psi^{[i]} < f_2(\varphi^c)\}$, set $U = \psi^{[j]}$,

$c= c+1$,

Call *Max_Algorithm,* let the solution be φ^c,

if $f_1(\varphi^c) \le f_1(\varphi^{PPK})$ then set $S = S \cup \{\varphi^c\}$ and go to Step 1 else go to Step 2.

Step 2. The set of nondominated points is *S.*

In each iteration of the above algorithm, *Max_Algorithm* is called. In the worst case, all edges have distinct weights and *Iterative_Algorithm* calls *Max_Algorithm* for each edge. Since the number of distinct edge weights is bounded by O(n), the complexity of *Iterative_Algorithm* is O(n^2).

Theorem 4.7. Given a Halin Graph, all nondominated points to *1-Σ 1-max TSP* can be found in O(n^2) .

Proof: The *Iterative_Algorithm* finds all nondominated points in O(n^2). □

4.3.3 *2-max TSP*

Finding a nondominated point

Finding a nondominated point of *2-max TSP* is easier than that of *1-Σ 1-max TSP*. We use *PPK* twice and find a nondominated point for a given upper bound on one of the objective values. We refer to this algorithm as *2Max_Algorithm.* This algorithm finds a nondominated point to *2-max TSP* in O(n).

2Max_Algorithm

Input: $w_1(e)$ and $w_2(e)$ for all $e \in E$, and U.

Output: A nondominated point $f(\varphi) = (f_1(\varphi), f_2(\varphi))$ with minimum $f_1(\varphi)$ where $f_2(\varphi) \le U$

Steps of the algorithm

Step 0. Using a sufficiently large number M, update the edge weights as follows:

$$w_1'(e) = \begin{cases} w_1(e) & \text{if } w_2(e) \le U \\ M & \text{if } w_2(e) > U \end{cases} \qquad e \in E$$

Step 1. Run *PPK* with $w_1'(e)$ weight set. Let the optimal tour be φ^* and the objective function value be $f_1(\varphi^*)$ and $f_2(\varphi^*)$.

Step 2. If $f_1(\varphi^*) \geq M$ then stop, there is no solution satisfying the given upper bound. Else go to Step 3.

Step 3. Using a sufficiently large number *M*, update the edge weights as follows:

$$w_2'(e) = \begin{cases} w_2(e) & \text{if } w_1(e) \leq f_1(\varphi^*) \\ M & \text{if } w_1(e) > f_1(\varphi^*) \end{cases} \qquad e \in E$$

Step 4. Run *PPK* with $w_2'(e)$ weight set. Let the optimal tour be φ^{**} and the objective function values be $f_1(\varphi^{**})$ and $f_2(\varphi^{**})$.

Step 5. Report the point $f(\varphi^{**})$ as the nondominated point satisfying the given upper bound.

Finding all nondominated points

This problem is similar to *1-Σ 1-max TSP*. *Iterative_Algorithm* can be modified by replacing *Max_Algorithm* with *PPK* in Step 0.c and replacing *Max_Algorithm* with *2Max_Algorithm* in Step 0.b and Step 1. We will refer to this algorithm as *2Iterative_Algorithm.* The complexity of identifying all nondominated points of *2-max TSP* is $O(n^2)$ by using *2Iterative_Algorithm*.

Up to now, we discussed three biobjective problems. In the following subsections, we will generalize the results to multiobjective problems.

4.3.4 p_1-Σ p_2-max TSP

There are p_1 TSP-type and p_2 BTSP-type objectives in this problem. For p_1=2, we know that finding a nondominated point is *NP*-Hard and finding all nondominated points is intractable. Since increasing p_1 does not simplify the problems, the same complexity results are valid for $p_1 \geq 2$. We next consider the remaining two cases: $p_1 = 1$ and $p_1 = 0$.

4.3.5 *1-Σ p-max TSP*

In Theorems 4.6 and 4.7, we showed that a nondominated point can be found in $O(n)$ for *1-Σ 1-max TSP* and all nondominated points for this problem can be found in $O(n^2)$. In this subsection, we generalize these results for *1-Σ p-max TSP*.

Finding a nondominated point

For this problem, we consider that an upper bound is introduced for each BTSP-type objective, and the TSP-type objective is minimized subject to these p upper bounds. A nondominated point can be obtained with a modification in Step 0 of *Max_Algorithm*. An updating is done for the distance matrix corresponding to the BTSP-type objective in the original *Max_Algorithm*. This updating operation should be done for all p BTSP-type objectives. The complexity of the updating is $O(pn)$. *Max_Algorithm* can be used to find a nondominated in $O(n)$. The overall complexity of finding a nondominated point is then $O(pn)$.

Finding all nondominated points

Each BTSP-type objective function can take $O(n)$ distinct values. In the worst case, there are $O(n^p)$ distinct combinations of objective function values for p BTSP-type objectives. On the other hand, the TSP-type objective function can have $O\left(2^{|E|}\right)$ distinct values. So we conclude that the number of the nondominated points is bounded by $O(n^p)$. For each combination of upper bounds of p BTSP-type objectives, the modified *Max_Algorithm* can be used. The set of nondominated points can be identified in $O(pn^{p+1})$.

4.3.6 *p-max TSP*

We showed that, for *2-max TSP*, a nondominated point can be found in $O(n)$ using *2Max_Algorithm* and all nondominated points can be identified in $O(n^2)$ using *2Iterative_Algorithm*. In this subsection, we generalize these results for *p-max TSP*.

Finding a nondominated point

For this problem, we consider that an upper bound is introduced for all BTSP-type objectives but the last one. The last BTSP-type objective is minimized subject to these p-1 upper bounds. A nondominated point can be obtained by the *2Max_Algorithm* after two modifications on this algorithm. An updating is required for the distance matrix corresponding to the BTSP-type objective in Step 0 of the *2Max_Algorithm*. This updating operation should be done for all p BTSP-type objectives. This can be done in O(pn). *PPK* is called in the first step of the algorithm. Steps 3 and 4 must be executed for each of the p-1 objectives in order to ensure finding a nondominated point. After these modifications, the complexity of the *2Max_Algorithm* is still O(pn).

Finding all nondominated points

Each BTSP-type objective function can take O(n) distinct values. Since we are minimizing one of the objectives, in the worst case, the remaining p-1 BTSP-type objectives may have $O(n^{p-1})$ distinct combinations of objective function values. Hence, the number of nondominated points is bounded by $O(n^{p-1})$ and all nondominated points can be identified in $O(pn^{p})$ in the worst case.

4.4 Discussions

In this chapter, we considered polynomially solvable special cases of two problems, TSP and BTSP. Although both problems are *NP*-Hard in general, there exist polynomial algorithms when these problems are defined on Halin graphs. We addressed the multiobjective versions of these problems with various combinations of objective functions.

We showed that, when there are two or more TSP-type objective functions in the problem then finding a nondominated point is *NP*-Hard and there are exponentially many nondominated points. However, if there is at most one TSP-type objective function in the problem and all remaining objectives are BTSP-type, then the

problem is polynomially solvable. We developed algorithms to find nondominated points.

To summarize, we showed the complexity results for all possible combinations of TSP and BTSP-type objectives for multiobjective problems on Halin graphs and we developed polynomial time algorithms where possible.

CHAPTER 5

AN EXACT ALGORITHM TO FIND ALL EXTREME SUPPORTED NONDOMINATED POINTS IN MULTIOBJECTIVE PROBLEMS

It is possible to find all extreme supported nondominated points of a biobjective integer programming problem in the objective space using the algorithm of Cohon (1978) and Aneja and Nair (1979). However, this algorithm is not directly applicable to problems with three or more objectives. In this chapter, we develop an exact algorithm to find all extreme supported nondominated points of a multiobjective problem. We propose several properties to improve the algorithm. We test our algorithm on the Assignment, the Knapsack and the Traveling Salesperson Problems with three and four objectives.

5.1 Introduction

Consider the following single objective integer program, $MOIP(\lambda)$, which has a weighted sum objective function:

$$\begin{aligned} &min\ \lambda Cx = \sum_{q=1}^{p} \lambda_q f_q(x) \\ &s.t. \quad x \in X \end{aligned}$$

where $\lambda \in \mathbb{R}^p_>$ and $\mathbb{R}^p_> = \{\lambda \in \mathbb{R}^p : \lambda_q > 0, q = 1, \ldots, p\}$. The optimal solution of $MOIP(\lambda)$ is an extreme supported nondominated point of *MOIP* for any $\lambda \in \mathbb{R}^p_>$. As we defined in Chapter 2, Y_E is the set of all extreme supported nondominated points.

For a multiobjective linear program, *MOLP*, all efficient solutions are supported (Steuer, 1986). There are algorithms to generate all efficient solutions of *MOLP*, see for example the ADBASE algorithm developed by Steuer (1989). Benson and Sun (2002) proposed an algorithm to find all nondominated points in the objective space for *MOLP*, instead of studying in the decision space.

However, research about this issue on *MOIP*s is very limited compared to that of *MOLP*. For biobjective integer problems, Cohon (1978) and Aneja and Nair (1979) developed similar algorithms to find all extreme supported nondominated points. They proposed a systematic way of varying the weights of the objective functions. However, to the best of our knowledge, only Przybylski, Gandibleux and Ehrgott (2007) propose an algorithm to find all extreme supported nondominated points of *MOIP* with three or more objective functions.

In this chapter, our aim is to develop an algorithm to find all extreme supported nondominated points of *MOIP*s. In Section 2, we review the algorithms of Cohon (1978) and Aneja and Nair (1979). We discuss the study and the algorithm of Przybylski, Gandibleux and Ehrgott (2007) in Section 3. In Section 4, we present additional definitions and our algorithm. In Section 5, we discuss possible improvements. In Section 6, we report the computational results on the test problems. In the last section, we conclude the chapter.

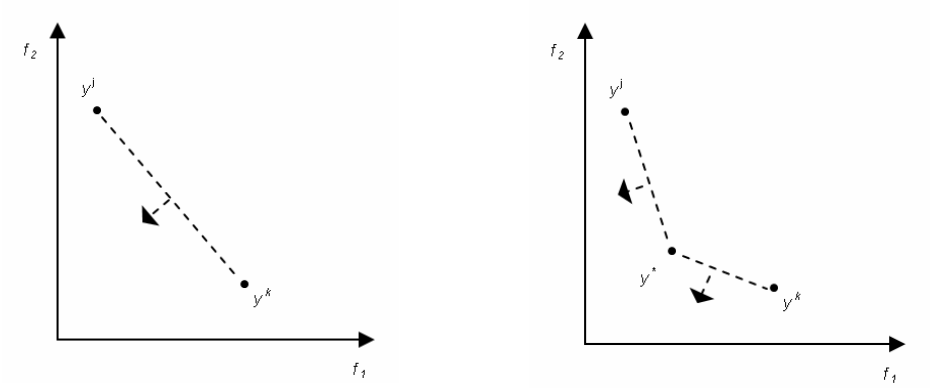
5.2 Algorithms for Biobjective Integer Problems

In this section, we review the algorithm developed by Cohon (1978) and Aneja and Nair (1979) for the biobjective integer problems. Cohon (1978) calls the algorithm

as Noninferior Set Estimation (NISE). We will call this algorithm as the *CAN* algorithm due to the initials of the authors.

CAN keeps a list *L*. The elements of *L* are extreme supported nondominated point pairs. In each iteration of *CAN*, a pair of extreme supported nondominated points, say (y^k, y^j), is selected from a list *L*. Consider points y^k and y^j in Figure 5.1a. The normal vector λ of the line passing through these points is calculated such that $\lambda y^k = \lambda y^j$. $MOIP(\lambda)$ is solved with this λ. Let y^* be the point corresponding to the optimal solution of $MOIP(\lambda)$. If $\lambda y^* < \lambda y^k$, as in Figure 5.1b, then $y^* \in Y_E$. New point y^* is recorded and two new pairs (y^k, y^*) and (y^*, y^j) are added to *L*. If $\lambda y^* = \lambda y^k$ then *CAN* concludes that no more extreme supported nondominated points can be obtained from this pair. At the end of the iteration, the pair (y^k, y^j) is removed from *L*. *CAN* stops when the list *L* is empty.

To initialize *CAN*, the pair (y^1, y^2) is generated such that $y_q^q = \min_{y^k \in Y_E} \{y_q^k\}$ q=1,2. These points can be obtained by solving $MOIP(\lambda)$ with $\lambda = (1-\varepsilon, \varepsilon)$ and $\lambda = (\varepsilon, 1-\varepsilon)$ where ε is a very small positive problem instance specific constant to avoid dominated points. Aneja and Nair (1979) showed that the algorithm stops exactly after $2|Y_E| - 3$ iterations if $|Y_E| > 2$.



(a)Stage (y^j,y^k) (b) Stages (y^j,y^*) and (y^*,y^k)

Figure 5.1 An example iteration of *CAN*

In biobjective problems, if the nondominated points are sorted in increasing order of their first objective function values, then they are naturally sorted in decreasing order of their second objective function values. This is a special property of biobjective problems and is not valid for problems with three or more objectives. By this property, for any pair $\left(y^k, y^j\right)$, both weights are strictly positive, $\lambda \in \mathbb{R}^2_>$.

Solanki, Appino and Cohon (1993) and Przybylski, Gandibleux and Ehrgott (2007) point out two difficulties in generalizing *CAN* to problems with $p \geq 3$ objectives. The first difficulty is the determination of initial points. Since *p* points are needed to define a hyper plane in $\mathbb{R}^p$, we have to select *p* points minimizing each objective function. If $p = 2$ then, for each objective *q*, there exists only one point $y^q \in Y_E$ such that $y^q_q = \min_{y^k \in Y_E} \left\{ y^k_q \right\}$. Due to the special property of the biobjective problems, $y^r_q = \max_{y^k \in Y_E} \left\{ y^k_q \right\}$ for $r \neq q$. However, if $p \geq 3$, then for each objective *q*, there may be more than one distinct point $y^q \in Y_E$ such that $y^q_q = \min_{y^k \in Y_E} \left\{ y^k_q \right\}$. Thus, the selection of the initial problems may be problematic. The second difficulty is about the normal vector of the hyperplane passing through *p* points. In biobjective problems, the normal vector always has positive components. However for $p \geq 3$, it may have

some negative components. Using such a λ vector in $MOIP(\lambda)$ may result in dominated points.

We will use an example developed by Tenfelde-Podehl (2003) throughout this chapter to explain the algorithms and their details. Consider an assignment problem with *p* objectives:

$$\text{"}min\text{"}\ Cx = \left(\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{1} x_{ij}, \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{2} x_{ij}, \ldots, \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{p} x_{ij} \right)$$

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, \ldots, n$$

$$x_{ij} \in \{0,1\} \quad i = 1, \ldots, n, \ \ j = 1, \ldots, n$$

This problem has three objective functions and the following cost matrices:

$$C^1 = \begin{pmatrix} 3 & 6 & 4 & 5 \\ 2 & 3 & 5 & 4 \\ 3 & 5 & 4 & 2 \\ 4 & 5 & 3 & 6 \end{pmatrix},\ C^2 = \begin{pmatrix} 2 & 3 & 5 & 4 \\ 5 & 3 & 4 & 3 \\ 5 & 2 & 6 & 4 \\ 4 & 5 & 2 & 5 \end{pmatrix} \text{ and } C^3 = \begin{pmatrix} 4 & 2 & 4 & 2 \\ 4 & 2 & 4 & 6 \\ 4 & 2 & 6 & 3 \\ 2 & 4 & 5 & 3 \end{pmatrix}.$$

There are four extreme supported nondominated points in this example problem: $y^1 = (11,11,14)$, $y^2 = (15,9,17)$, $y^3 = (19,14,10)$ and $y^4 = (13,16,11)$. The first three points are the unique optimal points of the corresponding single objective problems. Przybylski, Gandibleux and Ehrgott (2007) used this example and showed that a direct implementation of *CAN* is not able to find these four points. By solving single objective problems, the first three points are found. However, the plane passing through these three points has a negative element in its normal, and y^4 cannot be found.

5.3 The Algorithm of Przybylski, Gandibleux and Ehrgott (2007)

In this section, we discuss the algorithm proposed by Przybylski, Gandibleux and Ehrgott (2007) to find Y_E of a MOIP with $p \geq 3$. We call this algorithm the *PGE* algorithm. They develop some properties for the weight space of a MOIP and use these properties in the *PGE* algorithm. Some of these properties will also be useful in the proofs related to our algorithm.

In the *PGE* algorithm, they use the weight space decomposition approach of Benson and Sun (2000). Benson and Sun (2002) developed an algorithm to find Y_E of a *MOLP*. However their algorithm is not applicable to *MOIP* since it uses some properties of linear programming.

The weight space decomposition approach uses a normalized weight space W^0 and decomposes it into subsets $W^0(y)$ for all $y \in Y_E$. Each subset $W^0(y)$ corresponds to the weight set where *y* is the point corresponding to the optimal solution of the $MOIP(\lambda)$.

$$W^0 = \left\{ \lambda \in \mathbb{R}^p_>, \sum_{i=1}^{p} \lambda_i = 1 \right\} \text{ and } W^0(y) = \left\{ \lambda \in W^0 : \lambda y \leq \lambda y' : y' \in Y_E \right\}$$

where $y \in Y_E$.

Benson and Sun (2000) showed that $W^0 = \bigcup_{y \in Y_E} W^0(y)$.

Let us define *ConvY* as the convex hull of $Y = \left\{ y \in \mathbb{Z}^p : y = Cx, x \in X \right\}$. Przybylski, Gandibleux and Ehrgott (2007) developed the following properties for a *MOIP* with *p* objectives. Note that the dimension of W^0 is *p*-1.

Proposition 5.1. (Przybylski, Gandibleux and Ehrgott, 2007). $W^0(y)$ is a convex polytope.

Proposition 5.2. (Przybylski, Gandibleux and Ehrgott, 2007). The nondominated point y is an extreme nondominated point of *ConvY* if and only if $W^0(y)$ has dimension p-1.

Definition 5.1. (Przybylski, Gandibleux and Ehrgott, 2007). Two extreme nondominated points y^1 and y^2 are adjacent if and only if $W^0(y^1) \cap W^0(y^2)$ is a polytope of dimension p-2.

The weight space decomposition cannot be used during the search of Y_E, since Y_E is not completely known. Let us define Y_E' as the set of known solutions at some iteration of the algorithm. Przybylski, Gandibleux and Ehrgott (2007) proposed to decompose the weight space properly by using Y_E' as follows:

$$W_p^0(y) = \{\lambda \in W^0 : \lambda y \leq \lambda y' : y' \in Y_E'\} \text{ where } y \in Y_E'.$$

The *PGE* algorithm keeps the information on $W_p^0(y)$ for all $y \in Y_E'$. For each solution $y \in Y_E'$, *PGE* searches for new solutions at the boundaries of $W_p^0(y)$ and updates the proper decomposition information as new solutions are added to Y_E'. For example, if p=3 , $y^1, y^2 \in Y_E'$ and they are adjacent, then $W_p^0(y^1) \cap W_p^0(y^2)$ is a line segment. *PGE* searches for new solutions on this line segment by solving biobjective problems. Hence *PGE* utilizes *CAN* for this purpose. Originally, *CAN* works in the two dimensional objective space. However *PGE* uses *CAN* in the two dimensional weight space of a *MOIP* with three objectives. Similarly, multiobjective integer problems with four or more objectives are also recursively reduced to biobjective problems in their weight spaces and solved using *CAN*.

5.4 An Exact Algorithm

In this section, we develop an exact algorithm to find all extreme supported nondominated points of a multiobjective integer problem with three or more objectives. We first provide the additional definitions and notation. We then

introduce a set of dummy points and their effects in the weight and objective spaces. Finally, we introduce the algorithm.

5.4.1 Additional Definitions

We first present the definitions of *valid inequality*, *face* and *facet* from Nemhauser and Wolsey (1998). Using these, we define a *nondominated face* and a *nondominated facet*.

Definition 5.2. (Nemhauser and Wolsey, 1988). The inequality $\lambda y \le \lambda_0$ (or (λ, λ_0)) is called a valid inequality for *ConvY* if it is satisfied by all points in *ConvY*.

Definition 5.3. (Nemhauser and Wolsey, 1988). If (λ, λ_0) is a valid inequality for *ConvY*, $F = \{y \in ConvY : \lambda y = \lambda_0\}$, F is called a face of *ConvY*.

Definition 5.4. (Nemhauser and Wolsey, 1988). A face F of *ConvY* is a facet of *ConvY* if $dim(F) = dim(ConvY) - 1$.

Definition 5.5. A nondominated face is a face of *ConvY* with $\lambda \in \mathbb{R}^p_>$.

Definition 5.6. A nondominated facet is a facet of *ConvY* with $\lambda \in \mathbb{R}^p_>$.

Let us define q-dimensional nondominated faces of *ConvY* as $SF(q)$. Then the nondominated frontier *NDF* is defined as:

$$NDF = \bigcup_{q=0}^{p-1} SF(q).$$

Using only the nondominated facets may not be enough to define the nondominated frontier, i.e., there may be cases where $NDF \neq SF(p-1)$. For example, the nondominated frontier of the linear relaxation of the example problem is:

$$NDF = \{y : y = \alpha y^1 + (1-\alpha) y^2, 0 \le \alpha \le 1\} \cup \left\{y : y = \alpha^1 y^1 + \alpha^2 y^3 + \alpha^3 y^4, \alpha^i \ge 0, \sum_{i=1}^{3} \alpha^i = 1\right\}$$

In this problem, *NDF* consists of a facet and a *p*-2 dimensional face. In Figure 5.2, the nondominated frontier (5.2a) and the weight space decomposition (5.2b) of the example problem is given.

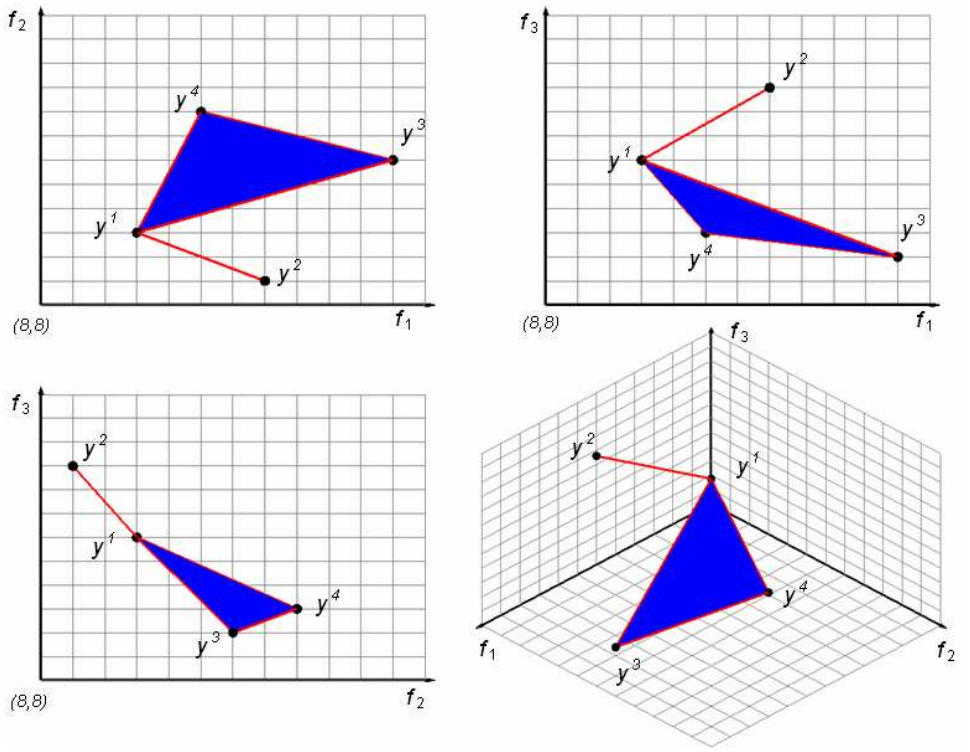
Let us define dummy points in the objective space as follows:

$$m^q = M \times e_q \text{ for } q = 1,..., p,$$

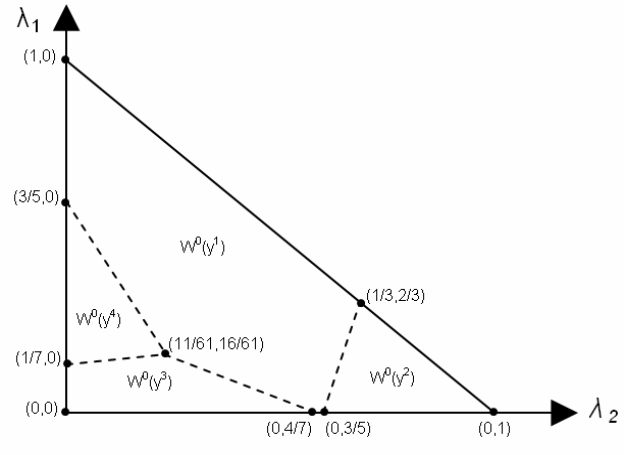
where $e_q = (0,...,0,1,0,...,0)$ is the q^{th} unit vector and *M* is a large number. These points have *p*-1 components equal to zero, and q^{th} component equal to *M*. We will derive a lower bound for *M* in Section 5.4.2. These are infeasible dummy points and there exists no $y \in Y_E$ dominating these points since we assume that $f_q(x) > 0$ for $q = 1,..., p$. Let us define two sets:

$$Y_M = \bigcup_{q=1}^{p} m^q \text{ and } Y_{EM} = Y_E \cup Y_M .$$

Introducing the dummy points has important effects on the weight and objective spaces. We first mention the effects on the weight space and provide Theorems 5.1 and 5.3.



(a) The Nondominated Frontier



(b) Weight Space Decomposition

Figure 5.2 Properties of the example problem

The weight space decomposition in Figure 5.2b is done for $y \in Y_E$. We propose to consider the dummy points in the weight space decomposition, in addition to the points $y \in Y_E$. Let $Boundary\left(W^0\right) = \left\{\lambda \in W^0 : \prod_{q=1}^{p} \lambda_q = 0\right\}$, i.e., at least one λ_q is equal to zero. The effect of introducing dummy points is given in the following theorem.

Theorem 5.1. If $W^0 = \bigcup_{y \in Y_{EM}} W^0(y)$, then $W^0(y) \cap Boundary\left(W^0\right) = \varnothing$ for all $y \in Y_E$.

Proof: On the boundaries of W^0, at least one of the weights, say λ_q, is equal to 0. Since $f_q(x) > 0$ and $\lambda y > 0$ for all $y \in Y_E$ and $\lambda \in W^0$, we have $\lambda m^q = 0$. Corresponding boundary is in $W^0\left(m^q\right)$. □

We first mention the effect of the above theorem on the weight space on a biobjective problem and then generalize it. If there are two objectives, then W^0 and weight sets $W^0(y)$ are line segments. Only two points, say y^1 and y^2, have common boundaries with W^0 and these points are adjacent to only one other point. Every point $y \in Y_E$, except y^1 and y^2, are adjacent to exactly two other points in Y_E. These adjacent points determine the boundaries of the weight set decomposition. By introducing points m^1 and m^2, we crop some portions of $W^0\left(y^1\right)$ and $W^0\left(y^2\right)$ such that they do not have common boundaries with W^0. So every point in Y_E is adjacent to exactly two points in Y_{EM}. For a problem with *p* objectives, there may be a point in Y_E such that it is adjacent to only one other point in Y_E. However, the upper bound on the number of points adjacent to $y \in Y_E$ is $|Y_E| - 1$, not *p*.

Let us define the sets *A*(*y*) and *NA*(*y*) for $y \in Y_{EM}$ as:

$$A(y) = \left\{y' \in Y_{EM} : y \text{ and } y' \text{ are adjacent}\right\}.$$

$$NA(y) = \{y' \in Y_{EM} : y \text{ and } y' \text{ are not adjacent}\}.$$

By definition, $A(y) \cup NA(y) \cup \{y\} = Y_{EM}$ and $A(y) \cap NA(y) = \varnothing$ for any $y \in Y_{EM}$. In Theorem 5.3, we prove that every point $y \in Y_E$ is adjacent to at least p points in Y_{EM}.

Theorem 5.2. (Kalai, 1993). Every q-dimensional polytope has at least q+1 facets.

Theorem 5.3. Every point $y \in Y_E$ is adjacent to at least p points in Y_{EM}.

Proof: For a given $y \in Y_E$, $W^0(y)$ is a convex polytope with dimension p-1 (by Propositions 5.1 and 5.2). Moreover, $W^0(y) \bigcap W^0(y')$ is a polytope with dimension p-2 for all $y' \in A(y)$ (by Definition 5.1). In order to define a polytope of dimension p-1, at least p facets (faces with dimension p-2) are required (by Theorem 5.2). Hence y must be adjacent to at least p points. Every point $y \in Y_E$ is adjacent to at least p points in Y_{EM}. □

Note that, all points adjacent to $y \in Y_E$ are not necessarily in Y_E and some of them may be in Y_M.

We observe that, if $Y = \varnothing$ then $Y_E = \varnothing$ because $Y_E \subseteq Y$. It is obvious that, if $Y \neq \varnothing$ then $Y_E \neq \varnothing$.

Corollary 5.1. If $Y_E \neq \varnothing$ then every point $y \in Y_{EM}$ is adjacent to at least p points in Y_{EM}.

Proof: By Theorem 5.3, we know this corollary holds for every $y \in Y_E$. If $Y_E = \varnothing$, then every dummy point $y \in Y_M$ is adjacent to the remaining p-1 dummy points. If

$|Y_E| = 1$, then every dummy point is also adjacent to this point due to Theorem 5.3, i.e. this point must be adjacent to at least *p* points. Adding more points to Y_E cannot decrease $|A(y)|$ for any $y \in Y_{EM}$. Hence, given that $Y_E \neq \varnothing$, every point $y \in Y_{EM}$ is adjacent to at least *p* points in Y_{EM}. □

Introducing dummy points also affects the structure of the objective space. Let us define $ConvY_{EM}$ as the convex hull of Y_{EM}, and nondominated frontier of $ConvY_{EM}$ as NDF_{EM}. We first prove that the dummy points are extreme supported nondominated points. We next show that we can define NDF_{EM} only with the nondominated facets, $SF(p)$ of $ConvY_{EM}$.

Theorem 5.4. The dummy point $m^q \in Y_M$ is an extreme supported nondominated point $ConvY_{EM}$.

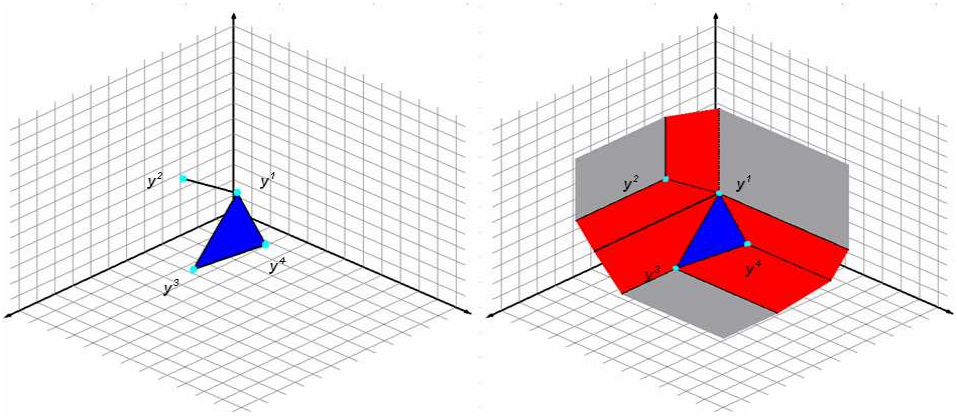
Proof: The dummy point m^q is nondominated since no $y \in Y_E$ can dominate m^q due to the assumption $f_q(x) > 0$ and no $m^k \in Y_M$, $k \neq q$ dominates m^q. This point is also an extreme point since its q^{th} objective function value *M* is the largest value in objective *q*. □

Theorem 5.5. $NDF_{EM} = SF(p)$.

Proof: Assume that there exists a face $F_q \in SF(q)$ and there exists no $F_p \in SF(p)$ such that $F_q \subseteq F_p$. This is possible if there exists a $y \in F_q$ such that $|A(y)| = q - 1$. Since $|A(y)| \geq p$ for $y \in Y_{EM}$ (by Theorem 5.3 and Corollary 5.3.1) it is not possible to have such a point $y \in Y_{EM}$. Hence nondominated frontier of $ConvY_{EM}$ can be defined by the set of nondominated facets. □

In Figure 5.3, we present the effect of dummy points in the objective space. *ConvY* (Figure 5.3a) is defined with a face and a facet. The face passes through points y^1

and y^2. The facet passes through the points y^1, y^3 and y^4. $ConvY_{EM}$ can be defined by using only facets. The blue facet is the same facet used in *ConvY*. The red facets are defined by two points in Y_E and one point in Y_{EM}. The grey facets are defined by one point in Y_E and two points in Y_{EM}. All 0-dimensional and 1-dimensional faces of *ConvY* are transformed to 2-dimensional facets by the use of the dummy points.



(a) *NDF* (b) NDF_{EM}

Figure 5.3 The effect of the dummy points in the objective space

5.4.2 The Algorithm

The main idea of the exact algorithm is very similar to that of *CAN*. We start with a set of initial points. At each iteration of the algorithm, we try to find new points in Y_E or identify new facets of $ConvY_{EM}$. The algorithm stops when no more points or facets can be identified. We call our exact algorithm as *ExA*.

The *CAN* algorithm uses pairs of extreme supported nondominated points and calculates the normal vector of the line passing through these points. Using two points is sufficient for the biobjective problems since two points can define a facet of the nondominated frontier. Similarly, we define a set $R = \{r^1, r^2, ..., r^p\}$ containing p

points where $r^q \in Y_E' \cup Y_M$ for $q = 1, \ldots, p$. We refer to these R sets as *stages*. *ExA* keeps track of three different lists of the stages. These lists are:

L : list of stages to be searched,

V : list of stages already searched (visited), and

F : list of facet defining stages.

Lists L, V and F are sets and their elements are stages, which are also sets with exactly p elements. F is a subset of V.

Steps of *ExA* is given in Figure 5.4. At the first step of the algorithm, we initialize the set Y_E' and lists F, L and V. Variable k corresponds to the cardinality of Y_E' and it is initially set to one. The important feature of this step is the initialization of L with the dummy points.

During the search of the algorithm, a stage R is selected from L and it is added to V at the second step. The normal of the hyperplane, λ, passing through the points of stage R, is calculated at the third step. If $\lambda \in \mathbb{R}^p_>$, then $MOIP(\lambda)$ is solved and optimal point r^* is obtained at Step 4.1. If $r^* \in R$ then we conclude that R is a facet defining stage and it is added to F at Step 4.2. If $r^* \notin R$ then, p new stages are generated by replacing r^* with each element of R. A newly generated stage is added to list L, if it is not already a member of lists L or V. Otherwise, it is discarded in order to prevent cycling between a subset of stages. If $r^* \notin Y_E'$, then it is a new point and is added to Y_E' as the k^{th} member. If a stage R has a negative component in its normal vector, then that stage is discarded at Step 5.

The stage R is removed from list L at Step 6, since a search is performed with this stage. At Step 7, *ExA* reports Y_E' and stops if there are no more stages to be visited, otherwise, the algorithm moves to Step 2 for a new search.

ExA	
Initialize	1. Set $Y_E' = \varnothing$, $k = 1$, $V = \varnothing, F = \varnothing, L = \left\{\left(m^1, m^2, ..., m^p\right)\right\}$.
Search	2. Select an element $R = \left\{r^1, r^2, ..., r^p\right\} \in L$ and set $V = V \cup \{R\}$. 3. Calculate λ such that $\lambda r^1 = \lambda r^2 = ... = \lambda r^p$. 4. If $\lambda \in \mathbb{R}^p_>$ 4.1. Solve problem $MOIP(\lambda)$ and let the optimal point be $r^* = \left(r_1^*, r_2^*, ..., r_p^*\right)$. 4.2. If $r^* \in R$ then set $F = F \cup \{R\}$. 4.3. If $r^* \notin R$ then 4.3.1. $L = L \cup \left\{\left\{r^1, ..., r^{p-1}, r^*\right\}, \left\{r^1, ..., r^*, r^p\right\}, ..., \left\{r^*, ..., r^{p-1}, r^p\right\}\right\}$ and $L = L - (L \cap V)$. 4.3.2. If $r^* \notin Y_E'$ then $y^k = r^*$, $Y_E' = Y_E' \cup \left\{y^k\right\}$ and *k=k+1.* 5. If $\lambda \notin \mathbb{R}^p_>$ then go to Step 6.
Control the loop	6. $L = L - \{R\}$. 7. If $L = \varnothing$ then report Y_E' and stop, otherwise go to Step 2.

Figure 5.4 Steps of *ExA*

In the following theorems, we prove that *ExA* finds all extreme supported nondominated points in finite iterations and does not find any other points.

Theorem 5.6. *ExA* ends in a finite number of iterations.

Proof: Whenever a stage R is selected, it is recorded to V in Step 2. If a generated stage (at Step 4.3.1) is already in V, then it is immediately removed from L. Hence a stage can be visited at most once during the algorithm. Number of stages visited is $|V|$ and it is bounded by:

$$|V| \leq \binom{|Y_E \cup Y_M|}{p} = \frac{|Y_E \cup Y_M|!}{p!(|Y_E \cup Y_M| - p)!}.$$

Hence *ExA* ends in a finite number of iteration. □

Note that although there is a finite number of iterations, $|Y_E|$ may be exponential in the problem size and *ExA* may require an exponential number of iterations.

Theorem 5.7. $r^* \in Y_E$ **.**

Proof: r^* is obtained by $MOIP(\lambda)$ using $\lambda \in \mathbb{R}^p_>$. □

Corollary 5.2. $Y'_E \subseteq Y_E$.

Proof: Follows directly by Theorem 5.7. □

Let $C_R = Conv\{r^1, r^2, ..., r^p, r^*\}$ denote the convex hull defined by $r^1, r^2, ..., r^p$ and r^* where $R = \{r^1, r^2, \ldots, r^p\}$ and r^* solves $MOIP(\lambda)$ for $\lambda \in \mathbb{R}^p_>$ such that $\lambda r^1 = \lambda r^2 = ... = \lambda r^p$.

Lemma 5.1. There exists no extreme supported nondominated point in the interior of C_R, $\text{int}\, C_R$. (In other words, there is no r' such that $r' \in Y_{EM} \cap \text{int}\, C_R$)

Proof: Assume that $r' \in \operatorname{int} C_R$. Then by Minkowski's Theorem

$$r' = \alpha^* r^* + \sum_{k=1}^{p} \alpha^k r^k \text{ for some } \alpha = \left(a^1, ..., a^p, a^*\right) \in \mathbb{R}_{\geq}^{p+1} \text{ such that}$$

$$\sum_{k=1}^{p} \alpha^k = 1 - \alpha^*.$$

Since r' can be expressed as a convex combination of other points in Y_{EM}

$$r' \notin Y_{EM} \text{ and } Y_{EM} \cap \operatorname{int} C_R = \varnothing \qquad \square$$

Theorem 5.8. Each $R \in F$ defines a nondominated facet of *ConvY*$_{EM}$.

Proof: A stage *R* is added to *F* if $\lambda \in \mathbb{R}_{>}^{p}$ and $r^* \in R$. There are *p* extreme points in *R* and they are affinely independent since $\lambda \in \mathbb{R}_{>}^{p}$. Hence the polyhedron defined by *R* has dimension *p*-1 and is a nondominated facet of *ConvY*$_{EM}$. $\square$

In the next theorem, we prove that *ExA* finds all extreme supported nondominated points. We need the following two definitions in the proof of the theorem.

$$B^{\geq}\left(\left\{r^1, r^2, ..., r^p\right\}\right) = B^{\geq}(R) = \left\{ y = \left(y_1, y_2, ..., y_p\right) \middle| \sum_{q=1}^{p} \lambda_q^R y_q \geq \lambda_0^R \right.$$
$$\left. \text{where } \lambda_0^R = \sum_{q=1}^{p} \lambda_q^R r_q^1 = \sum_{q=1}^{p} \lambda_q^R r_q^2 = ... = \sum_{q=1}^{p} \lambda_q^R r_q^p \right\}$$

We define $B^{\geq}(R)$ for $R \in F$. The hypervolume defined by $B^{\geq}(R)$ is the set of points dominated by any linear combination of the points in *R*.

$$B^{\leq}\left(\left\{r^1, r^2, ..., r^p\right\}\right) = B^{\leq}(R) = \left\{ y = \left(y_1, y_2, ..., y_p\right) \middle| y_q \leq \sum_{k=1}^{p} \alpha^k r_q^k \ \forall q \right.$$
$$\left. \text{where } \sum_{k=1}^{p} \alpha^k = 1, \alpha^k \geq 0 \ \forall k \right\}$$

We define $B^{\leq}(R)$ for $R \in F \cup L$. The hypervolume defined by $B^{\leq}(R)$ is the set of points dominating all convex combinations of the points in *R*.

Theorem 5.9. At the termination of *ExA*, $Y_E' = Y_E$.

Proof: By induction we will prove that if $y^k \in Y_E$ then the following term holds at each iteration of the algorithm:

$$y^k \in \bigcup_{R \in F \cup L} B^{\leq}(R).$$

At the first iteration $F \cup L = \left\{\left(m^1, m^2, ..., m^p\right)\right\}$, and the term holds since all feasible points lie in the defined hypervolume.

Assume that at some iteration of the algorithm the term holds. Then at that iteration, an element $R \in L$ is selected in Step 2 and the λ vector corresponding to stage *R* is calculated in Step 3.

If $\lambda \in \mathbb{R}^p_>$, then $MOIP(\lambda)$ is solved and the optimal point r^* is obtained in Step 4.1. If $r^* \in R$ then *R* is added to *F* (in Step 4.2) and removed from *L* (in Step 6) hence $F \cup L$ does not change. If $r^* \notin R$ then *R* is removed from *L* (in Step 6). In Step 4.3.1, *p* new stages are generated and the unvisited ones are added to *L*. By removing *R* and adding *p* new stages to *L*, some hypervolume defined by the term is removed, however, by Lemma 5.1, we know that there is no $y^k \in Y_E$ in the removed hypervolume.

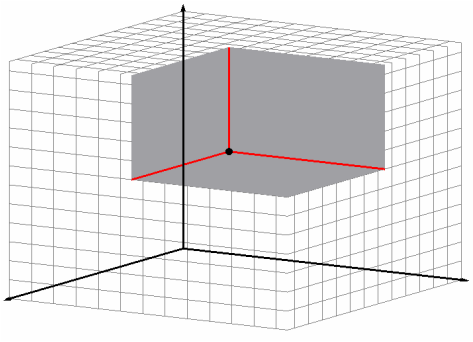
If $\lambda \notin \mathbb{R}^p_>$ then the hyperplane defined by *R* cannot define a nondominated facet. Hence *R* is removed from *L* (in Step 6).

Therefore after one iteration $y^k \in \bigcup_{R \in F \cup L} B^{\leq}(R)$ still holds.

The algorithm stops when $L = \varnothing$. At the end of the algorithm $F \cup L = F$. $y^k \in \bigcup_{R \in F} B^{\leq}(R)$ holds for all $y^k \in Y_E$. Also all feasible solutions are in $B^{\geq}(R)$ for all $R \in F$. So for all extreme supported nondominated points, $y^k \in B^{\geq}(R)$ for all $R \in F$. Moreover, $y^k \in \bigcap_{R \in F} B^{\geq}(R)$. Hence for every $y^k \in Y_E$, there exists a $R^k \in F$ such that $y^k \in B^{\geq}(R^k) \cap B^{\leq}(R^k)$. This completes the proof. □

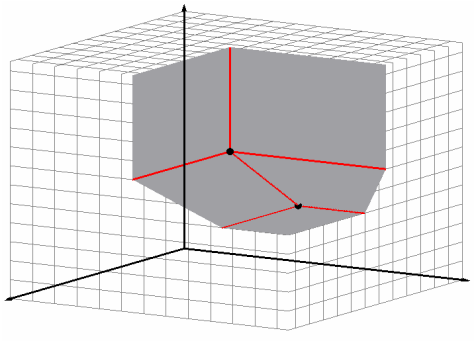
In Figure 5.5, we show the progress of *ExA* graphically on the example problem of Tenfelde-Podehl (2003). We only show a subset of the iterations of the algorithm. In the first iteration, we have $L = \{(m^1, m^2, m^3)\}$. In order to display enough detail, we use a range of [0,25] for the axes. Due to this reason, we cannot show the dummy points. Each plane in Figure 5.5a, passes through two dummy points and y^1. Grey planes represent stages in *L*. Blue planes are facets of $ConvY_{EM}$. In this example, the number of iterations and the number of visited stages |*V*| happened to be equal. This is because the stages generated in Step 4.3.1 are not listed in *L* or *V* and are added to *L*. However, this may not always happen and there will be instances where |*V*| is less than the number of iterations.

ExA works using the dummy points. We provide a lower bound for the *M* value used by the dummy points. We should mention that this lower bound is valid when all objective function values are strictly positive and integer valued. It can be adapted for the general case with some modifications, as discussed in Section 2.3. We first consider the biobjective case to explain the main idea, and then provide a generalization for the multiple objectives case.



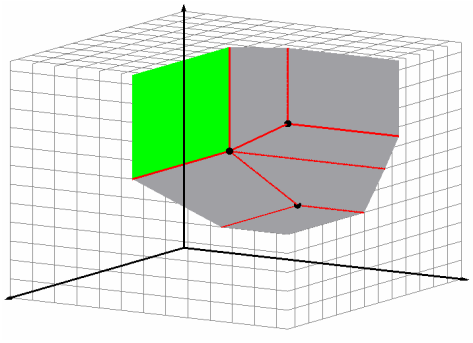
a) *Iteration*=1, |*L*|=3, |*V*|=1, |*F*|=0,

$Y'_E = \{y^1\}, |Y'_E| = 1$



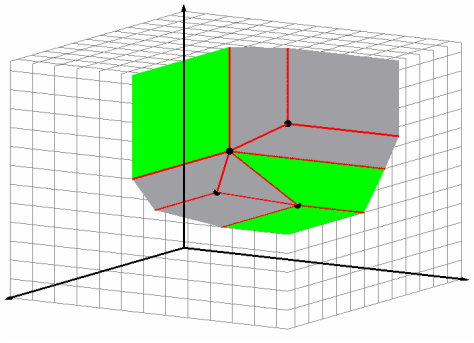
b) *Iteration*=2, |*L*|=5, |*V*|=2, |*F*|=0,

$Y'_E = \{y^1, y^3\}, |Y'_E| = 2$



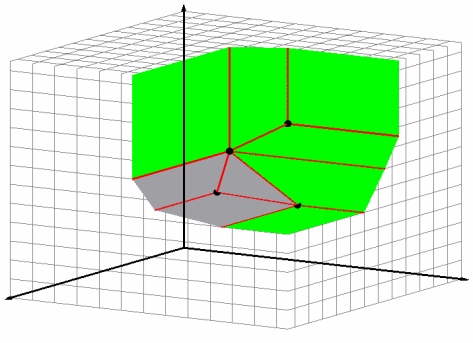
c) *Iteration*=4, |*L*|=6, |*V*|=4, |*F*|=1,

$Y'_E = \{y^1, y^3, y^2\}, |Y'_E| = 3$



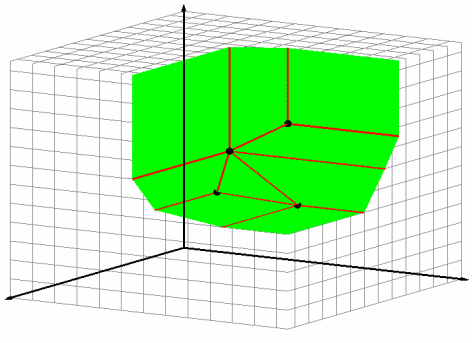
d) *Iteration*=7, |*L*|=6, |*V*|=7, |*F*|=3,

$Y'_E = \{y^1, y^3, y^2, y^4\}, |Y'_E| = 4$



e) *Iteration*=10, |*L*|=3, |*V*|=10, |*F*|=6,

$Y'_E = \{y^1, y^3, y^2, y^4\}, |Y'_E| = 4$



f) *Iteration*=13, |*L*|=0, |*V*|=13, |*F*|=9,

$Y'_E = Y_E = \{y^1, y^3, y^2, y^4\}, |Y_E| = 4$

Figure 5.5 Progress of *ExA*

Assume that LB_q and UB_q correspond to the lower and upper bounds of the q^{th} objective function, q=1,2. If UB_q is found among all efficient solutions, then it will lead to a better (smaller) M value. Since, this is not an easy problem, and since any UB_q value is sufficient for our purposes, we can find UB_q among all feasible solutions. The M value must be large enough so that no point $y \in Y_E$ should be convex dominated by stage $R = \left(m^q, y'\right)$ for any $y' \in Y_E$ and q=1,2. Consider the slope of the line passing through points $y^1, y^2 \in Y_E$ in Figure 5.6. The slope is the steepest when $y^1 = \left(LB_1 + 1, LB_1\right)$ and $y^2 = \left(LB_1, UB_2\right)$. In this case, the line and the f_2 axis intersect at the point $\left(0, \left[UB_2\left(LB_1 + 1\right) - \left(LB_1 LB_2\right)\right]\right)$. By a similar analysis on the f_1 axis, we conclude that M must be greater than $max\left\{\left[UB_2\left(LB_1 + 1\right) - \left(LB_1 LB_2\right)\right], \left[UB_1\left(LB_2 + 1\right) - \left(LB_1 LB_2\right)\right]\right\}$. Eliminating the terms with negative coefficients and replacing the lower bounds LB_1, LB_2 with the upper bounds UB_1, and UB_2 we can define amore conservative lower bound and define M as:

$$M \geq \left(UB_1 + 1\right)\left(UB_2 + 1\right)$$

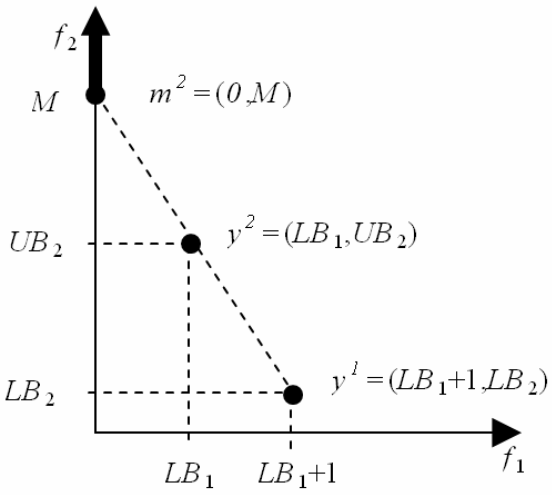


Figure 5.6 Calculating a lower bound for M

In the following theorem, we give a general lower bound to the *M* value.

Theorem 5.10: Given $y_q > 0$ and $y_q \in \mathbb{Z}$ for $q = 1,..., p$, for all $y \in Y$, and $LB_q \le y_q \le UB_q$ for all $y \in Y_E$. A lower bound for *M* is

$$M \ge \left(\sum_{q=1}^{p} UB_q \right) \left(\max_{q=1,\ldots,p} \{UB_q\} \right).$$

Proof: In order to obtain steepest edges, let us define the following set of nondominated points:

$$z^q = \left(LB_1, LB_2, \ldots, LB_{q-1}, LB_q + 1, LB_{q+1}, \ldots, LB_p \right) \text{ for } q=1,\ldots,p,$$

$$r^q = \left(UB_1, UB_2, \ldots, UB_{q-1}, LB_q, UB_{q+1}, \ldots, UB_p \right) \text{ for } q=1,\ldots,p, \text{ and}$$

dummy points, $m^q = Me_q$ for $q = 1,..., p$, where $e_q = (0,...,0,1,0,...,0)$ is the q^{th} unit vector.

The value of *M* should be large enough so that r^q should dominate any convex combination of the points $m^1, \ldots, m^{q-1}, z^q, m^{q+1}, \ldots, m^p$.

$$r^q \le \sum_{i=1, i \ne q}^{p} \alpha_i m^i + \alpha_q z^q \text{ where } \sum_{i=1}^{p} \alpha_i = 1,\ \alpha_i \ge 0\ \forall i.$$

This corresponds to the following set of inequalities.

$$UB_i \le \alpha_i M + \alpha_q LB_i \quad i = 1,..., p, i \ne q$$

$$LB_q \le \alpha_q \left(LB_q + 1 \right)$$

$$\sum_{i=1}^{p} \alpha_i = 1$$

For a given *q*, $\alpha_q \ge \dfrac{LB_q}{LB_q + 1}$. Let $\alpha_0 = \sum_{i \ne q} \alpha_i$. Then $\alpha_0 \le \dfrac{1}{LB_q + 1}$.

Summation of the *p*-1 constraints corresponding to $i \ne q$.

$$\sum_{i \ne q} UB_i \le M \sum_{i \ne q} \alpha_i + \alpha_q \sum_{i \ne q} LB_i$$

$$M \geq \frac{\sum_{i \neq q} UB_i - \alpha_q \sum_{i \neq q} LB_i}{\sum_{i \neq q} \alpha_i} = \frac{\sum_{i \neq q} UB_i - (1-\alpha_0) \sum_{i \neq q} LB_i}{\alpha_0} = \frac{\sum_{i \neq q} UB_i - \sum_{i \neq q} LB_i}{\alpha_0} + \sum_{i \neq q} LB_i$$

In order to set *M* as small as possible, use the maximum possible value of α_0. Then

$$M \geq (LB_q + 1)\left(\sum_{i \neq q} UB_i - \sum_{i \neq q} LB_i \right) + \sum_{i \neq q} LB_i$$

$$M \geq (LB_q + 1) \sum_{i \neq q} UB_i .$$

This term is valid for each *q*. In order to get a general lower bound, we need all upper and lower bound values. This term has to be calculated for each *q*, and the largest value should be selected.

In order to obtain an easy conservative value for *M*, we eliminate the terms with negative coefficients and replace $\sum_{i \neq q} UB_i$ with $\sum_{i=1}^{p} UB_i$. We also replace $LB_q + 1$ with $\max_{q=1,\ldots,p} \{UB_q\}$ since $LB_q + 1 \leq UB_q$ must hold for every *q*. Hence a conservative value for *M* can be constructed as follows:

$$M \geq \left(\sum_{q=1}^{p} UB_q \right) \left(\max_{q=1,\ldots,p} \{UB_q\} \right).$$ □

5.5 Improvements on the Exact Algorithm

In this section, we discuss the opportunities to improve *ExA*. During its search, *ExA* faces different situations. There are several types of information embedded in these situations, and we can use them to improve the algorithm. We first discuss the information embedded in nondominated facet defining stages. We next discuss a property which may decrease the number of $MOIP(\lambda)$s solved. We finally discuss different queue disciplines for selecting *R* from *L*.

5.5.1 Nondominated Facets

Let us define the following two sets for each $y \in Y_E$:

CF(*y*): The set of points $y' \in Y_E$ such that there exists a stage $R \in F$ (i.e. common facet) and $y, y' \in R$.

NF(*y*): The set of points $y' \in Y_E$ such that there exists no stage $R \in F$ (no common facet) and $y, y' \in R$.

We consider $y' \in CF(y)$ if there is a nondominated facet defining stage *R* and both y and y' are elements of *R*. The second set, *NF*(*y*) is the complement of *CF*(*y*). If there exists no nondominated facet defining stage *R* such that two points y and y' are elements of *R*, then $y' \in NF(y)$. By definition, $CF(y) \cap NF(y) = \varnothing$ and $CF(y) \cup NF(y) \cup \{y\} = Y_E$. Also, these sets are symmetric, i.e., if $y' \in CF(y)$ then $y \in CF(y')$ and if $y' \in NF(y)$ then $y \in NF(y')$.

We can obtain these two sets when we know all $y \in Y_E$, i.e. when we have complete information about the nondominated points and all nondominated facets. However, during the search, we only know a subset of Y_E and we must use the partial information gathered. Due to this reason, we define $CF'(y)$ and $NF'(y)$ for $y \in Y_E'$. If both points y and y' are elements of a facet defining stage then we update both sets; $CF'(y) = CF'(y) \cup \{y'\}$ and $CF'(y') = CF'(y') \cup \{y\}$. The sets $CF'(y)$ and $NF'(y)$ are subsets of the original sets, $CF'(y) \subseteq CF(y)$ and $NF'(y) \subseteq NF(y)$. The property $CF'(y) \cap NF'(y) = \varnothing$ still holds. However, the other property, $CF'(y) \cup NF'(y) \cup \{y\} = Y_E'$ may not hold since we add point y to the set $NF'(y')$ only if we are sure that these two points cannot be the members of a nondominated facet defining stage at the same time.

Theorem 5.11. If $\lambda \notin \mathbb{R}^p_>$ for some *R*, then there exists at least a pair of points $r^t, r^s \in R$, such that $r^s \in NF'\left(r^t\right)$.

Proof: If $\lambda \notin \mathbb{R}^p_>$, the elements of *R* cannot define a nondominated facet since by definition, the condition $\lambda \in \mathbb{R}^p_>$ must hold for nondominated facets. However, if $r^s \in CF'\left(r^t\right)$ holds for all pairs of elements of *R*, then *R* must define a nondominated facet. So we conclude that there must be at least a pair of points in *R* which cannot be on the same nondominated facet. □

Corollary 5.3. If $R \in F$ then $r^s \in CF'\left(r^t\right)$ for all pairs $r^t, r^s \in R$.

Proof: All points in *R* are elements of a nondominated facet. □

If $\lambda \in \mathbb{R}^p_>$, then in Step 4.1 of *ExA*, there are five possible cases considering r^*, λr^* and λ_0 where $\lambda_0 = \lambda r^k$ for each $r^k \in R$. These cases are:

Case 1: $\lambda r^* < \lambda_0$ and $r^* \notin Y'_E$

Case 2: $\lambda r^* < \lambda_0$ and $r^* \in Y'_E$

Case 3: $\lambda r^* = \lambda_0$ and $r^* \in R$

Case 4: $\lambda r^* = \lambda_0$ and $r^* \in Y'_E \setminus R$

Case 5: $\lambda r^* = \lambda_0$ and $r^* \notin Y'_E$

In the following theorems, we show the implications of these cases.

Theorem 5.12. If $\lambda \in \mathbb{R}^p_>$ and $\lambda r^* < \lambda_0$ then the elements of *R* do not define a facet.

Proof: A facet is a *p*-1 dimensional face, and a face is a valid inequality. However, condition $\lambda r^* < \lambda r^k$ is contradicting with the definition of a valid inequality. □

Corollary 5.4. If $\lambda \in \mathbb{R}^p_>$ and $\lambda r^* < \lambda_0$ then the elements of *R* do not define a nondominated facet.

Proof: Follows directly from Theorem 5.12. □

Theorem 5.13. If $\lambda \in \mathbb{R}^p_>$ and $\lambda r^* = \lambda_0$, then r^* and all points of *R* are on the same nondominated facet.

Proof: If the points in *R* did not define a nondominated facet, then $\lambda r^* < \lambda_0$ should hold (as proved in Theorem 5.12). There are *p* for cases 1, 2 and 3 and *p*+1 for cases 4 and 5 affinely independent points having the same objective function value for the given $\lambda \in \mathbb{R}^p_>$. Hence these points define a nondominated facet and for all pairs $r^t, r^s \in R \cup \{r^*\}$, $r^s \in CF'(r^t)$ holds. □

We can use the information gathered during the search of *ExA* by developing rules based on Theorems 5.11, 5.12, 5.13 and Corollary 5.4. We define the following rules:

Rule 1: After selecting stage $R = \{r^1, r^2, ..., r^p\} \in L$ in Step 2, check NF' information for all pairs of points in *R*. If there exists a pair $r^t, r^s \in R$ such that $r^s \in NF'(r^t)$ then *R* cannot be a nondominated facet defining stage. Add *R* to *V*, skip Steps 3, 4 and 5 and proceed to Step 6.

Rule 2: If $\lambda \notin \mathbb{R}^p_>$ for some *R*, and there exist points $r^u, r^v \in R$ such that $r^s \in CF'(r^t)$ for all (r^t, r^s) pairs for $r^t, r^s \in R$ except (r^u, r^v) pair, then set $r^u \in NF'(r^v)$ and $r^v \in NF'(r^u)$.

Rule 3: If Case 2 is observed and there exist points $r^u, r^v \in R$ such that $r^s \in CF'(r^t)$ for all (r^t, r^s) pairs for $r^t, r^s \in R$ except (r^u, r^v) pair, then set $r^u \in NF'(r^v)$ and $r^v \in NF'(r^u)$.

Rule 4: If Cases 3, 4 or 5 is observed then set $r^s \in CF'(r^t)$ for all (r^t, r^s) pairs such that $r^t, r^s \in R \cup \{r^*\}$.

5.5.2 Pre-Calculation

In this section, we define a property that may decrease the number of $MOIP(\lambda)$s solved during the search *ExA*. We refer to this property as *pre-calculation*.

Let us define $ConvY'_{EM}$ as the convex hull of $Y'_E \cup Y_M$. The aim of *ExA* is to close the gap between $ConvY'_{EM}$ and $ConvY_{EM}$. At the end of the algorithm, $Y'_E = Y_E$, and also $ConvY'_{EM}$ is equal to $ConvY_{EM}$. Consequently, in Step 4.1 of *ExA*, we solve $MOIP(\lambda)$ and obtain r^*. Using r^*, we define new stages and add them to *L*. However, in some cases (Cases 2, 3 and 4 in Section 5.1), r^* may be a known point, i.e., $r^* \in Y'_E$. In these cases, we would obtain r^* by simply searching the best point $y \in Y'_E$ for the given λ instead of solving $MOIP(\lambda)$. Although it depends on the size of the single objective problem, the performance of $MOIP(\lambda)$ solver used, and the cardinality of Y'_E, searching for the best point for the given weights could be much easier than solving a single objective integer program. We will discuss the trade-offs of pre-calculation in the computational results section.

Based on this observation, we modify Step 4.1 of *ExA* as follows:

4.1 *Pre-calculation,* proceed to Step 4.1.1

4.1.1 Let r^*_{pc} be such that $\lambda r^*_{pc} = \min_{y \in Y'_E} \{\lambda y\}$.

4.1.2 If $r^*_{pc} \notin R$ then $r^* = r^*_{pc}$, go to Step 4.3, otherwise go to Step 4.1.3

4.1.3 Solve $MOIP(\lambda)$ and let the optimal point be $r^* = (r^*_1, r^*_2, ..., r^*_p)$, go to Step 4.2.

In Step 4.1.1, we determine the best available point r^*_{pc} for the given λ. In Step 4.1.2, we check if $r^*_{pc} \in R$. If $r^*_{pc} \in R$ then either R is a nondominated facet defining stage, or the optimal point of this stage is a hitherto unknown point, i.e., $r^* \notin Y'_E$. Hence for this case, we proceed to Step 4.1.3 and solve $MOIP(\lambda)$. However, if $r^*_{pc} \notin R$, then we set $r^* = r^*_{pc}$ and proceed to Step 4.3 (we skip Step 4.2 since we already know that $r^* \notin R$).

When we apply the Pre-calculation property, point r^*_{pc} found in Step 4.1.1 and the optimal solution of $MOIP(\lambda)$, r^* may not be equal. This corresponds to Cases 1 and 5. This may result with the addition of some stages to *L*, which would have never been added to *L* in the original *ExA*. On the other hand, some stages that would have been visited by the original algorithm, may not be visited when using pre-calculation.

We still know that at the end of the algorithm, we will have $Y'_E = Y_E$ because, the hypervolume removed from the $\bigcup_{R \in F \cup L} B^{\leq}(R)$ term is equal to the interior of the convex hull defined by $\{r^1, r^2, \ldots, r^p, r^*_{pc}\}$. By Lemma 5.1, it is not possible to have an extreme supported nondominated point in this hypervolume.

5.5.3 Queuing Disciplines

In each iteration, *ExA* selects a stage *R* from list *L* in Step 2. In this section, we consider *L* as a queue of stages and discuss three queue disciplines for selecting *R*.

Before discussing how to select a stage *R* from *L*, let us mention how we add new stages to *L*. In Step 4.3.1, we add new stages to the end of *L* as follows:

$$L = L \cup \left\{ \{r^1, \ldots, r^{p-1}, r^*\}, \{r^1, \ldots, r^*, r^p\}, \ldots, \{r^*, \ldots, r^{p-1}, r^p\} \right\}$$

and we remove from L the stages that are already visited. Thus L is updated as $L = L - (L \cap V)$. This removal prevents the cycling of the algorithm.

The first queue discipline we use is *first in first out*. In this discipline, we select the first element of L in Step 2 as the new R stage and remove it from L in Step 6. The second queue discipline we use is *last in first out*. In this discipline, we select the last element of L in Step 2 as the new R and remove it from L in Step 6.

In Figure 5.7, we build a tree to represent the search of *ExA*. It is the tree of the example problem of Tenfelde-Podehl (2003) with three objectives. Each node of the tree corresponds to a stage R. The number in the first line is a unique number to refer to a stage easily. The second line corresponds to the set of points in R. The third line contains the point corresponding to the optimal solution of $MOIP(\lambda)$ for that stage.

At the beginning of the algorithm, only stage $\{m^1, m^2, m^3\}$ is in L and the optimal solution of $MOIP(\lambda)$ corresponds to y^1. We then remove the first stage from L and add three new stages: $\{y^1, m^2, m^3\}$, $\{m^1, y^1, m^3\}$ and $\{m^1, m^2, y^1\}$. Assume that we next select $R = \{y^1, m^2, m^3\}$. As seen from the figure, the optimal solution corresponds to y^1. We remove this stage from L and add it to F, since it is a nondominated facet defining stage. Note that, the leaf nodes (those without any offspring) are nondominated facet defining stages in this tree. However, in general, there are three different possibilities for being a leaf node. A stage is a leaf node if it

i) defines a nondominated facet stage,
ii) has negative elements in its normal vector, $\lambda \notin \mathbb{R}^p_>$, or
iii) has no unvisited offspring, i.e., there are nodes in the tree corresponding to its offspring.

Assume that, we next select $R = \{m^1, y^1, m^3\}$. We obtain y^2 and add three new stages to L: $\{y^2, y^1, m^3\}$, $\{m^1, y^2, m^3\}$ and $\{m^1, y^1, y^2\}$. Now there are four stages in L, 4th,

5^{th}, 6^{th} and 7^{th} stages. According to the first-in-first-out and the last-in-first-out disciplines, we should select 4^{th} and 7^{th} stages, respectively. As it can be seen, the first in first out and the last-in-first-out disciplines correspond to breadth first and depth first strategies of exploring the tree, respectively.

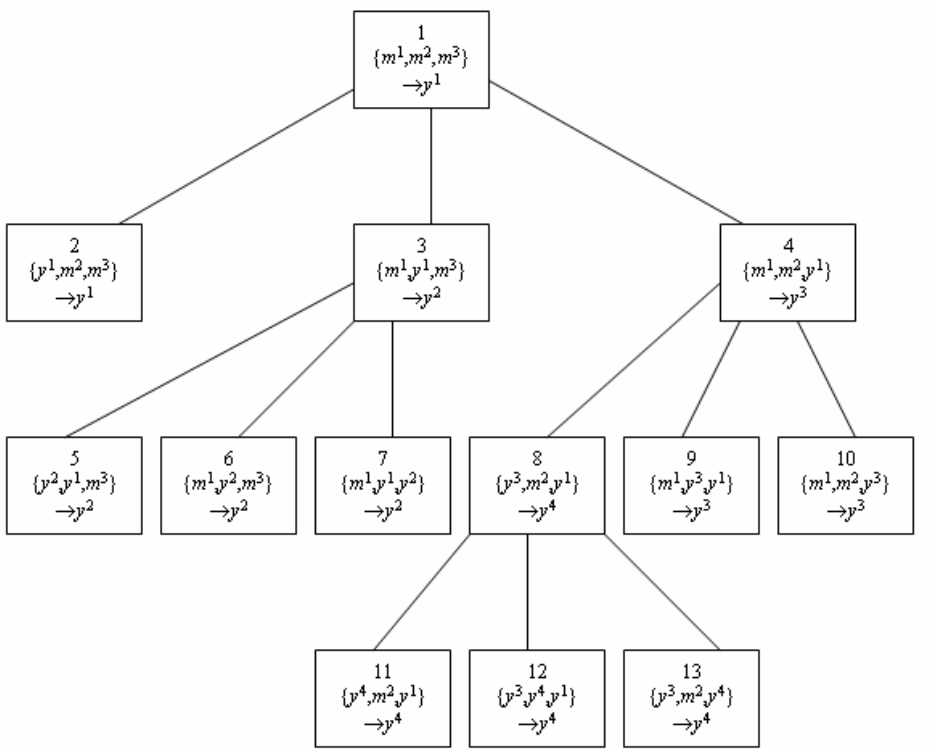


Figure 5.7 Tree structure of the example problem

Consider Rules 2 and 3 discussed in Section 5.5.1. These rules are applicable for a stage *R* if there exist points $r^u, r^v \in R$ such that $r^s \in CF'\left(r^t\right)$ for all $\left(r^t, r^s\right)$ pairs for all $r^s, r^t \in R$ except the $\left(r^u, r^v\right)$ pair. If all conditions are satisfied, we set $r^u \in NF'\left(r^v\right)$ and $r^v \in NF'\left(r^u\right)$. This information would be helpful in applying

Rule 1 at other stages. The earlier this type of stages are selected, the sooner information extracted can be used. Obtaining more information sooner will generally decrease the number of iterations required.

For a given stage, we can find the number of pairs $\left(r^t, r^s\right)$ satisfying $r^s \in CF'\left(r^t\right)$ as follows:

$$\frac{1}{2}\sum_{r^t, r^s \in R}\left|r^t \cap CF'\left(r^s\right)\right| .$$

We halve the summation since CF' sets are symmetric. There are $\binom{p}{2}$ pairs in any stage and Rules 2 and 3 require $\binom{p}{2}-1$ pairs in CF' sets. Hence, if we can select a stage having $\binom{p}{2}-1$ pairs in CF' sets, then we may have a chance to apply these rules. The third queue discipline searches for this type of stages. A stage $R \in L$ is a such candidate if

$$\frac{1}{2}\sum_{r^t, r^s \in R}\left|r^t \cap CF'\left(r^s\right)\right| = \binom{p}{2}-1 .$$

If there is no such candidate, this discipline utilizes first in first out discipline.

We solve a sample assignment problem with p=3 and n=20. We first consider *ExA* and do not gather any common facet information. In this case, the third queuing discipline is equivalent to the first one. Hence we have two different queuing disciplines. In both disciplines, *ExA* stops nearly after 14,800 iterations. In Figure 5.8, we plot the ratio $\frac{|Y_E'|}{|Y_E|}$ after every ten iterations for each discipline. Breadth first discipline finds all points before 1910th iteration. It tries to prove that $Y_E' = Y_E$ in the remaining iterations. At 1910th iteration, the depth first discipline found only 73% of the points. It finds all points before 14600th iteration.

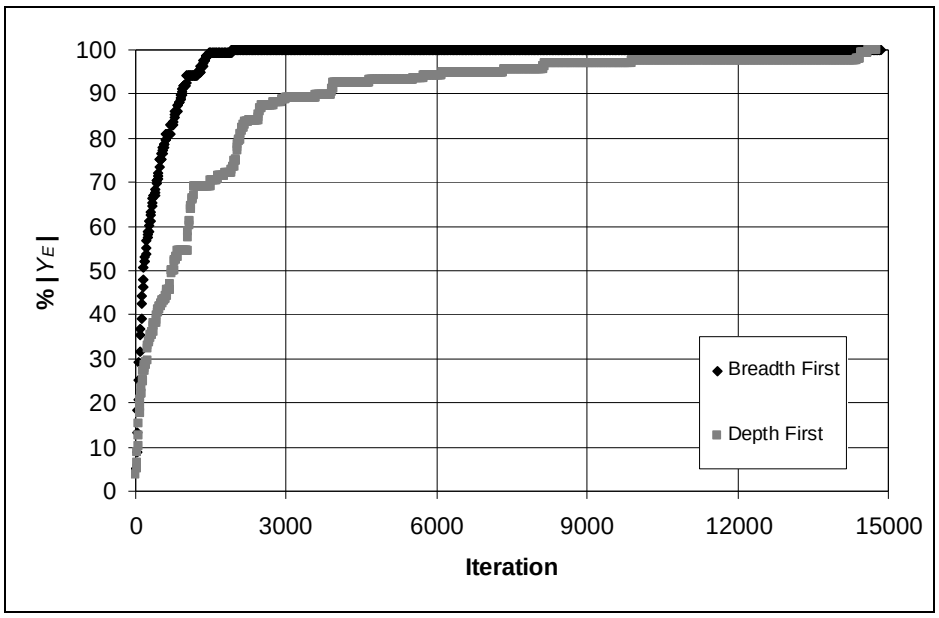


Figure 5.8 Change in the $\frac{|Y_E'|}{|Y_E|}$ ratio for different queuing disciplines on a sample problem.

We next solve the same sample problem with the nondominated facets property. In Figure 5.9, we show the effect of using the nondominated facet information by applying Rules 1-4 discussed in Section 5.5.1. We do not plot $\frac{|Y_E'|}{|Y_E|}$ for the third queuing discipline since it is very similar to the plot of breadth first discipline. Applying the rules does not change the pattern of the plots. However the number of iterations decreases substantially.

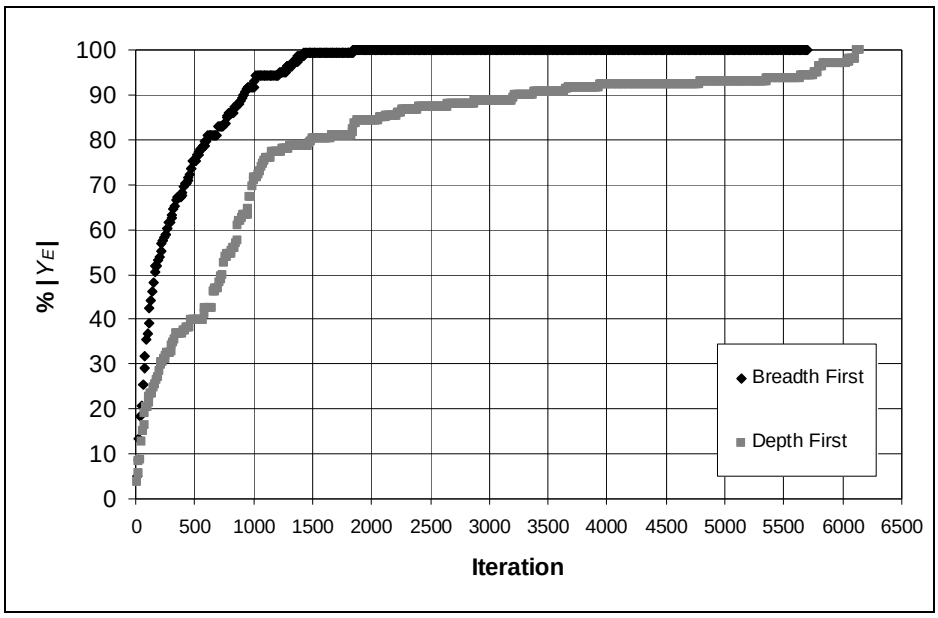


Figure 5.9 Change in the $\frac{|Y_E'|}{|Y_E|}$ ratio for different queuing disciplines on a sample problem where Rules 1-4 are used.

In Figure 5.10, we plot the first 100 points found by the breadth first and the depth first queuing disciplines. The blue points are found by breadth first discipline. They are spread over the nondominated frontier. The red points are found by depth first discipline. They are concentrated on a region of the nondominated frontier.

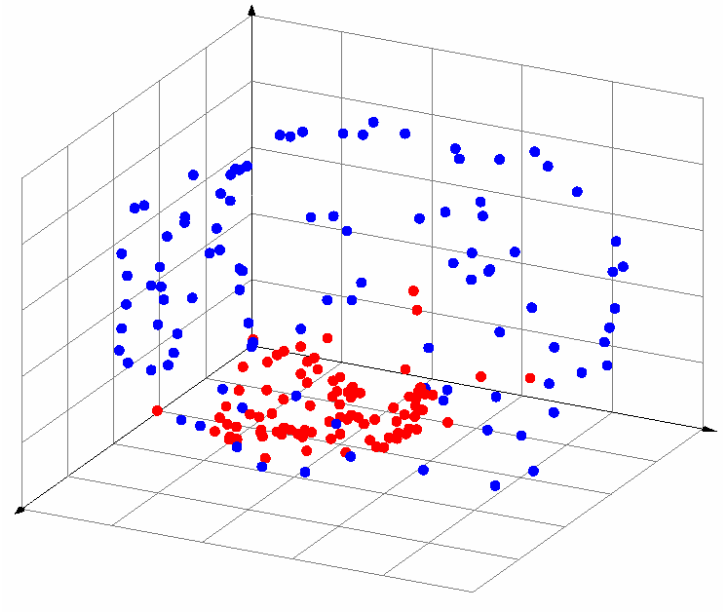


Figure 5.10 First 100 points found by BF and DF.

5.6 Computational Experiments

In order to test the performance of *ExA* and the proposed improvements to it, we solve three- and four-objective versions of three well-known combinatorial optimization problems: The Assignment Problem (AP), the Knapsack Problem (KP) and the Traveling Salesperson Problem (TSP).

In Section 5.6.1, we give the mathematical model of each test problem, random data generation schemes and implementation details. We report the results in Section 5.6.2.

5.6.1 Test Problems

The Assignment Problem (AP)

In AP, there are n jobs and n resources. The cost of assigning the i^{th} job to the j^{th} resource is c_{ij}. The aim is to assign each job to a different resource in such a way that

the total cost of the assignment is minimized. It is known that the constraint set of AP is unimodular and the optimal solution of the linear relaxation is equal to the optimal solution of the original problem.

In the multiobjective version of this problem, the cost of assigning the i^{th} job to the j^{th} resource with respect to the q^{th} objective is c_{ij}^{q}. The mathematical model of AP with p objectives is:

$$\text{"}min\text{"}\ Cx = \left(\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{1} x_{ij}, \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{2} x_{ij}, \ldots, \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{p} x_{ij} \right)$$

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, \ldots, n$$

$$x_{ij} \in \{0,1\} \quad i = 1, \ldots, n, \ \ j = 1, \ldots, n$$

where $x_{ij} = \begin{cases} 1 \text{ if job } i \text{ is assigned to resource } j \\ 0 \text{ otherwise} \end{cases}$

For AP, we use a random data generation scheme very similar to the one used by Przybylski, Gandibleux and Ehrgott (2007). They generate c_{ij}^{q} values from a discrete uniform distribution in the interval [0,20], where as we use the interval [1,20]in order to have strictly positive objective function values. We solve problems with 10, 20, 30 and 40 jobs.

The Knapsack Problem (KP)

In KP, there are n items and a knapsack with a known capacity, c. Each item j has a weight w_j and a value v_j. The aim is to select a subset of items in such a way that the total weight of the selected items does not exceed c while the total value of the selected items is maximized. *KP* is *NP*-Hard in the ordinary sense. We refer to Martello and Toth (1990) and Kellerer, Pferschy and Pisinger (2004) for further details on KP.

In the multiobjective version of KP, the value of the j^{th} item with respect to the q^{th} objective function is v_j^q. The mathematical model of KP with p objectives is

$$\text{"}max\text{"}\ \mathrm{V}x = \left(\sum_{j=1}^{n} v_j^1 x_j, \sum_{j=1}^{n} v_j^2 x_j, \ldots, \sum_{j=1}^{n} v_j^p x_j \right)$$

$$s.t. \quad \sum_{j=1}^{n} w_j x_j \le c$$

$$x_j \in \{0,1\} \qquad j = 1, \ldots, n$$

where $x_j = \begin{cases} 1 \text{ if item } i \text{ is selected} \\ 0 \text{ otherwise} \end{cases}$

ExA is developed for minimization problems. In order to apply *ExA* directly, we transform KP into a minimization problem:

$$\text{"}min\text{"}\ \mathrm{V}x = \left(UB_1 - \sum_{j=1}^{n} v_j^1 x_j, UB_2 - \sum_{j=1}^{n} v_j^2 x_j, \ldots, UB_p - \sum_{j=1}^{n} v_j^p x_j \right)$$

$$s.t. \quad \sum_{j=1}^{n} w_j x_j \le c$$

$$x_j \in \{0,1\} \qquad j = 1, \ldots, n$$

where $UB_q > \sum_{j=1}^{n} v_j^q x_j$ for all q and for all feasible solutions.

We use an upper bound strictly greater than the objective function values of all feasible solutions in order to ensure that objective function values of the minimization problem is strictly greater than zero.

We use the random data generation scheme used by Pamuk and Köksalan (2003). We generate w_j and v_j^q values from a discrete uniform distribution in the interval [60,100]. The capacity of the knapsack is c. The capacity is set as the nearest integer to $\frac{1}{2}\sum_{j=1}^{n} w_j$ in order to generate harder instances. We solve problems with 50, 75, 100, 150 and 200 items. We set $UB_q = \sum_{j=1}^{n} v_j^q$ since it is not possible to select all items at the same time with the capacity generation method used.

The Traveling Salesperson Problem (TSP)

In TSP, there are n cities. The distance between cities i and j is c_{ij}. A traveling salesperson is located at city 1 and has to plan a tour that visits each city exactly once. The salesperson's aim is to find a tour with the minimum total distance travelled. TSP is *NP*-Hard in the strong sense. We refer to Gutin and Punnen (2002) for further information on the TSP.

In the multiobjective version of the problem, the cost of traveling from city i to city j with respect to the q^{th} objective function is c_{ij}^q. The mathematical model of TSP with p objectives is

$$\text{"}min\text{"}\ \mathrm{C}x = \left(\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^1 x_{ij}, \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^2 x_{ij}, \ldots, \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^p x_{ij} \right)$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, \ldots, n$$

$$\sum_{\substack{(i,j)\in E \\ i\in U,\, j\in N\setminus U}} x_{ij} \geq 1 \ \text{ for } 2 \leq |U| \leq |N| - 2$$

$$x_{ij} \in \{0,1\} \quad i = 1, \ldots, n, \ \ j = 1, \ldots, n$$

where N is the set of all cities, U is a subset of N, E is the set of all city pairs and the decision variables are defined as

$$x_{ij} = \begin{cases} 1 \text{ if city } j \text{ is visited just after city } i \\ 0 \text{ otherwise} \end{cases}.$$

For TSP, we use a random data generation scheme similar to the one used in DIMACS STSP Implementation Challenge (see www.research.att.com/~dsj/chtsp). We generate the integer coordinates of the cities on a 1000 × 1000 square and calculate the Euclidian distances between cities. We generate p coordinates for each city and calculate p distance matrices. We solve problems with 5, 10, 15, 25 and 30 cities. We use Concorde, a special TSP solver developed by Applegate, Bixby, Chavatal and Cook to solve each TSP instance (see www.tsp.gatech.edu/concorde).

All objective function coefficients are integer valued in all three problems, but their weighted sums are not necessarily integers. We use a general-purpose solver for AP and KP where having rational coefficients for these two problems does not pose any concerns. However, Concorde uses only integer valued distances. Thus, we multiply the weighted sum values by a large number and round them to the nearest integer for TSP.

5.6.2 Computational Results

We code *ExA* on Microsoft Visual C++ 6.0 and test on a computer with Pentium M 1.6 GHz, 256 RAM and Microsoft Windows XP. We use Callable Library of CPLEX 8.1 for AP and KP and Concorde for TSP.

We generate 10 instances for each problem size–number of objective function combination. We solve these problems with *ExA*. We set the time limit as one hour and terminate the algorithm if it exceeds the time limit. We start solving small sized problems and increase the problem size. If *ExA* reaches the time limit in most of the instances then we stop increasing the problem size. In this section, we call *ExA* without any additional properties as *Base ExA*. We want to find as many points as possible during the time limit. Accordingly, *Base ExA* performs breadth first search.

In Table 5.1, we present the results of *Base ExA* for $p = 3$. The first two columns show the problem type and problem size, respectively. The third column shows the number of instances (out of 10) that *Base ExA* could not solve in one hour. If *Base ExA* stops before the time limit is reached, then $Y_E' = Y_E$, otherwise $Y_E' \subseteq Y_E$. We separately report the results for instances $Y_E' = Y_E$ and $Y_E' \subseteq Y_E$ because mathematical operations on their results do not make sense. The next four columns are for the instances where $Y_E' = Y_E$. The CPU column shows the average run time of the algorithm in seconds. Average numbers of extreme supported nondominated points are reported in the $|Y_E|$ column. The third column is $\frac{SC}{|Y_E|}$ where *SC* stands for the

number of solver calls, i.e., number of times $MOIP(\lambda)$ is solved. This column shows the average number of solver calls to find each extreme supported nondominated point. The $|V|$ column shows the number of stages visited during the search. The last four columns are for the instances where $Y_E' \subseteq Y_E$. We report the average $|Y_E'|$, $\frac{SC}{|Y_E'|}$ and $|V|$ values when the algorithm stops at the one hour time limit. At the end of each iteration, the algorithm checks *L* and stops if *L* is empty. However, if the algorithm stops due to the time limit, then *L* may not be empty. In the last column we report the average of $\frac{|V|}{|V|+|L|}$, i.e., the ratio of the number of visited stages to the number of opened and visited stages.

In the one hour time limit, *Base ExA* can solve APs up to 30 jobs, KPs up to 150 items and TSPs up to 25 nodes. The average CPU time and the average $|V|$ increases rapidly as problem size increases. Average $|Y_E|$ and average $\frac{SC}{|Y_E|}$ also increase with the problem size. It is observed that all indicators increase faster in AP and TSP compared to KP. We think that the number of decision variables may have an effect on this result. Note that the number of decision variables in KP is O(*n*) where as it is O(n^2) in AP and TSP.

We report result for $p = 4$ in Table 5.2. In general, results are similar to the $p = 3$ case. In the one hour time limit, *Base ExA* can solve APs with 10 jobs, KPs up to 100 items and TSPs with 10 nodes. The reason of these decreases is the rapid increase in average $|Y_E|$. Increasing *p* from 3 to 4 enormously effects Y_E. Consider AP with 20 jobs. Average $|Y_E|$ is nearly 150 for $p = 3$. For $p = 4$, *Base ExA* can search only half of the opened stages in the time limit but average $|Y_E'|$ is nearly 1000.

Table 5.1 Results of *Base ExA* ($p = 3$).

Problem	n	#	$Y'_E = Y_E$				$Y'_E \subseteq Y_E$			
			CPU (seconds)	$\lvert Y_E \rvert$	$\frac{SC}{\lvert Y_E \rvert}$	$\lvert V \rvert$	$\lvert Y'_E \rvert$	$\frac{SC}{\lvert Y'_E \rvert}$	$\lvert V \rvert$	$\frac{\lvert V \rvert}{\lvert V \rvert + \lvert L \rvert}$
AP	10	0	0.05	30.8	7.5	264.8				
	20	0	31.25	156.9	75.0	14369.5				
	30	1	1719.03	368.3	246.7	109894.8	397.0	367.1	171833.0	99.6
	40	10					640.5	214.0	154290.4	85.4
KP	50	0	4.39	48.4	14.5	978.0				
	75	0	44.60	82.2	32.6	3688.5				
	100	0	194.80	122.4	40.0	6201.0				
	150	2	1443.71	217.1	96.4	25652.8	309.0	86.3	30551.0	85.7
	200	10					429.8	70.4	33820.5	80.4
TSP	5	0	0.83	4.7	3.6	19.0				
	10	0	15.27	25.2	6.8	205.8				
	15	0	163.47	63.8	26.0	2165.1				
	25	3	2396.85	198.3	94.8	23152.4	258.7	101.9	29579.7	89.3
	30	10					316.1	60.3	21552.9	83.9

Table 5.2 Results of *Base ExA* ($p = 4$).

Problem	n	#	$Y'_E = Y_E$				$Y'_E \subseteq Y_E$			
			CPU (seconds)	$\lvert Y_E \rvert$	$\frac{SC}{\lvert Y_E \rvert}$	$\lvert V \rvert$	$\lvert Y'_E \rvert$	$\frac{SC}{\lvert Y'_E \rvert}$	$\lvert V \rvert$	$\frac{\lvert V \rvert}{\lvert V \rvert + \lvert L \rvert}$
AP	10	0	212.23	104.6	194.4	30734.8				
	20	10					980.7	104.9	131102.9	51.2
KP	50	1	618.98	129.6	174.0	37753.7	222.0	535.5	168326.0	94.7
	75	9	339.53	149.0	87.7	19169.0	415.8	255.6	133182.1	74.8
	100	9	1520.69	337.0	109.3	55784.0	529.8	135.6	86384.4	66.4
TSP	5	0	1.54	5.4	5.8	38.1				
	10	0	710.63	70.6	105.6	12430.3				
	15	10					297.0	130.3	48560.3	71.3

We discussed three improvements to *ExA* in Section 5.5. We must decide the best combination of them. We observe that using the nondominated facets property substantially decreases the CPU time whereas it only uses negligible additional memory (compared to *L*, *V* or *F* lists) for keeping the gathered information. Hence we use the nondominated facets property in the best combination. We perform a set of preliminary runs in order to decide the remaining properties.

We solve three instances for each combination and for each problem with p=3. We aim to select the problems that can be solved in nearly 1800 seconds. However, in order to decrease the variance of CPU times, we select the instances that can be solved in nearly 1900 seconds. In Table 5.3, we report average CPU times for each combination. BF, DF and BF+NF columns stand for breadth first, depth first and the third (breadth first plus nondominated facets) queuing disciplines, respectively. In the last column, we report the average CPU times of *Base ExA* in order to provide an insight into the effect of the nondominated facets property. We see that using the pre-calculation property with the depth first queuing discipline is the best combination for all problems. We apply the paired T-Test and observe that, statistically speaking this combination is significantly better than the others.

Table 5.3 CPU times (sec) of the preliminary runs

Problem	Pre calculation	Queuing Disciplines			*Base ExA* (BF)
		BF	DF	BF+NF	
AP	Not used	589.48	826.13	669.28	1927.20
	Used	524.56	441.82	573.93	-
KP	Not used	822.22	830.48	825.23	1916.42
	Used	109.51	89.22	111.74	-
TSP	Not used	858.20	704.77	853.04	1901.65
	Used	117.14	111.64	117.89	-

Let us call *ExA* with the pre-calculation and the nondominated facets properties and the depth first queuing discipline combination *Best ExA*. In Tables 5.4 and 5.5, we report the results for *Best ExA* for $p = 3$ and $p = 4$, respectively.

As expected, *Best ExA* performs better than *Base ExA*. *Base ExA* cannot solve 11 AP, 12 KP and 13 TSP instances with p = 3 in the one hour time limit. Among these instances, *Best ExA* solves 1 AP, 11 KP and 13 TSP instances. Similarly, for p = 4, *Base ExA* cannot solve 10 AP, 19 KP and 10 TSP instances. *Best ExA* solves 10 KP and 10 TSP instances among them. CPU times also decrease considerably with *Best ExA*. There is important decrease in the average number of solver calls per point found in Y_E. This is due to the pre-calculation property, which decreases the number of $MOIP(\lambda)$s solved, but increases the number of stages in L. However, in general, there is a decrease in average $|V|$ because the nondominated facets property eliminates many stages. *Best ExA* is more effective on KP and TSP compared to AP. That is, the improvements in CPU time and the number of instances solved in the one hour limit are much better for KP and TSP. This may be related to the computational complexities of the problems. Solving fewer $MOIP(\lambda)$s affects CPU time more for KP and TSP because they are *NP*-Hard problems. There is an interesting property on the average $|Y_E'|$ and $|V|$ values. For all problems, problem sizes and number of objectives, instances that cannot be solved in the one hour time limit have more extreme supported nondominated points than those can be solved, i.e., the average $|Y_E'|$ is greater than the average $|Y_E|$. This observation is also valid for average $|V|$ values. More extreme supported nondominated points may define more nondominated facets and the algorithms require more stages to search and define these nondominated facets.

Table 5.4 Results of *Best ExA* ($p = 3$).

Problem	n	#	$Y'_E = Y_E$				$Y'_E \subseteq Y_E$			
			CPU (seconds)	$\lvert Y_E \rvert$	$\frac{SC}{\lvert Y'_E \rvert}$	$\lvert V \rvert$	$\lvert Y'_E \rvert$	$\frac{SC}{\lvert Y'_E \rvert}$	$\lvert V \rvert$	$\frac{\lvert V \rvert}{\lvert V \rvert + \lvert L \rvert}$
AP	10	0	0.02	30.8	3.2	199.9				
	20	0	3.15	156.9	3.0	5429.2				
	30	0	490.80	371.2	3.0	58559.2				
	40	10					531.2	3.0	172176.1	92.7
KP	50	0	0.86	48.4	3.2	472.2				
	75	0	3.43	82.2	3.1	1212.8				
	100	0	10.48	122.4	3.1	2175.8				
	150	0	95.60	235.5	3.1	16700.5				
	200	1	1065.34	417.6	3.0	82621.1	540.0	2.3	172429.0	93.7
TSP	5	0	0.81	4.7	3.1	19.1				
	10	0	6.27	25.2	3.1	156.5				
	15	0	17.09	63.8	3.0	811.9				
	25	0	141.78	216.4	3.0	12840.2				
	30	0	336.16	316.1	3.0	28750.7				

Table 5.5 Results of *Best ExA* ($p = 4$).

Problem	n	#	$Y'_E = Y_E$				$Y'_E \subseteq Y_E$			
			CPU (seconds)	$\lvert Y_E \rvert$	$\frac{SC}{\lvert Y'_E \rvert}$	$\lvert V \rvert$	$\lvert Y'_E \rvert$	$\frac{SC}{\lvert Y'_E \rvert}$	$\lvert V \rvert$	$\frac{\lvert V \rvert}{\lvert V \rvert + \lvert L \rvert}$
AP	10	0	1.70	104.6	5.5	4110.2				
	20	10					701.8	6.1	165528.6	72.8
KP	50	0	16.84	138.8	5.3	8063.0				
	75	3	723.71	318.4	5.5	78448.6	380.3	4.8	187782.3	87.2
	100	6	417.18	389.5	5.3	55774.0	414.8	5.1	190350.7	89.6
TSP	5	0	1.14	5.4	4.5	36.8				
	10	0	31.76	70.6	5.2	2051.5				
	15	0	560.29	297.0	6.2	55119.6				

Przybylski, Gandibleux and Ehrgott (2007) proposed two versions for their algorithm and reported computational results for AP with $p = 3$ up to 50 jobs. Let us call their better algorithm *PGE*. In Table 5.6, we compare the performance of *PGE* and *Best ExA*. For problems with 10 jobs, *PGE* and *Best ExA* are performing similarly. However, *PGE* outperforms *Best ExA* for larger sized instances. We believe that $p = 3$ is a special case for *PGE* because the dimension of the weight space is two and *CAN* is directly applicable to solving biobjective problems in the weight space. *PGE* may not have this advantage with $p \geq 4$ and it may perform worse. In order to compare *PGE* and *Best ExA*, a computational test on several MOCO problems with different number of objective functions should be done. However, the coding of *PGE* for four or more objectives is not straight forward and conducting such a comparison may not easy.

Table 5.6 Comparison of *PGE* and *Best ExA* for AP ($p = 3$).

	PGE		*Best ExA*	
n	CPU (seconds)	$\lvert Y_E \rvert$	CPU (seconds)	$\lvert Y_E \rvert$
10	0.02	43	0.02	30.8
20	0.22	146	3.15	156.9
30	1.14	351	490.80	371.2
40	6.20	728	≥ 3600	≥ 640.5
50	16.48	1150		

ExA performs a search by generating new stages. We consider all possible stages as the search space of *ExA*. We provided an upper bound on the number of possible stages in Theorem 5.6 as:

$$|V| \le \binom{|Y_E \cup Y_M|}{p} = \frac{|Y_E \cup Y_M|!}{p!(|Y_E \cup Y_M| - p)!}.$$

This bound is a theoretical bound and represents the worst case. In practice *ExA* visits only a small portion of these stages. In Table 5.7, we compare the ratio of $|V|$ to its theoretical upper bound. We report the average of this ratio for the instances solved within the time limit. The ratio decreases as the problem size increases. Also for a given problem size, the ratio is smaller for $p = 4$. Increasing *n* or *p* increases $|Y_E|$. As $|Y_E|$ increases, any randomly selected two points are more unlikely to be the common members of a nondominated facet.

Table 5.7 Percent of the search space visited.

Problem	*n*	$p = 3$		$p = 4$	
		Base ExA	*Best ExA*	*Base ExA*	*Best ExA*
AP	10	4.786	3.704	0.413	0.079
	20	2.020	0.787		
	30	1.244	0.620		
	40				
KP	50	3.731	2.376	0.245	0.065
	75	2.676	1.049	0.087	0.015
	100	1.735	0.648	0.010	0.005
	150	1.363	0.590		
	200		0.635		
TSP	5	39.095	39.381	25.081	25.308
	10	5.746	4.732	0.714	0.206
	15	3.574	1.714		0.017
	25	1.593	0.626		
	30		0.524		

CHAPTER 6

GENERATING AN APPROXIMATE SET OF EXTREME SUPPORTED NONDOMINATED POINTS IN MULTIOBJECTIVE PROBLEMS

6.1 Introduction

In the previous chapter, we developed an exact algorithm, *ExA*, that generates all extreme supported nondominated points of a *MOIP* with *p* objectives. As the problem size or the number of objective functions increase, *ExA* needs more CPU time. At the end of *ExA*, $Y_E' = Y_E$. It is possible to stop the algorithm at some iteration and use Y_E' which is a subset of Y_E. However such an approach has important drawbacks. We do not know whether Y_E' is equal to Y_E or not. Moreover, we have no idea about how close Y_E' is to Y_E.

We observed that if we use the breadth first search strategy in *ExA*, then most of the points in Y_E are found at the beginning of the algorithm. The algorithm finds only a small portion of the points in the remaining iterations and mostly tries to prove that $Y_E' = Y_E$ (see Figure 5.8 and Figure 5.9). If we can measure how close Y_E' is to Y_E, then we may terminate the algorithm early.

In this chapter, we propose an approximation algorithm. This algorithm utilizes a lower bound set and an upper bound set for Y_E. It provides a proximity measure between these bound sets. Hence, we can stop the algorithm when the proximity measure is less than a predetermined level.

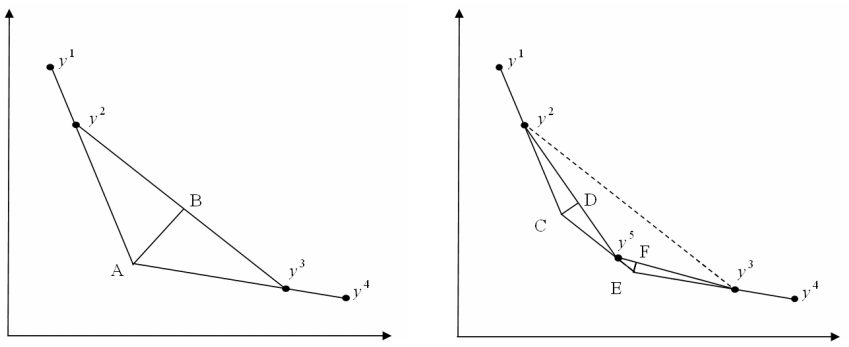
6.2 *CAN* as an Approximation Algorithm

Aneja and Nair (1979) proposed *CAN* as an exact algorithm to find all extreme supported nondominated points. However, Cohon (1978) discussed *CAN* not only as an exact algorithm but also as an approximation algorithm to find a subset of extreme supported nondominated points for bicriteria problems.

Cohon (1978) provided lower and upper bound sets for Y_E and a proximity measure between them. Consider Figure 6.1a. Assume that, points y^1, y^2, y^3 and y^4 are in Y_E' and stages $\left(y^1, y^2\right)$ and $\left(y^3, y^4\right)$ are facet defining. There is only stage $\left(y^2, y^3\right)$ in *L*. It is not possible to have extreme supported nondominated points in the southwest regions of lines passing through y^1 and y^2 and through y^3 and y^4. Also there is no extreme supported nondominated point in the convex hull defined by points in Y_E'. In the extreme case, we obtain point A by solving $MOIP(\lambda)$ for stage $\left(y^2, y^3\right)$. Let line segment [A,B] be perpendicular to the line passing through y^2 and y^3 and AB be the length of this line segment. If AB is smaller than a predetermined accuracy level then we can stop. Otherwise we should solve $MOIP(\lambda)$ for stage $\left(y^2, y^3\right)$. Assume that we want more accuracy and let y^5 in Figure 6.1b be the optimal solution of $MOIP(\lambda)$ for stage $\left(y^2, y^3\right)$. We remove stage $\left(y^2, y^3\right)$ from *L* and add two new stages: $\left(y^2, y^5\right)$ and $\left(y^3, y^5\right)$.We draw a line passing through y^5 that is parallel to the dashed line joining y^2 and y^3. Point y^5 is the optimal point of $MOIP(\lambda)$ for λ, which is equal to this line's normal vector. Hence, it is not possible to have an extreme supported nondominated point in the southwest of this line. We obtain two points as new extremes, C and E. If we solve $MOIP(\lambda)$ for stage $\left(y^2, y^5\right)$, in the

extreme case, we can obtain point C and the proximity of this point to the line passing through points y^2 and y^5 is CD. Similarly for stage $\left(y^3, y^5\right)$, the proximity is equal to EF. If the maximum of CD and EF is less than the desired accuracy level then we may stop the algorithm.

Note that the convex hull defined by points in Y_E' is used as the upper bound set for Y_E. The lines passing through points y^1 and y^2, C and E, and y^3 and y^4 shown in Figure 6.1b are used as lower bounds for Y_E.



a)Gap = $|AB|$ b) Gap = $min\{|CD|,|EF|\}$

Figure 6.1 Approximation of *CAN*

This approach works for biobjective problems. However, due to the difficulties discussed in Section 5.2, it is not directly applicable to problems with three or more objectives. Solanki, Appino and Cohon (1993) developed an approximation algorithm applicable to *MOLP*'s with p objectives. Their algorithm is an extension to the approximation version of *CAN* algorithm. They propose some approaches to solve the problems that arise when $p \geq 3$.

The algorithm and the approach we discuss in this chapter are quite different from that of Cohon (1978) and Solanki, Appino and Cohon (1993). Both approaches make approximations for each stage. Since the weight vectors are known for their cases, they operate in the objective space, as demonstrated for the biobjective case in Figure 6.1. In our approach, we know the lower and upper bounds in the objective space and we operate in the weight space.

6.3 Lower and Upper Bound Sets for Y_E

In this section, we develop lower and upper bound sets for Y_E. We define partial ideal points for stages and prove that these points define a lower bound set for Y_E. We next show that the points in Y_E' naturally define an upper bound set for Y_E.

The partial ideal point of stage R is $I(R)=\left(I_1(R), I_2(R),\ldots, I_p(R)\right)$ where $I_q(R)=\min_{r^k\in R}\left\{r_q^k\right\}$. The set of points dominated by the partial ideal point $I(R)$ is $B_I^{\geq}(R)=\left\{y=\left(y_1,y_2,\ldots,y_p\right)\middle| y_q\geq I_q(R)\ \ \forall q\right\}$ and the set of points dominating $I(R)$ is $B_I^{\leq}(R)=\left\{y=\left(y_1,y_2,\ldots,y_p\right)\middle| y_q\leq I_q(R)\ \ \forall q\right\}$. Note that $I(R)\leq r^k$ and $I(R)\neq r^k$ for all $r^k\in R$ since all elements of R are extreme supported nondominated points.

In the following theorem, we prove that at any iteration of *ExA*, all extreme supported nondominated points are in the union set of the points dominated by partial ideal points.

Theorem 6.1. $y^k\in\bigcup_{R\in F\cup L} B_I^{\geq}(R)$ for all $y^k\in Y_E$.

Proof: By induction, we will prove that the following term holds for all $y^k\in Y_E$ at each iteration of *ExA*:

$$y^k \in \bigcup_{R \in F \cup L} B_I^{\geq}(R).$$

At the first iteration, $F \cup L = \left\{\left(m^1, m^2, ..., m^p\right)\right\}$ and the term holds since $I(R) = (0, ..., 0)$.

Assume that the term holds at some iteration of the algorithm. Then at that iteration, an element $R \in L$ is selected in Step 2 and the λ vector corresponding to stage *R* is calculated in Step 3.

If $\lambda \notin \mathbb{R}^p_>$, then *R* is removed from *L*, since it does not define a nondominated facet.

If $\lambda \in \mathbb{R}^p_>$, then $MOIP(\lambda)$ is solved and the optimal solution, r^*, is obtained in Step 4.1. If $r^* \in R$ then *R* is added to *F* (in Step 4.2) and removed from *L* (in Step 6) hence $F \cup L$ does not change. If $r^* \notin R$ then *R* is removed from *L* (in Step 6). In Step 4.3.1, *p* new stages are generated and the unvisited ones are added to *L*.

Let us define the k^{th} new stage as $R^k = \left(r^1, \ldots, r^{k-1}, r^*, r^{k+1}, \ldots, r^p\right)$, i.e., we replace the k^{th} element of *R* with r^* and obtain R^k. In order to analyze the possible changes in $y^k \in \bigcup_{R \in F \cup L} B_I^{\geq}(R)$ term, we must consider $I(R)$ and $I\left(R^k\right)$. There are two cases in the calculation of $I(R)$:

i) There exists a $r^k \in R$ such that $r_q^k > I_q(R)\ \forall q$, i.e., point r^k does not contribute to partial ideal point and is strictly dominated by it. Since $I\left(R^k\right) \leq I(R)$ then $B_I^{\geq}\left(R^k\right) \supseteq B_I^{\geq}(R)$. The term holds in the next iteration.

ii) There is no $r^k \in R$ such that $r_q^k > I_q(R)\ \forall q$. This means, every element of *R* has at least one objective function value equal to that of the partial ideal point. Without loss of generality we assume that $r_q^q = I_q(R)$. If there exits a *q* such that $r_q^* \leq r_q^q$ then $I\left(R^q\right) \leq I(R)$ and the term holds in the next iteration because $B_I^{\geq}\left(R^q\right) \supseteq B_I^{\geq}(R)$. However, if $r_q^* > r_q^q$ for all *q* then we define the

convex hull $S = conv\left(I\left(R^1\right), I\left(R^2\right), \ldots, I\left(R^p\right), r^*\right)$. In this case, S is the region removed from the term. However, it is not possible to have a point y such that $y \neq r^*$ and $y \in Y_E \cap S$, otherwise r^* will be dominated by that y.

Therefore after one iteration, $y^k \in \bigcup_{R \in F \cup L} B_I^{\geq}(R)$ still holds. □

Corollary 6.1. For each $y^k \in Y_E$, there exists a stage $R \in F \cup L$, such that $I(R) \leq y^k$.

Proof: Follows directly from Theorem 6.1. □

The term $\bigcup_{R \in F \cup L} B_I^{\geq}(R)$ defines a nonconvex hypervolume. In the next theorem, we show that there exists no $y^k \in Y_E$ dominating the partial ideal point of a stage $R \in F \cup L$.

Theorem 6.2. For a given stage $R \in F \cup L$ such that $\lambda^R \in \mathbb{R}_>^p$, $Y_E \cap B_I^{\leq}(R) = \varnothing$.

Proof (by contradiction): Assume that there exists a $y \in Y_E$ and $Y_E \cap B_I^{\leq}(R) = y$. Since $I(R) \leq r^k$ and $I(R) \neq r^k$ for all $r^k \in R$, and $y \in B_I^{\leq}(R)$.

$$\lambda^R y \leq \lambda^R I(R) < \lambda^R r^k \text{ for all } r^k \in R.$$

Moreover, since $I(R)$ dominates every $r^k \in R$, we can write the above term for any $\lambda \in W^0$.

$$\lambda y \leq \lambda I(R) < \lambda r^k \text{ for all } r^k \in R.$$

So y must dominate all $r^k \in R$. However we know that $r^k \in Y_E$ for all $r^k \in R$ hence it cannot be a dominated point. There is a contradiction and we conclude that no such y exists. □

In the following two theorems, we propose lower and upper bounds on the weighted objective function value of $MOIP(\lambda)$, for a given $\lambda \in \mathbb{R}^p_>$.

Theorem 6.3. For a given $\lambda \in \mathbb{R}^p_>$, $\min_{R \in F \cup L} \{\lambda I(R)\} < MOIP(\lambda)$.

Proof: Let the optimal solution of $MOIP(\lambda)$ be *y*. We know that $y \in Y_E$ and by Corollary 6.1, we know that each $y \in Y_E$ is dominated by a partial ideal point of a stage $R \in F \cup L$. For the given $\lambda \in \mathbb{R}^p_>$, the minimum weighted objective function value of partial ideal points must be strictly less than that of any $y \in Y_E$. Otherwise, some $y \in Y_E$ must weakly dominate a partial ideal point. □

Theorem 6.4. For a given $\lambda \in \mathbb{R}^p_>$, $MOIP(\lambda) \leq \min_{y^k \in Y'_E} \{\lambda y^k\}$.

Proof: Let the optimal solution of $MOIP(\lambda)$ be *y* and $\lambda y^* = \min_{y^k \in Y'_E} \{\lambda y^k\}$. It is not possible to have $\lambda y > \lambda y^*$. Since in that case, there exists a solution y^* better than *y* for the given λ. However, this contradicts with the assumption that *y* is the optimal solution of $MOIP(\lambda)$. □

Hence by Theorems 6.3 and 6.4, we know that for all $\lambda \in \mathbb{R}^p_>$

$$\min_{R \in F \cup L} \{\lambda I(R)\} < MOIP(\lambda) \leq \min_{y^k \in Y'_E} \{\lambda y^k\}$$

We can consider the set $\bigcup_{R \in F \cup L} \{I(R)\}$ as the lower bound set and Y'_E as the upper bound set for Y_E.

6.4 An Approximation Algorithm

In this section, we present an approach to measure the proximity between lower and upper bound sets. We then propose an approximation algorithm using this measure.

This algorithm is a variant of *ExA* with a special property, which ensures that the proximity values follow a nonincreasing pattern throughout the algorithm. By this property, if the proximity is below a predetermined value, we conclude that Y_E' is close enough to Y_E and stop the algorithm. We call this approximation algorithm *ApA*.

6.4.1 The Approximation Approach

We know that for all $\lambda \in \mathbb{R}^p_>$, $\min_{R \in F \cup L} \{\lambda I(R)\} < MOIP(\lambda) \leq \min_{y^k \in Y_E'} \{\lambda y^k\}$ holds. For a given $\lambda \in \mathbb{R}^p_>$, we can identify the value of best known point $UB = \min_{y^k \in Y_E'} \{\lambda y^k\}$ as an upper bound for the optimal value of $MOIP(\lambda)$. Similarly, any point must have a value strictly greater than the lower bound, $LB = \min_{R \in F \cup L} \{\lambda I(R)\}$. If the gap between *LB* and *UB* is less than an acceptable level, then we do not need to solve $MOIP(\lambda)$, instead we can use the best known point in Y_E'. Let us define the gap between *LB* and *UB* for a given $\lambda \in \mathbb{R}^p_>$ as

$$\frac{UB - LB}{UB} = \frac{\min_{y^k \in Y_E'} \{\lambda y^k\} - \min_{R \in F \cup L} \{\lambda I(R)\}}{\min_{y^k \in Y_E'} \{\lambda y^k\}} = 1 - \frac{\min_{R \in F \cup L} \{\lambda I(R)\}}{\min_{y^k \in Y_E'} \{\lambda y^k\}}.$$

We prefer smaller values of this ratio, which means that the bounds are closer. We use a ratio term instead of $UB - LB$ due to following reasons. Using the ratio of *UB-LB* to *UB* eliminates the effect of the objective function ranges. This is an important advantage because we may not have enough information about these ranges at the beginning of the search process. Simply subtracting two values may be misleading because the ranges of objective function values may change from one instance to another. In other words, while the ratio is unitless, the difference is not. Also λ is not necessarily a normalized vector and the gap obtained by subtracting the two values will be different for λ vectors normalized to different constants, i.e., 1 or 100. The ratio definition eliminates this problem as well.

In the approximation algorithm, we assume that λ is not known but the lower and upper bound sets are given. Then we define the proximity measure, *pm*, as

$$pm = max\left\{\frac{UB - LB}{UB}\right\} = 1 - \min_{\lambda \in \mathbb{R}^p_>}\left\{\frac{\min\limits_{R \in F \cup L}\{\lambda I(R)\}}{\min\limits_{y^k \in Y'_E}\{\lambda y^k\}}\right\}.$$

The proximity measure finds the worst $\lambda \in \mathbb{R}^p_>$ such that the gap is maximum. Let us define the optimal solution to this problem as

$$pm = 1 - \frac{\lambda^{pm} I\left(R^{pm}\right)}{\lambda^{pm} y^{pm}}.$$

Since partial ideal points dominate points in Y'_E, $pm > 0$. In the worst case, partial ideal point can be the origin and $\lambda^{pm} I\left(R^{pm}\right) = 0$ then $pm = 1$. Hence $0 < pm \leq 1$. In its current form, we need to solve a nonlinear problem to find the optimal *pm* value. In Section 4.3, we discuss a solution method to obtain the optimal *pm* value.

Assume that, we calculate the *pm* value at Step 2 in each iteration of *ExA* and record this value. At some iteration, we may observe an increase in the *pm* value, which means that the proximity between lower and upper bound sets increased.

We can explain this situation as follows. After an iteration, Y'_E is either unchanged or a new point is added to it. For a given λ^{pm}, it is not possible to have a larger *UB* value compared to that of previous iteration. Because the minimum of a set does not increase by adding new elements to it. For *LB*, we expect a similar property since we expect the hypervolume defined by the term $\bigcup\limits_{R \in F \cup L} B_I^{\geq}(R)$ to get smaller as the algorithm proceeds. However, this is not the case and $\bigcup\limits_{R \in F \cup L} B_I^{\geq}(R)$ may expand after an iteration. Assume that we select stage *R* in Step 2 and obtain r^*. If $r^* \in B_I^{\geq}(R)$, then the hypervolume gets smaller as shown in the proof of Theorem 6.1. However, if $r^* \notin B_I^{\geq}(R)$ then the hypervolume may increase and $\min\limits_{R \in F \cup L}\left\{\lambda^{pm} I(R)\right\} \leq \lambda^{pm} I\left(R^{pm}\right)$. Hence, *LB* may decrease and there is a possibility to get a larger *pm* value in the next iteration.

We resolve this problem of fluctuating pattern in the *pm* value by introducing two new lists, *L*2 and *V*2, in addition to *L*, *V* and *F* lists. These lists are such that they guarantee adherence to a nonincreasing property for the *pm* value through iterations. By Theorem 6.1 we know that $y^k \in \bigcup_{R \in F \cup L} B_I^{\geq}(R)$ holds at any iteration of the algorithm for all $y^k \in Y_E$. Assume that, at some iteration of the algorithm, we set $L2 = L$ and $V2 = V$. Since *L*2 is equal to *L*, $y^k \in \bigcup_{R \in F \cup L2} B_I^{\geq}(R)$ also holds and we can use *L*2 in the *pm* calculation.

Let *pmL* and *pmL2* be the *pm* values obtained using *L* and *L*2, respectively as follows:

$$pmL = 1 - \min_{\lambda \in \mathbb{R}^p_>} \left\{ \frac{\min_{R \in F \cup L} \{\lambda I(R)\}}{\min_{y^k \in Y'_E} \{\lambda y^k\}} \right\} \text{ and } pmL2 = 1 - \min_{\lambda \in \mathbb{R}^p_>} \left\{ \frac{\min_{R \in F \cup L2} \{\lambda I(R)\}}{\min_{y^k \in Y'_E} \{\lambda y^k\}} \right\}.$$

We update *L*2, *V*2 and *pmL*2, only if $pmL < pmL2$. Therefore *pmL* may fluctuate through the iterations but *pmL2* follow a nonincreasing pattern.

6.4.2 Steps of the Algorithm

We provide the steps of *ApA* in Figure 6.2. Steps of *ApA* are similar to those of *ExA*. However, there are some important differences. We only discuss the new steps and the changes in the algorithm. In Step 1, we initialize *L*2 and *V2* lists as well. In Step 2, we calculate *pmL* and update *L*2, *V*2 and *pmL*2 if *L* is a better lower bound set. We assume that ε^{app} is a predetermined positive scalar representing the desired proximity between the bound sets. In Step 3, we compare *pmL*2 and ε^{app} and if $pmL2 \leq \varepsilon^{app}$, then we conclude that lower and upper bound sets are close enough. In this case, *ApA* stops and reports the points in Y'_E. The rest of the algorithm is similar to *ExA*.

ApA	
Initialize	1. Set $Y_E^{\varepsilon} = \varnothing$, $k = 0$, $V = \varnothing, F = \varnothing, L = \left\{\left(m^1, m^2, ..., m^p\right)\right\}, L2 = L, V2 = V$.
Search	2. Calculate $pmL = 1 - \min\limits_{\lambda \in \mathbb{R}^p_>} \left\{ \dfrac{\min\limits_{R \in F \cup L} \{\lambda I(R)\}}{\min\limits_{y^k \in Y'_E} \{\lambda y^k\}} \right\}$, if $pmL < pmL2$ then $L2 = L$, $V2 = V$ and $pmL2 = pmL$. 3. If $pmL2 \le \varepsilon^{app}$ then stop and report Y'_E. 4. Select a stage $R = \left\{r^1, r^2, ..., r^p\right\} \in L$ and set $V = V \cup \{R\}$. 5. Calculate λ such that $\lambda r^1 = \lambda r^2 = ... = \lambda r^p$. 6. If $\lambda \in \mathbb{R}^p_>$ 6.1. Solve problem $MOIP(\lambda)$ and let the optimal point be $r^* = \left(r_1^*, r_2^*, ..., r_p^*\right)$. 6.2. If $r^* \in R$ then set $F = F \cup \{R\}$. 6.3. If $r^* \notin R$ then 6.3.1. $L = L \cup \left\{\left\{r^1, ..., r^{p-1}, r^*\right\}, ..., \left\{r^*, ..., r^{p-1}, r^p\right\}\right\}$ and $L = L - (L \cap V)$. 6.3.2. If $r^* \notin Y'_E$ then $y^k = r^*$, $Y'_E = Y'_E \cup \left\{y^k\right\}$ and *k=k+1.* 7. $\lambda \notin \mathbb{R}^p_>$ then go to Step 8.
Loop	8. $L = L - \{R\}$. 9. If $L = \varnothing$ then stop and report Y_E, otherwise go to Step 2.

Figure 6.2 Steps of *ApA.*

6.4.3 Calculation of the Proximity Measure

Calculation of the *pm* value is crucial in *ApA* because the algorithm compares *pmL2* and ε^{app} values and decides whether to stop or to proceed. In this section, we discuss how we can solve the following problem and calculate the *pm* value.

$$pmL = 1 - \min_{\lambda \in \mathbb{R}^p_>} \left\{ \frac{\min_{R \in F \cup L} \{\lambda I(R)\}}{\min_{y^k \in Y'_E} \{\lambda y^k\}} \right\}.$$

In this problem, we can skip the constant term 1 during the optimization and minimize the second term. At the end of the optimization, we can convert the optimal solution to *pmL* value. We use $pmL' = 1 - pmL$ in the following mathematical models where

$$pmL' = \min_{\lambda \in \mathbb{R}^p_>} \left\{ \frac{\min_{R \in F \cup L} \{\lambda I(R)\}}{\min_{y^k \in Y'_E} \{\lambda y^k\}} \right\}.$$

In this term, we only impose $\lambda \in \mathbb{R}^p_>$. Hence we can set the denominator term to a constant and minimize the numerator term. We set the denominator to 1 in order to keep the range of *pmL* unchanged, i.e., $0 < pmL \leq 1$. The corresponding nonlinear programming problem is, *NLP*:

NLP

$$\min \left\{ \min_{R \in F \cup L} \{\lambda I(R)\} \right\}$$

s.t.

$$\min_{y^k \in Y'_E} \{\lambda y^k\} = 1$$

$$\lambda \in \mathbb{R}^p_>$$

We introduce a set of binary variables and change *NLP* into a mixed integer programming problem, *MIP*. The optimal solution value of *MIP* is pmL'.

MIP *min LB*

s.t.

$$\lambda y^k \geq 1 \qquad y^k \in Y_E'$$

$$\lambda I(R) \geq LB \qquad R \in F \cup L$$

$$\lambda I(R) - M(1 - B_R) \leq LB \qquad R \in F \cup L$$

$$\sum_{R \in F \cup L2} B_R \geq 1$$

$$\lambda \in \mathbb{R}^p_>$$

$$B_R \in \{0,1\} \qquad R \in F \cup L$$

where M is a sufficiently large number and binary variable B_R is defined as follows:

$$B_R = \begin{cases} 1 \text{ if } \lambda I(R) = LB \\ 0 \text{ otherwise} \end{cases}$$

In *MIP*, we define $\lambda I(R) \geq LB$ for each stage $R \in F \cup L$. Using the binary variables, we force the model to have $\lambda I(R) \leq LB$ for at least one stage $R \in F \cup L$. The number of binary variables in *MIP* increases as $|F \cup L|$ increases. This may result in longer solution times. However, we need to solve *MIP* frequently to measure the gap between the lower and upper bound sets. If we can reduce the size of *MIP*, we can reduce the total CPU time required by *ApA*.

Property 6.1. Consider two stages R^1 and R^2 such that $R^1, R^2 \in F \cup L$. If $I(R^1) \leq I(R^2)$ and $I(R^1) \neq I(R^2)$ then $\lambda I(R^1) < \lambda I(R^2)$ for any $\lambda \in \mathbb{R}^p_>$.

Using this property, we can reduce the size of *MIP*. Let us define *NI* as the set of stages whose partial ideal points are nondominated.

$$NI = \{R : R \in F \cup L, \nexists R' \in F \cup L \text{ such that } I(R') \leq I(R)\}.$$

We obtain the reduced model, *MIP_Reduced*, by replacing $F \cup L$ set with *NI* set as follows:

MIP_Reduced $\min LB$

s.t.

$$\lambda y^k \geq 1 \qquad y^k \in Y_E'$$

$$\lambda I(R) \geq LB \qquad R \in NI$$

$$\lambda I(R) - M(1 - B_R) \leq LB \qquad R \in NI$$

$$\sum_{R \in NI} B_R \geq 1$$

$$\lambda \in \mathbb{R}^p_>$$

$$B_R \in \{0,1\} \qquad R \in NI$$

As *ApA* proceeds, there will be more stages in the *NI* set and solving *MIP_Reduced* would require more CPU time. It may be even more time consuming than solving $MOIP(\lambda)$. Hence, we decompose *MIP_Reduced* into $|NI|$ linear programming problems. Consider the following LP, *pm(R)* for a stage $R \in NI$ and let the optimal solutions of *pm(R)* and *MIP_Reduced* be LB_R and LB^*, respectively.

pm(R) $\min \lambda I(R)$

s.t.

$$\lambda y^k \geq 1 \quad y^k \in Y_E'$$

$$\lambda \in \mathbb{R}^p_>$$

Theorem 6.5: $\min_{R \in NI} \{LB_R\} = LB^*$.

Proof: Let $LB_R^* = \min_{R \in NI} \{LB_R\}$ and $LB_R^* = \lambda^* I(R^*)$.

Since $LB_R^* = \min_{R \in NI} \{LB_R\}$ then $LB_R^* = \lambda^* I(R^*) \leq \lambda^R I(R) = LB_R$ where λ^R is the optimal weight vector of *pm(R)*. Moreover $\lambda^* I(R^*) \leq \lambda^R I(R) \leq \lambda^* I(R)$ holds because λ^* is not necessarily equal to λ^R. Hence setting $B_{R^*} = 1$ and $\lambda = \lambda^*$ is a feasible solution for *MIP_Reduced*. The optimal solution of $pm(R^*)$ is $LB_R^* = \lambda^* I(R^*)$ and we cannot obtain a better objective function value with a more restricted problem *MIP_Reduced*. □

This decomposition prevents us from solving a mixed integer programming problem frequently during *ApA*. Moreover, it brings another important advantage. The optimal solution of *pm(R)* may change only if a new point is added to Y_E' because we add a constraint to *pm*(*R*) for each point in Y_E' . So we do not need to solve *pm(R)* until a new point is added to Y_E'. Obviously, we must solve *pm(R)* for each new stage added to *NI*.

In *ApA*, we assume that we do not know Y_E and calculate the proximity between lower and upper bound sets. However, if Y_E is known, then we can calculate the real proximity measure, pm^{real} for a given Y_E'. Although, approximating a set which is already known does not make sense in practice, the comparison of *pmL*2 and pm^{real} values provide an insight on the tightness of the lower bound set. We provide a comparison of these two values in Section 6.7.

If we know Y_E, then we can calculate the pm^{real} value. We solve the following linear program for each $y \in Y_E \setminus Y_E'$, which is similar to *pm(R)*.

$$
\begin{aligned}
pm(y) \quad & \min \lambda y \\
& s.t. \\
& \lambda y^k \geq 1 \quad y^k \in Y_E' \\
& \lambda \in \mathbb{R}^p_{>}
\end{aligned}
$$

Let the optimal solution of *pm(y)* be LB_y, then the real proximity is

$$pm^{real} = 1 - \min_{y \in Y_E \setminus Y_E'} \{LB_y\}.$$

Since each $y \in Y_E \setminus Y_E'$ is dominated by a partial ideal point of a stage $pmL2 > pm^{real}$.

6.5 Tightening the Lower Bound Set

In this section, we discuss some properties to help us to tighten the lower bound set. By tightening the lower bound set, we may decrease the proximity between lower and upper bound sets.

6.5.1 Stages with Dummy Points

We start *ApA* with stage $\left(m^1, m^2, ..., m^p\right)$ having *p* dummy points. In later iterations, the algorithm generates new stages and some of these stages may have dummy points as elements. If a stage *R* has two or more dummy points then $I(R) = \{0, \ldots, 0\}$. If there is only one dummy point in a stage, then *p*-1 components of the partial ideal point are zero.

The stages with dummy points increase the value of *pm* substantially, since *p* or *p*-1 components of their partial ideal points are zero. However, we can improve these partial ideal points and the *pm* value by setting lower bounds on all objective function values. Let us define the lower bound on q^{th} objective function as $LB_q \le y_q$ for all $y \in Y$ and for all *q*. Hence we can redefine the partial ideal point of stage *R* as $I(R) = \{I_1(R), I_2(R), \ldots, I_p(R)\}$ where $I_q(R) = min\left\{LB_q, \min_{r^k \in R}\{r_q^k\}\right\}$.

For a given problem, we can find some underestimates of LB_q for each objective function and use them as LB_q values in the algorithm. However we can determine exact LB_q values during the algorithm and obtain better *pm* values.

Let y^0 be the first point found by the algorithm and R^q be the stage obtained from stage $\left(m^1, m^2, ..., m^p\right)$ by replacing y^0. Let m^q and y^q be the optimal solution of stage R^q. The weight vector corresponding to stage R^q is $\lambda^q = \left(\varepsilon, ..., 1-(p-1)\varepsilon, \ldots, \varepsilon\right)$ where ε is a very small positive scalar. Then $LB_q = y_q^q$.

6.5.2 Nondominated Facet Defining Stages

In Theorem 6.1, we proved that $y^k \in \bigcup_{R \in F \cup L} B_I^{\geq}(R)$ holds for all $y^k \in Y_E$. Using this property, we can obtain a lower bound for any weight vector at any iteration of the

algorithm. In Theorem 6.6, we show that we can tighten the lower bound by cropping some portion of $\bigcup_{R\in F\cup L} B_I^{\geq}(R)$.

Theorem 6.6. $y^k \in \left[\bigcup_{R\in L} B_I^{\geq}(R)\right] \cup \left[\bigcup_{R\in F} B^{\geq}(R)\right]$ for all $y^k \in Y_E$.

Proof: For a given stage $R=\left(r^1,\ldots,r^p\right)\in F$, consider two hypervolumes $B_I^{\geq}(R)$ and $B^{\geq}(R)$. Let us define $B_I^{\geq}(R)\setminus B^{\geq}(R)=S$. Note that we can also define S as follows:

$S=Conv\left(I(R),r^1,\ldots,r^p\right)/\left\{r^1,\ldots,r^p\right\}$.

It is not possible to have $y\in Y_E$ in *S*, because *R* is a facet defining stage. Hence $S\cap Y_E=\varnothing$ and we can use $B^{\geq}(R)$ term instead of $B_I^{\geq}(R)$ for nondominated facet defining stages. Moreover,

$y^k \in \left[\bigcup_{R\in L} B_I^{\geq}(R)\right] \cup \left[\bigcup_{R\in F} B^{\geq}(R)\right]$ holds for all $y^k \in Y_E$. □

By the above theorem, we do not need to solve *pm(R)* for facet defining stages, instead we may solve the following problem for each $r\in R$.

pm(r) $\quad \min \lambda r$

$\quad s.t.$

$\quad \lambda y^k \geq 1 \quad y^k \in Y_E'$

$\quad \lambda \in \mathbb{R}_{>}^{p}$

Since $r\in Y_E'$, the optimal solution of *pm(r)* is equal to 1 for each point in a nondominated facet defining stage. Hence we can skip the stages in *F* and consider only the stages in *L* for the *pm* value calculation.

6.5.3 Shifting Partial Ideal Points

We can also improve the *pm* value by shifting some of the partial ideal points. Consider a stage $R \in L$. There are two cases in the calculation of the partial ideal point of *R* as discussed in the proof of Theorem 6.1.

The first case is when there exists a $r^k \in R$ such that $r_q^k > I_q(R)\ \forall q$, i.e., r^k is not contributing to the partial ideal point and *I(R)* strictly dominates r^k. In this case, we can shift the partial ideal point by 1 unit as $I_q^s(R) = \min_{r^u \in R}\{r_q^u\} + 1$ because it is not possible to have a point $y \in Y_E \setminus Y_E'$ that dominates $I_q^s(R)$. If such a point $y \in Y_E \setminus Y_E'$ existed, it would also dominate r^k because $r^k = I_q^s(R)$ in the best case.

If there is no $r^k \in R$ such that $r_q^k > I_q(R)\ \forall q$, i.e., all points in *R* are contributing to partial ideal point, then no shifting is possible. To demonstrate, consider an example with $p = 3$, $R = (r^1, r^2, r^3)$ where points are $r^1 = (a, b+1, c+1)$, $r^2 = (a+1, b, c+1)$ and $r^3 = (a+1, b+1, c)$. Then $I(R) = (a, b, c)$ and $I^S(R) = (a+1, b+1, c+1)$. There may be a nondominated point $y = (a, b, c+2)$. This point dominates the shifted partial ideal point but not the original partial ideal point.

6.6 Improvements on the Approximation Algorithm

In this section, we discuss some properties in order to improve the performance of *ApA*. We develop a variant of pre-calculation property discussed in Chapter 5. We propose to use a different queuing discipline for selecting the next stage in Step 4 of *ApA*. We discuss a policy to replace *L* and *V* with *L*2 and *V*2.

6.6.1 Pre-Calculation with Partial Ideal Points

In Section 5.5.2, we discussed the pre-calculation property in order to decrease the number of $MOIP(\lambda)$'s solved. In pre-calculation, for a given R and the corresponding weight vector λ, we search for point r^*_{pc} such that $\lambda r^*_{pc} = \min_{y \in Y'_E} \{\lambda y\}$ and $r^*_{pc} \notin R$.

In this section, we assume that a stage R and the corresponding weight vector λ are given. However, our aim is not to find the minimum valued point for the given λ. Instead, we search for a point which will help us to reduce the hypervolume defined by $B^{\geq}_I(R)$. Hence, we may improve *pmL*.

We refer to this property as *ideal-pre-calculation*. We search for a point $r^*_{ideal} \in Y'_E$ which is not in R and dominated by partial ideal point of stage R, $I(R)$, i.e., $r^*_{ideal} \in \left(Y'_E \cap B^{\geq}_I(R)\right) \setminus R$. If there are many points in the defined hypervolume then we break the ties as follows:

$\lambda r^*_{ideal} = \min_{y \in \left(Y'_E \cap B^{\geq}_I(R)\right) \setminus R} \{\lambda y\}$.

We treat r^*_{ideal} as the optimal point corresponding to R. We remove R from L and add up to p new stages to L. As discussed in the proof of Theorem 6.1, we remove the convex hull $S = conv\left(I(R^1), I(R^2), \ldots, I(R^p), r^*_{ideal}\right)$ from $B^{\geq}_I(R)$ where $R^k = \left(r^1, \ldots, r^{k-1}, r^*_{ideal}, r^{k+1}, \ldots, r^p\right)$. That is, we replace the k^{th} element of R with r^*_{ideal} and obtain R^k.

We modify Step 6.1 of *ApA* as follows:

6.1 *Ideal-pre-calculation,* proceed to Step 6.1.1

6.1.1 Let r^*_{ideal} be such that $\lambda r^*_{ideal} = \min_{y \in \left(Y'_E \cap B^{\geq}_I(R)\right) \setminus R} \{\lambda y\}$.

6.1.2 If there exists such a point then $r^*_{ideal} = r^*$, go to Step 6.3.1, otherwise go to Step 6.1.3.

6.1.3 Solve $MOIP(\lambda)$ and let the optimal point be $r^* = (r^*_1, r^*_2, ..., r^*_p)$, go to Step 6.2.

In Step 6.1.1, we search the best available point r^*_{ideal}. In Step 6.1.2, if there is such a point then we set it as r^* and go Step 6.3.1, otherwise we proceed to Step 6.1.3 and solve $MOIP(\lambda)$.

Above, we discussed the use of the ideal-pre-calculation property for a given stage. This property deals with one stage at each iteration. However, *ApA* may add up to *p* new stages to *L*. Since the approximation approach considers all stages in *L* for *pm* value calculation, we can use this property to reorganize all stages in *L*. Consider the following algorithm. For each stage in *L* we search for r^*_{ideal} point. If we can find such a point we update *L* and *V*.

Reorganize_L

1. For each $R \in L$ perform the following steps.

1.1. Calculate λ such that $\lambda r^1 = \lambda r^2 = ... = \lambda r^p$.

1.2. Let r^*_{ideal} be such that $\lambda r^*_{ideal} = \min_{y \in (Y'_E \cap B^{\geq}_I(R)) \setminus R} \{\lambda y\}$.

1.3. If there exists such a point then $r^*_{ideal} = r^*$ and go to Step 1.4.
Otherwise go to Step 1.

1.4. $L = L \cup \{\{r^1, ..., r^{p-1}, r^*_{ideal}\}, \{r^1, ..., r^*_{ideal}, r^p\}, ..., \{r^*_{ideal}, ..., r^{p-1}, r^p\}\}$

$L = L - (L \cap V)$ and $V = V \cup \{R\}$.

6.6.2 Queuing Discipline

In Section 5.5.3, we discussed three queuing disciplines to select the next stage *R* from *L* for *ExA*. Those disciplines are all applicable to *ApA*. However, in *ApA*, our

main concern is to decrease the *pm* value by closing the gap between lower and upper bound sets. With this motivation, we define a new queuing discipline.

At each iteration of *ApA*, we first calculate *pmL* in Step 2. We select next stage in Step 4. Consider *pm*(*R*) (see Section 6.4.3) used for *pmL* calculation

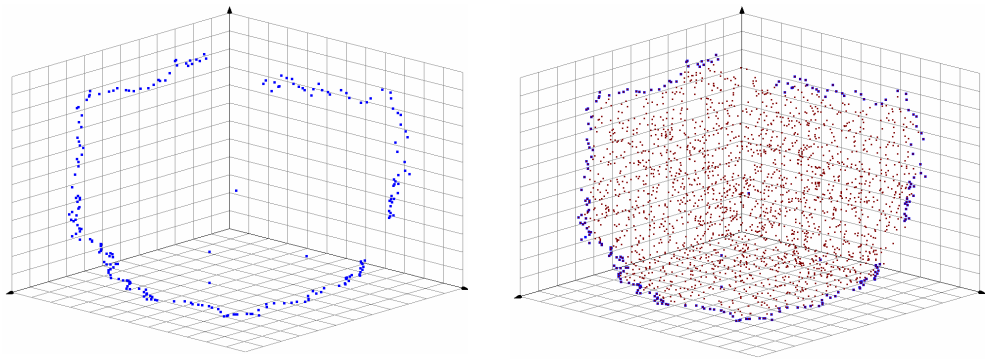
$$
\begin{aligned}
pm(R) \quad & \min \ \lambda I\left(R\right) \\
& s.t. \\
& \lambda y^k \geq 1 \quad y^k \in Y_E' \\
& \lambda \in \mathbb{R}^p_{>}
\end{aligned}
$$

The optimal solution of *pm*(*R*) is LB_R and $pmL = 1 - \min_{R \in L}\{LB_R\}$. Let $pmL = 1 - LB_{R^C}$. We refer to stage R^c as *critical stage*. This critical stage determines *pmL* and we cannot improve *pmL* if we do not select R^c as the next stage. Hence *ApA* selects critical stage as the next stage in every iteration.

As we discussed in Section 6.5.1, stages with dummy points have partial ideal points having *p*-1 or *p* zero components. We proposed to improve these partial ideal points by replacing zero components with lower bounds. In preliminary runs, we observed that these stages still have small *pm*(*R*) values and they are selected as the critical stage at the early iterations of *ApA*. Based on this observation, we modify our queuing discipline. We first select stages with dummy points from *L*, if there is no such stage in *L* then we select the critical stage.

Using this discipline has two advantages. We know that *pmL* is very high if there are stages with dummy points. Hence, we do not calculate *pmL* until all stages with dummy points are visited and removed from *L*. The second advantage is much more important. If there are dummy points in a stage, then the corresponding weight vector λ has some components very close to zero, except the first stage, $\left(m^1, m^2, ..., m^p\right)$. Hence *ApA* firstly determines the boundaries of the nondominated frontier. We consider a sample AP with $p = 3$ in Figure 6.3. We plot extreme supported nondominated points found only considering the stages with dummy

points in Figure 6.3a and a subset of Y_E in Figure 6.3b. Since one or more components of λ is very close to zero for the stages with dummy points, we have the advantage to analyze the tradeoffs between the other objective functions with larger components in λ.



a) Optimal points found by stages with dummy points

b) A subset of Y_E

Figure 6.3. Effect of the new queuing discipline

6.6.3 Recovery Policy

We developed *L*2, *V*2 and *pmL*2 in order to keep the best state of *L*, *V* and *pmL*, respectively, during the search of *ApA*. In some iterations, *pmL* may get larger, which means the hypervolume defined by $\bigcup_{R \in F \cup L} B_I^{\geq}(R)$ gets larger. When we face such a case, we do not need to let the algorithm increase the hypervolume. Since we keep the best state of the algorithm in *L*2 and *V*2, we can return to this state by setting $L = L2$ and $V = V2$. *ApA* restarts its search from the best state again. We refer this policy as the *recovery policy*.

Using the recovery policy may result in the cycling of the algorithm. Consider the following case. At some iteration, we update *L*2, *V*2 and *pmL*2 and select the next

stage R. The optimal point is $r^* \notin R$ but $r^* \in Y_E'$. Moreover, $r^* \notin B_I^{\geq}(R)$. We add new stages to L and remove R from L. Hypervolume defined by $\bigcup_{R \in F \cup L} B_I^{\geq}(R)$ increases. In the next iteration, *pmL* is larger than *pmL*2 and the recovery policy sets $L = L2$ and $V = V2$. *ApA* starts cycling in such a case.

In order to prevent the cycling of *ApA*, we propose a simple rule. We record $Y2 = |Y_E'|$ when *L*2, *V*2 and *pmL*2 are updated. We do not apply the recovery policy before a new point is added to Y_E' after the last update. Hence, *ApA* be recovered if $|Y_E'| > Y2$. Note that this rule prevents the cycling of *ApA* for the case discussed in the above paragraph. Having at least one new point in Y_E' since the last update has two advantages. We may have a chance to reorganize *L* with the help of new points. This may improve the lower bound set and decrease *pmL*. Another advantage of new points is the possible improvement in the upper bound set. Since we add new constraints to *pm*(*R*) for each new point, we may have a chance to decrease *pmL*.

Using recovery policy forces *ApA* to improve *pmL*2. This policy promotes improving moves of *ApA*, i.e., removing some hypervolume from $\bigcup_{R \in F \cup L} B_I^{\geq}(R)$. However, *ApA* may reach to a better state if it can first apply some nonimproving moves. We propose to apply recovery policy if *ApA* finds a predetermined number of new points after the last update. In computational tests, we use different numbers for this policy.

6.7 Computational Test

In this section, we conduct a computational test on *ApA*, and discuss the results of the test. We code *ApA* on Microsoft Visual C++ 6.0 and test on a computer with Pentium M 1.6 GHz, 256 RAM and Microsoft Windows XP. We use Callable Library of CPLEX 8.1.

We test *ApA* using the assignment problem with three objective functions. We use the same random data generation scheme discussed in Chapter 5 and solve problems with 30, 40, 50 and 60 jobs. We skip problems with 10 and 20 jobs since *Best ExA* solves them in less than 8 seconds of CPU time. We add problems with 50 and 60 jobs instead.

We apply three lower bound set tightening properties discussed in Section 6.5 to *ApA* because these properties can be processed in negligible CPU times. We also implement the improvements discussed in Section 6.6. After preliminary runs, we decided to apply the ideal-pre-calculation property. We also call *Reorganize_L* in *ApA*. In preliminary runs, we tested results of calling this algorithm in every Q iterations for $Q = \{1,10,25,50,100\}$ .We decided to call *Reorganize_L* in every 50 iterations because it is expensive to call it frequently in terms of CPU time. It also increases the number of stages L too much, which makes the *pm* value calculation harder. We use 5, 10, 15 and 20 as rule number required to apply the recovery policy in addition to its original value 1.

We run *ApA* for 5000 iterations. We report the results in Table 6.1. The first column is the problem size and the second column is the rule number for recovery policy. After every thousand iterations, we report the average *pmL*2 value. Average CPU time is reported in the last column. Note that, CPU times are reasonable, considering the fact that *Best ExA* solves AP with 30 jobs in nearly 490 seconds and cannot solve AP with 40 jobs in 3600 seconds. As rule number increases CPU time increases and *pmL2* values decreases. However, we do not observe a trend between problem size and *pmL2*. Table 6.2 is very similar to Table 6.1. In this table, we report the average ratio of $|Y_E'|$ to $|Y_E|$. As rule number increases the average ratio increases however this increase does not affect *pmL*2 value.

Table 6.1 Results of *ApA*.

n	Rule Number	*pmL2* (%)					CPU
		1000	2000	3000	4000	5000	
30	1	23.61	23.11	23.04	22.71	22.57	41.91
	5	21.94	21.58	21.36	21.24	21.18	93.13
	10	21.93	21.47	21.29	21.21	21.13	108.03
	15	22.16	21.82	21.68	21.64	21.63	115.00
	20	21.67	21.34	21.23	21.18	21.07	116.35
40	1	24.19	23.58	23.49	23.32	23.26	51.05
	5	22.28	21.97	21.90	21.86	21.80	128.07
	10	21.68	21.36	21.26	21.16	21.13	184.49
	15	21.63	21.31	21.24	21.21	21.19	196.85
	20	21.55	21.11	21.06	21.02	20.97	227.03
50	1	25.22	24.39	24.30	24.24	24.21	74.31
	5	23.03	22.62	22.25	22.09	22.05	285.14
	10	22.14	21.64	21.57	21.49	21.46	432.88
	15	21.97	21.47	21.42	21.38	21.37	438.93
	20	21.75	21.27	21.15	21.09	21.08	525.01
60	1	26.28	25.05	24.77	24.48	24.42	133.85
	5	23.80	23.39	23.22	23.10	22.94	439.59
	10	22.67	22.04	21.94	21.89	21.85	724.58
	15	23.27	22.64	22.59	22.45	22.44	739.86
	20	22.20	21.81	21.67	21.63	21.60	1031.03

Table 6.2 Percent of points found by *ApA*.

n	Rule Number	$\%\left(\lvert Y_E'\rvert / \lvert Y_E\rvert\right)$					CPU
		1000	2000	3000	4000	5000	
30	1	44.0	50.2	53.6	58.0	61.1	41.9
	5	55.0	65.4	72.4	76.8	80.2	93.1
	10	56.7	67.3	74.2	78.4	81.7	108.0
	15	57.9	69.6	75.1	79.2	82.5	115.0
	20	59.1	69.2	75.2	80.2	82.9	116.4
40	1	28.0	33.3	36.1	38.4	40.7	51.1
	5	37.8	47.3	52.5	56.1	59.3	128.1
	10	41.6	51.0	56.4	60.8	64.1	184.5
	15	42.5	52.0	57.6	61.5	64.6	196.8
	20	43.2	53.1	58.4	62.2	65.3	227.0
50	1	27.4	32.7	35.4	37.3	39.3	74.3
	5	37.1	47.7	55.6	61.4	65.0	285.1
	10	42.7	56.2	63.5	68.8	72.9	432.9
	15	43.1	57.2	64.2	69.4	73.6	438.9
	20	44.4	58.3	65.9	71.4	75.2	525.0
60	1	27.5	34.0	37.0	39.9	41.8	133.8
	5	37.1	48.5	56.0	62.3	68.4	439.6
	10	40.7	57.2	66.0	72.2	77.9	724.6
	15	42.2	58.0	65.8	72.3	77.3	739.9
	20	44.4	61.1	71.1	78.3	83.2	1031.0

In Table 6.3, we compare the *pmL*2 obtained after 5000 iterations to the pm^{real} values. As seen from the table, the pm^{real} values are very small compared to the *pmL*2 values. We use same upper bound sets in the calculation of both values. Hence we conclude that the lower bound set used by *ApA* is not tight enough.

Table 6.3 Performance of *pmL2* (rule number = 15).

	pmL2			pm^{real}		
n	Min	Ave	Max	Min	Ave	Max
30	17.06	21.63	25.59	0.29	0.91	1.33
40	16.50	21.19	25.10	0.49	1.14	1.71
50	19.73	21.37	23.00	0.64	1.30	1.79
60	20.55	22.44	24.91	0.96	1.35	2.03

6.8 Discussions

In this chapter, we developed an approximation algorithm to find a subset of extreme supported nondominated points of a MOIP. We proposed lower and upper bound sets for Y_E and provide a worst case proximity measure between the bound sets. We discussed a number of properties to improve the algorithm and the lower bound sets. We tested the algorithm on a MOCO problem.

CHAPTER 7

CONCLUSIONS AND FURTHER RESEARCH

In this thesis, we studied multiobjective combinatorial optimization problems. We can organize our study under two main topics. The first topic is about polynomially solvable cases of the Traveling Salesperson Problem (TSP) and the Bottleneck Traveling Salesperson Problem (BTSP). We considered multiobjective versions of these problems. To the best of our knowledge, there is no other study in the literature considering the solvable cases of TSP or BTSP with multiple objectives. Our second topic is generating extreme supported nondominated points of multiobjective integer programming problems. We developed algorithms to find such points of an integer programming problem with any number of objective functions.

We considered two solvable cases of TSP and BTSP: pyramidal tours and Halin graphs. For pyramidal tours, we studied the multiobjective TSP on a set of special distance matrices and showed some properties of nondominated points. We developed a pseudo-polynomial dynamic program to find a nondominated point when all distance matrices are in the same class. For the biobjective case, we developed an approach to find all nondominated points. We also demonstrated that the optimal tours of bottleneck types of Van der Veen matrices and Demidenko

matrices are not necessarily pyramidal. Hence the developments are not applicable to these cases.

For the Halin graphs, we addressed multiobjective problems with various combinations of TSP and BTSP–type objective functions. We showed that, if there are two or more TSP-type objective functions in the problem, then finding a nondominated point is *NP*-Hard, and there are exponentially many nondominated points. However, if there is at most one TSP-type objective function and all remaining objectives are BTSP-type, then the problem is polynomially solvable. We developed algorithms to find the nondominated points.

A future research topic is to study polynomially solvable cases of combinatorial problems in general. The approaches developed in this thesis may prove useful in some of those problems as well. Further analyzing the computational complexities of the studied problems is another future research topic.

In our second topic, we developed two algorithms for generating the extreme supported nondominated points of a multiobjective integer programming problem with any number of objective functions. The first algorithm is an exact algorithm and it finds all such points. This algorithm finds only extreme supported nondominated points and stops after a finite number of iterations. We proposed several improvements on this algorithm and tested on three well-known combinatorial optimization problems.

The second algorithm is an approximation algorithm and finds only a subset of the extreme supported nondominated points. The approximation algorithm keeps lower and upper bound sets for these points. The main feature of this algorithm is its worst case proximity measure between lower and upper bound sets. We proposed an approach that provides a nonincreasing proximity measure. We tested our approach on a well-known combinatorial optimization problem.

Finding the set of all extreme supported nondominated points or an approximation for it requires extensive computational effort, because there may be too many such points even for moderate size problems. It may be reasonable to focus on regions of nondominated frontier that are more interesting to the decision maker. Incorporating the preferences of the decision maker into both algorithms is an interesting future research direction.

We developed some properties in order to improve the performances of *ExA* and *ApA*. It may be possible to develop more properties using the multidimensional nature of the problems. The resulting algorithms may be capable of solving larger instances.

Another research direction is to conduct an experimental study on MOCO problems. In this study, we can figure out some properties on the cardinality of the nondominated set and that of extreme supported nondominated points.

The approximation algorithm uses the partial ideal points as the lower bound set. We proposed some properties to tighten these bounds. However, these bounds, in their current state, are not tight enough for practical purposes. We plan to work on improving the performance of these bounds. Also another research direction is to incorporate other approximation methods with our exact algorithm.

REFERENCES

Aneja, Y.P., Nair, K.P., 1979. Bicriteria transportation problem, *Management Science*, 25(1), 73- 78.

Benson, H.P., Sun, E., 2000. Outcome space partition of the weight set in multiobjective linear programming, *Journal of Optimization Theory and Applications*, 105, 17–36.

Benson, H.P., Sun, E., 2002. A weight decomposition algorithm for finding all efficient extreme points in the outcome set of a multiple objective linear program, *European Journal of Operational Research*, 139, 26–41.

Bondy, J.A., Murty, U.S.R., 1979. *Graph Theory with Applications*. North-Holland, New York.

Burkard, R.E., Deineko, V.G., 2004. On the Euclidean TSP with permuted Van Der Veen matrix, *Information Processing Letters*, 91, 259-262.

Burkard, R.E., Deineko, V.G., Van Dal, R., Van der Veen, J.A.A., Woeginger, G.J., 1998. Well-solvable special cases of the traveling salesman problem: a survey, *SIAM Review*, 40(3), 496-546.

Burkard, R.E., Sandholzer, W., 1991. Efficiently solvable special cases of bottleneck traveling salesman problems, *Discrete Applied Mathematics*, 32, 61-76.

Burkard, R.E., Klinz, B., Rudolf, R., 1996. Perspectives of Monge properties in optimization, *Discrete Applied Mathematics*, 70, 95-161.

Cohon, J.L., 1978. *Multiobjective Programming and Planning.* Academic Press, New York.

Coullard, C.R., Rais, A., Rardin, R.L., Wagner, D.K., 1993. Linear-time algorithms for the 2-connected Steiner subgraph problem on special classes of graphs, *Networks*, 23, 195-206.

Cournejols, G., Naddef, D., Pulleyblank, W.R., 1983. Halin graphs and the traveling salesman problem, *Mathematical Programming*, 26, 287-294.

Deineko, W.G., Woeginger, G.J., 2000. A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem, *Mathematical Programming*, 87, 519-542.

Ehrgott M., 2000. Multicriteria optimization, *Lecture Notes in Economics and Mathematical Systems*, Springer, Berlin.

Ehrgott M., Gandibleux, X., 2000. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22, 425-460.

Ehrgott M., Gandibleux, X., 2002. Multiobjective combinatorial optimization – theory, methodology, and applications. Ehrgott, M., ed. *Multiple criteria optimization: state of the art annotated bibliographic surveys*, Kluwer Academic Publishers, USA, 369-444.

Ehrgott M., Gandibleux, X., 2004. Approximative solution methods for multiobjective combinatorial optimization, *Sociedad de Estadistica e Investigacion Operative, TOP*, 12,1-89.

Ehrgott M., Gandibleux, X., 2007. Bound sets for biobjective combinatorial optimization problems, *Computers & Operations Research*, 34, 2674-2694.

Gamarnik, D., Lewenstein, M., Sviridenko, M., 2004. An improved upper bound for the TSP in cubic 3-edge-connected graphs, *Operations Research Letters*, **33**(5)**,** 467-474.

Gilmore, R.C., Lawler, E.L., Shmoys, D.B., 1985. Well-solved special cases. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds.), *The traveling salesman problem*, Wiley, Chichester, 87-143.

Gutin, G., Punnen, A.P. (eds), 2002. *The traveling salesman problem and its variations.* Kluwer Academic Publishers, Netherlands.

Gutin G., Yeo, A., Zverovitch, A., 2002. Exponential Neighborhoods and Domination Analysis for The TSP. G. Gutin, A.P. Punnen, eds. *The Traveling Salesman Problem and Its Variations,* Kluwer Academic Publishers, Dordrecht, 223-256.

Kabadi. S.N., 2002. Polynomially Solvable Cases of the TSP. Gutin, G., Punnen, A.P. (eds.), *The traveling salesman problem and its variations.* Kluwer Academic Publishers, Netherlands, 489-583.

Kabadi. S.N., Punnen, A.P., 2002. The bottleneck TSP. Gutin, G., Punnen, A.P. (eds.), *The traveling salesman problem and its variations.* Kluwer Academic Publishers, Netherlands, 697-736.

Kalai, G. 1993. Some aspects of the combinatorial theory of convex polytopes. Internet. Available from http://www.ma.huji.ac.il/~kalai/papers.html; accessed 25.01.2008

Kellerer,H., Pferschy,U., Pisinger, D. 2004. *Knapsack Problems.* Springer, Berlin

Korte, B., Vygen J., 2002. *Combinatorial optimization theory and algorithms.* Springer, Berlin.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds,), 1985. *The traveling salesman problem.* Wiley, Chichester.

Martello, S., Toth, P., 1990. *Knapsack problems: algorithms and computer implementations*. Wiley, Chichester.

Nemhauser, G., Wolsey, L.A., 1988. *Integer and combinatorial optimization.* John Wiley & Sons, USA.

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

Phillips, J.M., Punnen, A.P., Kabadi, S.N., 1998. A linear time algorithm for the bottleneck travelling salesman problem on a Halin graph, *Information Processing Letters*, 67, 105-110.

Przybylski, A., Gandibleux, X., Ehrgott, M., 2007. Two recursive algorithms for finding all nondominated extreme points in the outcome set of a multi-objective integer programme. Technical Report, Universite de Nantes.

Punnen. A.P., 2002a. *The traveling salesman problem: applications, formulations and variations.* Gutin, G., Punnen, A.P. (eds.), *The traveling salesman problem and its variations.* Kluwer Academic Publishers, Netherlands, 1-28.

Punnen. A.P., 2002b. *Computational complexity*. Gutin, G., Punnen, A.P. (eds.), *The traveling salesman problem and its variations.* Kluwer Academic Publishers, Netherlands, 754-760.

Ruzika S., Wiecek, M.M., 2005. Approximation Methods in Multiobjective Programming. *Journal of optimization theory and applications*, 126(3), 473-501.

Solanki, R. S., Appino, P. A., Cohon, J. L., 1993. Approximating the Noninferior Set in Multiobjective Linear Programming Problems, *European Journal of Operational Research*, 68, 356–373.

Steuer, R.E., 1986. *Multiple criteria optimization: theory, computation and application.* John Wiley & Sons, USA.

Tenfelde-Podehl, D., 2003. A recursive algorithm for multiobjective combinatorial problems with Q criteria. Technical Report, University of Graz.

Vairaktarakis, G.L., 2003. On Gilmore-Gomory's open question for the bottleneck TSP, *Operations Research Letters*, **33**(6), 483-491.

Van der Veen, J.A.A., 1993. An O(n) algorithm to solve the bottleneck traveling salesman problem restricted to ordered product matrices, *Discrete Applied Mathematics*, 47, 57-75.

Van der Veen, J.A.A., 1994. A new class of pyramidally solvable symmetric traveling salesman problems, *SIAM Journal Discrete Mathematics*, 7, 585-592.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Özpeynirci, Özgür
Nationality: Turkish (TC)
Date and Place of Birth: 01 February 1979, Konya
Marital Status: Married

EDUCATION

Degree	Institution	Year of Graduation
MS	Industrial Engineering	2004
Minor Degree	METU Sociology	2002
BS	METU Industrial Engineering	2002
High School	Mersin Science High School	1997

WORK EXPERIENCE

Year	Place	Enrollment
2002-2007	TÜBİTAK	Expert Researcher
2001-2002	TÜBİTAK	Part-time Employee
1999-2002	METU Industrial Engineering	Student Assistant

FOREIGN LANGUAGE

Advanced English

PUBLICATIONS

Özpeynirci Özgür, Köksalan Murat, "Performance Evaluation using Data Envelopment Analysis in the Presence of Time Lags", *Journal of Productivity Analysis*, 27, 221-229, 2007.

Özpeynirci Özgür, Süral Haldun, "Comment on Berman and Huang (2004): Minisum collection depots location problem reduces to the p-median problem", *Journal of the Operational Research Society*, 58(10), 1395-1396, 2007.

Solyalı Oğuz, **Özpeynirci Özgür**, "Operational fixed job scheduling problem under spread time constraints: a branch-and-price algorithm", *International Journal of Production Research,* to appear, 1-17, 2007.