EXACT AND REPRESENTATION METHODS FOR MULTIOBJECTIVE OPTIMIZATION PROBLEMS

by

Gökhan Kirlik

A Thesis Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

in

Industrial Engineering and Operations Management

Koç University

August, 2014

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

Gökhan Kirlik

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

Prof. Dr. Serpil Sayın
College of Administrative Sciences and Economics, Koç University

---

Assoc. Prof. Dr. Yalçın Akçay
College of Administrative Sciences and Economics, Koç University

---

Prof. Dr. İ. Kuban Altınel
Department of Industrial Engineering, Boğaziçi University

PD Dr. Marco Laumanns
Department of Management, Technology, and Economics, ETH Zurich

---

Assoc. Prof. Dr. Emre Aper Yıldırım
Department of Industrial Engineering, Koç University

Date: _____

To my mother

ABSTRACT

Many real-world decision-making situations involve simultaneous consideration of conflicting objectives. When a mathematical programming framework is utilized to model such problems, the result is a multiobjective optimization problem, which no longer possesses a unique optimal objective function value. In multiobjective optimization, the set of efficient solutions is used instead of the optimal solution. An efficient solution has the property that no improvement on any objective is possible without sacrificing at least another objective. The solution to a multiobjective optimization problem consists of the efficient set which portrays all relevant trade-off information to a decision maker. Contributions to the theory of multiobjective optimization date back to the 1970s. During the last two decades there has been significant progress in terms of practically implementable algorithms to solve several versions of the multiple objective optimization problem. Among these is the bicriteria case that corresponds to optimization of two objective functions and a number of well-studied discrete optimization problems with two and sometimes three objectives. However, enumerating the efficient set and enabling the decision maker to find a most-preferred solution within the efficient set remains a challenge for the general case. In this work, we revisit the theory of multiple objective optimization with the goal of building algorithms that are capable of solving problems with more than two objectives. Our main concern is to develop methods that enumerate the efficient set for multiobjective optimization problems. We develop an algorithm that enumerates the entire set of solutions for multiple objective discrete optimization problems. However in general the efficient set is not easy to deal with, so it might be better to generate

a fine subset of the efficient set. Such sets are called representations. We consider finding representations of the efficient set. Although representations of the efficient set can be found in many different ways, imposing quality guarantees has been a challenge. Our efforts in this direction lead to a bilevel programming-based subproblem. By using the bilevel formulation, we propose an algorithm to generate representations that satisfy the specified error factor. We test the algorithm on multiobjective linear programming problems.

ÖZETÇE

Birçok gerçek hayat karar verme problemi birbiriyle çelişen birden fazla amacın dikkate alınmasını gerektirir. Karar verme sürecinde matematiksel programlama kullanıldığında, bu problem çok amaçlı eniyileme problemine dönüşür. Çok amaçlı eniyileme probleminde birden fazla amaç fonksiyonu dikkate alındığından eniyi çözüm yerine etkin çözüm kullanılmaktadır. Bir etkin çözümün bir amaç fonksiyonunun iyileştirilebilmesi için diğer amaç fonksiyonlarından birinin kötüleştirilmesi gerekmektedir. Çok amaçlı eniyileme problemine ait bütün etkin çözümler etkin çözüm kümesi olarak tanımlanır ve karar vericiye bütün uygun ödünleşme bilgisini sunar. Çok amaçlı eniyileme teorisine yapılan katkılar 1970'lere kadar uzanmaktadır. Diğer taraftan son 20 yılda ise farklı tipteki çok amaçlı eniyileme problemleri için pratik olarak uygulanabilir algoritmalar geliştirilmiştir. Bu çalışmalarda iki amaç fonksiyonlu problemler ve iki veya üç amaç fonskiyonuna sahip olan kesikli eniyileme problemleri dikkate alınmıştır. Bu tezde ikiden fazla amaç fonksiyonlu problemleri çözebilmek için gerekli olan teorik altyapı oluşturulmuştur. Daha sonrasında bu sonuçlar kullanılarak etkin çözüm kümesini türeten yeni bir yöntem önerilmiştir. Fakat etkin çözüm kümesi genellikle büyük bir kümedir ve bütün olarak değerlendirilmesi zordur. Bu nedenle etkin çözüm kümesi yerine bu kümenin bir alt kümesinin türetilmesi tercih edilebilmektedir. Bu özelliğe sahip olan kümelere temsili etkin çözüm kümesi denilmektedir. Temsili etkin çözüm kümesi birçok farklı yöntemle bulunabilir. Fakat bu kümenin belirlenen bir kalite garantisini sağlaması ile tanımlanan problem oldukça zordur. Bu kapsamda iki-seviyeli eniyileme problemleri kullanılmıştır. Bu tezde iki-seviyeli eniyileme probleminden faydalanılarak çok amaçlı eniyileme problemleri için temsili etkin çözüm

kümesini belirli bir kalite garantisi ile türetebilecek bir algoritma önerilmiştir. Bu algoritma çok amaçlı doğrusal programlama problemleri üzerinde test edilmiştir.

ACKNOWLEDGMENTS

First, I would like to thank my supervisor, Dr. Serpil Sayın for her great supervision and invaluable support during my Ph.D. study. It was a privilege to work with Dr. Sayın.

I thank Dr. Alper Erdoğan and Dr. Emre Alper Yıldırım for insightful comments and suggestions in all progress presentations. I thank Dr. Yalçın Akçay and Prof. Kuban Altınel for being in my thesis committee and supportive comments on my thesis. I also thank Dr. Marco Laumanns for his sincere interest in my research and for being a mentor in the IBM PhD Fellowship Awards program and a thesis committee member in my Ph.D. defense.

I thank Önay (Batur), Caner (Canyakmaz), Burak (Kaleci), Uğur (Kaplan), Zehra (Önen), Yiğit Can (Ören), Can (Öz) and Tolgahan (Yılmaz) for their support and encouragement. They are not only colleagues but also lifelong friends.

I thank my mother for her unconditional support and encouragement. I would like to dedicate this dissertation to my mother.

I was supported by the IBM PhD Fellowship Award Program for the academic year 2013-2014. I thank IBM Corporation for this support.

This thesis is supported by TUBITAK (Scientific & Technical Research Council of Turkey), project no. 112M217.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

$\mathbb{R}^n$	Set of n-dimensional real vectors
$\mathbb{Z}^n$	Set of n-dimensional integer numbers
f	Objective function
p	Number of objective functions
n	Number of decision variables
$\mathcal{X}$	Feasible set ($\mathcal{X} \subseteq \mathbb{R}^n$)
$\mathcal{Y}$	Outcome space ($\mathcal{Y} \subseteq \mathbb{R}^p$)
x	Decision vector
y	Mapping of a solution in the outcome space
$\mathcal{X}_E$	Efficient set ($\mathcal{X}_E \subseteq \mathcal{X}$)
$\mathcal{Y}_N$	Nondominated set ($\mathcal{Y}_N \subseteq \mathcal{Y}$)
y^I	Ideal point, $y^I \in \mathbb{R}^p$
y^N	Nadir point, $y^N \in \mathbb{R}^p$
MOLP	Multiobjective linear programming
MODO	Multiobjective discrete optimization problem
MOKP	Multiobjective knapsack problem
MOAP	Multiobjective assignment problem
MOILP	Multiobjective integer linear programming
BLP	Bilevel linear programming
ILP	Integer linear programming

Nomenclature *(continued)*

SVM	Support vector machine

Chapter 1

INTRODUCTION

Many real life decision making problems take into account a single objective. However, most of decision making problems have multiple objectives by nature. Hence it is necessary to consider several conflicting objectives in the decision making process. When a mathematical programming framework is utilized to model such problems, the result is multiobjective optimization which is a generalization of traditional single-objective optimization.

Optimization problems with multiple objectives can be found in a wide variety of applications. In financial engineering, portfolio optimization is a well-studied problem, and the problem has two objectives by definition. Portfolio optimization aims to determine the weights of various assets to be held in a portfolio that maximize the expected value of portfolio returns, while minimizing the risk that is measured by the standard deviation of portfolio returns in the classical model [133]. Hence, solution techniques that are used to solve multiobjective optimization methods have been applied to the portfolio optimization problem, as in [59], [9] and [36]. Additionally, new objective functions may be considered. For instance, in [7], the portfolio optimization problem has three objectives: risk, return and the number of securities in the portfolio. In healthcare, intensity-modulated radiation treatment planning selects the beam angles and computes the intensity of the beams to maximize the dosage to tumor while minimizing the dosage to organs-at-risk [153]. Bertsimas et al. deal with the radiation treatment planning problem by combining several objectives into a single objective

[28]. In data mining, support vector machines (SVM) construct a hyperplane with the maximum-margin that separates the data into two classes while minimizing empirical errors [46]. In other words, SVM classification problem has two objectives that are maximization of the margin and minimization of the empirical errors. [11] and [10] consider SVM classification as an optimization problem with multiple objectives. In supply chain management, sustainability concerns have led to considerations of cost effectiveness and greenhouse gas emissions [193]. In other words, environmentally conscious supply chain management problems consider maximization of the profits and minimization of the greenhouse gas emissions simultaneously. Scheduling is one of the most studied problems in combinatorial optimization and has several applications that consider more than one objective [12]. For example, Dhaenens et al. solve a flow-shop scheduling problem with three objectives that are completion time of the last job (makespan), total tardiness and maximum tardiness [57]. These are only a few applications of multiobjective optimization. Many others can be found, e.g. in routing [190], service systems [126], airline operations [128], and engineering design [134]. These applications denote the widespread applicability and usefulness of multiobjective optimization.

In a single-objective optimization model, there is a single optimal objective function and therefore comparing feasible solutions based on this value is straight forward. On the other hand, in multiobjective optimization, the set of efficient solutions is used instead of the optimal solution. An efficient solution has the property that no improvement in any objective is possible without sacrificing at least one other objective. The set of all efficient solutions is called the efficient set. This set portrays all relevant trade-off information to a decision maker. The decision maker should evaluate these solutions and select the most preferred one according to his/her preferences.

In this thesis, we develop exact and representative solution methodologies for multiobjective optimization problem (MOP). In exact methods, the aim is to obtain

either all efficient solutions in the decision space or all nondominated solutions[1] in the outcome space. Several exact algorithms have been proposed to solve multiobjective optimization problems, but most of them are limited to two objective functions. A multiobjective optimization problem with two objective functions is called a bicriteria optimization problem (BOP). BOP is a well studied problem because of the simplicity of the parametric search. When the number of objective functions is increased from two to three, the problem gets more complicated. As Ehrgott and Gandibleux state regarding multiobjective optimization, "three is more than two plus one" [68].

Generating the efficient set gives all relevant information to the decision maker. The efficient set of a MOP can rarely be defined by a closed-form formula and generating all efficient solutions can be time consuming. Thus, instead of obtaining the entire efficient set, generating a finite discrete subset of it, which is called a representative set, may be a better way to deal with the problem. In this context, a representation method tries to determine a subset of the efficient set that satisfies some quality measures. These measures include coverage level of the efficient set, diversity of the solutions and the number of solutions [160]. Our main goal is to develop exact and representation methods for multiobjective optimization problems with any number of objective functions.

Along with the general MOP, we also consider the problem with discrete variables which is called multiobjective discrete optimization (MODO) problem. In MODO, the main issue is to develop effective procedures to generate efficient solutions. Generally, scalarization methods are utilized to obtain efficient solutions for MODO problems, which formulate a single-objective optimization problem such that optimal solutions to the single-objective optimization problem are efficient solutions to the MOP [66]. In this thesis, we develop a new algorithm to generate all nondominated solutions for a MODO problem with any number of objective functions [114]. In this algorithm, the

[1]The set of nondominated solutions is the image of the efficient solutions set in the outcome space.

search is managed over $(p-1)$-dimensional rectangles where p represents the number of objectives in the problem. For each rectangle, two-stage optimization problems are solved where the first stage problem is an ε-constraint scalarization. The ε-constraint method retains one of the p objective functions as the objective function, while the remaining $p-1$ are turned into constraints [93]. The method searches $(p-1)$-dimensional space exhaustively, and guarantees to find all nondominated solutions in a finite number of iterations. We compare our method with former studies on multiobjective knapsack and assignment problem instances with up to five objectives.

Another interesting research topic for MODO problems is finding the nadir point. The nadir point is constructed from the worst objective values over the efficient set of a multiobjective optimization problem. Obtaining the nadir point is generally a hard problem [71]. The nadir point is an important element of multiobjective optimization, because all components of this point define the upper bound of the efficient set. In fact, there are some methods that require the nadir point as an input, and the nadir point is also useful for finding the representative sets. Hence, determination of the nadir point has been studied extensively and several exact and heuristic methods have been proposed for the problem [65]. In this thesis, we characterize the determination of nadir point with two-stage subproblems [113]. Based on the characterization result, we present an algorithm that searches the $(p-2)$-dimensional parametric space exhaustively. Also, the algorithm utilizes the nadir point underestimator information to eliminate some portion of the search space beforehand. We show that the algorithm guarantees to find the nadir point for MODO problems with any number of objective functions in a finite number of iterations. We test the method on multiobjective knapsack, assignment and integer linear programming problems.

The efficient set of a MOP is generally a large set to deal with. For example, although MODO problems with bounded efficient sets have finite number of efficient solutions, the size of the efficient set might still be too large [114]. The decision maker intends to choose an efficient solution from the efficient set. As the size of the efficient

set increases, the decision maker has to work on a larger set. Additionally, obtaining an efficient solution has a cost and it is not negligible. Due to these reasons, instead of generating the entire efficient set, obtaining a finite subset, a representative set, is an interesting problem to study.

In general, a representative set should contain solutions from every portion of the nondominated set without missing any region. This assessment criteria is defined as coverage error which is one of the well-known quality measures for representations [160]. In this thesis, we generate a representative set that satisfies the specified coverage error requirement, i.e. the minimum distance between the worst represented point in the nondominated set and the representative set should be less than a specified coverage level. We utilize p-dimensional rectangles to search the outcome space. During the search, some of the rectangles can be eliminated when they satisfy the specified coverage error. We show that the algorithm terminates in a finite number of iterations, and generates a representative set that satisfies specified coverage error level for MODO problems. We test the algorithm on multiobjective knapsack and assignment problems.

The proposed algorithm generates representative sets with specified coverage errors, however the algorithm is limited to solve MODO problems. Hence, we generalize the representation algorithm for the continuous multiobjective optimization problem with any number of objective functions. In this algorithm, we utilize bilevel optimization to find a nondominated solution in a given rectangle. We test the method on multiobjective linear programming (MOLP) problems. When the aim is to obtain a nondominated solution in a rectangle for a MOLP problem, general bilevel programming problem turns into a bilevel linear programming (BLP) problem. We use the penalty approach [132] and an integer programming reformulation [84] to solve the BLP problem. Both methods utilize KKT optimality conditions to turn the BLP problem into a single level optimization problem [55] which is referred to as a mathematical program with complementarity constraints [129]. In the penalty approach,

complemantarity constraints of KKT optimality conditions are moved into the objective function with a penalty parameter. In integer programming reformulation, complemantarity constraints are turned into linear constraints by using binary variables and a sufficiently large coefficient [14]. Finally we use this algorithm to obtain better accuracy in support vector machine (SVM) classification on imbalanced data set problem. Askan and Sayin model the SVM classification on imbalanced data set problem as a MOLP with three objectives [10]. We generate representative sets for this problem, and determine the best performing hyperplane that separates the data.

The organization of this thesis is as follows. In Chapter 2, we present the literature review for the exact and representation solution methods for MODO problems with necessary definitions and notation. In Chapter 3, we present a new algorithm to generate all nondominated solutions for MODO problems. In Chapter 4, we give an algorithm to determine the nadir point for MODO problems. In Chapter 5, we present a representation method that generates representative sets with specified coverage error for multiobjective discrete optimization problems. In Chapter 6, we model the problem of finding a nondominated solution in a rectangle as a bilevel optimization problem. By using this result, we present a representation algorithm for multiobjective optimization problems. Our conclusions and directions for further research are presented in Chapter 7.

Chapter 2

LITERATURE REVIEW

In this chapter, we present notation and definitions, and literature review on exact and representation methods for MOP problems. We start by giving a general formulation of multiobjective problems. We study multiobjective optimization problems with discrete variables in detail. In Section 2.2, we give the formulation of MODO problems, and well-studied MODO problems. In Section 2.3, the exact solution methods for multiobjective optimization problems are covered under four different subsections. These are are linear scalarization method, ε-constraint method, min-max approaches and other exact solution methods. In Section 2.4, we give the definitions of the ideal and nadir points that define the boundaries of the efficient set. We also present the literature on exact solution methods that compute the nadir point. Finally, we discuss representation methods with and without quality guarantees in Section 2.5.

2.1. Multiobjective Optimization

In this section, the basic definitions and the notation related to MOP and the necessary foundations for the methods described in the forthcoming chapters are established.

In multiobjective optimization p objective functions $f_j(x) : \mathbb{R}^n \to \mathbb{R}$ for $j = 1, \ldots, p$ have to be optimized. In this thesis, without loss of generality, we assume that each objective is minimized. The feasible set, which is defined by a set of constraints, is denoted as $\mathcal{X}$. Any solution $x \in \mathcal{X}$ is represented with $x \in \mathbb{R}^n$. Each feasible solution $x \in \mathcal{X}$ is mapped into its corresponding objective vector $y = f(x)$ and

$\mathcal{Y} = \{y \in \mathbb{R}^p : y = f(x) \text{ for some } x \in \mathcal{X}\}$ is referred to as the set of feasible outcomes in the objective space. In mathematical terms, MOP is defined as:

$$\begin{aligned} (\textbf{MOP}) \quad \min \quad & f(x) = [f_1(x), \ldots, f_p(x)] \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned} \tag{2.1}$$

In the above formulation, when the decision variables are continuous, and the objective functions and constraints are linear, the MOP turns into a MOLP problem [171]. If the objectives or the constraints incorporate any nonlinear term, then the problem is called a multiobjective nonlinear programming problem [140]. In some problems, the decision variables can be discrete. Any MOP with discrete variables is called a MODO problem which is discussed in more detail in the following section. Multiobjective optimization problems can also be classified with respect to the number of objective functions. MOP with two objectives is the special case of MOP, and it is called as BOP.

For single criterion optimization, the concept of optimality is well-defined. However, due to conflicting objectives, MOP is expected to have more than one solution. These solutions are called efficient solutions. We give the definitions of weakly efficient and efficient solution below. Note that our main interest is to generate efficient solutions.

Definition 1. *A solution $x^* \in \mathcal{X}$ is called* **weakly efficient** *if there exists no feasible solution $x \in \mathcal{X}$ such that $f_j(x) < f_j(x^*)$ for all $j \in \{1, \ldots, p\}$.*

Definition 2. *A solution $x^* \in \mathcal{X}$ is called an* **efficient solution** *if there exists no feasible solution $x \in \mathcal{X}$ such that $f_j(x) \leq f_j(x^*)$ for all $j \in \{1, \ldots, p\}$ and there exists $\hat{\jmath} \in \{1, \ldots, p\}$ such that $f_{\hat{\jmath}}(x) < f_{\hat{\jmath}}(x^*)$. For an efficient solution x^*, $f(x^*) \in \mathbb{R}^p$ is referred to as a* **nondominated solution** *in the outcome space.*

The set of all efficient solutions for MOP is called the efficient set and is denoted as $\mathcal{X}_E$. The image of the efficient set in the objective space is called the nondominated

set and is denoted as $\mathcal{Y}_N$, i.e. $\mathcal{Y}_N = \{y \in \mathbb{R}^p : y = f(x) \text{ for some } x \in \mathcal{X}_E\}$.

Note that any efficient solution may be an optimal solution for MOP. In contrast, there may exist feasible solutions that are never optimal to MOP. For such a solution, it is possible to improve one of the objective functions without sacrificing another objective. These are called dominated solutions. A formal definition is given below.

Definition 3. *A solution $f(x^*) \in \mathcal{Y}$ is called a* **dominated solution** *if there exists a feasible solution $x \in \mathcal{X}$ such that $f_j(x) \leq f_j(x^*)$ for all $j \in \{1, \ldots, p\}$ and there exists $\hat{\jmath} \in \{1, \ldots, p\}$ such that $f_{\hat{\jmath}}(x) < f_{\hat{\jmath}}(x^*)$, $f(x) \in \mathcal{Y}$ dominates $f(x^*) \in \mathcal{Y}$.*

In Figure 2.1, the outcome space of a bicriteria discrete optimization problem is given. Gray shaded region represents the convex hull of the outcome space, and the convex hull of $\mathcal{Y}$ is denoted as $conv(\mathcal{Y})$. Circles with bold borderline are the nondominated solutions. Hence, the nondominated set of this problem is $\mathcal{Y}_N = \{y_1, y_2, y_3, y_4, y_5, y_6\}$. Additionally, the solutions with labels 7, 8, 13 and 14 are images of some weakly efficient solutions since there exists no solution which is better in both objectives. Weakly nondominated solutions may be dominated by nondominated solutions and are indeed not desirable. Nevertheless, it is important to make the distinction from a theoretical point of view because some methods may deliver weakly nondominated solutions unless some additional measure is taken.

In nonconvex multiobjective optimization problems, such as MODO problems, an important classification exists for the efficient solutions: supported efficient solutions and nonsupported efficient solutions. The nonsupported nondominated vectors are located inside the $conv(\mathcal{Y})$ in the objective space, while the supported vectors are found on the boundaries of the $conv(\mathcal{Y})$. Definitions are given below:

Definition 4. *Let $x \in \mathcal{X}_E$. If there exist $\lambda > 0$, $\lambda \in \mathbb{R}^p$, such that $x \in X_E$ is an optimal solution of (2.2) (see Section 2.3.1 for the details of the formulation), then x is called a supported efficient solution and $y = f(x)$ is called* **supported nondominated solution**.

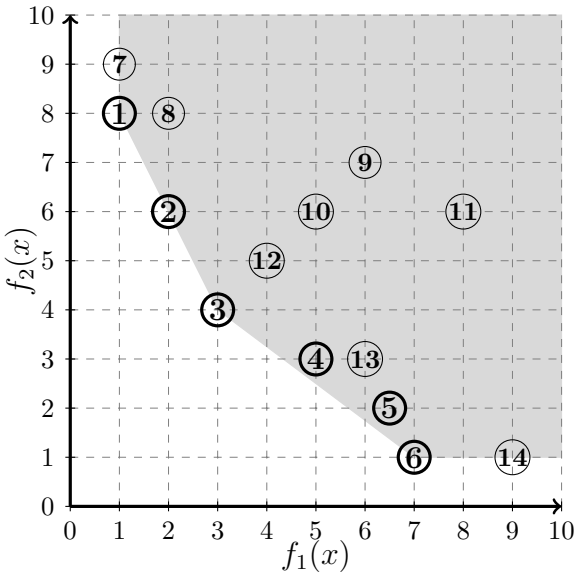


Figure 2.1: Outcome space of a bicriteria discrete optimization problem.

$$\begin{aligned} P(\lambda) \quad & \min \quad \lambda^T f(x) \\ & \text{s.t.} \quad x \in \mathcal{X} \end{aligned} \tag{2.2}$$

The set of of supported efficient solutions is also divided into two subsets that are extreme supported efficient solutions and nonextreme efficient solutions. Mapping of these in the outcome space are called extreme supported nondominated solutions and nonextreme supported nondominated solutions, respectively. If $y = f(x)$ is an extreme point of $conv(\mathcal{Y})$, then it is called an extreme supported nondominated solution. The remaining supported nondominated solutions are called nonextreme supported nondominated solutions.

In nonconvex multiobjective optimization problems, there may exist some solutions which are not optimal to $P(\lambda)$ for some $\lambda > 0$. These solutions are called nonsupported efficient solutions. Note that union of supported and nonsupported efficient solutions forms the efficient set. In Figure 2.1, the image of supported efficient solutions are y_1, y_2, y_3 and y_6, and all these solutions are on the boundary of

$conv(\mathcal{Y})$. The extreme points of $conv(\mathcal{Y})$ are y_1, y_2 and y_6, i.e. extreme supported nondominated solutions of the problem. The nondominated solutions with label 4 and 5 are in the interior of $conv(\mathcal{Y})$. Thus, y_4 and y_5 are images of nonsupported efficient solutions.

Another important formulation that is related to multiobjective optimization is the lexicographic optimization problem. For a given priority order, the lexicographic optimization program requires solving p single-objective optimization problems [19]. At first, the objective function in the first order is minimized over the feasible set of the problem. This objective function is bounded by the optimal objective value of the first single-objective optimization problem. Then, the objective function in the second order is minimized over the updated feasible region. The additional objective bound constraint is added iteratively to obtain the lexicographically optimal solution. More importantly, any lexicographically optimal solution of a MOP is efficient without any convexity assumption [65]. While the essential feature of efficiency is the existence of trade-off between objectives, lexicographic optimization considers lexicographic order to compare objective vectors in the objective space.

In the rest of thesis, some of the formulations require a metric $d(y^a, y^b)$ definition on $\mathbb{R}^p$ which is used in measuring the distance between outcomes in the objective space. The following group of functions are metrics on $\mathbb{R}^p$.

Definition 5. *For* $y^a, y^b \in \mathbb{R}^p$

$$L_q(y^a, y^b) = \begin{cases} \left(\sum_{j=1}^{p} |y_j^a - y_j^b|^q\right)^{1/q} & \text{for } 1 \leq q < \infty \\ \max_{j=1,\ldots,p} |y_j^a - y_j^b| & \text{for } q = \infty \end{cases}$$

is referred to as the L_q*-norm.*

2.2. Multiobjective Discrete Optimization

Discrete optimization is an extensively studied field of mathematical optimization with respect to the applications and the solution methods, due to its wide applicability in real world problems. Discrete optimization problem with more than one objective function is called MODO problem. As most discrete optimization problems are computationally difficult to solve with a single objective function [88], solving MODO problems is also difficult [67]. MODO is a special case of MOP where all variables are discrete. Therefore, the feasible set of MODO is also discrete. In mathematical terms, MODO is defined as:

$$\begin{aligned} \textbf{(MODO)} \quad \min \quad & f(x) = [f_1(x), \ldots, f_p(x)] \\ \text{s.t.} \quad & x \in \mathcal{X} \\ & x \in \mathbb{Z}^n \end{aligned}$$

Since the feasible set of MODO is discrete, if the efficient set of MODO is bounded, then $\mathcal{X}_E$ has a finite number of elements. Therefore, it is possible to enumerate all efficient solutions of a MODO problem. In Chapter 3, we propose an algorithm to generate all nondominated solutions of MODO problems.

Three comprehensive surveys have been presented in [180], [67] and [68] so far related to applications and solution methodologies for MODO problems. In the following subsection, we review MODO problems with special structure. Following that, we discuss multiobjective integer linear programming problem (MOILP). MOILP is a special case of MODO where all objective functions and constraints are linear.

2.2.1 Multiobjective discrete optimization problems with special structures

Some MODO problems that are widely studied are multiobjective knapsack problem [186], multiobjective assignment problem [150], multiobjective network flow problem [94], multiobjective travelling salesman problem [104], multiobjective transporta-

tion problem [8], multiobjective facility location problem [41], and multiobjective shortest path problem [87]. MODO problems are generally difficult to solve. As an example, shortest path problem is in the class of easy problems. However, multiobjective shortest path problem is intractable, i.e. the number of efficient solutions may be exponential in the number of nodes [96]. Since these problems are difficult, some approximation results have also been published [147, 64, 69].

In this thesis, we use multiobjective knapsack and multiobjective assignment problem from the class of MODO problems with special structures as test instances. Hence, these two problems are explained in detail.

The knapsack problem is one of the well studied discrete optimization problems in the literature [109]. Moreover, some real-world applications such as capital budgeting [156] have been modeled as multiobjective knapsack problems. The multiobjective knapsack problem consists of a postive integer capacity W with n objects. Each object r has a positive integer weight w_r and p nonnegative integer profits v_r. Decision variable x_r denotes whether item r is selected for the knapsack or not.

$$(\textbf{MOKP}) \quad \max \quad \sum_{r=1}^{n} v_r^j x_r \quad j = \{1, \ldots, p\} \tag{2.3}$$

$$\text{s.t.} \quad \sum_{r=1}^{n} w_r x_r \leq W \tag{2.4}$$

$$x_r \in \{0, 1\} \quad r = 1, \ldots, n \tag{2.5}$$

Equation (2.3) is the set of p objective functions and each objective function denotes the total profit of chosen items. Equation (2.4) is the capacity constraint. The total weight of selected objects has to be less than or equal to the knapsack's capacity. Equation (2.5) represents the binary integrality constraints.

Several exact solution methods have been proposed to solve multiobjective knapsack problem. Klamroth and Wiecek present several dynamic programming formu-

lations [116]. Visée et al. [186] use a two phase algorithm is used to generate the efficient set. In the two-phase algortihm, they obtain the nonsupported efficient solutions by using a branch-and-bound procedure. Captivo et al. [38] present a labeling algorithm that is based on a transformation of the problem into a bicriteria shortest path problem. Bazgan et al. [17] propose an approach based on dynamic programming. Figueira et al. [83] present another labeling algorithm to find all nondominated solutions.

The assignment problem aims to obtain optimal assignments between a set of agents $r \in \{1, \ldots, n\}$ and a set of tasks $l \in \{1 \ldots, n\}$ where each assignment has a nonnegative cost c_{rl}. The multiobjective assignment problem is formulated as follows,

$$(\textbf{MOAP}) \quad \min \quad \sum_{r=1}^{n} \sum_{l=1}^{n} c_{rl}^{k} x_{rl} \quad j = \{1, \ldots, p\} \tag{2.6}$$

$$\text{s.t.} \quad \sum_{l=1}^{n} x_{rl} = 1 \quad r = 1, \ldots, n \tag{2.7}$$

$$\sum_{r=1}^{n} x_{rl} = 1 \quad l = 1, \ldots, n \tag{2.8}$$

$$x_{rl} \in \{0, 1\} \quad r = 1, \ldots, n;\ l = 1, \ldots, n \tag{2.9}$$

The formulation consists of p objective functions where all objective coefficients are nonnegative integers. The decision variable x_{rl} takes a value of one if agent r is assigned to task l. Equation (2.6) represents the set of objective functions where each minimizes total assignment cost with different cost coefficients. Equation (2.7) ensures that each agent is assigned to exactly one task, and similarly equation (2.8) guarantees that each task is assigned to exactly one agent. Equation (2.9) is the binary integrality constraints.

Early studies for multiobjective optimization assignment problem only deal with supported efficient solutions by combining the several objectives with positive coef-

ficients [68]. Malhotra et al. [130] use the two dual problems to generate the set of efficient solutions for bicriteria assignment problem. Przybylski et al. present a two-phase algorithm to generate the efficient set of a bicriteria assignment problem in [150]. In the first-phase, the algorithm generates all supported efficient solutions based on dichotomic scheme proposed by Aneja and Nair [8]. In the second-phase, all nonsupported efficient solutions are enumerated in an improved version of [180]. Same authors generalize this method to deal with the multiobjective assignment problem with any number of objectives [152]. This algorithm is also a two-phase method, in the first phase extreme supported efficient solutions are generated by using the solution methodology presented in [151]. In the following phase, the algorithm enumerates remaining efficient solutions.

2.2.2 The multiobjective integer linear programming problems

We also test our proposed algorithms on general multiobjective integer linear programming problems. MOILP is a special case of MODO where all objective functions and constraints are linear. Here, m and n represent the number of constraints and number of variables, respectively, and x is the decision vector of the problem. Given coefficients of the objective functions c_l^j, the technical coefficients a_{rl}, and right-hand side values b_r where $r \in \{1,\ldots,m\}$, $l \in \{1,\ldots,n\}$, and $j \in \{1,\ldots,p\}$, MOILP problem is defined as follows.

$$(\textbf{MOILP}) \quad \max \quad \sum_{l=1}^{n} c_l^j x_l \quad j = 1,\ldots,p \tag{2.10}$$

$$\text{s.t.} \quad \sum_{l=1}^{n} a_{rl} x_l \leq b_r \quad r = 1,\ldots,m \tag{2.11}$$

$$x_l \geq 0 \text{ and integer} \quad l = 1,\ldots,n. \tag{2.12}$$

Equation (2.10) is the set of p objective functions. Equation (2.11) represents

the set of constraints of MOILP problem. Equation (2.5) represents the integrality constraints. Evans [78] and Teghem and Kunsch [178] present early surveys about the exact solution methods for MOILP.

2.3. Exact Methods for Multiobjective Optimization Problems

In this section, we discuss the exact solution methods for multiobjective optimization problems. The exact methods are reviewed in four subsections. The subsections are linear scalarization, ε-constraint method, min-max methods and other solution methods.

2.3.1 Linear scalarization

One straightforward way of solving a multiobjective optimization problem is by reducing it to a single-objective optimization problem using a weighted sum formulation (or linear scalarization) that combines multiple objectives [195]. The formulation of the linear scalarization method is given in (2.2). For a given $w > 0$, the optimal solution of the weighted sum formulation $P(\lambda)$ is an efficient solution [195]. Since the aim of the exact methods is to generate the efficient set, it is also necessary to show that efficient solutions are optimal to $P(\lambda)$ for some $\lambda > 0$. For this purpose consider the following theorem.

Theorem 1. *Let $\mathcal{X} \subset \mathbb{R}^n$ be convex and assume $f_j : \mathcal{X} \to \mathbb{R}$ are convex for all $k = 1, \ldots, p$. Then, $x \in \mathcal{X}$ is efficient if and only if x is an optimal solution of $P(\lambda)$ with $\lambda > 0$ [89].*

Although the theorem is restricted to convexity, it indicates that $P(\lambda)$ is able to enumerate all efficient solutions for convex multiobjective optimization problems.

A special case of the convex MOP is a multiobjective problem with linear objectives and constraints which is referred to as MOLP. We briefly present results for MOLP in the following paragraph.

Yu and Zeleney present some theoretical results for MOLP in [194]. They show that the set of all nondominated solutions is a subset of the convex hull of extreme nondominated solutions. Finally, the authors propose multiobjective version of the simplex method, and use this method to identify the nondominated set. Isermann gives an algorithm that enumerates all efficient solutions of MOLP in three steps [101]. The algorithm obtains an initial basic (extreme) efficient solution by solving a simple linear programming problem, then all basic efficient solutions are established. Finally, all efficient solutions are constructed by using the results of the previous step. Ecker and Kouada propose an algorithm to deal with extreme efficient solutions for MOLP in [61]. Aneja and Nair propose an algorithm to generate all extreme efficient solutions for bicriteria linear programming problems [8] by using weighted sum scalarization. Their method starts with two extreme nondominated solutions which define the boundaries of the nondominated set. The algorithm searches for a new extreme nondominated solution in between previously obtained points by constructing a new weight. When all intervals are investigated the algorithm is terminated. Aneja and Nair's algorithm can be used to obtain extreme nondominated solutions of any bicriteria linear programming problem. However, it cannot be applied to MOLP problems with any number of objective functions.

Weighted sum scalarization can be used in MODO problems. However as the feasible set is nonconvex, it is not possible to obtain the entire efficient set by using linear scalarization. Only supported efficient solutions can be obtained by using the weighted sum formulation. Nondominated solutions which are located in the interior of $conv(\mathcal{Y})$ cannot be generated. For bicriteria MODO problems, Aneja and Nair's algorithm [8] can be used to generate all extreme supported efficient solutions as in [181].

All these approaches can be used to solve bicriteria problems relatively easily. For MOP with any number of objective functions, Benson and Sun [27] generalize Aneja and Nair's algorithm to obtain extreme efficient solutions of MOLP problems.

They show that weight space of each extreme efficient solution is convex for MOLP problems. In other words, there exists a weight space $\hat{w}$ which is a subset of positive orthant of $\mathbb{R}^p$ and convex such that for any $w \in \hat{w}$ the efficient solution x^* is optimal to formulation $P(w)$. Benson and Sun present an algorithm to obtain all extreme efficient solutions for MOLP by using the convexity property of the search space. Based on these results, Przybylski et al. [151] present a recursive algorithm to generate all extreme supported efficient solutions of MODO problems. In this algorithm, MODO with p objectives problem is reduced $(p-1), (p-2), \ldots, 2$ objective problems, recursively. They test the recursive algorithm on three-objective assignment and knapsack problems. Özpeynirci and Köksalan [146] also use the weight space decomposition result to generate all extreme supported efficient solutions for MODO problems. They test the computational performance of the approach on multiobjective assignment, knapsack, and traveling salesperson problems.

2.3.2 ε-Constraint method

Another well-known technique to solve multiobjective optimization problems is the ε-constraint method, introduced by Haimes in 1971 [93]. In this method, one of the objectives is chosen as the objective function and the others are transformed into constraints. An extensive discussion of this method can be found in [40]. For some $k \in \{1, \ldots, p\}$ and $\varepsilon \in \mathbb{R}^{p-1}$, ε-constraint formulation, $P_k(\varepsilon)$, is as follows,

$$\begin{aligned} \mathbf{P_k}(\varepsilon) \quad \min \quad & f_k(x) \\ \text{s.t.} \quad & f_j(x) \leq \varepsilon_j \quad j = 1, \ldots, p;\ j \neq k \\ & x \in \mathcal{X}. \end{aligned} \tag{2.13}$$

In the above formulation, the right-hand sides of the constraints, $\varepsilon \in \mathbb{R}^{p-1}$, are quantities in the objective space. The following theorem shows that optimal solution of the ε-constraint method is a weakly efficient solution [65].

Theorem 2. *For any $\varepsilon \in \mathbb{R}^{p-1}$, let $x^* \in \mathcal{X}$ be the optimal solution of $P_k(\epsilon)$ for some $k \in \{1, \ldots, p\}$, then x^* is a weakly efficient solution.*

Proof. Let x^* be the optimal solution to the $P_k(\varepsilon)$. Assume that x^* is not a weakly efficient solution, then there exists a solution $x' \in \mathcal{X}$ such that $f(x') < f(x^*)$. Since $f_j(x') < f_j(x^*) \leq \varepsilon_j$ for $j = 1, \ldots, p$ and $j \neq k$, x' is feasible to $P_k(\varepsilon)$. This contradicts the optimality of x^*, because $f_k(x') < f_k(x^*)$. Therefore, the optimal solution of the $P_k(\varepsilon)$ is a weakly efficient solution. □

This is a useful result for the ε-constraint method; however, we are interested in efficient solutions. Different methods have been proposed to avoid weakly efficient solutions while using the ε-constraint approach. First method is lexicographic optimization [19] that requires solving p subsequent subproblems, and its characterization result was presented in [18]. Laumanns et al. [122] use lexicographic optimization to deal with weakly efficient solutions. The hybrid approach is a combination of weighted sum scalarization with the ε-constraint method [92]. Augmentation suggests incorporating the sum of other objective functions of the subproblem by utilizing a sufficiently small weight [171] which is applied in ε-constraint method by Mavrotas [136] and Ozlen and Azizoglu [144].

In Theorem 2, it is shown that for any efficient solution $x \in \mathcal{X}_E$ there exists $\varepsilon \in \mathbb{R}^{p-1}$ such that x^* is optimal to the ε-constraint method [39].

Theorem 3. *Let $x^* \in \mathcal{X}_E$. Then there exist an $\hat{\varepsilon} \in \mathbb{R}^p$ such that x^* is an optimal solution of $P_k(\hat{\varepsilon})$ for some $k \in \{1, \ldots, p\}$.*

Proof. Let $\hat{\varepsilon} = f(x^*)$. Assume x^* is not an optimal solution of $P_k(\hat{\varepsilon})$ for some $k \in \{1, \ldots, p\}$. Then there exist a solution $x' \in \mathcal{X}$ that solves $P_k(\hat{\varepsilon})$. Since x' is optimal to $P_k(\hat{\varepsilon})$, $f_k(x) < f_k(x^*)$ and $f_j(x') \leq \hat{\varepsilon}_j = f_j(x^*)$ for all $j \in \{1, \ldots, p\}$ and $j \neq k$. This implies that x' dominated x^*. Hence, $x^* \notin \mathcal{X}_E$. □

Theorems 2 and 3 show that with appropriate choices of $\varepsilon \in \mathbb{R}^{p-1}$, all efficient solutions can be obtained by using ε-constraint method without any convexity assumption [40]. Due to the wide-range of applicability, the method is utilized in various multiobjective optimization problems including discrete problems.

For the linear case, Benson defines a characterization for the bicriteria linear programming problem in [20]. The author also presents a procedure to obtain the nondominated set in a systematic way by means of sensitivity analysis. The formulation used in the procedure is not referred to as the ε-constraint method; however it is the same as the ε-constraint method with two objective functions.

Since the ε-constraint method finds an entire efficient set without any convexity assumption, it is also applied to numerous MODO problems. Most of these studies consider bicriteria discrete optimization problems because of the simplicity of the parametric search for the bicriteria problems. In the bicriteria case, one of the objectives is taken as the objective function and the other one as a constraint. Then it is sufficient to search single dimensional space exhaustively to obtain all nondominated solutions. Bérubé et al. use the ε-constraint method to obtain the efficient set of bicriteria traveling salesman problem in [29] where the objective functions are maximizing the collected prize and minimizing the total travel cost.

While most of the ε-constraint methods consider bicriteria case, only a few of them apply the method for more than two objective functions. The first method is proposed by Laumanns et al. in which an adaptive scheme for the ε-constraint method is utilized to obtain all nondominated solutions [122]. They search for efficient solutions inside the $(p-1)$-dimensional grid which partitions the whole objective space. Unlike ε-constraint method, they use two-sided bounds, and solve lexicographic optimization problems to deal with weak efficiency. For a given $\varepsilon' \in \mathbb{R}^{p-1}$ and $\varepsilon \in \mathbb{R}^{p-1}$ where $\varepsilon' < \varepsilon$, the lexicographic ε-constraint method used in [122] is given in (2.14). An optimal solution of this formulation may not be efficient. Therefore, some mechanisms are devised in order not to generate dominated solutions. The drawback of this

method is the memory issue. Number of grids may become unmanageable, especially in large-size problems.

$$\begin{aligned} \text{lex}\min \quad & [f_1(x), \ldots, f_p(x)] \\ \text{s.t.} \quad & \varepsilon' \leq f_j(x) < \varepsilon_j \quad j = 2, \ldots, p \\ & x \in \mathcal{X}. \end{aligned} \tag{2.14}$$

In [136], Mavrotas introduce the augmented form of the ε-constraint method. The formulation of the method is as follows,

$$\begin{aligned} \min \quad & f_1(x) + \rho \sum_{j=2}^{p} s_j \\ \text{s.t.} \quad & f_j(x) + s_j = \varepsilon_j \quad j = 2, \ldots, p \\ & x \in \mathcal{X}. \end{aligned} \tag{2.15}$$

Unlike the original ε-constraint formulation, (2.15) generates an efficient solution due to the augmented form of the objective function. Mavrotas utilizes $(p-1)$ dimensional grid to generate several efficient solutions. The boundaries of nondominated set is estimated by using payoff table result. Afterwards, this bounded search space is divided into grids according to decision maker's tolerance expectation. For each grid, method solves (2.15) to obtain an efficient solution. However, the payoff table underestimates the upper levels of the search space, so even if the tolerance is kept very small, this method does not guarantee generating the entire efficient set. Additionally, the scalar ρ may cause numerical problems. By definition, ρ is a sufficiently small positive real number. On the other hand, ρ should be large enough to obtain an efficient solution. Mavrotas and Florios [139] improve the augmented ε-constraint formulation given in (2.15). The improved formulation exploits the information from the slack variables in every iteration. They also give an algorithm to generate all nondominated solutions for MODO problems on condition that the nadir point is known. Another variant of (2.15) is given in [197]. Authors enhance the nondominated set

generation algorithm with two innovations which are early exit and bouncing steps.

Özlen and Azizoğlu also use the augmented ε-constraint method in a recursive algorithm to generate all nondominated solutions of MODO problems [144]. In this method, initially ranges for the nondominated solutions in the outcome space are obtained by minimizing and maximizing each objective function over the feasible set. Then, by applying the range information in the constrained problems the authors generate all nondominated solutions. This method is improved by Ozlen et al. in [145]. They reduce the number of models solved by keeping track of solved subproblems and their solutions.

Lokman and Köksalan give a method to generate all nondominated solutions for MODO problems [127]. The method uses the augmented ε-constraint method to obtain nondominated solutions, and in each iteration the method excludes the regions that are dominated by the previously generated nondominated solutions. They give two algorithms for this method. In the first one, the algorithm generates new solutions by solving models with additional binary variables and constraints. The second algorithm utilizes a search procedure to find the next solution without incorporating additional binary variables.

2.3.3 Min-max approaches

Another generic solution methodology to deal with both convex and nonconvex multiobjective optimization problems is the weighted norm approach. The formulation is as follows,

$$P(w,q) \quad \min \quad \left(\sum_{j=1}^{p} w_j \left\| f_j(x) - y_j^I \right\|_q \right) \qquad \text{s.t.} \quad x \in \mathcal{X} \tag{2.16}$$

where $1 \leq q < \infty$, y^I is the ideal point (see Section 2.4), and $w_j \geq 0$ for all $j \in \{1, \ldots, p\}$. In (2.16), q represents the L_q-norm. When $q = 1$ the weighted norm

formulation reduces to linear scalarization.

The main idea of this method is to find a solution as close as possible to the ideal point. L_q-norm is used to define the distance between the solution and the ideal point. If either $P(w,q)$ formulation with $w \geq 0$ has a unique optimal solution or $w > 0$ for any $1 \leq q < \infty$, then the weighted norm formulation generates an efficient solution. Consider Theorem 4 for the efficiency results of $P(w,q)$ formulation [39].

Theorem 4. *Let x^* solve $P(w,q)$ for any $1 \leq q < \infty$ with $w > 0$ when either*

1. *x^* is a unique solution of $P(w,q)$, or*

2. *$w_j > 0$ for all $j = 1, \ldots, p$*

holds. Then x^* is an efficient solution.

Proof. Let x^* be the optimal solution of $P(w,q)$ for any $1 \leq q < \infty$ and for some $w \geq 0$. Thus,

$$\sum_{j=1}^{p} w_j \left(\left\| f_j(x) - y_j^I \right\|_q - \left\| f_j(x^*) - y_j^I \right\|_q \right) \geq 0 \quad \forall x \in \mathcal{X} \tag{2.17}$$

Assume that x^* is not an efficient solution, then there exist $x' \in \mathcal{X}$ that dominates x^*. This implies that, $f_j(x') \leq f_j(x^*)$ for all $j = 1, \ldots, p$ with at least one strict inequality. Since y^I is the ideal point, by definition $y_j^I \leq f_j(x)\ \forall x \in \mathcal{X}$. Hence, for any $1 \leq q < \infty$, $\left\| f_j(x') - y_j^I \right\|_q \leq \left\| f_j(x^*) - y_j^I \right\|_q$ for all j with strict inequality for at least one $j \in \{1, \ldots, p\}$. Since $w \geq 0$,

$$\sum_{j=1}^{p} w_j \left(\left\| f_j(x') - y_j^I \right\|_q - \left\| f_j(x^*) - y_j^I \right\|_q \right) \leq 0 \tag{2.18}$$

(i) If x^* is the unique optimal to $P(w,q)$ for any $1 \leq q < \infty$ and for some $w \geq 0$, then strict inequality occurs in (2.17) which contradicts (2.18). **(ii)** If we take $w > 0$

in $P(w,q)$, then strict inequality occurs in (2.18) which contradicts (2.17). Hence, if either (i) or (ii) holds, then $x^* \in \mathcal{X}_E$. □

Theorem 4 shows that the solution of the weighted norm minimization is efficient, and the optimal solution of the $P(w,q)$ formulation is also referred to as a compromise solution where $w \geq 0$ and $1 \leq q < \infty$ [196].

Up to now, we have considered the weighted norm formulation for $1 \leq q < \infty$ case. When we set $q = \infty$, the weighted norm becomes weighted Tchebycheff norm (L_∞-norm). This formulation is referred to as the weighted Tchebycheff scalarization and was introduced by Bowman in [34]. The weighted Tchebycheff formulation is as follows,

$$\begin{aligned} P_{Tch_1}(w) \quad \min \quad & \left(\sum_{j=1}^{p} w_j \left\| f_j(x) - y_j^I \right\|_\infty \right) \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned} \tag{2.19}$$

where $w_j > 0$ for all $j \in \{1, \ldots, p\}$. The formulation (2.19) is equivalent to the following formulation,

$$\begin{aligned} P_{Tch_2}(w) \quad \min \quad & \left(\max_{j=1,\ldots,p} w_j \left(f_j(x) - y_j^I \right) \right) \\ \text{s.t.} \quad & x \in \mathcal{X}. \end{aligned} \tag{2.20}$$

Weighted Tchebycheff formulation given in (2.20) is nonlinear. However, it can be linearized by adding one extra variable and p new constraints. The linear formulation of the weighted Tchebycheff method is given as follows.

$$\begin{aligned} P_{Tch_3}(w) \quad \min \quad & z \\ \text{s.t.} \quad & z \geq w_j (f_j(x) - y_j^I) \quad j = 1, \ldots, p \\ & x \in \mathcal{X} \end{aligned} \tag{2.21}$$

In (2.21), a nonnegative variable[2] $z \in \mathbb{R}$ is included with p constraints. All three weighted Tchebycheff formulations are equivalent. The following theorem shows that the optimal solution of the weighted Tchebycheff formulation is a weakly efficient solution [39].

Theorem 5. *For $w > 0$, the optimal solution of the $P_{Tch}(w)$ is weakly efficient.*

Proof. Let x^* be an optimal solution of $P_{Tch}(w)$. Assume that x^* is not a weakly efficient solution. Then, there exist $x' \in \mathcal{X}$ such that $f_j(x') < f_j(x^*)$ for all $j \in \{1, \ldots, p\}$. By definition, $(f_j(x) - f_j^I) \geq 0$ for all $x \in \mathcal{X}$ and $w > 0$. So, we have $\max\limits_{j=1,\ldots,p} w_j(f_j(x') - f_j^I) < \max\limits_{j=1,\ldots,p} w_j(f_j(x^*) - f_j^I)$, which contradicts optimality of x^* to $P_{Tch}(w)$. Therefore, x^* is a weakly efficient solution. □

The characterization result of the weighted Tchebycheff formulation was presented in [34]. This implies that the entire set of efficient solutions can be obtained by parameterizing the objective functions using the Tchebycheff norm. Eswaran et al. propose an algorithm based on these findings for bicriteria problems in [76]. Their algorithm finds the entire efficient set under a slightly restrictive assumption referred to as uniform dominance (see Definition 6). Ralphs et al. present different ways to relax the uniform dominance assumption in [154].

Definition 6. *An efficient set is said to be uniformly dominant if, for every dominated solution $x' \in \mathcal{X}$, there exists an efficient solution $x^* \in \mathcal{X}$ such that $f_j(x') < f_j(x^*)$ for all $j \in \{1, \ldots, p\}$ [76].*

One way to remove uniform dominance assumption in weighted Tchebycheff method is to use augmented Tchebycheff norm which is defined in [172]. The augmented Tchebycheff norm is defined as

$$\|y_1, \ldots, y_p\|_\infty^{w,\rho} = \max_{j=1,\ldots,p} \{w_j|y_j|\} + \rho \sum_{j=1}^{p} |y_j| \tag{2.22}$$

[2]Since $w > 0$ and $f_j(x) - y_j^I \geq 0$ for all $j \in \{1, \ldots, p\}$ and for all $x \in \mathcal{X}$, $z \geq 0$.

where ρ is a small positive number.

The idea is to generate an outcome that is closest to the ideal point along one edge of the optimal level line as measured by both the L_∞-norm and the L_1-norm. For a given $\rho > 0$ and $w > 0$, the formulation that generates the feasible outcome closest to the ideal point under this metric is as follows,

$$\begin{aligned} P_{aTch}(w) \quad \min \quad & z + \rho \sum_{j=1}^{p} (f_j(x) - y_j^I) \\ \text{s.t.} \quad & z \geq w_j (f_j(x) - y_j^I) \quad j = 1, \ldots, p \\ & x \in \mathcal{X}. \end{aligned} \tag{2.23}$$

Unlike weighted Tchebycheff formulation, the optimal solution of the $P_{aTch}(w)$ generates an efficient solution for given $w > 0$ without any assumption. The characterization result of the method is similar to weighted Tchebycheff method and is presented in [172]. The detailed theoretical results for this method can be found in [171]. Additionally, this method is widely used in interactive solution approaches [4].

As we mentioned in augmented ε-constraint formulation, choosing a proper value for ρ can be problematic. In (2.23), too small a ρ can cause numerical difficulties because the weight of the second term in the objective function can lose significance with respect to the first term. This situation can lead to generation of weakly dominated outcomes despite the augmented objective. On the other hand, augmented Tchebycheff may not be optimal for some nondominated solutions, i.e. some nondominated solutions are unreachable, if ρ is not a sufficiently small coefficient. For discrete bicriteria optimization problems, Dächert et al. present a method for a problem dependent determination of all parameters of the augmented weighted Tchebycheff norm such that all nondominated solutions can be found and ρ is as large as possible. However, this result has not been generalized for the MOP with any number of objectives yet.

In this subsection, all methods that we discussed are also referred to as reference point ($r \in \mathbb{R}^p$) methods [189]. In this context, the reference point of the weighted

Tchebycheff method is the ideal point ($r = y^I$). Another method called min-max approach is introduced by Sayın and Kouvelis where the reference point is the origin, i.e. $r_j = 0$ for $j = 1, \ldots, p$, instead of the ideal point [163]. This method is also referred to as robust optimization in [120]. For a given positive weight vector $w \in \mathbb{R}^p$, the formulation of min-max method is as follows,

$$\begin{aligned} P_{mm}(w) \quad \min \quad & \max_{j=1,\ldots,p} w_j f_j(x) \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned} \tag{2.24}$$

For any $w > 0$, the optimal solution of $P_{mm}(w)$ is weakly efficient. Sayın and Kouvelis present a second stage subproblem to obtain an efficient solution to the problem without the uniform dominance assumption. The optimal solution of the two stage problems is efficient [163]. Authors also show that there exists $w \in \mathbb{R}^p$ for any efficient solution which is optimal to two-stage mathematical programs. They propose an algorithm based on their findings. The algorithm is able to generate all efficient solutions for bicriteria discrete optimization problems by searching over weights.

Along with exact methods several interactive approaches have been presented that uses weighted Tchebycheff norm for multiobjective mixed integer linear programming problems. Steuer and Choo introduce augmented Tchebycheff formulation in [172], and they propose a filtering procedure as an interactive method. Durso proposes an interactive branch-and-bound method method in which augmented weighted Tchebycheff metric is utilized [60]. Karaivanova et al. [107] present an adaptation of the interactive method offered by Steuer and Choo [172] in which the augmented weighted Tchebycheff subproblems are solved heuristically. In [169], Solanki presents an adaptation method where the method is a variation of the noninferior set estimation (NISE) method developed by Cohon et al. for bicriteria linear programs in [43]. The method seeks to generate a representative subset of nondominated solutions by combining the NISE's key features with weighted Tchebycheff subproblems.

Finally, approximation results that use Tchebycheff metric for nonconvex bicriteria problems are presented in [164]. These results are generalized for the multiobjective case in [165].

2.3.4 Other solution methods

In this section, we present the remaining solution methodologies for nonconvex multiobjective optimization problems that could not be classified in the previous subsections.

A large number of studies have been presented to solve MOP problems so far. However, only few of them consider nonconvex multiobjective problems. In the early studies, generally branch-and-bound method is used to obtain the efficient set, see [30], [115], [188], [131] and [137]. Among these studies, while [30], [115] and [188] solve binary multiobjective optimization problems, [131], [137] and [138] deal with mixed binary linear programming problems. Recently, Vincent et al. [185] introduce several improvements to early branch-and-bound methods using better bound sets and branching strategies. In [2], another branch-and-bound method is proposed to obtain the nondominated set. Authors consider two types of nodes in the search. The first set of nodes aim to obtain integer feasible solutions. The second type of nodes remove the dominated integer vectors. Note that [131] is an interactive approach, and the rest of them are exact methods.

One of the mostly used methods to obtain both supported and nonsupported efficient solutions for MODO problems is the two-phase method. Ulungu and Teghem [181] introduce this method to determine the entire efficient set for bicriteria problems in two steps. In the first phase, all extreme supported efficient solutions are obtained by using the algorithm given in [8]. In the second phase, remaining efficient solutions are determined by exploring all the triangles, underlying each pair of adjacent extreme supported efficient solutions. Inside the triangles nonextreme supported and nonsupported efficient solutions are obtained by using branch-and-bound method.

Przybylski et al. generalize this method to generate all efficient solutions for MODO problems with p objective functions in [152].

Another generalization of the two-phase method is presented by Tenfelde-Podehl in [179]. This algorithm determines the ideal and the nadir points for p-objective problem by using the efficient set with $(p-1)$ objectives. Then subspaces are determined with these solutions. For every subspace, a single objective problem is solved. When a new solution is found, the search space is split and new searches are launched. This method stops once all the search spaces have been examined and no new solution is found. This method is applied to three-objective quadratic assignment problem. Based on this idea, an exact approach named parallel partitioning method is proposed to solve bicriteria combinatorial optimization problems in [124]. This method determines all nondominated solutions in three stages. In the first stage, the ideal and the nadir points are computed to establish the boundaries of the search space. In the second stage, well distributed nondominated solutions are searched in order to divide the search space. The third stage consists of finding other efficient solutions by reducing the search space using solutions found during the second stage. Computational performance of this method is shown by using bicriteria flow-shop problem. This method is generalized to obtain nondominated sets of MODO problems with any number of objective functions in [57]. This method is applied to three-objective flow-shop problems.

A completely different solution strategy for MOILP problems to generate all dominated points with p objective functions is presented by Sylva and Crema in [174]. The main idea of this method is to obtain a new efficient solution by removing the dominated space by previously obtained efficient solutions. However, removing some portion of the feasible set requires inclusion of additional constraints and binary variables. The improved version of this algorithm is presented by the same authors in [176] which requires less number of constraints compared to the previous approach. Authors compare these two methods on multiobjective generalized assignment prob-

lem instances with two and three objectives, and results show that improved version performs better than the previous one with respect to solution time.

Recently, Dächert and Klamroth propose a box algorithm to determine the entire nondominated set of MOILP problems with three objectives. Authors show that the number of scalarized subproblems to be solved is bounded by $3|N| - 2$ where N represents the number nondominated solutions [48].

2.4. Computation of the Nadir Point

The nadir point consists of worst objective values attained over the efficient set. Obtaining the nadir point is generally a hard problem [71]. Along with the relatively easy to obtain ideal point, the nadir point is an important element of MOP, because these points define lower and upper bounds of the efficient set. In fact, there are some methods that require the nadir point as input, especially among interactive approaches such as in [141, 91, 119]. Hence, determination of the nadir point has been studied extensively and several exact and heuristic methods have been proposed for the problem [71].

The ideal and the nadir points, denoted as y^I and y^N respectively, determine the bounds of the nondominated set. For any $y \in \mathcal{Y}_N$ these points satisfy $y_j^I \leq y_j \leq y_j^N$ for $j = 1, \ldots, p$. Mathematical definitions of the ideal and the nadir point are given below.

Definition 7. *The point* $y^I = (y_1^I, \ldots, y_p^I)$ *given by*

$$y_j^I = \min_{x \in \mathcal{X}_E} f_j(x) = \min_{y \in \mathcal{Y}_N} y_j \quad j = 1, \ldots, p \tag{2.25}$$

is called the ideal point.

From a multiobjective point of view, computation of the ideal point can be considered as easy [65]. It is well-known that $y_j^I = \min_{x \in \mathcal{X}_E} f_j(x) = \min_{x \in \mathcal{X}} f_j(x)$ for

$j = 1, \ldots, p$.

Definition 8. *The point* $y^N = (y_1^N, \ldots, y_p^N)$ *given by*

$$y_j^N = \max_{x \in \mathcal{X}_E} f_j(x) = \max_{y \in \mathcal{Y}_N} y_j \quad j = 1, \ldots, p$$

is called the nadir point.

Overestimator of the nadir point can be obtained by relaxing the feasible set of the nadir point. A formulation is given below where the feasible set is defined by the original constraints instead of the efficient set.

$$y_j^U = \max_{x \in \mathcal{X}} f_j(x) = \max_{y \in \mathcal{Y}} y_j \quad j = 1, \ldots, p$$

Obviously, y^U is an overestimator for the nadir point y^N, i.e. $y_j^N \leq y_j^U$ for all $j \in \{1, \ldots, p\}$.

Obtaining the nadir point is a challenging task except for the case $p = 2$. In bicriteria optimization, the worst value of the second objective function is attained among solutions that minimize the first objective function and vice versa, which makes it easy to compute. The well-known payoff table solution [155] can be considered as a generalization of this approach for $p > 2$.

Let T represent the payoff table. This table consists of p rows and p columns and is computed by solving single criterion optimization problems. For $m \in \{1, \ldots, p\}$, let x_m^* be an optimal solution to the lexicographic optimization problem [18] where m^{th} objective function is first in the order. Then the entry in the m^{th} row and the k^{th} column of the payoff table is given by $T_{mk} = f_k(x_m^*)$ for $k \in \{1, \ldots, p\}$. The payoff table estimate y_k^{PT} for the k^{th} entry of the nadir point is obtained as follows,

$$y_k^{PT} = \max_{m=1,\ldots,p} T_{mk} = \max_{m=1,\ldots,p} f_k(x_m^*) \quad k = 1, \ldots, p. \tag{2.26}$$

When $p > 2$, payoff table approach can generate only a heuristic solution instead

of the exact nadir point, i.e. $y_j^{PT} \leq y_j^N$ for all $j \in \{1, \ldots, p\}$. Obtaining the payoff table T requires solving p lexicographic optimization problems, i.e. p^2 single-objective optimization problems. Hence complexity of obtaining y^{PT} is $O(p^2 \cdot t)$ where t is the running time of the single-objective optimizer.

Earlier studies on the nadir point were proposed for MOLP problems. Isermann and Steuer propose three different approaches to compute the nadir point for MOLP [102]. The first obtains the nadir point after computing all efficient solutions. The second solves a large primal-dual feasible program with nonlinear constraints. The third is a simplex-based procedure using the fact that the efficient extreme points are connected by efficient edges. Recently, Alves and Costa present a method to compute the nadir point for MOLP by using weight space search [5]. This method determines, for each objective function, the region of the weight space associated with the efficient solutions that have a value in that criterion worse than already known. A new efficient solution is computed with weighted sum formulation using a weight vector picked from the region. The search continues until the region is empty.

Obtaining the nadir point is a special case of the problem of optimization over the efficient set. This is a global optimization problem [21] and has been addressed in several studies. Several of these studies consider the optimization of a linear function over the efficient set of a MOLP, for instance as in [21, 22, 23, 24, 62, 51, 25, 161]. While [50] considers a nonlinear function, [32] takes into account the minimization of a quasi-concave function over the efficient set of a MOLP. Maximization of convex, concave and quadratic functions over the efficient set of a convex mathematical program is presented in [6]. Recently, [33] presents a method that minimizes a convex function over the efficient set of a convex MOP. Survey of existing algorithms for optimization over the efficient set is given in [192]. Although these procedures are theoretically able to compute the nadir values, they are rather complex algorithms with limited or no computational verification [71]. Generally, optimization over efficient set studies consider convex MOP efficient sets. As exceptions, [1] and [103]

optimize a linear function over a multiobjective integer efficient set. Other methods exist in the evolutionary algorithms literature where they try to estimate the nadir point accurately [52].

The most naive way to compute the nadir point is to obtain the entire efficient set. This is computationally demanding, and is not appropriate, for instance, for algorithms that require the nadir point as an input to compute the entire efficient set. Ehrgott and Tenfelde-Podehl [71] present an important result to compute the nadir point for MOP in a computationally more affordable way. The method removes one objective function from the objective functions vector and enumerates all efficient solutions of the remaining MOP with $(p-1)$ objective functions. After repeating for each objective function, these solutions are merged into one set. Some solutions in this set that may not be efficient for the original MOP are eliminated. Finding the supremum of the set for each objective function results in the nadir point for MOP. This method requires an algorithm that generates the efficient set of MOP with $(p-1)$ objective functions and computational verification of the algorithm is not reported in [71].

For MODO problems, another way to obtain the nadir point is to use optimization over integer efficient set algorithms. As an example, in [1], different types of cuts are imposed to improve the objective value at each iteration. The algorithm by Jorge [103] is based on solving more constrained integer linear programs progressively. Both algorithms rely on adding cuts to the single-objective optimization problem, and avoid explicit enumeration of the efficient set. However, depending on the instance, the additional cuts may make the problem computationally overwhelming.

2.5. Representation Methods

Generally the nondominated set is a large set to deal with. For example, a MODO problem with bounded efficient set has finite number of solutions. On the other

hand, the size of the nondominated set may increase to thousands even in mid-sized MODO problems [114]. Decision maker intends to choose an efficient solution among all efficient solutions. As the size of efficient set increases, decision maker has to work on a larger set. It is not meaningful to generate too many efficient solutions. After some point, nondominated solutions inside the outcome space will be too close, so differentiating them become an issue [80]. Because of these reasons, instead of generating all nondominated solutions, generating a subset of the nondominated set may be more meaningful. A finite discrete subset of the nondominated set, $\mathcal{Y}_N$, is called as representative set, and is denoted as $\mathcal{Y}_R$.

A representative set is a subset of a given multiobjective optimization problem's efficient set. Hence, every solution in the representative set is still efficient. Some methods aim to approximate the nondominated set with an approximation factor. These solution methods are called approximation methods [70, 158]. In addition to these methods, another class of solution methods is evolutionary algorithms. The goal of these methods is to get as close as possible to the exact nondominated set [42]. From the decision maker's perspective, representations can be more preferable than approximated solution sets. Discrete representations present a finite and manageable number of solutions to the decision maker, while approximations do not limit the number of solutions. Besides, the solutions in a discrete representation are efficient for MOP this is not necessarily true for approximated solution sets [79].

In the following subsection, methods that generate arbitrary representative sets are reviewed. Afterwards, quality measures and representative methods with quality guarantees are discussed.

2.5.1 Finding arbitrary representative sets

Several methods have been proposed to generate a representation of true nondominated set. In most of these studies, resulting representations are arbitrary in terms of satisfying quality measures. These kind of methods generally aim to obtain the most

diverse subset of the nondominated set. In this subsection, we review the methods that generate arbitrary representations.

Helbig suggests an approach for producing a discrete representation of the efficient sets for biobjective programs. The convex hull of the individual objective minima is discretized and these points are used as the reference points in the min-max method. Helbig presents a method for choosing the discretized points so that the maximum Euclidean distance between a point in the true Pareto set and a point in the representation is at most a prespecified value [98]. Benson and Sayin developed a global shooting procedure that seeks to find global representations of MOP with a compact feasible set by constructing a special simplex that contains the feasible criterion space [26]. Das and Denis propose a boundary intersection method for finding several nondominated solutions for general multiobjective optimization problem [49]. This method is able to generate an evenly distributed set of solutions in the nondominated set. Shao and Ehrgott propose a method that combines the global shooting and normal boundary intersection methods to determine well distributed nondominated solutions for MOLP problem [167]. The method produces evenly distributed nondominated solutions without missing parts of the nondominated set. Karasakal and Koksalan suggest a method to obtain discrete representation of the nondominated set for multiobjective linear programming problem [108]. First, method finds an approximate surface of the nondominated set. Then, this surface is discretized with equidistant points. These are projected onto the nondominated set and they form discrete representation. Eichfelder [73, 74] gives a method to determine the representative set for multiobjective optimization problem considering the spacing of generated nondominated solutions which is based on Pascoletti and Serafini scalarization [148].

Fu and Diwekar present an approach to obtain representation of the nondominated set based on efficient sampling methods used in uncertainty analysis [85]. Caballero and Hernández present a new method to estimate the weakly efficient set for mul-

tiobjective linear fractional programming problem. The method is able to control distances between solutions from the estimation set [37].

Kim and De Weck give a method to generate a well-distributed representation for bicriteria nonconvex problems [111]. Authors use weighted sum method with additional inequality constraints to find efficient solutions inside nonconvex regions of the efficient set. Kim and De Weck generalize the method for multiobjective optimization problems with any number of objectives in [112]. Sylva and Crema present an algorithm for generating well-dispersed nondominated solutions for multiobjective mixed integer linear programming problem [175]. Starting from an initial nondominated solution, at each iteration the procedure finds a new one that maximizes the L_∞-norm distance to the set dominated by all the previously found nondominated solutions. The optimal solution of this mathematical program may not be efficient. Hence, they use another subproblem to generate the closest nondominated solution to the mapping of the previous solution. Masin and Bukchin propose a method to obtain a set of nondominated solutions with the maximum diversity for mixed-integer and combinatorial optimization problems [135]. The method iteratively includes a new nondominated solution to the set which is most diverse from the others.

Leyffer formulates the problem of finding a maximally uniform representation of the nondominated set with a given number of solutions for convex multiobjective optimization problems as a mathematical program with complementarity constraints [125]. A similar problem is considered by Faulkenberg et al. in [81]. They present two methods for generating discrete representations with equidistant points for bicriteria optimization problems. In the first method, authors utilize ε-constraint method with an additional constraint to control the spacing of generated solutions. In the second method, they use bilevel programming formulation. While upper level formulation controls the spacing, lower level formulation generates the nondominated solutions. Pereyra et al. give a method for uniform sampling of the nondominated set for bicriteria optimization problems by using equal spacing constraints on the objective

values [149].

2.5.2 Finding representative sets with quality guarantees

In general, the aim in finding a representation is to obtain a "good" subset of the efficient set. Since the meaning of "good" is ambiguous, different quality measures are defined. Recent surveys for the quality measures of representative sets are presented in [79, 80]. Sayın presents three different measures to evaluate a representation in [160] that are listed as follows:

1. **Coverage:** All of the efficient solutions of the problem must be well-represented. A globally-representative subset of $\mathcal{X}_E$ should contain points from every portion of the efficient set without missing any region.

2. **Uniformity:** The representation should be uniform or it should not be clustered in region(s) of the frontier. In the name of uniformity, an ideal representation may contain points that are exactly at the same distance to each other.

3. **Cardinality:** The representation should consist of a reasonable number of elements. Too many points require some processing before being studied by the decision maker. Including one more efficient solution into the representation has a cost, so that the size of the representation should be as small as possible.

In any representation, the cardinality can be determined just by examining the size of the representation. However, the coverage error and the uniformity level need to be defined mathematically. In [160], given a representation $\mathcal{Y}_R$ of a set $\mathcal{Y}_N$, the coverage error is given by the quantity

$$\alpha^* = \max_{\bar{y} \in \mathcal{Y}_N} \min_{y \in \mathcal{Y}_R} d(\bar{y}, y). \tag{2.27}$$

As seen in (2.27), any fixed element $\bar{y} \in \mathcal{Y}_N$ is represented by the closest point to $\bar{y}$ in the representation $\mathcal{Y}_R$. The worst representative point in the nondominated set defines the coverage error performance of the representation. Similarly, the uniformity level δ is determined by the quantity

$$\delta = \min_{\substack{\bar{y}, y \in \mathcal{Y}_R \\ \bar{y} \neq y}} d(\bar{y}, y). \tag{2.28}$$

Equation (2.28) calculates the closest two points in the representation for a given distance metric. Briefly, α^* is a parameter that signifies how precisely the efficient set is being approximated, and δ implies that points within a representation do not get closer to each other more than a δ amount.

Among these measures, we focus on the coverage measure. We aim to generate a representative set with a specified coverage error $\alpha \in \mathbb{R}$. Definition of a α-representation is given below.

Definition 9. *A set $\mathcal{Y}_R$ is called an α-representation of $\mathcal{Y}_N$ if $|\mathcal{Y}_R| < \infty$, $\mathcal{Y}_R \subseteq \mathcal{Y}_N$ and the optimal objective value of* (2.27) *is less than or equal to α, i.e. $\alpha^* \leq \alpha$.*

A few studies consider methods that generate representations with specified level of error. In [162], Sayin gives a method for multiobjective linear problems to produce a discrete representation with a specified quality guarantee or the maximum coverage possible given a target cardinality by using a mathematical program.

Sayin and Kouvelis, [163, 120] give a method to obtain all nondominated solutions of multiobjective optimization problem. They utilize parametric search over weights with min-max type subproblems. Additionally, they propose a modification in their algorithm to obtain a representative sample of the efficient set based on a parameter. The quality guarantee of the representation is controlled by continuing to refine an interval between two previously generated nondominated solutions until its length falls below a specified value.

Hamacher et al. present a box algorithm to generate a representation of the efficient set for bicriteria discrete optimization problems [95, 157]. Initially, the algorithm computes the boundaries of the nondominated set that also defines the initial box. Boxes are formed with consecutive nondominated solutions. For each box, the algorithm solves the ε-constraint method to generate a nondominated solution. The presepecified error of the representation is calculated as the area of the largest of these rectangles. The representation is refined until the accuracy is met.

Eusebio and Figueira propose a new algorithm to find a representation of the nondominated set for bicriteria integer network flow problem [77]. The algorithm solves a sequence of ε-constraint problems with a branch-and-bound algorithm to find a subset of the nondominated set. At each iteration, the algorithm finds a solution by using ε-constraint problem, and adds to the representation until it is guaranteed that the representation has the desired quality. Vaz et al. introduce several algorithms for finding a representation for a bicriteria combinatorial optimization problem considering uniformity, coverage and the ε-indicator measures [182].

Chapter 3

A NEW ALGORITHM FOR GENERATING ALL NONDOMINATED SOLUTIONS FOR MULTIOBJECTIVE DISCRETE OPTIMIZATION PROBLEMS

In this chapter, a new algorithm is proposed to generate all nondominated solutions for MODO problems with any number of objective functions. We reviewed exact solutions methods for MODO problems with p objectives in Section 2.3. One possible way is to find solutions by removing the dominated space by previously obtained efficient solutions [174, 176, 127]. Another way is to search the outcome space or the projection of the outcome space exhaustively by using grids [122, 139], recursion [179, 144] or boxes [48]. In our algorithm, we manage the search over $(p-1)$-dimensional rectangles which is similar to the $(p-1)$-dimensional grid definition by Laumanns et al. [122]. For each rectangle, the algorithm solves two-stage optimization problems to obtain an efficient solution where the first stage problem is ε-constrained scalarization. In the second stage, a simpler model is solved to deal with weakly efficient solutions. The proposed method searches the $(p-1)$-dimensional space exhaustively to generate all nondominated solutions. The contribution of the method lies in the way rectangles are defined and tracked. The algorithm is compared with former studies on multiobjective knapsack and multiobjective assignment problem instances. The method is highly competitive in terms of solution time and the number of optimization models solved. In the following section, we present the theoretical background. In Section 3.2, the algorithm and associated theoretical results are given. In Section 3.3, computational results and comparison with previous

algorithms are presented on multiobjective knapsack and multiobjective assignment problem instances. Finally, conclusions are presented in Section 3.4.

3.1. Theoretical Background

We aim to obtain all nondominated solutions for MODO problems. We use the ε-constraint method to obtain efficient solutions. We utilize two-stage formulations to avoid weakly efficient solutions. For any $\varepsilon \in \mathbb{R}^{p-1}$, two-stage ε-constraint formulations, $P_k(\varepsilon)$ and $Q_k(\varepsilon)$ for some $k \in \{1, \ldots, p\}$, are defined as follows.

$$\begin{aligned} \mathbf{P_k}(\varepsilon) \quad \mathbf{z} = \min \quad & f_k(x) \\ \text{s.t.} \quad & f_j(x) \leq \varepsilon_j \quad j = 1, \ldots, p \text{ and } j \neq k \\ & x \in \mathcal{X}. \end{aligned}$$

Let z^* be the optimal objective value of the subproblem $P_k(\varepsilon)$ and consider the second stage formulation $Q_k(\varepsilon)$.

$$\begin{aligned} \mathbf{Q_k}(\varepsilon) \quad \min \quad & \sum_{j=1}^{p} f_j(x) \\ \text{s.t.} \quad & f_j(x) \leq \varepsilon_j \quad j = 1, \ldots, p; \text{ and } j \neq k \\ & f_k(x) = z^* \\ & x \in \mathcal{X}. \end{aligned}$$

Let x^* be an optimal solution of two-stage formulations $P_k(\varepsilon)$ and $Q_k(\varepsilon)$. We will show that x^* is always efficient for any $\varepsilon \in \mathbb{R}^{p-1}$, and any efficient solution of MODO problem can be obtained by using two-stage programs.

Theorem 6. *For $\varepsilon \in \mathbb{R}^{p-1}$, an optimal solution to the two-stage formulations $P_k(\varepsilon)$*

and $Q_k(\varepsilon)$ is efficient.

Proof. Let $\varepsilon \in \mathbb{R}^{p-1}$ and let x^* be an optimal solution to $P_k(\varepsilon)$ and $Q_k(\varepsilon)$ for some $k \in \{1, \ldots, p\}$. Suppose x^* is not efficient. Then there exists a solution $x' \in \mathcal{X}$ such that $f_j(x') \leq f_j(x^*)$ for all $j = 1, \ldots, p$ and $f_{\hat{\jmath}}(x') < f_{\hat{\jmath}}(x^*)$ for some $\hat{\jmath} \in \{1, \ldots, p\}$. Since $f_j(x') \leq f_j(x^*) \leq \varepsilon_j$ for $j = 1, \ldots, p$ and $j \neq k$, x' is feasible for $P_k(\varepsilon)$. If $f_k(x') < f_k(x^*)$, this contradicts the optimality of x^*. Then $f_k(x') = f_k(x^*) = z^*$ must hold. Therefore, x' is also feasible to $Q_k(\varepsilon)$. Summing over all j yields $\sum_{j=1}^{p} f_j(x') < \sum_{j=1}^{p} f_j(x^*)$, which contradicts optimality of x^* to $Q_k(\varepsilon)$. Therefore, the optimal solution x^* of the two-stage formulations is efficient. □

Theorem 7. *For any efficient solution x^* of MODO, there exists an $\varepsilon \in \mathbb{R}^{p-1}$ such that x^* is an optimal solution to two-stage formulations $P_k(\varepsilon)$ and $Q_k(\varepsilon)$ for some $k \in \{1, \ldots, p\}$.*

Proof. For any $k \in \{1, \ldots, p\}$, let $\hat{\varepsilon} = (\ldots, f_{k-1}(x^*), f_{k+1}(x^*), \ldots)$, $\hat{\varepsilon} \in \mathbb{R}^{p-1}$, and suppose that x^* does not solve two-stage formulations $P_k(\hat{\varepsilon})$ and $Q_k(\hat{\varepsilon})$. Let x' be an optimal solution of two-stage programs with $\hat{\varepsilon}$. Note that x^* is feasible to $P_k(\hat{\varepsilon})$. Then either $f_k(x') < f_k(x^*)$ or $f_k(x') = f_k(x^*)$ and $\sum_{j=1}^{p} f_j(x') < \sum_{j=1}^{p} f_j(x^*)$. Since $f_j(x') \leq f_j(x^*) = \hat{\varepsilon}_j$ for all $j \in \{1, \ldots, p\}$ and $f_{\hat{\jmath}}(x') < f_{\hat{\jmath}}(x^*) = \hat{\varepsilon}_{\hat{\jmath}}$ for some $\hat{\jmath} \in \{1, \ldots, p\}$, x' dominates x^*, which contradicts that x^* is an efficient solution. □

Theorems 6 and 7 show that with appropriate choices of $\varepsilon \in \mathbb{R}^{p-1}$ all efficient solutions can be obtained by using two-stage formulations. In the characterization, $\varepsilon_j \in \mathbb{R}$ values are equal to the actual objective values of the efficient solution. In Figure 3.1, all nondominated solutions in $\mathbb{R}^3$ and the projection of these points onto $\mathbb{R}^2$ for a three objective assignment problem with 5 items are given. For this example, we set $k = 1$ in $P_k(\varepsilon)$, so ε_j values are on the $f_2 - f_3$ plane. As seen in the figure, for each nondominated solution in $\mathbb{R}^3$ there exists at least one rectangle such that every $\varepsilon \in \mathbb{R}^2$ inside this rectangle generates the nondominated solution as an optimal solution to

two-stage problems. This implies that the nondominated set can be obtained by searching over $\varepsilon \in \mathbb{R}^{p-1}$ vectors in the projected space, i.e. $(p-1)$-dimensional space. The challenge is in managing this structure when the nondominated set is not known but is to be obtained.

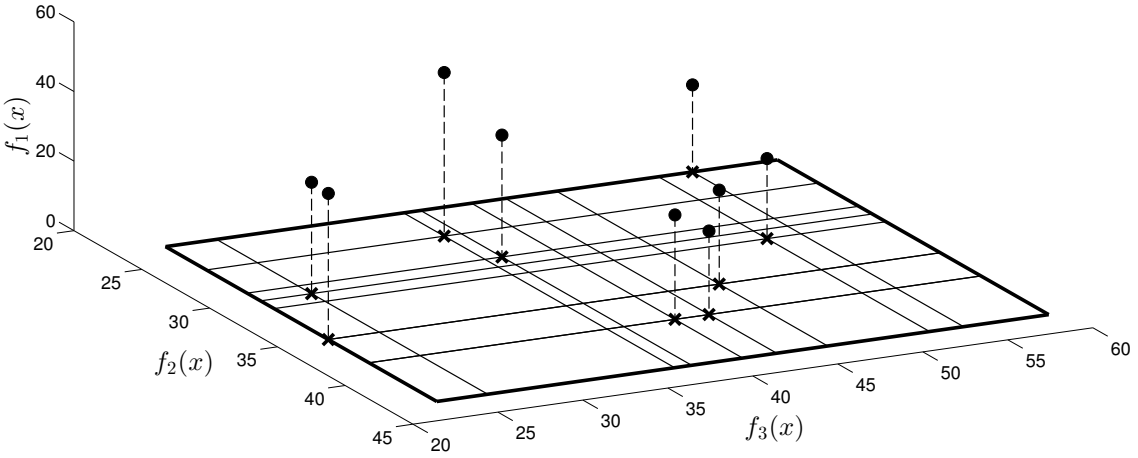


Figure 3.1: Nondominated solutions in $\mathbb{R}^3$ and projection of the points onto the $f_2 - f_3$ plane.

For $p = 2$, a single parameter $\varepsilon \in \mathbb{R}$ suffices to parameterize the subproblems. By changing ε systematically, the entire nondominated set can be obtained. For $p = 3$, however, the parametric space consists of $\mathbb{R}^2$, and even for this case, how $\mathbb{R}^2$ should be exhaustively searched so that all nondominated solutions are obtained is not clear. Below we present a new algorithm which relies on $P_k(\varepsilon)$ and $Q_k(\varepsilon)$ and introduce a structure of partitioning $\mathbb{R}^{p-1}$ based on rectangles.

3.2. Finding the Nondominated Set Using the ε-Constraint Method

The results in the previous section show that all efficient solutions of the MODO problem with p objective functions can be obtained by performing a search over

$\varepsilon \in \mathbb{R}^{p-1}$ and solving single-objective two-stage mathematical programs, $P_k(\varepsilon)$ and $Q_k(\varepsilon)$, for some $k \in \{1, \ldots, p\}$. To simplify the presentation of the proposed method, without loss of generality assume that $k = 1$. Since ε is in $(p-1)$-dimensional space, the search is managed in $\mathbb{R}^{p-1}$. For ease of notation, we define $\bar{y} = \bar{f}(x) \in \mathbb{R}^{p-1}$ where $\bar{f}(x) = (f_2(x), \ldots, f_p(x))$. Figure 3.1 shows the projection of the nondominated solutions onto the $f_2 - f_3$ plane where the outcome space is in $\mathbb{R}^3$ and the search space is in $\mathbb{R}^2$. The projected points form $(p-1)$-dimensional rectangles in the search space. In the figure, a rectangle's lower and upper bound are determined by the $\bar{y}$ values which are the projections of nondominated solutions.

A rectangle's lower and upper vertices are denoted as $l \in \mathbb{R}^{p-1}$ and $u \in \mathbb{R}^{p-1}$, respectively. With these bounds, a rectangle is defined as $R(l, u) = \{\bar{y} \in \mathbb{R}^{p-1}, l \leq \bar{y} \leq u\}$ [100]. During the search, a set of rectangles should be maintained, the i^{th} rectangle is denoted as R_i with its lower and upper vertices $l^i, u^i \in \mathbb{R}^{p-1}$. All rectangles that need to be searched are kept in list L, i.e. $L = \bigcup_{i=1}^{|L|} \{R_i\}$ where $|L|$ represents the number of rectangles in L.

The search is initialized with a single rectangle that covers $(p-1)$-dimensional space. Indeed, if the problem is bounded, then it is possible to define the boundaries of the search space for each dimension by first solving small subproblems. Lower bounds for each dimension can be obtained by minimizing each objective function over the feasible set $\mathcal{X}$. This returns the j^{th} element of the ideal point (y^I). Similarly, upper bounds (y_j^U) can be obtained by maximizing each objective function over $\mathcal{X}$ for all $j \in \{1, \ldots, p\}$. y^U defines an upper bound for each efficient solution in $\mathcal{X}_E$, i.e. $f_j(x) \leq y_j^U$ for $j = 1, \ldots, p$ for each $x \in \mathcal{X}$. Since the search is managed over $\varepsilon \in \mathbb{R}^{p-1}$, we define $\bar{y}^I$ and $\bar{y}^U$ in $\mathbb{R}^{p-1}$ where $\bar{y}^I = (y_2^I, \ldots, y_p^I)$ and $\bar{y}^U = (y_2^U, \ldots, y_p^U)$.

Instead of finite bounds for the initial rectangle, our algorithm can be adjusted to allow for $\mathbb{R}^{p-1}$ itself. However, we want to implement a selection procedure based on a volume measure. Hence, we introduce finite bounds. We define a volume measure

associated with rectangle R_i as

$$V_i = \prod_{j=1}^{p-1} (u_j^i - \bar{y}_j^I). \tag{3.1}$$

Note that this is not the volume of R_i itself, but the volume of the rectangle defined by $\bar{y}^I$ as the lower vertex and the upper vertex u^i of R_i. In the rest of the chapter, when we refer to the volume of the rectangle, measure V_i is implied.

An efficient solution is obtained by solving $P_1(u^i)$ and $Q_1(u^i)$ associated with $R_i(l^i, u^i)$. When a new nondominated solution is found, some of the rectangles in the list L are divided into smaller disjoint rectangles. Although each rectangle is defined by using lower and upper vertices, the two-stage formulation is solved only considering the upper bounds. Some of the feasible regions in $P_1(u^i)$ with different rectangles may overlap. Therefore it might be better to process a rectangle with larger volume before a rectangle with a smaller volume. This rule aims to select rectangles in a way to improve total processing time. Nevertheless, the proposed algorithm does not critically depend on such a prioritization rule. In principle, an arbitrary rectangle can be chosen from the list L instead of processing one with the largest volume.

We label the proposed method as Searching Over Rectangles for Generating the Nondominated Set (SOR-GNS). The SOR-GNS starts with a single $(p-1)$-dimensional rectangle where the lower and upper vertices of the rectangle are $\bar{y}^I$ and $\bar{y}^U$, respectively. In each iteration, the algorithm picks a rectangle R_i with the largest volume. Then $P_1(u^i)$ and $Q_1(u^i)$ are solved. The technical statement of the SOR-GNS is as follows.

Algorithm SOR-GNS

Input: MODO problem, i.e. objective functions $f_j(x)$ for $j = 1, \ldots, p$, and feasible set $\mathcal{X}$.

Output: Nondominated set ($\mathcal{Y}_N$) of MODO problem.

Step-1 Initialize the nondominated solutions list, $\mathcal{Y}_N = \emptyset$, and the rectangles search list, $L = \{R(\bar{y}^I, \bar{y}^U)\}$.

Step-2 If L is empty, go to Step-4. Otherwise, pick a rectangle $R_i(l^i, u^i)$ with the highest V_i from the list L. Solve the first stage formulation with the upper vertex of R_i, i.e. $P_1(u^i)$.

Step-3 If $P_1(u^i)$ is feasible, than solve $Q_1(u^i)$. Let x^* denote an optimal solution.

- If $f(x^*) \notin \mathcal{Y}_N$, $\mathcal{Y}_N = \mathcal{Y}_N \cup \{f(x^*)\}$. Now, apply the rectangular subdivision process for each $R_s \in L$. Pick rectangle R_s from list L. Set $L = L \setminus \{R_s\}$. Define an index set for R_s as $C_s = \{j \in \{1, \ldots, p-1\} : l^s_j < \bar{f}_j(x^*) < u^s_j\}$. Initialize list L' for the rectangular subdivision process, $L' = \{R_s\}$.

  For Each $(j \in C_s)$

  Initialize list L'', $L'' = \emptyset$.

  For Each $(R_t \in L')$

  $R_1 = \{\bar{y} \in R_t : \bar{y}_j \leq \bar{f}_j(x^*)\}$

  $R_2 = \{\bar{y} \in R_t : \bar{y}_j \geq \bar{f}_j(x^*)\}$

  $L'' = L'' \cup \{R_1\} \cup \{R_2\}$

  end

  $L' = L''$

  end

  $L = L \cup L'$. Remove rectangles that lie in $R(\bar{f}(x^*), u^i)$.

- Else (If $f(x^*) \in \mathcal{Y}_N$), remove rectangles that lie in $R(\bar{f}(x^*), u^i)$.

Else (If $P_{k,m}(u^i)$ is infeasible), remove rectangles that lie in $R(\bar{y}^I, u^i)$.

Go to Step-2.

Step-4 Return the nondominated set $\mathcal{Y}_N$ and stop.

Depending on the solution of the two-stage formulations, three cases can occur. First a new nondominated solution may be obtained, and let x^* be an associated efficient solution. Initially, the outcome of x^* in the objective space $f(x^*)$ is inserted into nondominated solutions list $\mathcal{Y}_N$. Then, among all rectangles in L, if $\bar{f}_j(x^*)$ is in between the bounds of rectangle R_i for the j^{th} axis, then R_i is split into two rectangles along the j^{th} axis.

After updating the list, some of the rectangles can be removed because the volume between the upper vertex of the rectangle and $\bar{f}(x^*)$ does not need to be searched (see Lemma 1). Therefore, any rectangle $R_s \in L$, if $R_s \subseteq R(\bar{f}(x^*), u^i)$, then $L = L \setminus \{R_s\}$.

The second possible case after solving a set of two-stage programs in SOR-GNS is that an optimal solution x^* is obtained, but $f(x^*)$ has already been found at an earlier iteration ($f(x^*) \in \mathcal{Y}_N$). The reason for this situation is that different $\varepsilon \in \mathbb{R}^{p-1}$ values may return the same nondominated solution when problems $P_1(\varepsilon)$ and $Q_1(\varepsilon)$ are solved as seen in Figure 3.1. As in the previous case, rectangles that lie in between $\bar{f}(x^*)$ and current rectangle's upper vertex u_i are removed.

The third possible case is the infeasibility of the two-stage programs with u^i. Since every $\varepsilon \in R(\bar{y}^I, u^i)$ would lead to an infeasible $P_1(\varepsilon)$ formulation (see Lemma 2), all such rectangles are removed. The algorithm is terminated when the list of rectangles is empty. At termination all nondominated solutions ($\mathcal{Y}_N$) are obtained (see Theorem 13).

The steps of the algorithm have been presented so far. Now, we need to show that the proposed algorithm generates all nondominated solutions in a finite number of iterations. Below, we first show that no nondominated solution maps into the removed rectangles.

Lemma 1. *At any point in the algorithm, let x^* be an optimal solution of two-stage programs $P_1(u^i)$ and $Q_1(u^i)$ where $u^i \in \mathbb{R}^{p-1}$ is the upper vertex of a rectangle. Then, there is no nondominated solution that is mapped into $R(\bar{f}(x^*), u^i)$, other than $f(x^*)$.*

Proof. Assume to the contrary that there exist an efficient solution $x' \in \mathcal{X}$ that is mapped into the rectangle $R(\bar{f}(x^*), u^i)$. Then, $\bar{f}_j(x^*) \leq \bar{f}_j(x') \leq u^i_j$ for $j = 1, \ldots, p-1$. This implies that x' is feasible to $P_1(u^i)$, and we know that x^* is optimal to $P_1(u^i)$. Then, $f_1(x^*) \leq f_1(x')$. Unless $f_j(x') = f_j(x^*)$ for all $j = 1, \ldots, p$, x^* dominates x'. Therefore, there exists no nondominated solution that is mapped inside the rectangle $R(\bar{f}(x^*), u^i)$ other than $f(x^*)$. □

This shows that when the algorithm obtains x^* as an optimal solution to two-stage problems, there exists no nondominated solution that projects into rectangle $R(\bar{f}(x^*), u^i)$ where u^i is the upper vertex of the current rectangle. The lemma applies to both the first and the second case in SOR-GNS.

The final case of the proposed algorithm is the infeasibility of the two-stage programs. Given upper vertex u^i for the rectangle, if $P_1(u^i)$ is feasible, then $Q_1(u^i)$ is also feasible. Therefore, infeasiblity can be observed only in the first stage problem. Let the upper bound of the current rectangle be $u^i \in \mathbb{R}^{p-1}$. If $P_1(u^i)$ is infeasible, then the rectangles inside $R(\bar{y}^I, u^i)$ are removed. Lemma 2 addresses the infeasibility case.

Lemma 2. *If $P_1(u^i)$ is infeasible, then there is no nondominated solution that is mapped into the rectangle $R(\bar{y}^I, u^i)$.*

Proof. Assume to the contrary that there exist an efficient solution $x' \in \mathcal{X}$ such that $f_j(x') \leq u^i_j$ for $j = 2, \ldots, p$. This implies that x' is feasible to $P_1(u^i)$ which contradicts that $P_1(u^i)$ has no feasible solution. Therefore there exist no nondominated solution that is mapped into $R(\bar{y}^I, u^i)$. □

Example: The key aspects of the algorithm are shown on an illustrative example with three objectives. The first case is represented in Figure 3.2. First, the initial rectangle $R(\bar{y}^I, \bar{y}^U)$ is determined which is shown in bold in Figure 3.2a. Then, the two-stage formulation is solved with $\bar{y}^U$. The f_1 value of the resulting nondominated solution is y^I_1. The single rectangle is split into four rectangles with the new

nondominated solution. The first iteration is completed by removing the rectangle $R((\bar{y}_2^1, \bar{y}_3^1), (\bar{y}_2^U, \bar{y}_3^U))$ by using Lemma 1. In the following iteration, the volume measure indicates to process the rectangle $R((\bar{y}_2^I, \bar{y}_3^1), (\bar{y}_2^1, \bar{y}_3^U))$ (rectangle with dotted pattern) in Figure 3.2a. Then the two-stage formulation is solved with the upper vertex of this rectangle. The resulting nondominated solution and new rectangles are shown in Figure 3.2b. The shaded volume corresponds to removed rectangles in the former iterations. In the third iteration, a new nondominated solution is obtained as seen in Figure 3.2c. Unlike previous iterations, the new nondominated solution labeled with 3 is outside of the chosen rectangle.

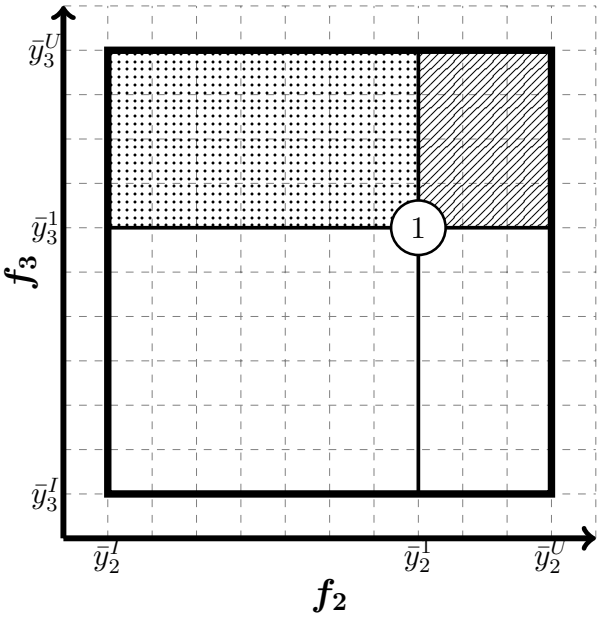
For the following iteration, the rectangle $R((\bar{y}_2^3, \bar{y}_3^1), (\bar{y}_2^1, \bar{y}_3^2))$ is picked as indicated in Figure 3.2c. This rectangle leads to the point labelled with 3 again. Since the nondominated solution has already been obtained before, all rectangles inside $R((\bar{y}_2^3, \bar{y}_3^3), (\bar{y}_2^1, \bar{y}_3^2))$ can be removed. Therefore, rectangle $R((\bar{y}_2^3, \bar{y}_3^1), (\bar{y}_2^1, \bar{y}_3^2))$ is removed as shown in Figure 3.3.

The final case of the algorithm is illustrated in Figure 3.4 which is the fifth iteration of the algorithm on the three-objective sample problem. The highest volume rectangle is $R((\bar{y}_2^2, \bar{y}_3^1), (\bar{y}_2^3, \bar{y}_3^2))$ in Figure 3.3. The first stage is solved with the upper vertex of this rectangle, and this model is infeasible. By Lemma 2 all rectangles inside $R((\bar{y}_2^I, \bar{y}_3^I), (\bar{y}_2^3, \bar{y}_3^2))$ are removed as shown in Figure 3.4. List L is updated accordingly. After this iteration, since L still contains three rectangles, the algorithm will continue with picking $R((\bar{y}_2^I, \bar{y}_3^2), (\bar{y}_2^2, \bar{y}_3^U))$ in Figure 3.4 which has the highest volume measure.

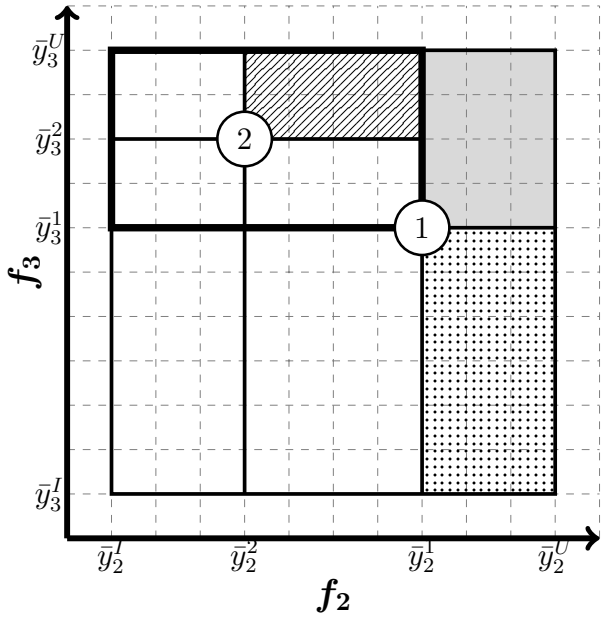
We have shown that in each iteration the algorithm removes at least one rectangle from list L, and no other nondominated solution projects into removed rectangles. To show the completeness of the proposed algorithm, Theorem 8 is given below.

Theorem 8. *SOR-GNS generates the entire nondominated set.*

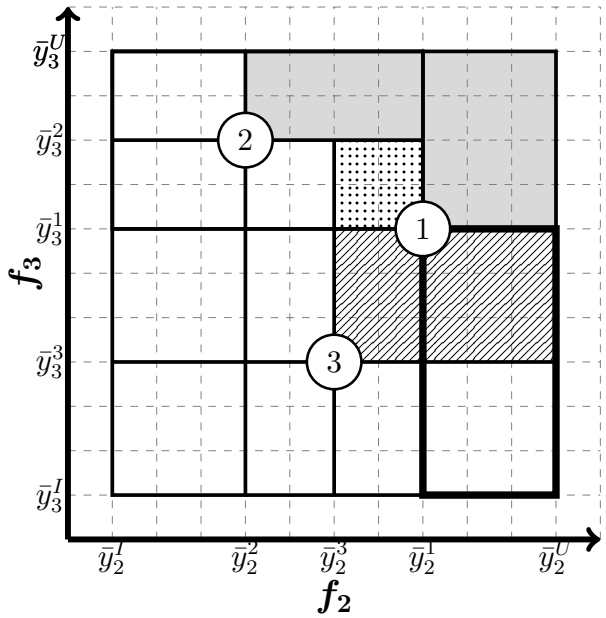
Proof. Two-stage mathematical programs $P_1(u^i)$ and $Q_1(u^i)$ either find an efficient solution (see Theorem 6) or they are infeasible for right-hand-side vector u^i. Therefore,



(a) Initial search space and the first nondominated solution.



(b) Second iteration.



(c) Third iteration.

Figure 3.2: Case-1: A new nondominated solution is obtained.

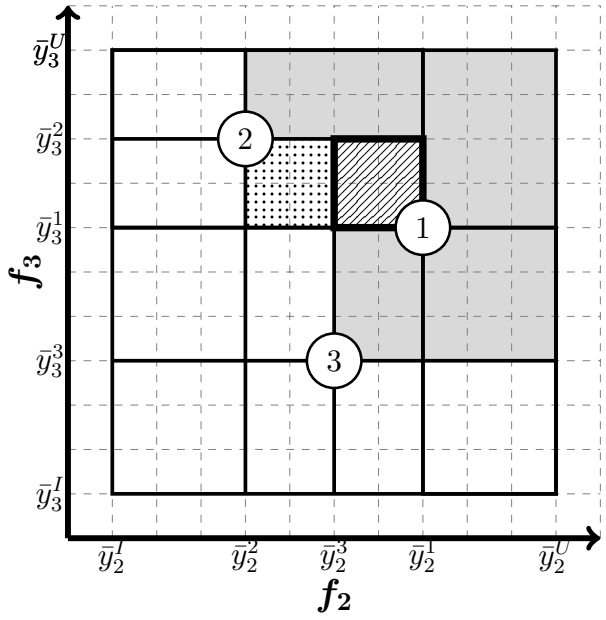


Figure 3.3: Case-2: Resulting nondominated solution has already been found.

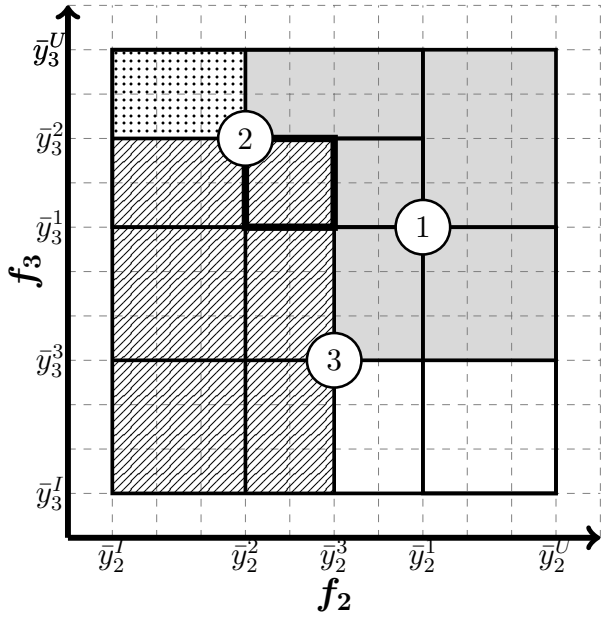


Figure 3.4: Case-3: Infeasiblity of the first stage problem.

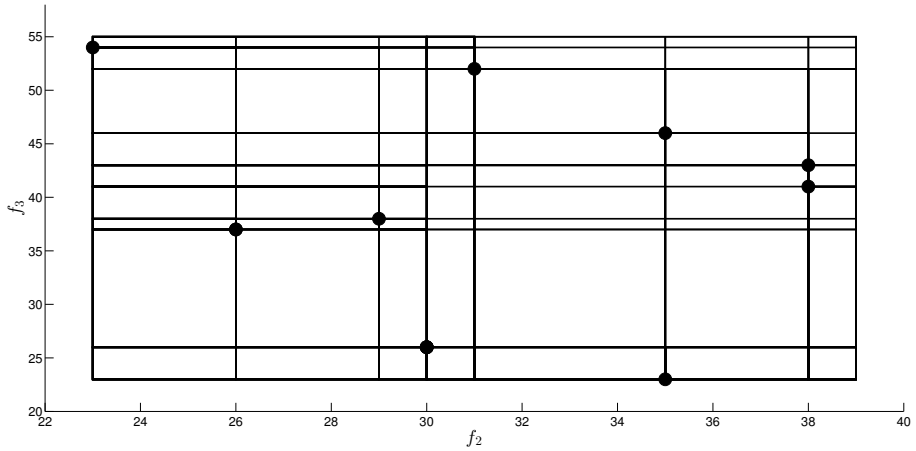
all points in $\mathcal{Y}_N$ are nondominated solutions. Let $\bar{\mathcal{Y}}_N$ contain the projections of all nondominated solutions to R^{p-1} with first coordinate f_1 removed. Then $\bar{\mathcal{Y}}_N \subseteq R(\bar{y}^I, \bar{y}^U)$ by definition of $\bar{y}^I$ and $\bar{y}^U$. In the proposed algorithm this space is searched exhaustively, and in each iteration some part of the initial rectangle is removed. From Lemma 1 and 2, removed volumes do not include images of a nondominated solution. From the characterization result, there exists $\varepsilon \in \mathbb{R}^{p-1}$ for each nondominated solution. Therefore, when the termination condition occurs in SOR-GNS, all nondominated solution are obtained. □

In Theorem 9, we show that the proposed algorithm terminates in a finite number of iterations.

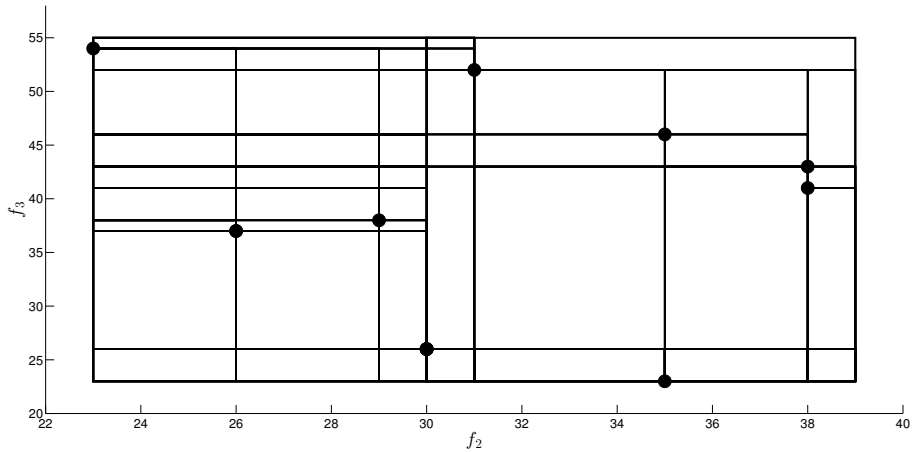
Theorem 9. *SOR-GNS is finite.*

Proof. Since we deal with MODO problems with bounded efficient sets, they have a finite number of nondominated solutions which is represented with $|\mathcal{Y}_N|$. In the worst case, each axis in $\mathbb{R}^{p-1}$ is divided into $(|\mathcal{Y}_N| + 1)$ rectangles. Since the search space is $(p-1)$-dimensional, $|L| \leq (|\mathcal{Y}_N| + 1)^{p-1}$ where $|L|$ denotes the size of the rectangle list. This means that the number of rectangles is also finite. Besides, in each iteration at least one rectangle is removed from L. This implies that L will be empty in a finite number of iterations. Hence, SOR-GNS is finite. □

In Figure 3.5, the projections of all nondominated solutions of problem instance given in Figure 3.1 into $\mathbb{R}^2$ are shown. When the search space is divided along the image of each nondominated solution, Figure 3.5a is obtained. This is the worst case in terms of generated rectangles to be explored by SOR-GNS. In Figure 3.5b, the actual rectangles generated by SOR-GNS are shown.



(a) Projection of nondominated solutions and corresponding rectangles.



(b) Rectangles that are generated by SOR-GNS.

Figure 3.5: Rectangles generated by the projection of nondominated solutions and SOR-GNS.

3.3. Computational Results

The proposed method is compared with three different methods which are presented in [174], [121] and [144]. The main idea of the method presented in [174] is to obtain a new efficient solution by removing the dominated space by the previously obtained efficient solutions. The method includes additional constraints and binary variables to remove the dominated space. In [121], Laumanns et al. propose a method in which they utilize an adaptive scheme for the ε-constraint method to obtain the entire efficient set. They search for efficient solutions by using $(p-1)$-dimensional grid which partitions a $(p-1)$-dimensional projection of the objective space. For each grid, the method solves an ε-constraint subproblem with a second stage formulation to deal with the weakly efficient solutions. Laumanns et al. utilize the same search strategy in [122]. Unlike [121] in [122], they use ε-constraint method with two-sided bounds to obtain a solution inside the $(p-1)$-dimensional grid and solve lexicographic optimization problem. [144] is a recursive algorithm that is proposed by Ozlen and Azizoglu. The authors use the augmented form for the ε-constraint formulation. The algorithm applies the range information in the constrained problems to generate all nondominated solutions. The common property of these methods is that all can obtain the entire nondominated set for MODO problems with any number of objective functions. All algorithms except for the adaptive ε-constraint method have been implemented in C++. Laumanns et al. provide the source code of their algorithm which is implemented in C [121]. Subproblems are solved by using IBM CPLEX 12.4 [47]. All tests were conducted on a shared cluster with Intel Xeon 2.3 GHz CPU and 4 GB memory limit with Linux operating system. These algorithms are tested on the multiobjective knapsack problem and multiobjective assignment problem instances. Different problem categories are generated based on problem size. There are 10 instances in each category. Tables 3.1-3.3 summarize computational results. In these tables, the first row for each category reports averages. The numbers in the second

row correspond to the minimum and maximum for each category.

We report CPU time and number of solved models statistics to compare the four different methods. Sylva and Crema [174] solve a single model with additional disjunctive constraints to obtain a nondominated solution. Ozlen and Azizoglu [144] apply augmented form of the ε-constraint method so that a single model solution is sufficient to obtain a nondominated solution. Laumanns et al. [121] and SOR-GNS applies two-stage formulation so it requires to solve two submodels per nondominated solution. Since solving the remaining second stage of two-stage formulation is relatively easy after solving the first submodel, we also count each of them as one. Computation of each instance is interrupted after 25000 CPU seconds. If a blank cell appears in a table, it indicates that none of the 10 instances could be completed within the time limit.

3.3.1 The multiobjective knapsack problem

The multiobjective knapsack problem is explained in Section 2.2. The multiobjective knapsack problem instances are generated for $p = 3$, 4 and 5 cases. The multiobjective knapsack problem consists of n objects, and v_r^j (nonnegative integer profits) and w_r (positive integer weights) are random integers drawn from the interval $[1, 1000]$ where $j = 1, \ldots, p$ and $r = 1, \ldots, n$. The capacity of the knapsack is calculated as $W = \left\lceil 0.5 \sum_{j=1}^{n} w_j \right\rceil$. For $p = 3$ case, we have generated 10 instances with a size varying from 10 to 100 with increments of 10. The number of models solved by each method is given in Table 3.1 along with other results.

As seen in Table 3.1, the number of nondominated solutions grows with the size of the problems. The difference in the number of solved models and the computation time is considerable among different methods. In particular, SOR-GNS outperforms the other methods in both comparison criteria. The difference between the performance of SOR-GNS and other methods increases with increasing size of the problem.

Table 3.1: Comparing the solution methods on the multiobjective knapsack problem with $p = 3$.

		Sylva and Crema		**Laumanns et al.**		**Özlen and Azizoglu**		**SOR-GNS**	
n	$\lvert\mathcal{Y}_N\rvert$	Model	Time	Model	Time	Model	Time	Model	Time
10	9.8	10.8	0.2	86.1	0.9	47.1	0.3	18.6	0.1
	[5, 25]	[6, 26]	[0.0, 1.0]	[28, 337]	[0.1, 5.2]	[16, 169]	[0.1, 1.3]	[9, 49]	[0.0, 0.5]
20	38.0	39.0	12.9	704.5	9.0	298.9	2.8	74.8	0.8
	[22, 63]	[23, 64]	[1.9, 46.1]	[305, 1429]	[2.8, 19.5]	[130, 718]	[1.1, 5.5]	[43, 124]	[0.3, 1.5]
30	115.8	**{1}**[1] 100.1	1423.7	6308.5	100.5	1380.3	20.1	230.0	3.7
	[62, 266]	[63, 197]	[106.3, 8904.4]	[1595, 24945]	[19.7, 353.9]	[423, 4053]	[3.9, 63.0]	[122, 528]	[1.4, 9.4]
40	311.2			41173.1	841.0	5590.9	126.7	617.2	17.7
	[136, 541]			[8552, 107716]	[130.8, 2273.6]	[1340, 13204]	[31.1, 313.8]	[270, 1067]	[6.0, 38.0]
50	444.2			72463.1	1778.8	10056.6	279.6	878.9	30.2
	[178, 803]			[10706, 221452]	[226.6, 5629.0]	[2758, 17059]	[48.5, 713.8]	[355, 1572]	[6.9, 69.0]
60	917.1			278030.4	6740.3	28050.3	1283.5	1815.6	101.2
	[550, 1283]			[109844, 468524]	[1570.4, 14716.1]	[13243, 50165]	[344.9, 2371.5]	[1091, 2539]	[38.8, 170.7]
70	1643.4			**{4}**[1] 406547.0	12699.1	57163.9	2705.2	3237.0	292.7
	[1019, 2632]			[290156, 601585]	[9335.7, 16835.8]	[22481, 115509]	[1012.1, 7103.9]	[2027, 5208]	[109.5, 756.2]
80	2295.8			**{9}**[1] 620336.0	18176.7	115394.6	4866.8	4532.6	437.5
	[1383, 3315]			[620336, 620336]	[18176.7, 18176.7]	[48588, 232648]	[1907.0, 10347.1]	[2733, 6543]	[234.8, 753.6]
90	3107.8			**{9}**[1] 278707.0	10405.9	**{1}**[1] 135282.0	10940.8	6125.9	936.4
	[1195, 5568]			[278707, 278707]	[10405.9, 10405.9]	[54212, 252885]	[2209.8, 20912.7]	[2373, 10967]	[131.1, 2754.6]
100	5849.0					**{5}**[1] 178531.6	10811.7	11536.2	3222.0
	[2193, 10179]					[78825, 254722]	[3524.8, 15985.1]	[4360, 20042]	[494.0, 8912.5]

[1]Number of instances that could not be solved in 25000s of CPU time.

For example, when $n = 100$, SOR-GNS is the only method that solves all 10 instances successfully. Another effectiveness criterion for the methods is the number of mathematical models solved to obtain one efficient solution. For SOR-GNS, the ratio of number of solved models and number nondominated solutions is 1.97 on average, i.e. SOR-GNS requires 1.97 model solutions per nondominated solution. Besides, in none of the instances does this ratio exceeds 1.99.

In addition to $p = 3$ knapsack instances, we test SOR-GNS for $p = 4$ and $p = 5$ to see how it behaves for larger p. As seen in Table 3.1, Sylva and Crema [174] and Laumanns et al. [122] methods cannot solve larger multiobjective knapsack problem instances even if $p = 3$. Hence, we solve $p = 4$ instances by using the recursive algorithm given in [144] and SOR-GNS. The four-objective knapsack problem results are given in Table 3.2. To the best of our knowledge, this is the first study that generates both supported and unsupported nondominated solutions for four-objective knapsack problems up to $n = 40$.

Table 3.2: Comparing the solution methods on the multiobjective knapsack problem with $p = 4$.

		Özlen and Azizoğlu		**SOR-GNS**	
n	$\lvert\mathcal{Y}_N\rvert$	Model	Time	Model	Time
10	11.6	273.0	0.6	40.1	0.2
	[5, 22]	[43, 734]	[0.0, 2.1]	[14, 82]	[0.0, 0.7]
20	136.8	37118.7	531.7	656.4	46.7
	[17, 325]	[552, 135400]	[2.1, 2287.4]	[56, 1743]	[0.4, 272.6]
30	397.6	**{3}**[1] 123314.1	4673.7	1990.9	988.1
	[186, 735]	[34871, 441368]	[630.0, 21630.9]	[817, 3689]	[34.0, 4479.2]
40	790.6			**{5}**[1] 3959.4	5263.0
	[508, 1248]			[2362, 6144]	[794.9, 16242.3]

[1]Number of instances that could not be solved in 25000s of CPU time.

For $p = 4$, the number of nondominated solutions more than doubles compared to

the $p = 3$ case. Moreover, the gap between the CPU times of the recursive algorithm and SOR-GNS widens. The number of models solved increases drastically for the recursive algorithm and none of the $n = 40$ instances can be solved.

We solve five-objective knapsack problem instances by using SOR-GNS only. For $p = 5$, instances become unmanageable beyond $n = 20$. For $n = 10$, the average number of nondominated solutions is 16.2, and the average CPU time is 1.0s. For $n = 20$, the average number of nondominated solutions is 161.2, and the average CPU time is 5084.1s while the maximum is 21304.4s. Another way to solve small-sized instances is to use enumeration. In this case, for $n = 20$ the average and maximum CPU time is 60.3 and 84.3, respectively. Although enumeration outperforms SOR-GNS for $n = 20$, it is also not applicable beyond $n = 20$.

Unlike generic multiobjective methods that have been referred to so far, there are some studies presented to solve only multiobjective knapsack problems. In one recent study, Bazgan et al. proposed an approach based on dynamic programming [17]. In another one, a labeling algorithm was used to find all nondominated outcomes in [83]. Although these two studies are problem specific algorithms, we make an informal comparison with respect to CPU time. Since their computational environment is different from ours, we scale the reported CPU times to our computation setting by using SPEC benchmark results [170]. Additionally, both studies generate their instances similar to our random generation setting with same sizes. In small-sized instances the performance of Bazgan et al.'s algorithm is better than our approach. The difference between CPU times decreases with the increase in instance size. When $n = 110$, SOR-GNS outperforms Bazgan's algorithm [17]. Figueira et al. solve knapsack instances up to 20 with $p = 3$ to 7 [83]. For $p = 3$ and $p = 4$, SOR-GNS outperforms Figueira et al.'s algorithm. However, for large values of p, their algorithm seems to perform better.

3.3.2 The multiobjective assignment problem

The multiobjective assignment problem is explained in Section 2.2. Test problem instances for the multiobjective assignment problem are formed in sizes varying from 5 to 50 with increments of 5 and p is set equal to 3. The objective function coefficients are randomly generated integers in the interval $[1, 20]$. The test problems are generated with the same parameters given in [152], and this is the only study that considers the multiobjective assignment problem for more than two objective functions. Przybylski et al. [152] generalize the two-phase method presented in [181] for the $p > 2$ case. The number of models solved and CPU times of each method for the multiobjective assignment problem instances are given in Table 3.3.

As seen in Table 3.3, the number of nondominated solutions is increasing more rapidly compared to knapsack instances. The performance difference between Özlen and Azizoglu's algorithm and SOR-GNS is not as much as it is for the knapsack problem. Nevertheless, the proposed method is almost twice as fast as Özlen and Azizoglu's algorithm in all instances. The number of problems solved per nondominated solution decreases to 1.45 on average compared to knapsack instances while the maximum ratio is 1.79.

As a side remark, the performance of Özlen and Azizoglu's algorithm is highly affected by the distance between y^I and y^U, due to its search mechanism. This distance is smaller in assignment instances compared to knapsack instances. Therefore, their algorithm has a poorer performance in knapsack instances, but SOR-GNS is not affected by the variations in objective function value ranges. More importantly, Özlen and Azizoglu's algorithm uses the augmented form of the ε-constraint method, and in their formulation while the coefficient of one of the objective functions is one, remaining objective coefficients are products of inverse of ranges. Hence coefficients may take values which are close to zero, and the resulting solution can be weakly efficient. In one knapsack instance, the total number of such points is found to be 91.

Table 3.3: Comparing solution methods on the multiobjective assignment problem for $p = 3$.

		Sylva and Crema		**Laumanns et al.**		**Özlen and Azizoglu**		**SOR-GNS**	
n	$\lvert\mathcal{Y}_N\rvert$	Model	Time	Model	Time	Model	Time	Model	Time
5	14.1	15.1	0.4	112.2	0.4	54.9	0.1	28.2	0.1
	[5, 23]	[6, 24]	[0.0, 1.4]	[24, 280]	[0.1, 1.4]	[20, 96]	[0.0, 0.3]	[12, 47]	[0.1, 0.3]
10	176.8	**{2}**[1] 154.6	6331.5	3376.8	68.2	1046.2	14.1	294.6	6.7
	[65, 316]	[66, 237]	[104.8, 21923.1]	[843, 5737]	[14.2, 115.5]	[250, 1796]	[3.6, 23.7]	[103, 525]	[2.1, 11.6]
15	674.9			13276.9	472.3	3522.0	94.9	1080.1	45.5
	[395, 1109]			[7811, 17625]	[277.3, 679.5]	[1885, 5510]	[44.5, 159.6]	[665, 1737]	[26.1, 80.1]
20	1860.5			29861.6	1778.3	7742.9	337.8	2742.2	184.5
	[889, 2520]			[17388, 36662]	[863.4, 2246.4]	[4917, 9753]	[215.8, 434.6]	[1356, 3651]	[88.9, 251.8]
25	3567.8			51592.7	4513.6	14399.2	939.5	5098.1	483.0
	[1968, 5385]			[36077, 65864]	[1856.7, 8551.1]	[9284, 19304]	[570.4, 1304.8]	[2912, 7598]	[268.2, 728.8]
30	6181.3			**{1}**[1] 74043.4	17939.2	21501.3	2024.1	8377.9	1136.4
	[5264, 8451]			[57316, 87718]	[9027.1, 23453.2]	[16625, 24231]	[1412.7, 2386.5]	[7144, 11430]	[904.0, 1565.3]
35	8972.3			**{9}**[1] 105978.0	22343.1	29963.0	3767.4	12067.5	2118.6
	[6621, 10862]			[105978, 105978]	[22343.1, 22343.1]	[22435, 34818]	[2763.4, 4616.8]	[9154, 14673]	[1527.7, 2624.7]
40	14679.7					42517.8	7155.7	19231.9	4244.7
	[10199, 18110]					[30539, 54838]	[4917.7, 9760.3]	[13279, 24397]	[3079.5, 5663.1]
45	17702.2					52326.6	11075.8	22919.0	6345.5
	[13899, 22659]					[43142, 60705]	[8867.5, 13531.5]	[18279, 29167]	[5009.7, 8202.6]
50	24916.8					66981.1	18677.7	31541.6	11516.9
	[19249, 28453]					[56582, 74170]	[14950.2, 21715.9]	[25092, 35608]	[8706.5, 13048.9]

[1]Number of instances that could not be solved in 25000s of CPU time.

This does not seem to happen in the assignment problem instances.

3.4. Conclusion

In this chapter, we have proposed an algorithm to solve multiobjective discrete optimization problems with any number of objective functions. We showed that the algorithm can generate the entire nondominated set in a finite number of steps. Our method uses the well-established ε-constraint scalarization and is based on a partitioning mechanism that searches the $(p-1)$-dimensional constraint space exhaustively. The proposed method is compared with previous algorithms, and is seen to outperform all of them on the experimented problem instances. The number of models solved per nondominated solution may be a better comparison criterion for the scalarization methods than the computation time statistics. The proposed algorithm solved at most 1.99 subproblems per nondominated solution on the test problems. This ratio is highly competitive compared to previous studies. Nevertheless, as problem size increases and the number of nondominated solutions grows, the requirements of the approach become unrealistically high. Hence, in Chapter 5, we modify the algorithm to generate representations of the nondominated set with desired quality guarantees.

Chapter 4

COMPUTING THE NADIR POINT FOR MULTIOBJECTIVE DISCRETE OPTIMIZATION PROBLEMS

In this chapter, we investigate the problem of finding the nadir point for MODO. We present an algorithm to compute nadir values for MODO with p objective functions. The nadir point is constructed from the worst objective values over the efficient set of a multiobjective optimization problem. We reviewed the solution methods that compute the nadir point for MOP problems in Section 2.4. These solution methods can be classified into two groups. In the first group, computation of the nadir point is considered as an optimization over the efficient set problem [1, 103]. The other group of methods require exhaustive search of the outcome space [71] or the weight space [5]. Our algorithm is based on exhaustive search of $(p-2)$-dimensional space which is the projection of the outcome space. We partition this space into $(p-2)$-dimensional rectangles to search the space entirely. During the search, a two-stage ε-constraint scalarization is used to obtain nondominated solutions. The method guarantees to find the nadir point in a finite number of iterations. We compare our algorithm with two earlier studies from the literature. We give numerical results for all algorithms on multiobjective knapsack, assignment and integer linear programming problems. Our algorithm is able to obtain the nadir point for relatively large problem instances with up to five-objectives. In the following section, we present the theoretical basis for our approach. In Section 4.2, our algorithm to compute the nadir point for MODO is given. In Section 4.3, computational results for our algorithm, Ehrgott

and Tenfelde-Podehl's algorithm [71], and Jorge's algorithm [103] are given on multiobjective knapsack, assignment and integer linear programming problem instances. Finally, conclusions are presented in Section 4.4.

4.1. Theoretical Background

For the presentation of our approach, we use an unattainable overestimator of the nadir point which we define as $y_j^U = \max_{x\in\mathcal{X}} f_j(x) + \delta$ for $j = 1, \ldots, p$ where $\delta > 0$. Clearly, $y_j^N < y_j^U$ for $j = 1, \ldots, p$.

The ε-constraint method suggests retaining one of the p objective functions as the objective function while remaining $(p-1)$ are turned into constraints [93]. The weakly efficient solutions thus found then can be eliminated by lexicographic, augmented or two-stage subproblems as recently implemented in [122, 144, 114]. Here, we propose a slight modification of the ε-constraint method for the purpose of seeking nondominated solutions that determine the k^{th} component of the nadir point. For any $\varepsilon \in \mathbb{R}^{p-2}$, two-stage ε-constraint formulations, $P_{k,m}(\varepsilon)$ and $Q_{k,m}(\varepsilon)$ for some $k, m \in \{1, \ldots, p\}$ and $k \neq m$, are defined as follows.

$$
\begin{aligned}
\mathbf{P_{k,m}}(\varepsilon) \quad \mathbf{z} = \min \quad & f_m(x) \\
\text{s.t.} \quad & f_k(x) \leq y_k^U \\
& f_j(x) \leq \varepsilon_j \quad j \in C \\
& x \in \mathcal{X}.
\end{aligned}
$$

Above, set C is defined as $C = \{1, \ldots, p\} \setminus \{k, m\}$. Let z^* be the optimal objective value of subproblem $P_{k,m}(\varepsilon)$ and consider the second stage formulation $Q_{k,m}(\varepsilon)$.

$$\begin{aligned} \mathbf{Q_{k,m}}(\varepsilon) \quad \min \quad & \sum_{j=1}^{p} f_j(x) \\ \text{s.t.} \quad & f_m(x) = z^* \\ & f_k(x) \leq y_k^U \\ & f_j(x) \leq \varepsilon_j \quad j \in C \\ & x \in \mathcal{X}. \end{aligned}$$

Let x^* be an optimal solution to two-stage formulations $P_{k,m}(\varepsilon)$ and $Q_{k,m}(\varepsilon)$ where $k, m \in \{1, \ldots, p\}$ and $k \neq m$. That x^* is always efficient for any $\varepsilon \in \mathbb{R}^{p-2}$ follows easily from earlier results [114]. Besides, whenever $f_k(x^*) = y_k^N$, there exist an $\varepsilon \in \mathbb{R}^{p-2}$ such that x^* is optimal to two-stage programs.

Theorem 10. *Let $k, m \in \{1, \ldots, p\}$ and $k \neq m$. For $\varepsilon \in \mathbb{R}^{p-2}$, an optimal solution to the two-stage formulations $P_{k,m}(\varepsilon)$ and $Q_{k,m}(\varepsilon)$ is efficient.*

Theorem 11. *Let $k, m \in \{1, \ldots, p\}$ and $k \neq m$. For an efficient solution x^* where $f_k(x^*) = y_k^N$, there exists an $\varepsilon \in \mathbb{R}^{p-2}$ such that x^* is an optimal solution to two-stage formulations $P_{k,m}(\varepsilon)$ and $Q_{k,m}(\varepsilon)$.*

Proof. Let $k, m \in \{1, \ldots, p\}$ and $k \neq m$. Suppose x^* is a feasible solution such that $f_k(x^*) = y_k^N$. Set $\hat{\varepsilon}_j = f_j(x^*)$ for each $j \in C$. Note that x^* is feasible to $P_{k,m}(\hat{\varepsilon})$. Suppose that x^* does not solve two-stage formulations $P_{k,m}(\hat{\varepsilon})$ and $Q_{k,m}(\hat{\varepsilon})$. Let x' be an optimal solution to two-stage programs with $\hat{\varepsilon} \in \mathbb{R}^{p-2}$. Then either $f_m(x') < f_m(x^*)$ or $f_m(x') = f_m(x^*)$ and $\sum_{j=1}^{p} f_j(x') < \sum_{j=1}^{p} f_j(x^*)$. Since $f_j(x') \leq \hat{\varepsilon}_j = f_j(x^*)$ for all $j \in \{1, \ldots, p\} \setminus \{k\}$ and $f_k(x') \leq f_k(x^*) = y_k^N$, $f_{\hat{\jmath}}(x') < f_{\hat{\jmath}}(x^*)$ for some $\hat{\jmath} \in \{1, \ldots, p\}$. This implies that x' dominates x^*, which contradicts that x^* is an efficient solution. □

Remark 1. *Theorem 11 shows that y_k^N, $k \in \{1, \ldots, p\}$, can be computed by using the two-stage formulation with $m \in \{1, \ldots, p\} \setminus \{k\}$ and $\varepsilon_j \in \mathbb{R}$ such that $j \in C$.*

Hence, the search space of this result, is the $(p-2)$-dimensional space, and has one less dimension than the search space of enumerating all nondominated solutions by using ε-constraint method [114]. In terms of dimension of the problem, this theorem is similar to Ehrgott and Tenfelde-Podehl's result [71] since they find the nadir point by generating efficient sets of p different MOPs with $(p-1)$ objectives. On the other hand, their result relies on the union of efficient sets, while Theorem 11 determines the nadir point component-wise. Later on, we remove some portion of the initial search space by using the payoff table estimate.

By Theorems 10 and 11, with appropriate choice of $\varepsilon \in \mathbb{R}^{p-2}$ the k^{th} component of the nadir point can be obtained by solving the two-stage formulations. To facilitate a search for the appropriate ε values, we can partition the search space into $(p-2)$-dimensional rectangles. A similar search was implemented in the previous chapter with $(p-1)$-dimensional rectangles to obtain the entire nondominated set. Lower and upper vertices of a rectangle are denoted as $l \in \mathbb{R}^{p-2}$ and $u \in \mathbb{R}^{p-2}$, respectively. With these bounds, a rectangle is defined as $R(l,u) = \{\bar{y} \in \mathbb{R}^{p-2} : l \leq \bar{y} \leq u\}$. For ease of notation, we define $\bar{y} = \bar{f}(x) \in \mathbb{R}^{p-2}$ such that $\bar{f}_j(x) = f_j(x)$ for all $j \in C$, i.e. $\bar{y} = \bar{f}(x)$ is the projection of $f(x) \in \mathbb{R}^p$ onto $\mathbb{R}^{p-2}$ with the m^{th} and k^{th} components removed. Define a rectangle $R(\bar{y}^I, \bar{y}^U)$ whose lower and upper vertices are projections of y^I and y^U onto $\mathbb{R}^{p-2}$, respectively. By definition of y^I and y^U, $y^I_j \leq f_j(x) < y^U_j$ for each $j \in \{1,\ldots,p\}$ and for all $x \in \mathcal{X}$. Hence, all efficient solutions are mapped into $R(\bar{y}^I, \bar{y}^U)$.

The idea is to start with a sufficiently large rectangle that contains the projection of the nondominated set, devise a method of refining the rectangles and solve two-stage problems repeatedly. We will now describe how payoff table information can be used to narrow down the search space.

Let $m \in \{1,\ldots,p\}$ be such that $T_{mk} = y^{PT}_k$, i.e. m is the row that generates the k^{th} component of y^{PT}. Let x^*_m denote an efficient solution that maps into the m^{th} row of the payoff table. Define $\varphi = \bar{f}(x^*_m) \in \mathbb{R}^{p-2}$ where $\varphi_j = f_j(x^*_m)$ for each $j \in C$,

i.e. φ is the projection of $f(x^*_m) \in \mathbb{R}^p$ onto $\mathbb{R}^{p-2}$.

Our next result essentially states that $R(\varphi, \bar{y}^U)$ can be excluded from the search space. This reduces our search space to a difference of two rectangles. We now argue that our search space can be represented as a union of several rectangles because $R(\bar{y}^I, \bar{y}^U)$ can be divided into several non-overlapping (in the sense that the interiors do not intersect) subrectangles by pivoting on $\varphi \in \mathbb{R}^{p-2}$. Let $C' = \{j \in C : \bar{y}^I_j < \varphi_j\}$. Consider a rectangular bisection process in which a rectangle is subdivided into two subrectangles by means of a hyperplane $H_j = \{\bar{y} \in \mathbb{R}^{p-2} : \bar{y}_j = \varphi_j\}$ where $j \in C'$. When such a process is applied repeatedly, it generates a family of rectangles which are partitions of the original rectangle [100, 123]. An element of this family is of the form

$$\begin{aligned} R^J = \{\bar{y} \in \mathbb{R}^{p-2} : \bar{y}^I_j \leq \bar{y}_j \leq \varphi_j \;\; j \in J \\ \varphi_j \leq \bar{y}_j \leq \bar{y}^U_j \;\; j \in C \setminus J\} \end{aligned} \tag{4.1}$$

where $J \subseteq C'$. Therefore, union of all such rectangles constitute the initial search space, i.e. $R(\bar{y}^I, \bar{y}^U) = \bigcup_{J \subseteq C'} R^J$. The rectangle $R(\varphi, \bar{y}^U)$ is computed when $J = \emptyset$ in (4.1). Then the initial search space U_R, given by

$$U_R = \bigcup_{\substack{J \subseteq C' \\ J \neq \emptyset}} R^J \tag{4.2}$$

contains partitions of $R(\bar{y}^I, \bar{y}^U)$ except for $R(\varphi, \bar{y}^U)$.

The result below shows that the nadir point y^N is projected into U_R where $U_R \subseteq R(\bar{y}^I, \bar{y}^U) \subseteq \mathbb{R}^{p-2}$.

Theorem 12. *Let $m \in \{1, \ldots, p\}$ be such that $T_{mk} = y^{PT}_k$. Let $x^*_m \in \mathcal{X}_E$ be such that $f_j(x^*_m) = T_{mj}$ for each $j \in \{1, \ldots, p\}$. Define $\varphi = \bar{f}(x^*_m) \in \mathbb{R}^{p-2}$. For an efficient solution x' where $f_k(x') = y^N_k$, $\bar{f}(x') \in U_R$.*

Proof. Let $x' \in \mathcal{X}$ be an efficient solution such that $f_k(x') = y^N_k$. Assume to the contrary that $f(x')$ is projected outside of U_R. By definition of y^I and y^U, $\bar{f}(x) \in$

$R(\bar{y}^I, \bar{y}^U)$ for all $x \in \mathcal{X}$. This implies that $\bar{f}(x') \in R(\varphi, \bar{y}^U)$. Then $\varphi_j = f_j(x^*_m) \leq f_j(x') \leq y^U_j$ for all $j \in C$. Since $\varphi \in R(\bar{y}^I, \varphi) \subseteq U_R$ and $\bar{f}(x') \notin U_R$, there exists $\hat{\jmath} \in C$ such that $f_{\hat{\jmath}}(x^*_m) < f_{\hat{\jmath}}(x')$. By definition of ideal and nadir point, $f_m(x^*_m) = y^I_m \leq f_m(x')$ and $f_k(x^*_m) = y^{PT}_k \leq f_k(x') = y^N_k$. Since $f_j(x^*_m) \leq f_j(x')$ for all $j \in C$ with at least one strict inequality, x^*_m dominates x'. This contradicts efficiency of x'. □

By Theorems 10, 11 and 12, the k^{th} component of the nadir point (y^N_k) can be obtained by an exhaustive search of the region U_R. The search space is represented by a union of $(p-2)$-dimensional rectangles. Unless the number of objective functions is less than or equal to 3, the search space is non-convex. For $p = 2$, the search space is a point. This means that y^{PT} is equal to y^N, which is a known fact. When $p = 3$, the search space becomes a single dimensional interval for the k^{th} component of the nadir point.

In the next section, we provide an algorithm that conducts this search for MODO problems. While results presented so far are applicable to any MOP, our search procedure requires $\mathcal{Y}_N$ to be a discrete set.

4.2. Finding the Nadir Point for a MODO Problem

In this section, we present an algorithm to search the region U_R exhaustively for all nondominated solutions that map into this region. Since $\varepsilon \in \mathbb{R}^{p-2}$, the search is managed over $(p-2)$-dimensional rectangles. During the search, a set of rectangles should be maintained. The i^{th} rectangle is denoted as R_i with its lower and upper vertices $l^i, u^i \in \mathbb{R}^{p-2}$. All rectangles that need to be searched are kept in list L, i.e. $L = \bigcup_{i=1}^{|L|} \{R_i\}$ where $|L|$ represents the cardinality of L.

Initially, we obtain the efficient solution x^*_m by using the payoff table, where $f_k(x^*_m) = y^{PT}_k$ and $f_m(x^*_m) = y^I_m$. Given objective function indices $\{k, m\}$ and x^*_m, the search space U_R can be constructed by using (4.1) and (4.2), and L is initialized

as U_R.

After the initialization phase, $L = \emptyset$ implies that y^{PT} is the nadir point. If L is a non-empty set, than we need to process the rectangles in an order. For prioritization, we define a volume-related measure associated with rectangle R_i as

$$V_i = \prod_{j \in C} (u_j^i - \bar{y}_j^I). \tag{4.3}$$

Note that this is not the volume of R_i itself, but the volume of the rectangle defined by $\bar{y}^I$ as the lower vertex and the upper vertex u^i of R_i. Theoretically, this prioritization rule can be replaced by any other list processing rule.

The technical statement of the proposed algorithm, which we label as Nadir Point Determination Algorithm (NPDA), is as follows.

Algorithm NPDA

Input: MODO problem, i.e. objective functions $f_j(x)$ for $j = \{1, \ldots, p\}$, and feasible set $\mathcal{X}$

Output: Nadir point (y^N) of MODO problem.

Step-0 Obtain the payoff table T, and initialize the nadir point with the payoff table estimate, i.e. $y_j^N = y_j^{PT}$ for $j = 1, \ldots, p$. Set $k = 1$.

Step-1 Initialize the nondominated solutions list, $\mathcal{Y}_P^k$[3]$= \emptyset$. Pick $m \in \{1, \ldots, p\}$ and efficient solution x_m^* such that $f_k(x_m^*) = y_k^{PT}$. Generate the initial rectangle list L by using (4.1) and (4.2) where $\varphi_j = f_j(x_m^*)$ for each $j \in C$.

Step-2 If L is empty, go to Step-4. Otherwise, pick a rectangle $R_i(l^i, u^i)$ with highest V_i from the list L. Solve the first stage formulation with the upper vertex of R_i, i.e. $P_{k,m}(u^i)$.

[3] $\mathcal{Y}_P^k$ represents the list of nondominated solutions until y_k^N is obtained. At termination, $\mathcal{Y}_P^k$ may not include all nondominated solutions, i.e. $\mathcal{Y}_P^k \subseteq \mathcal{Y}_N$ for each $k \in \{1, \ldots, p\}$.

Step-3 If $P_{k,m}(u^i)$ is feasible, than solve $Q_{k,m}(u^i)$. Let x^* denote an optimal solution.

- If $f(x^*) \notin \mathcal{Y}_P^k$, $\mathcal{Y}_P^k = \mathcal{Y}_P^k \cup \{f(x^*)\}$. If $f_k(x^*) > y_k^N$, update the k^{th} component of the nadir point, $y_k^N = f_k(x^*)$. Now, apply the rectangular subdivision process for each $R_s \in L$. Pick rectangle R_s from list L. Set $L = L \setminus \{R_s\}$. Define an index set for R_s as $C'_s = \{j \in C : l_j^s < \bar{f}_j(x^*) < u_j^s\}$. Initialize list L' for the rectangular subdivision process, $L' = \{R_s\}$.

  For Each $(j \in C'_s)$

  Initialize list L'', $L'' = \emptyset$.

  For Each $(R_t \in L')$

  $R_1 = \{\bar{y} \in R_t : \bar{y}_j \leq f_j(x^*)\}$

  $R_2 = \{\bar{y} \in R_t : \bar{y}_j \geq f_j(x^*)\}$

  $L'' = L'' \cup \{R_1\} \cup \{R_2\}$

  end

  $L' = L''$

  end

  $L = L \cup L'$. Remove rectangles that lie in $R(\bar{f}(x^*), u^i)$.

- Else (If $f(x^*) \in \mathcal{Y}_P^k$), remove rectangles that lie in $R(\bar{f}(x^*), u^i)$.

Else (If $P_{k,m}(u^i)$ is infeasible), remove rectangles that lie in $R(\bar{y}^I, u^i)$.

Go to Step-2.

Step-4 $k = k + 1$. If $k \leq p$, go to Step-1. Otherwise, return the nadir point y^N and stop.

NPDA starts with obtaining the payoff table and an efficient solution that corresponds to each component of y^{PT}. Then initial rectangle list L is generated. In each iteration, the algorithm picks a rectangle R_i with the largest V_i value. With the upper vertex of R_i, $u^i \in \mathbb{R}^{p-2}$, the two-stage programs are solved. Until termination,

the k^{th} component of the nadir point is determined. This procedure is repeated for each $k \in \{1, \ldots, p\}$ to obtain the nadir point y^N.

While solving the two-stage formulations with $u^i \in \mathbb{R}^{p-2}$, three cases can occur. First a new nondominated solution may be obtained, and let x^* be an associated efficient solution. Initially, $f(x^*)$ is inserted into nondominated solutions list $\mathcal{Y}_P^k$. Then, among all rectangles in L, if $\bar{f}_j(x^*)$ is in between the bounds of rectangle R_s for the j^{th} axis, then R_s is split into two rectangles along the j^{th} axis where $j \in C$.

After updating the list, some of the rectangles can be removed because the region between the upper vertex of the rectangle R_i and $\bar{f}(x)$ does not need to be searched (see Lemma 3). In this case, all rectangles that lie in $R(\bar{f}(x^*), u^i)$ are removed, i.e. for any $R_s \in L$ such that $R_s \subseteq R(\bar{f}(x^*), u^i)$, $L = L \setminus \{R_s\}$.

Lemma 3. *Let $k, m \in \{1, \ldots, p\}$ and $k \neq m$, and let x^* be an optimal solution to two-stage programs $P_{k,m}(u^i)$ and $Q_{k,m}(u^i)$ where $u^i \in \mathbb{R}^{p-2}$ is the upper vertex of a rectangle. Then, there is no nondominated solution that is projected into $R(\bar{f}(x^*), u^i)$, other than $f(x^*)$.*

Proof. Assume to the contrary that there exists an efficient solution $x' \in \mathcal{X}$ that is mapped into the rectangle $R(\bar{f}(x^*), u^i)$. Then, $\bar{f}_j(x^*) \leq \bar{f}_j(x') \leq u_j^i$ for each $j \in C$, and $f_k(x') \leq y_k^U$. This implies that x' is feasible to $P_{k,m}(u^i)$, and we know that x^* is optimal to $P_{k,m}(u^i)$. Then, $f_m(x^*) \leq f_m(x')$. Unless $f_j(x') = f_j(x^*)$ for all $j = 1, \ldots, p$, x^* dominates x'. Therefore, there exists no nondominated solution that is mapped into the rectangle $R(\bar{f}(x^*), u^i)$ other than $f(x^*)$. □

The second possible case after solving a set of two-stage programs in NPDA is that an optimal solution x^* is obtained, however $f(x^*)$ has already been found at an earlier iteration, i.e. $f(x^*) \in \mathcal{Y}_P^k$. The reason for this situation is that different $\varepsilon \in \mathbb{R}^{p-2}$ values may return the same nondominated solution. In this case, Lemma 3 still applies, and rectangles that lie in between $\bar{f}(x^*)$ and current rectangle's upper vertex u^i are removed.

The third possible case is the infeasibility of the two-stage programs. Given upper vertex u^i for the rectangle, if $P_{k,m}(u^i)$ is feasible, then $Q_{k,m}(u^i)$ is also feasible. Therefore, infeasibility can be observed only in the first stage problem. If the first stage is infeasible for the rectangle's upper vertex u^i, every $\varepsilon \in R(\bar{y}^I, u^i)$ would lead to an infeasible $P_{k,m}(\varepsilon)$ formulation. Hence, all rectangles inside $R(\bar{y}^I, u^i)$ are removed. Lemma 4 addresses the infeasibility case.

Lemma 4. *Let $k, m \in \{1, \ldots, p\}$ and $k \neq m$, if $P_{k,m}(u^i)$ is infeasible, then there is no nondominated solution that is projected into the rectangle $R(\bar{y}^I, u^i)$.*

Proof. Assume to the contrary that there exists an efficient solution $x' \in \mathcal{X}$ such that $f_j(x') \leq u^i_j$ for each $j \in C$, and $y_k(x') \leq y^U_k$. This implies that x' is feasible to $P_{k,m}(u^i)$ which contradicts that $P_{k,m}(u^i)$ has no feasible solution. Therefore there exists no nondominated solution that is mapped into $R(\bar{y}^I, u^i)$. □

We have shown that in each iteration the algorithm removes at least one rectangle from list L, and no other nondominated solution projects into removed rectangles. Similar proofs are given in [114]. Theorems 13 and 14 will show completeness and finiteness of the algorithm after an illustrative example on building and managing rectangles.

Example. Consider a four-objective problem ($p = 4$) where the third component ($k = 3$) of the nadir point, y^N_3, is being computed. Let the third component of the payoff table estimate be obtained with $x^*_4 \in \mathcal{X}_E$, i.e. $m = 4$, $f_3(x^*_4) = y^{PT}_3$ and $f_4(x^*_4) = y^I_4$. Since $k = 3$ and $m = 4$, the index set C is equal to $\{1, 2\}$. The projection of $f(x^*_4)$ onto $\bar{y}_1 - \bar{y}_2$ plane is denoted as φ, and suppose that $\bar{y}^I_j < \varphi_j$ for all $j \in C$. Hence, $C' = \{1, 2\}$. The initial search space that contains 3 rectangles is $U_R = R((\bar{y}^I_1, \varphi_2), (\varphi_1, \bar{y}^U_2)) \cup R((\varphi_1, \bar{y}^I_2), (\bar{y}^U_1, \varphi_2)) \cup R((\bar{y}^I_1, \bar{y}^I_2), (\varphi_1, \varphi_2))$ as shown in Figure 4.1. In this figure shaded region shows the removed space, which is $R((\varphi_1, \varphi_2), (\bar{y}^U_1, \bar{y}^U_2))$.

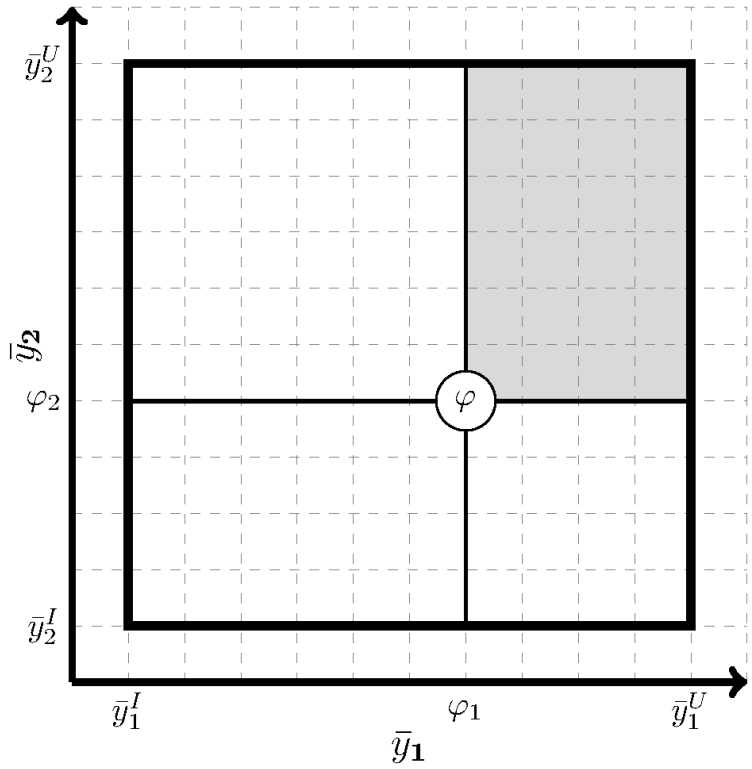


Figure 4.1: Initial search space and the portion removed by using payoff estimate.

After the initialization phase, the procedure continues by picking $R((\bar{y}_1^I, \varphi_2), (\varphi_1, \bar{y}_2^U))$. Two-stage formulation is solved for the upper vertex of this rectangle, and the projection of the resulting nondominated solution is $\bar{y}^1$. Rectangular subdivision process pivoting on $\bar{y}^1$ is shown in Figure 4.2. After the rectangular subdivision process, rectangles that lie in $R((\bar{y}_1^1, \bar{y}_2^1), (\varphi_1, \bar{y}_2^U))$ can be removed. Thus two rectangles, $R((\bar{y}_1^1, \varphi_2), (\varphi_1, \bar{y}_2^U))$ and $R((\bar{y}_1^1, \bar{y}_2^1), (\varphi_1, \varphi_2))$, are removed.

Theorem 13. *NPDA finds the nadir point y^N.*

Proof. Two-stage mathematical programs either find an efficient solution (see Theorem 10) or they are infeasible for right-hand-side vector u^i. Therefore, all points in $\mathcal{Y}_P^k$ for each $k \in \{1, \ldots, p\}$, that are obtained during the search are nondominated. In NPDA, the search is initialized with a set of rectangles that forms the search space

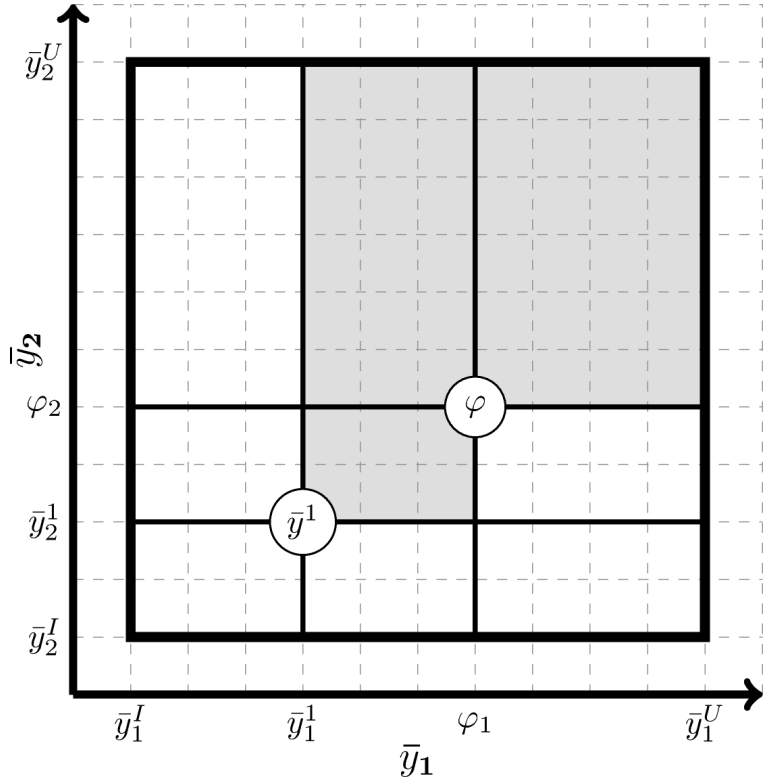


Figure 4.2: Rectangular subdivision process pivoting on $\bar{y}^1$ and removed rectangles.

U_R. By Theorem 12, an efficient solution x^* that satisfies $f_k(x^*) = y_k^N$ is mapped into this region. In NPDA, this space is searched exhaustively, and in each iteration some part of the initial rectangle is removed. From Lemma 3 and 4, removed regions do not include projection of a nondominated solution. By Theorem 11, there exists $\varepsilon \in \mathbb{R}^{p-2}$ such that x^* is optimal to a two-stage formulation whenever $f_k(x^*) = y_k^N$. Since NPDA considers each $k \in \{1, \ldots, p\}$, at termination NPDA obtains the nadir point y^N. □

We also need to argue the finiteness of NPDA. In Theorem 14, we show that the proposed algorithm terminates in a finite number of iterations.

Theorem 14. *NPDA is finite.*

Proof. By assumption, the nondominated set $\mathcal{Y}_N$ includes a finite number of elements. Since the list of nondominated solutions until y_k^N is obtained ($\mathcal{Y}_P^k$) is a subset of $\mathcal{Y}_N$, $\mathcal{Y}_P^k$ also has a finite number of elements, $|\mathcal{Y}_P^k|$. In the worst case, each axis in $\mathbb{R}^{p-2}$ is divided into $(|\mathcal{Y}_P^k|+1)$ intervals. Since the search space is $(p-2)$-dimensional, $|L| \leq (|\mathcal{Y}_P^k|+1)^{p-2}$ where $|L|$ denotes the size of the rectangle list. This means that the number of rectangles is finite. Besides, in each iteration at least one rectangle is removed from L. This implies that L will be empty in a finite number of iterations. Hence, NPDA is finite. □

The algorithm terminates when the list of rectangles is empty. Theorems 13 and 14 imply that NPDA finds the nadir point y^N in a finite number of iterations.

4.3. Computational Results

NPDA, Ehrgott and Tenfelde-Podehl's algorithm (ETPA) [71], and Jorge's algorithm (JA) [103] are tested on various MODO problem instances. Ehrgott and Tenfelde-Podehl showed that it suffices to generate efficient sets of p different MODOs with $(p-1)$ objective functions to compute the nadir point. Hence, their method requires an algorithm to enumerate all efficient solutions for MODO with $(p-1)$ objective functions. Therefore, we modify SOR-GNS (see Chapter 3) so as to generate the entire efficient set by finding all alternative optimal solutions for each subproblem. We choose SOR-GNS, because it performs well in terms of solution time [114], and utilizes a search structure similar to the one presented here, making a comparison more straight forward in terms of highlighting the additional benefits of NPDA. In the computational results, ETPA [71] with modified SOR-GNS is denoted as ETPA+. The proposed method is also compared with a recent solution algorithm for the problem of optimizing a linear function over the integer efficient set, which is presented by Jorge in [103]. JA iteratively cuts previously obtained nondominated solutions until the k^{th} component of the nadir point is obtained. Hence, the solution algorithm adds

p binary variables and $p+1$ constraints for each nondominated solution to the model.

All algorithms are implemented in C++. Subproblems are solved by using IBM CPLEX 12.4 [47]. All tests were conducted on a shared cluster with Intel Xeon 2.3 GHz CPU and 4 GB memory limit with Linux operating system. These algorithms are tested on the multiobjective knapsack problem, multiobjective assignment problem, and multiobjective integer linear programming problem instances. We use the same multiobjective knapsack problem and multiobjective assignment problem instances that were generated to test SOR-GNS in Chapter 3. Test problems have different categories based on problem size. There are 10 instances for each problem category. The results of each category are presented in two rows. In the first row, for each category, the average over 10 instances is given. The numbers in the second row correspond to the minimum and maximum for each category, respectively. Computation of each instance is interrupted after 100,000 CPU seconds. A blank cell in a table indicates that none of the 10 instances could be completed within the time limit.

For NPDA, we report total size of the list $\mathcal{Y}_P^k$ over all components of the nadir point, i.e. $\sum_{k=1}^{p} |\mathcal{Y}_P^k|$ is reported. Number of models solved and CPU time statistics are given for all algorithms.

4.3.1 The multiobjective knapsack problem

For the multiobjective optimization problem with $p = 3$, there are 10 categories with a size varying from 10 to 100 with increments of 10. The solution statistics are given in Table 4.1.

As seen in Table 4.1, NPDA outperforms both algorithms in terms of CPU time results. We observe that JA cannot solve all instances in the given time limit. When we compare NPDA with ETPA+, the number of solved models in each problem category for both methods are almost equal, but ETPA+ enumerates all alternative solutions for the given subproblem. In all instances, NPDA is able to obtain the nadir point in a shorter period of time.

Table 4.1: Comparing the solution methods on the multiobjective knapsack problem with $p = 3$.

	ETPA+		**JA**		**NPDA**		
n	#Models	CPU Time (s)	#Models	CPU Time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU Time (s)
10	15.9	0.1	9.9	0.2	9.9	12.9	0.1
	[10, 25]	[0.0, 0.1]	[8, 12]	[0.1, 0.2]	[4, 19]	[7, 22]	[0.0, 0.1]
20	35.9	0.4	23.1	1.4	30.0	33.0	0.2
	[28, 44]	[0.3, 0.4]	[18, 32]	[0.9, 2.8]	[22, 38]	[25, 41]	[0.1, 0.3]
30	61.0	1.0	39.4	8.4	55.1	58.1	0.5
	[46, 80]	[0.7, 1.4]	[22, 72]	[2.3, 28.7]	[40, 74]	[43, 77]	[0.3, 0.7]
40	104.7	3.4	70.8	57.6	99.0	102.0	1.3
	[77, 169]	[1.9, 7.5]	[42, 130]	[9.2, 253.5]	[71, 163]	[74, 166]	[0.8, 2.6]
50	138.7	6.4	89.2	132.5	133.1	136.1	2.3
	[82, 201]	[2.2, 15.7]	[42, 173]	[12.1, 625.5]	[76, 196]	[79, 199]	[0.9, 5.2]
60	188.6	12.9	154.2	830.8	183.3	186.3	3.8
	[153, 236]	[7.4, 23.8]	[110, 218]	[136.3, 2411.7]	[147, 232]	[150, 235]	[2.4, 6.0]
70	241.1	27.4	194.7	1782.2	237.6	240.6	6.5
	[194, 295]	[16.0, 34.9]	[119, 333]	[176.7, 5370.0]	[190, 302]	[193, 305]	[4.3, 8.2]
80	329.2	51.5	238.7	3993.1	327.2	330.2	11.9
	[215, 431]	[27.1, 76.9]	[135, 378]	[230.7, 16912.9]	[212, 427]	[215, 430]	[7.5, 16.7]
90	362.5	62.0	298.2	7976.8	359.1	362.1	13.9
	[258, 457]	[32.0, 92.0]	[137, 472]	[178.4, 24084.0]	[253, 456]	[256, 459]	[7.4, 21.1]
100	470.5	103.7	405.3	**{3}**[1] 24052.6	468.8	471.8	22.8
	[330, 651]	[59.3, 181.7]	[205, 613]	[1205.2, 63326.5]	[326, 658]	[329, 661]	[12.9, 39.8]

[1]Number of instances that could not be solved in 100,000 CPU seconds.

For $p = 4$ and $p = 5$ cases, we have generated 4 and 3 categories with a size varying from 10 to 40 and 10 to 30 with increments of 10, respectively. The solution statistics for both cases are given in Table 4.2.

For $p = 4$ case, NPDA still outperforms other two methods. In these instances, the solution time of NPDA and ETPA+ are in reasonable limits. This is not the case for JA. When we switch to $p = 5$ test instances, JA outperforms NPDA with the difference being important for $n = 30$. However, NPDA still outperforms ETPA+ for both $p = 4$ and $p = 5$ cases.

4.3.2 The multiobjective assignment problem

For the multiobjective assignment problem with $p = 3$, there are 10 categories with a size varying from 5 to 50 with increments of 5. The solution statistics are given in Table 4.3.

As seen in Table 4.3, NPDA computes the nadir point faster than ETPA+, and the performance difference is increasing with the increase in problem size. Another observation is that CPU time difference between NPDA and ETPA+ is larger in assignment problem instances compared to knapsack problem instances with $p = 3$. JA spends too much time to find the nadir on multiobjective assignment problems, and the algorithm cannot solve more than half of the instances in 100,000 CPU seconds. The assignment problem instances have larger number of nondominated solutions compared to knapsack problem instances (see [114] for the statistics). Hence, JA requires to add many binary variables and constraints to the model, and the model becomes unsolvable even if the problem size is $n = 25$.

We note that similar multiobjective knapsack and assignment problem instances were generated and their nondominated sets were enumerated in [114]. While the average number of nondominated solutions for the largest three-objective knapsack ($p = 3$ and $n = 100$) and assignment ($p = 3$ and $n = 50$) problem instances is 5849.0 and 24916.8, NPDA has $\sum_{k=1}^{p} |\mathcal{Y}_P^k|$ equal to 470 and 700.8, respectively. This implies

Table 4.2: Comparing the solution methods on the multiobjective knapsack problem with $p = 4$ and $p = 5$.

		ETPA+		**JA**		**NPDA**		
	n	#Models	CPU Time (s)	#Models	CPU Time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU Time (s)
$p = 4$	10	59.8	0.3	14.7	0.2	26.0	55.8	0.2
		[32, 94]	[0.1, 0.6]	[12, 21]	[0.1, 0.4]	[12, 43]	[28, 90]	[0.1, 0.4]
	20	407.9	7.4	40.5	7.1	200.7	404.4	4.6
		[90, 729]	[0.9, 14.1]	[17, 58]	[0.7, 20.9]	[41, 361]	[86, 723]	[0.7, 9.0]
	30	942.0	27.9	93.3	104.2	469.3	938.5	15.2
		[554, 1526]	[11.8, 51.5]	[57, 134]	[19.5, 236.8]	[274, 762]	[549, 1522]	[6.4, 27.7]
	40	2898.4	230.1	179.7	2641.5	1459.5	2903.5	98.2
		[1368, 6055]	[55.0, 615.8]	[106, 287]	[105.9, 10259.5]	[689, 3066]	[1371, 6096]	[24.5, 259.3]
$p = 5$	10	230.1	1.3	21.7	0.4	56.4	224.8	0.9
		[86, 454]	[0.2, 3.1]	[13, 25]	[0.2, 0.6]	[21, 108]	[81, 446]	[0.1, 2.3]
	20	1952.7	75.5	56.0	17.3	424.8	1948.2	60.0
		[862, 4770]	[13.1, 326.9]	[40, 73]	[3.8, 67.1]	[210, 958]	[857, 4769]	[9.5, 284.0]
	30	12630.0	20064.8	164.4	2353.6	2423.5	12648.9	18593.9
		[1617, 23718]	[44.8, 74862.9]	[75, 233]	[60.4, 9333.0]	[369, 4407]	[1614, 23754]	[26.6, 71104.3]

Table 4.3: Comparing the solution methods on the multiobjective assignment problem with $p = 3$.

	ETPA+		**JA**		**NPDA**		
n	#Models	CPU Time (s)	#Models	CPU Time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU Time (s)
5	16.4	0.1	45.0	17.0	10.8	13.8	0.1
	[11, 25]	[0.0, 0.1]	[30, 61]	[5.9, 26.1]	[5, 19]	[8, 22]	[0.0, 0.1]
10	60.2	1.2	145.0	491.4	62.5	65.5	0.9
	[38, 79]	[0.7, 1.8]	[101, 188]	[199.6, 945.9]	[41, 80]	[44, 83]	[0.6, 1.1]
15	103.9	5.0	376.5	7231.3	118.1	121.1	2.8
	[95, 120]	[4.5, 5.9]	[212, 505]	[1685.2, 12507.4]	[104, 140]	[107, 143]	[2.3, 3.6]
20	169.0	15.2	376.5	7315.1	202.9	205.9	7.2
	[129, 206]	[10.8, 19.0]	[212, 505]	[1716.8, 13093.7]	[155, 255]	[158, 258]	[5.6, 9.4]
25	199.3	33.0	644.3	**{4}**[1] 45850.1	248.4	251.4	13.2
	[164, 231]	[26.0, 38.6]	[458, 806]	[20841.2, 80308.3]	[195, 291]	[198, 294]	[10.3, 15.6]
30	268.9	67.5			370.4	373.4	26.2
	[241, 311]	[59.8, 80.0]			[330, 417]	[333, 420]	[23.4, 29.6]
35	312.3	124.5			438.5	441.5	40.5
	[282, 346]	[100.7, 139.4]			[386, 524]	[389, 527]	[33.7, 49.3]
40	359.5	205.8			536.1	539.1	51.6
	[329, 404]	[189.8, 221.9]			[462, 616]	[465, 619]	[44.8, 56.6]
45	390.0	321.4			602.9	605.9	74.1
	[354, 421]	[289.8, 349.8]			[516, 713]	[519, 716]	[65.3, 91.1]
50	427.9	507.8			700.8	703.8	105.1
	[399, 442]	[458.1, 541.9]			[649, 783]	[652, 786]	[86.9, 116.3]

[1]Number of instances that could not be solved in 100,000 CPU seconds.

that NPDA enumerates a small portion of $\mathcal{Y}_N$ in the process.

4.3.3 The multiobjective integer linear problem

We test the algorithms on general MOILP problem instances. The MOILP problem is explained in Section 2.2. In MOILP, m and n represent the number of constraints and number of variables. We consider MOILP problems with $p = 3, 4$ and 5, $m = 5, 10, \ldots, 50$ and $n = 2m$ for each m. The parameters of the model are randomly generated integer numbers with ranges similar to used in [5]. The coefficients of the objective functions (c_l^j) are generated in the ranges $[-100, -1]$ and $[0, 100]$ with probability 0.2 and 0.8, respectively. The technical coefficients (a_{rl}) are generated in the ranges $[-100, -1]$ with probability 0.1, $[1, 100]$ with probability 0.8, and $a_{rl} = 0$ with probability 0.1. Finally, right-hand side value (b_r) of each constraint is also generated randomly in the range of 100 and $\sum_{l=1}^{n} a_{rl}$. According to this scheme, it is possible for a generated MOILP instance to have an unbounded efficient set. Only one such instance was encountered in the $p = 5$ category and was discarded as reported in Table 4.6.

For $p = 3$ case, we test all methods with up to 50 constraints and 100 decision variables. The results of these tests are given in Table 4.4.

As seen in Table 4.4, the performance of NPDA and ETPA+ are closer compared to knapsack and assignment instances. Nevertheless, NPDA performs better than both methods in all instances. Additionally, NPDA solves all instances in an acceptable period of time even if the problem size increases to 50×100. For $p = 4$ case, we test all methods with up to 80 decision variables. The results of these tests are given in Table 4.5.

For $p = 4$ case, one out of 10 instances in the 40×80 category cannot be solved by NPDA in the given time limit. In this category, JA outperforms NPDA in small-sized instances. However, NPDA determines the nadir points faster than JA in most of the medium-sized instances.

Table 4.4: Comparing the solution methods on the MOILP problems with $p = 3$.

	ETPA+		**JA**		**NPDA**		
$m \times n$	#Models	CPU Time (s)	#Models	CPU Time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU Time (s)
5×10	30.7	0.4	11.5	0.2	25.2	28.2	0.3
	[8, 83]	[0.1, 1.1]	[4, 16]	[0.0, 0.4]	[2, 80]	[5, 83]	[0.0, 0.7]
10×20	30.9	1.2	16.7	0.8	25.1	28.1	0.6
	[15, 51]	[0.3, 2.9]	[8, 35]	[0.2, 2.1]	[9, 46]	[12, 49]	[0.2, 1.1]
15×30	47.5	11.4	26.3	8.4	42.4	45.4	4.9
	[26, 79]	[1.5, 32.2]	[17, 35]	[1.8, 18.1]	[20, 74]	[23, 77]	[0.6, 12.4]
20×40	63.9	47.8	30.9	46.7	59.3	62.3	21.8
	[28, 125]	[8.0, 153.2]	[16, 57]	[5.1, 307.6]	[24, 121]	[27, 124]	[3.5, 74.3]
25×50	71.6	130.1	45.5	142.8	66.6	69.6	71.8
	[37, 122]	[5.8, 414.6]	[22, 74]	[6.0, 660.7]	[31, 117]	[34, 120]	[3.5, 253.0]
30×60	71.5	257.9	53.9	1143.7	67.1	70.1	169.7
	[46, 130]	[37.6, 1317.2]	[22, 141]	[10.3, 9917.0]	[41, 127]	[44, 130]	[16.0, 1058.3]
35×70	81.9	495.4	62.1	1200.7	78.1	81.1	323.6
	[27, 140]	[22.3, 1130.2]	[15, 158]	[10.5, 6321.4]	[22, 139]	[25, 142]	[7.9, 803.5]
40×80	76.4	568.1	49.4	1013.4	71.4	74.4	405.5
	[47, 146]	[59.9, 2317.6]	[26, 95]	[64.5, 3213.0]	[41, 142]	[44, 145]	[18.5, 1803.7]
45×90	101.6	3589.5	67.0	9225.7	98.0	101.0	2537.1
	[40, 190]	[112.1, 16020.1]	[27, 147]	[86.8, 67455.5]	[36, 190]	[39, 193]	[47.9, 12227.9]
50×100	94.6	1738.5	56.6	4003.6	89.3	92.3	1187.1
	[69, 138]	[420.8, 3956.6]	[33, 106]	[454.5, 17602.4]	[63, 135]	[66, 138]	[235.7, 2827.3]

Table 4.5: Comparing the solution methods on the MOILP problems with $p = 4$.

	ETPA+		**JA**		**NPDA**		
$m \times n$	#Models	CPU Time (s)	#Models	CPU Time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU Time (s)
5×10	318.8	4.9	18.1	0.7	157.4	314.1	3.8
	[52, 899]	[0.5, 14.3]	[8, 40]	[0.1, 3.1]	[22, 452]	[48, 895]	[0.4, 10.4]
10×20	635.4	33.0	37.1	11.6	320.7	633.4	20.2
	[190, 1016]	[7.0, 81.3]	[17, 53]	[0.8, 40.4]	[92, 513]	[186, 1012]	[4.6, 46.2]
15×30	1124.0	500.9	58.4	324.2	580.0	1131.2	233.3
	[162, 3882]	[7.2, 2539.9]	[24, 164]	[2.4, 2593.0]	[80, 2037]	[160, 3972]	[4.4, 1074.8]
20×40	1152.1	1175.2	60.5	374.6	592.2	1151.0	635.5
	[388, 2827]	[71.0, 6307.7]	[32, 139]	[23.3, 1210.3]	[196, 1474]	[389, 2803]	[38.0, 3725.0]
25×50	1030.0	3522.3	56.4	3169.2	527.1	1033.0	2200.4
	[212, 3206]	[45.2, 15961.0]	[25, 118]	[10.3, 24968.0]	[103, 1657]	[209, 3221]	[28.6, 10423.8]
30×60	2907.9	17114.5	110.9	**{3}**[1] 10950.3	1507.1	2924.7	10950.6
	[1801, 6725]	[2056.2, 43766.8]	[67, 180]	[409.6, 34053.4]	[935, 3492]	[1801, 6781]	[1038.4, 29720.1]
35×70	2545.4	20121.8	76.3	**{3}**[1] 4173.1	1313.4	2556.5	14355.1
	[1047, 5463]	[935.6, 43698.6]	[45, 125]	[61.8, 15699.3]	[531, 2829]	[1039, 5496]	[393.2, 33628.6]
40×80	1768.2	**{2}**[1] 19503.5	100.6	**{1}**[1] 14654.3	1124.3	2191.8	**{1}**[1] 20196.6
	[1060, 2786]	[8613.4, 42920.2]	[55, 189]	[762.7, 63661.2]	[539, 2856]	[1061, 5510]	[3907.7, 88401.0]

[1]Number of instances that could not be solved in 100,000 CPU seconds.

Finally, algorithms are tested on the randomly generated MOILP problem instances with $p = 5$. For this group of test, the instances are generated with up to 40 decision variables. The results of these tests are given in Table 4.6. JA is able to solve all instances for $p = 5$ case. With higher number of objective functions, Jorge's algorithm works better than NPDA; however, we have to note that these are small problems in terms of number of variables and constraints. Since NPDA searches the space through $(p-2)$-dimensional rectangles, NPDA's complexity is exponential in number of objective functions. On the other hand, the performance of JA is affected by the size of the nondominated set in general. Therefore, JA does not strongly react to number of objective functions as long as the problem size remains relatively small otherwise and the nondominated set does not grow.

In summary, NPDA outperforms ETPA+ in all instances independent of the problem type and size, despite the fact that we utilize the best performing algorithm to obtain the efficient set in ETPA+. NPDA also outperforms JA except for $p = 5$ cases for which only small size problems can be reported.

4.3.4 Payoff estimate and nadir point comparison

In this subsection, we wish to explore the quality of payoff estimation in discrete optimization problems. For a given problem type with N instances and p objective functions, we compute

$$\Delta = 100 \times \sum_{i=1}^{N} \sum_{j=1}^{p} \left(\frac{y_{ij}^N - y_{ij}^{PT}}{y_{ij}^N - y_{ij}^I} \right) \Big/ (N \times p). \tag{4.4}$$

Δ can be interpreted as the normalized distance between a payoff table estimate and a nadir point computed independently for each objective function. We calculate Δ for multiobjective knapsack and MOILP problem instances.

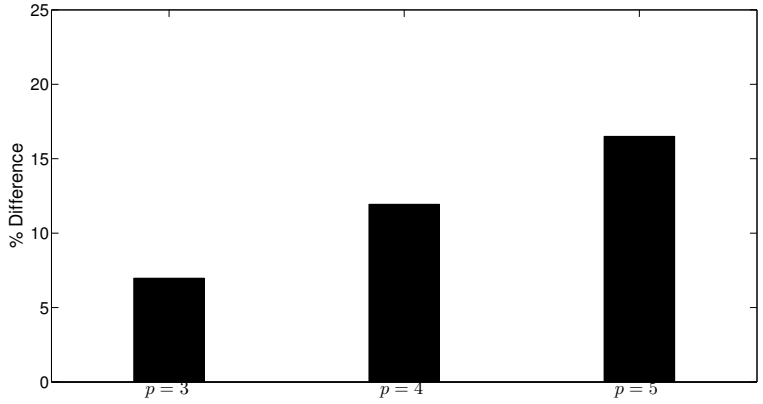
As seen in Figure 4.3, for both problem types, the average distance between the payoff table estimate and the nadir point is increasing with the number of objective

Table 4.6: Comparing the solution methods on the MOILP problems with $p = 5$.

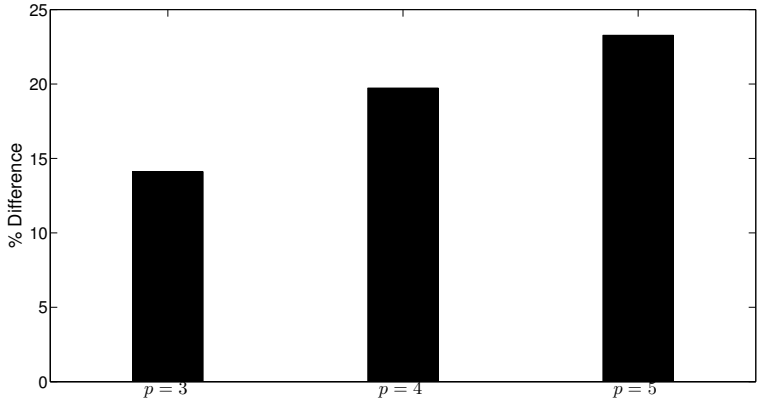
	ETPA+		**JA**		**NPDA**		
$m \times n$	#Models	CPU Time (s)	#Models	CPU Time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU Time (s)
{1}[2] 5×10	2102.6	129.8	25.1	1.6	457.3	2109.8	121.7
	[169, 6255]	[0.6, 529.4]	[0, 50]	[0.0, 5.7]	[43, 1294]	[164, 6413]	[0.7, 548.7]
10×20	8153.4	3637.9	51.7	84.2	1634.6	8174.6	3387.3
	[2018, 13197]	[81.5, 12294.7]	[17, 96]	[0.6, 561.5]	[426, 2580]	[2024, 13199]	[55.2, 12753.8]
15×30	6062.5	**{2}**[1] 1756.3	99.0	4778.0	1254.9	6078.1	**{2}**[1] 1246.3
	[2707, 8608]	[212.5, 3720.9]	[34, 377]	[5.5, 45078.6]	[619, 1744]	[2697, 8613]	[150.9, 2532.3]
20×40	7906.4	**{2}**[1] 8146.3	103.2	6264.9	1626.6	7951.2	**{2}**[1] 4972.8
	[1668, 17950]	[188.5, 32858.1]	[43, 242]	[16.9, 39210.2]	[372, 3694]	[1653, 18172]	[118.4, 21093.9]

[1]Number of instances that could not be solved in 100,000 CPU seconds.

[2]Number of unbounded instances.



(a) Multiobjective knapsack problem with $p = 3, 4$ and 5.



(b) Multiobjective integer linear programming problem with $p = 3, 4$ and 5.

Figure 4.3: Percentage difference between y^{PT} and y^N.

functions. On average, the payoff table estimate quality is worse in MOILP than in knapsack problems. This might be one of the underlying factors that lead to closer computational performance of NPDA and ETPA+ in MOILP instances.

4.4. Conclusion

In this chapter, we presented a new algorithm to find the nadir point of multiobjective discrete optimization problems with p objective functions. The proposed algorithm NPDA is based on an exhaustive search of the $(p-2)$-dimensional space. NPDA guarantees to find the nadir point exactly in a finite number of steps. We also compute Ehrgott and Tenfelde-Podehl's nadir point determination approach, and Jorge's solution method for the optimization over integer efficient set problem. All algorithms are tested on various types of discrete optimization problems with different sizes. In computational results, we see that NPDA outperforms Ehrgott and Tenfelde-Podehl's algorithm in all test instances by varying margins. NPDA also outperforms Jorge's algorithm in three and four-objective optimization problems. When $p = 5$, Jorge's algorithm displays better performance over problem sizes that can be solved within the enforced time limit.

Chapter 5

GENERATING REPRESENTATIVE SETS FOR MULTIOBJECTIVE DISCRETE OPTIMIZATION PROBLEMS WITH SPECIFIED COVERAGE ERRORS

In this chapter, we present an algorithm to generate representations with a quality guarantee for multiobjective discrete optimization problem with any number of objectives. In this algorithm, the outcome space is searched over p-dimensional rectangles, and for each rectangle two-stage mathematical programs are solved to find nondominated solutions. Since we aim to generate representations with a quality guarantee, any rectangle that satisfies the desired error level is eliminated from the search space. The algorithm is tested on multiobjective knapsack and multiobjective assignment problem instances with different error factors.

5.1. Introduction

Generating all nondominated solutions for multiobjective discrete optimization problem is the most desirable one. However, even bicriteria combinatorial optimization problems are intractable [166], i.e. the number of efficient solutions grows exponentially with respect to the size of the problem instance. Hence, for MODO problems, generating finite subset of all nondominated solutions may be more meaningful.

Generating representative set with specified error level has been considered in only four studies so far. These studies are only limited to handle the bicriteria case. Sayın and Kouvelis propose a representation method to obtain representations with a quality guarantee for bicriteria discrete optimization problems [163]. Hamacher et al. present

a box algorithm to generate a representation of the efficient set for discrete bicriteria problems [95]. Eusebio and Figueira propose a new algorithm to find a representation with desired quality for bicriteria integer network flow problems [77]. Vaz et al. introduce several algorithms for finding a representation for bicriteria combinatorial optimization problems considering uniformity, coverage and the ε-indicator measures [182].

In this chapter, we study generating representations with specified coverage errors for MODO problems with any number of objectives. We present an algorithm to search the outcome space with p-dimensional rectangles. We solve two-stage optimization problems to find nondominated solutions. During the search, any rectangle that satisfies the desired coverage error level is removed from the search list. The algorithm is tested on multiobjective knapsack and multiobjective assignment problems.

In the following section, computing the coverage is discussed. In Section 5.3, the proposed algorithm and associated theoretical results are given. In Section 5.4, computational results are presented on multiobjective knapsack and multiobjective assignment problem instances. Conclusions are presented in Section 5.5.

5.2. Computing the Coverage Error

Coverage error determines the maximum distance between a point in the nondominated set and its closest neighbor in the representation [80]. Our aim is to generate representative sets for MODO problems with a specified coverage error $\alpha \in \mathbb{R}$. Recall that given representative set $\mathcal{Y}_R$ and nondominated set $\mathcal{Y}_N$ the coverage error is calculated as follows,

$$z = \max_{\bar{y} \in \mathcal{Y}_N} \min_{y \in \mathcal{Y}_R} d(y, \bar{y}). \tag{5.1}$$

Let z^* be the optimal objective value of (5.1). Note that, the formulation for the coverage is same as the one-sided Hausdorff distance [97]. Here, the goal is to

generate $\mathcal{Y}_R$ that satisfies desired coverage level α, but we do not have any information about $\mathcal{Y}_N$ beforehand. Hence, we utilize the rectangles that partition the outcome space. A rectangle's lower and upper vertices are denoted as $l \in \mathbb{R}^p$ and $u \in \mathbb{R}^p$, respectively. With these bounds, a rectangle is defined as $R(l,u) = \{y \in \mathbb{R}^p, l \leq y \leq u\}$. Instead of $\mathcal{Y}_N$, we calculate the Hausdorff distance between the rectangle R and the representation $\mathcal{Y}_R$. The coverage error level of $R \subseteq \mathbb{R}^p$ by the representative set $\mathcal{Y}_R$ is computed as follows,

$$\lambda^* = \max_{\bar{y} \in R} \min_{y \in \mathcal{Y}_R} d(y, \bar{y}). \tag{5.2}$$

Let λ^* be the optimal objective value of (5.2). If $\lambda^* \leq \alpha$, then the rectangle satisfies the desired coverage error level α, and can be eliminated. The formulation (5.2) is a bilevel programming problem, and it can be turned into a single-level problem since $\mathcal{Y}_R$ is a discrete set. Let N represent the number of solutions in $\mathcal{Y}_R$. Given a metric (d), the coverage error formulation in (5.2) can be formulated as follows,

$$\begin{aligned} P_C(d) \quad \max \quad & \lambda \\ \text{s.t.} \quad & \lambda \leq d(\bar{y}, y^i) \quad i = 1, \ldots, N \\ & \bar{y} \in R. \end{aligned} \tag{5.3}$$

We use L_q-norm to measure the distance, and all norms are convex and nonlinear [63]. Hence, (5.3) is a challenging global optimization problem [100]. When the distance metric is the L_1-norm or the L_∞-norm, (5.3) can be formulated as a mixed integer linear programming problem [160]. Let M denote a sufficiently large positive number. Let $\lambda \in \mathbb{R}$, $(d_1, \ldots, d_N) \in \mathbb{R}^N$, $x, u^i, o^i \in \mathbb{R}^p$, and $t^i, s^i \in \{0,1\}^p$ for $i = 1, \ldots, N$ denote the variables. For $d = L_\infty$, the mixed integer linear programming formulation of (5.3) is as follows,

$$
\begin{aligned}
P_C(L_\infty) \quad \max \quad & \lambda \\
\text{s.t.} \quad & \lambda - d_i \leq 0 \quad i = 1, \ldots, N \\
& -d_i + \bar{y}_j + u_j^i = y_j^i \quad j = 1, \ldots, p;\ i = 1, \ldots, N \\
& d_i + \bar{y}_j - o_j^i = y_j^i \quad j = 1, \ldots, p;\ i = 1, \ldots, N \\
& u_j^i - M t_j^i \leq 0 \quad j = 1, \ldots, p;\ i = 1, \ldots, N \\
& o_j^i - M s_j^i \leq 0 \quad j = 1, \ldots, p;\ i = 1, \ldots, N \\
& \sum_{j=1}^{p} (t_j^i + s_j^i) \leq 2p - 1 \\
& \bar{y}_j \leq u_j \quad j = 1, \ldots, p \\
& \bar{y}_j \geq l_j \quad j = 1, \ldots, p \\
& \lambda \geq 0 \\
& u_j^i, o_j^i \geq 0 \quad j = 1, \ldots, p;\ i = 1, \ldots, N \\
& t_j^i, s_j^i \in \{0, 1\} \quad j = 1, \ldots, p;\ i = 1, \ldots, N \\
& d_i \geq 0 \quad i = 1, \ldots, N.
\end{aligned}
$$

The formulation $P_C(L_\infty)$ is linear, but it incorporates $2Np$ binary variables.

We need to compute the coverage error of each rectangle in the partition. Hence, it is not meaningful to solve a nonconvex optimization problem to decide whether rectangle R is covered or not. Then, we consider the following min-max formulation,

$$\gamma = \min_{y \in \mathcal{Y}_R} \max_{\bar{y} \in R} d(y, \bar{y}). \tag{5.4}$$

While calculating the coverage error, (5.2) considers all nondominated solutions in $\mathcal{Y}_R$, but (5.4) takes into account the closest nondominated solution from $\mathcal{Y}_R$. Let λ^* and γ^* be the optimal objective value of (5.2) and (5.4), respectively. For a given representation $\mathcal{Y}_R$ and a rectangle R, (5.4) computes an upper bound for the original coverage error formulation, i.e. $\lambda^* \leq \gamma^*$. Although (5.4) is a conservative way to calculate the coverage error, it is still valid in terms of assessing the measure without sacrificing the quality guarantee.

(5.4) is a bilevel optimization problem. For a given nondominated solution $y^i \in \mathcal{Y}_R$, the lower level problem maximizes the distance between the rectangle R and y^i. For any $y^i \in \mathcal{Y}_R$, let d_i be the optimal objective value of the lower level problem, then the lower level problem can be formulated as follows,

$$d_i = \max_{\bar{y} \in R} d(y^i, \bar{y}). \tag{5.5}$$

In the above formulation, d_i is the maximum distance between $y^i \in \mathcal{Y}_R$ and the rectangle R. Then the optimal objective value of (5.4) is $\gamma^* = \min_{i=1,\ldots,N} d_i$.

Since L_q-norm is convex, in general maximization of a convex function over a convex polytope is a global optimization problem [100]. The feasible set of (5.5) is a bounded convex polyhedral set in $\mathbb{R}^p$. Hence, the optimal solution is one of the extreme points of the feasible set [100]. However, we need to enumerate all extreme points of the rectangle to attain the optimal solution where full-dimensional rectangle in $\mathbb{R}^p$ has 2^p number of vertices.

Bodlaender et al. show that maximization of squared Euclidean norm over a rectangular parallelotop is polynomially solvable [31]. In our case, the problem is polynomially solvable for not only Euclidean norm but also any L_q-norm [100]. Given rectangle R with lower ($l \in \mathbb{R}^p$) and upper ($u \in \mathbb{R}^p$) vertices, the optimal solution of (5.5) is characterized as follows,

$$\bar{y}_j = \begin{cases} u_j & |u_j - y_j| \geq |l_j - y_j| \\ l_j & \text{otherwise} \end{cases} \qquad j = 1, \ldots, p.$$

5.3. Finding a Representative Set with Specified Coverage Error

In this section, we present an algorithm to generate representative sets for MODO problems with specified coverage errors. The search methodology of the algorithm is similar to SOR-GNS given in Chapter 3. SOR-GNS searches the projection of the

outcome space onto $(p-1)$-dimensional space to find all nondominated solutions. Our aim is to generate representations, and we need to compute coverage error of the rectangles in the outcome space, $\mathcal{Y} \subseteq \mathbb{R}^p$. Hence, we manage the search in the outcome space.

For each rectangle R in the partition, we solve a two-stage formulation where the first stage is the well-known ε-constraint formulation [93]. For a given rectangle $R(l,u) \subseteq \mathbb{R}^p$ and for some $k \in \{1,\ldots,p\}$, $\varepsilon \in \mathbb{R}^{p-1}$ defined as $\varepsilon_j = u_j$ for all $j \in \{1,\ldots,p\} \setminus \{k\}$. For a given $\varepsilon \in \mathbb{R}^{p-1}$ and for some $k \in \{1,\ldots,p\}$, the two-stage mathematical programs, $P_k(\varepsilon)$ and $Q_k(\varepsilon)$, are defined in Chapter 3. We use the two-stage formulation to obtain nondominated solutions for the representation. The search algorithm is also similar to SOR-GNS; however, instead of an exhaustive generation of nondominated set, a representation is sought. Here, our aim is to generate a representation with a specified quality error, and we intend to decrease the coverage error level of the worst represented rectangle by $\mathcal{Y}_R$. Hence, we define a new rectangle selection rule instead of the volume-based measure used in Chapter 3. This rule is based on coverage error and is expressed as follows,

$$R^* = \arg\max_{\hat{R} \in L} \max_{\bar{y} \in \hat{R}} \min_{y \in \mathcal{Y}_R} d(\bar{y}, y). \tag{5.6}$$

This formulation finds the worst represented rectangle by the current $\mathcal{Y}_R$. Among all rectangles in the current list L, coverage error of R^* is the highest. If we reduce the coverage error of R^*, the coverage error of L either decreases or remains the same. The second case occurs when there exists another rectangle with the same coverage error level with R^* after a subdivision takes place.

The technical statements of the proposed representative set generation algorithm that is labeled as RSGA-MODO are given below.

Input: Desired coverage error α, MODO problem, i.e. objective functions $f_j(x)$

for $j = \{1, \ldots, p\}$, and feasible set $\mathcal{X}$.

Output: α-representation of the nondominated set ($\mathcal{Y}_R$).

Step-0: Obtain the payoff table estimate y^{PT}. $\mathcal{Y}_R = \emptyset$. Initialize the rectangle list $L = \{R(y^I, y^U)\}$.

Step-1: If L is empty, go to Step-3. Otherwise, pick the worst representative rectangle $R_i \subseteq \mathbb{R}^p$. Solve the first stage formulation with $\varepsilon \in \mathbb{R}^{p-1}$ where $\varepsilon_j = u^i_j$ for $j \in \{1, \ldots, p\} \setminus \{k\}$, $P_k(\varepsilon)$.

Step-2:

- If $P_k(\varepsilon)$ is feasible, than solve $Q_k(\varepsilon)$. Let x^* denote an optimal solution. If $f(x^*) \notin \mathcal{Y}_R$, $\mathcal{Y}_R = \mathcal{Y}_R \cup \{f(x^*)\}$. Apply the rectangular subdivision process for each $R_s \in L$.

  Remove rectangles that lie in $R(f(x^*), y^U)$ or $R(y^I, f(x^*))$.
- Else (If $P_k(\varepsilon)$ is infeasible), define $u' \in \mathbb{R}^p$ as $u'_k = y^U_k$ and $u'_j = u^i_j$ for $j \in \{1, \ldots, p\} \setminus \{k\}$. Remove rectangles that lie in $R(\bar{y}^I, u')$.
- Check the coverage error of each rectangle in the list L. For each $R_s \in L$, $\min\limits_{y \in \mathcal{Y}_R} \max\limits_{\bar{y} \in R_s} d(y, \bar{y}) \leq \alpha$, than $L = L \setminus \{R_s\}$.
- Go to Step-1.

Step-3: Return representative set $\mathcal{Y}_R$ and stop.

The search is initialized with a single rectangle that covers the outcome space. In each iteration, algorithm picks a rectangle with the worst coverage error from the list L, and solves two-stage formulation for the upper vertex of this rectangle with the k^{th} component removed. Algorithm also checks each rectangle in the list L to see whether the rectangle is covered by the representative set or not. If the coverage error of the rectangle is less than or equal to desired error, then the rectangle is removed. Now, we need to show that, RSGA-MODO generates an α-representation of the nondominated set, and RSGA-MODO terminates in a finite number of iterations. In Lemma 5 and Lemma 6, we show that rectangles removed with optimality of the

two-stage formulation do not contain any nondominated solution.

Lemma 5. *Let x^* be an optimal solution of two-stage programs $P_k(\varepsilon)$ and $Q_k(\varepsilon)$ where $\varepsilon \in \mathbb{R}^{p-1}$. Then, there is no nondominated solution in $R(f(x^*), y^U)$ other than $f(x^*)$.*

Proof. Assume to the contrary that there exist an efficient solution $x' \in \mathcal{X}$ such that $f(x') \in R(f(x^*), y^U)$. Hence, $f_j(x^*) \leq f_j(x')$ for $j = 1, \ldots, p$. Since $f(x') \neq f(x^*)$, there exists $\hat{\jmath} \in \{1, \ldots, p\}$ such that $f_{\hat{\jmath}}(x^*) < f_{\hat{\jmath}}(x')$. This contradicts efficiency of x'. □

Lemma 6. *Let x^* be an optimal solution of two-stage programs $P_k(\varepsilon)$ and $Q_k(\varepsilon)$ where $\varepsilon \in \mathbb{R}^{p-1}$. Then, there is no nondominated solution in $R(y^I, f(x^*))$ other than $f(x^*)$.*

Proof. Assume to the contrary that there exist an efficient solution $x' \in \mathcal{X}$ such that $f(x') \in R(y^I, f(x^*))$. Hence, $f_j(x') \leq f_j(x^*)$ for $j = 1, \ldots, p$. Since $f(x') \neq f(x^*)$, there exists $\hat{\jmath} \in \{1, \ldots, p\}$ such that $f_{\hat{\jmath}}(x') < f_{\hat{\jmath}}(x^*)$. This contradicts efficiency of x^*. □

In Lemma 7, we show that removed rectangles with infeasibility of the first-stage problem do not contain any nondominated solution.

Lemma 7. *If $P_k(\varepsilon)$ is infeasible, then there is no nondominated solution in the rectangle $R(y^I, u)$ where $u_k = y^U_k$ and $u_j = \varepsilon_j$ for $j \in \{1, \ldots, p\} \setminus \{k\}$.*

Proof. Assume to the contrary that there exist an efficient solution $x' \in \mathcal{X}$ such that $y^I_j \leq f_j(x') \leq \varepsilon_j$ for $j = 1, \ldots, p$ and $j \neq k$, and $y^I_k \leq f_k(x') \leq y^U_k$. This implies that x' is feasible to $P_k(\varepsilon)$ which contradicts that $P_k(\varepsilon)$ has no feasible solution. □

In Theorem 15, we show that RSGA-MODO generates a representation that satisfies specified coverage error α.

Theorem 15. *RSGA-MODO generates an α-representation for a MODO problem.*

Proof. Two-stage mathematical programs $P_k(\varepsilon)$ and $Q_k(\varepsilon)$ either find an efficient solution or they are infeasible for any $\varepsilon \in \mathbb{R}^{p-1}$ and for some $k \in \{1, \ldots, p\}$. Therefore, all points in $\mathcal{Y}_R$ are nondominated solutions. By definition of y^I and y^U, all nondominated solutions are in $R(y^I, y^U)$, $\mathcal{Y}_N \subseteq R(y^I, y^U)$. In the proposed algorithm, some part of this rectangle is eliminated in two different ways. The first way is up to optimality or infeasibility of two-stage formulation, and we showed that removed volumes do not include a nondominated solution. The other way to remove a rectangle is to use coverage error information. Since our distance measure calculation provides an upper bound for the actual coverage error formulation, no point in the removed rectangle violates the coverage error factor α. From the characterization result, there exists $\varepsilon \in \mathbb{R}^{p-1}$ for each nondominated solution. Therefore, until the termination condition occurs in the proposed algorithm, an α-representative set is generated. □

In Theorem 16, we show that the algorithm terminates in a finite number of iterations.

Theorem 16. *RSGA-MODO is finite.*

Proof. Since we deal with bounded MODO problems, they have a finite number of nondominated solutions which is represented with $|\mathcal{Y}_N|$. In the worst case, each axis in $\mathbb{R}^p$ is divided into $(|\mathcal{Y}_N| + 1)$ rectangles. Since the search space is p-dimensional, $|L| \leq (|\mathcal{Y}_N| + 1)^p$ where $|L|$ denotes the size of the rectangle list. This means that the number of rectangles is also finite. Besides, in each iteration at least one rectangle is removed from L. This implies that L will be empty in a finite number of iterations. Hence, the proposed algorithm is finite. □

In Theorems 15 and 16, we showed that RSGA-MODO generates an α-representation of the nondominated set or MODO problem with any number of objective functions in a finite number of iterations.

5.4. Computational Results

RSGA-MODO is tested on multiobjective knapsack and multiobjective assignment problems with three objectives. We use the same multiobjective knapsack problem and multiobjective assignment problem instances that were generated to test SOR-GNS in Chapter 3. Since the nondominated set of these problems have already been obtained, we can compute actual coverage errors of the resulting representations. We use L_1-norm and L_∞-norm in the computational tests. Additionally, to test the effect of specified coverage level on RSGA-MODO, we generate representative sets with different factors, $\alpha = 5\%$, 10% and 20%. RSGA-MODO is compared with SOR-GNS (see Chapter 3) which is the best performing nondominated set enumeration algorithm in terms of solution time.

In the computational results, we report cardinality of the representative set, number of models solved and CPU time. Additionally, we compute the coverage error of the resulting representative set $\mathcal{Y}_R$ by using (5.1). In the following tables, the z^* column shows the coverage error of resulting nondominated set which is obtained by using RSGA-MODO.

As seen in Table 5.1, even for $\alpha = 5\%$, using RSGA-MODO is beneficial compared to the exact method in terms of solution time. In Table 5.2, the results of the RSGA-MODO with L_∞-norm on multiobjective knapsack problem are given.

When we use L_∞-norm and $\alpha = 5\%$, the exact algorithm outperforms RSGA-MODO in terms of solution time. On the other hand, the resulting coverage errors are closer to desired coverage level α. In Table 5.3 and 5.4, the results of the RSGA-MODO on multiobjective assignment problem are given.

RSGA-MODO performs better in assignment problems compared to knapsack problems, because of size of the efficient set. RSGA-MODO estimates the bounds of the nondominated set by using payoff table, and computes the upper bound for the coverage error to eliminate some rectangles. Despite these downsides, in some of the

Table 5.1: RSGA-MODO results with L_1-norm on the multiobjective knapsack problem with $p = 3$.

	5%-Representation				**10%-Representation**				**20%-Representation**			
n	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)
10	0.00	9.8	29.4	0.2	0.53	9.7	27.5	0.2	1.34	9.4	22.6	0.1
	[0.00,0.00]	[5,25]	[11;92]	[0.0,0.8]	[0.00,5.26]	[5,24]	[11;81]	[0.0,0.9]	[0.00,7.96]	[5,22]	[9;58]	[0.0,0.4]
20	0.52	37.5	129.6	1.6	5.00	32.1	91.5	0.8	7.42	27.9	81.1	0.5
	[0.00,2.17]	[22,61]	[78;203]	[0.8,3.2]	[1.63,7.08]	[21,48]	[64;121]	[0.5,1.4]	[1.63,11.95]	[21,35]	[57;115]	[0.4,0.9]
30	3.35	87.7	274.8	5.5	6.51	63.5	210.3	2.3	7.69	61.4	208.2	1.9
	[1.32,4.17]	[59,153]	[184;337]	[2.8,10.9]	[5.28,7.92]	[46,90]	[116;284]	[1.3,2.8]	[5.45,11.19]	[43,85]	[111;284]	[1.1,2.4]
40	4.05	156.3	497.5	15.0	7.07	121.6	447.3	8.6	7.96	120.2	446.1	6.9
	[3.50,4.32]	[100,225]	[266;958]	[6.4,32.9]	[6.11,8.34]	[78,208]	[231;924]	[3.9,21.8]	[6.11,11.38]	[78,208]	[232;926]	[3.4,16.1]
50	4.07	188.2	679.3	27.4	6.78	159.1	637.6	15.6	7.11	157.4	636.1	11.1
	[3.46,4.52]	[118,256]	[281;1805]	[7.3,64.3]	[4.33,8.39]	[97,254]	[262;1806]	[5.8,49.8]	[4.33,10.31]	[97,254]	[262;1802]	[4.4,31.4]
60	4.27	264.3	917.9	61.5	7.48	230.3	876.1	29.8	8.20	228.3	875.5	20.3
	[4.00,4.58]	[208,327]	[714;1270]	[31.6,123.8]	[5.79,9.23]	[182,297]	[636;1177]	[15.8,59.6]	[5.79,10.32]	[179,292]	[623;1179]	[10.8,31.1]
70	4.30	352.9	1423.5	137.6	7.55	305.9	1404.6	60.6	8.00	304.8	1403.7	39.2
	[4.06,4.54]	[280,450]	[831;2801]	[66.7,305.1]	[5.56,8.85]	[228,409]	[714;2790]	[28.5,98.1]	[5.56,10.72]	[236,408]	[750;2776]	[20.9,61.4]
80	4.35	451.8	1593.3	352.5	7.71	395.1	1523.3	132.1	8.56	391.5	1521.7	62.4
	[4.18,4.50]	[327,615]	[1237;2265]	[110.7,779.5]	[6.68,9.12]	[299,581]	[1149;2243]	[61.2,266.7]	[6.68,10.30]	[299,585]	[1147;2239]	[41.9,111.8]
90	4.38	498.6	2444.2	505.5	7.75	451.6	2377.1	192.0	8.16	449.1	2372.0	99.5
	[3.94,4.63]	[374,663]	[1527;4956]	[112.5,1046.0]	[5.12,9.09]	[360,576]	[1461;4875]	[64.5,363.3]	[5.12,10.72]	[360,574]	[1464;4880]	[45.1,227.0]
100	4.46	664.5	2698.3	1391.1	8.48	582.4	2580.3	488.1	9.43	576.3	2674.3	225.4
	[4.25,4.69]	[422,999]	[1420;5994]	[226.9,3162.4]	[7.23,9.26]	[375,922]	[1364;5845]	[142.5,1024.4]	[7.23,12.54]	[371,934]	[1359;5832]	[85.8,404.2]

Table 5.2: RSGA-MODO results with L_∞-norm on the multiobjective knapsack problem with $p = 3$.

	5%-Representation				**10%-Representation**				**20%-Representation**			
n	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)
10	0.00	9.8	35.6	0.2	0.69	9.7	34.4	0.2	0.92	9.5	31.3	0.2
	[0.00,0.00]	[5,25]	[12;92]	[0.0,0.8]	[0.00,6.88]	[5,24]	[12;85]	[0.0,0.7]	[0.00,9.22]	[5,22]	[12;73]	[0.0,0.5]
20	0.17	37.9	167.3	1.8	3.21	36.2	134.8	1.3	11.62	29.8	92.5	0.7
	[0.00,1.74]	[22,63]	[96;302]	[0.8,4.2]	[0.00,7.85]	[22,60]	[91;174]	[0.7,2.1]	[2.33,17.89]	[21,38]	[60;131]	[0.4,1.1]
30	1.84	107.8	450.3	8.5	7.58	79.6	243.6	4.2	12.97	64.7	202.8	2.5
	[0.00,4.60]	[62,220]	[281;774]	[3.3,21.4]	[5.74,8.65]	[56,130]	[177;373]	[2.1,8.7]	[9.28,16.70]	[46,96]	[139;316]	[1.3,3.9]
40	4.22	224.5	726.1	28.0	8.23	147.4	399.8	16.2	12.48	132.6	382.1	9.0
	[3.54,4.63]	[124,320]	[453;979]	[10.5,68.0]	[6.96,9.63]	[89,232]	[236;711]	[5.9,38.4]	[8.16,16.21]	[77,230]	[184;661]	[3.2,21.1]
50	4.40	274.4	806.4	48.4	7.97	190.1	542.3	29.5	10.89	182.0	533.3	16.1
	[3.59,4.67]	[151,423]	[616;1176]	[14.5,114.2]	[6.69,9.26]	[111,262]	[382;795]	[8.8,62.8]	[8.29,15.35]	[105,249]	[331;780]	[6.1,37.5]
60	4.50	401.7	1064.2	133.7	8.24	293.5	831.7	77.3	10.46	276.2	798.5	33.3
	[3.99,4.89]	[317,495]	[794;1378]	[49.2,228.6]	[7.08,9.12]	[231,382]	[606;977]	[38.0,160.0]	[8.45,13.24]	[220,345]	[561;1144]	[19.5,58.4]
70	4.59	507.1	1223.0	324.6	8.07	382.9	998.6	150.2	11.15	374.1	1024.5	66.7
	[4.23,4.91]	[429,673]	[925;1630]	[155.9,757.1]	[7.26,8.93]	[306,451]	[783;1317]	[83.9,261.6]	[8.85,13.31]	[272,461]	[751;1403]	[35.4,127.2]
80	4.63	638.8	1497.2	846.6	8.31	510.1	1281.3	445.7	11.10	488.5	1260.4	132.4
	[4.34,4.79]	[470,824]	[1307;1949]	[285.5,1807.1]	[7.54,9.30]	[382,681]	[1070;1702]	[177.0,792.4]	[7.79,14.02]	[381,644]	[1057;1648]	[59.8,260.9]
90	4.63	738.3	1795.6	1425.6	8.24	596.7	1581.4	879.7	12.37	551.2	1474.9	213.7
	[4.37,4.90]	[511,982]	[1330;2366]	[500.7,3168.3]	[6.50,9.33]	[432,872]	[1045;2318]	[172.7,3176.5]	[9.16,15.00]	[425,838]	[1081;2313]	[89.6,515.6]
100	4.73	1024.0	2419.5	5658.7	8.54	804.8	2088.4	1995.3	11.95	765.7	1970.8	551.1
	[4.59,4.88]	[619,1476]	[1600;3152]	[602.7,16974.0]	[7.81,9.53]	[588,1218]	[1481;2882]	[523.9,5967.5]	[6.99,15.92]	[567,1150]	[1292;2815]	[206.9,1943.2]

Table 5.3: RSGA-MODO results with L_1-norm on the multiobjective assignment problem with $p = 3$.

	5%-Representation				**10%-Representation**				**20%-Representation**			
n	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)
10	0.00	14.1	40.8	0.1	0.69	14.0	34.2	0.1	6.80	12.2	22.0	0.1
	[0.00,0.00]	[5,23]	[9;75]	[0.0,0.3]	[0.00,4.29]	[5,22]	[9;53]	[0.0,0.2]	[0.00,13.01]	[5,18]	[9;30]	[0.0,0.1]
20	3.83	122.1	247.1	4.6	7.19	75.9	173.0	2.5	10.84	62.8	137.4	2.6
	[3.26,4.42]	[62,173]	[134;339]	[2.4,6.8]	[6.12,8.64]	[47,115]	[89;263]	[1.3,4.2]	[8.05,13.57]	[41,90]	[82;219]	[1.6,4.2]
30	4.21	222.4	476.8	15.8	7.97	151.0	380.4	11.5	11.31	136.0	348.6	11.4
	[3.88,4.58]	[176,267]	[310;697]	[9.6,21.5]	[7.05,8.90]	[118,178]	[210;615]	[6.2,18.0]	[9.15,14.18]	[109,164]	[199;537]	[7.6,15.0]
40	4.34	345.9	860.8	57.8	8.31	251.3	715.1	37.2	10.68	234.7	709.9	36.6
	[4.16,4.53]	[221,400]	[480;1188]	[22.4,81.0]	[7.15,9.19]	[167,297]	[455;1109]	[22.0,59.4]	[9.18,12.63]	[161,294]	[417;1260]	[23.3,51.5]
50	4.43	443.3	1057.8	126.7	8.70	323.7	900.4	84.5	12.17	301.4	771.5	76.1
	[4.21,4.70]	[364,517]	[661;1609]	[75.2,193.2]	[7.64,9.41]	[263,404]	[510;1508]	[53.8,121.0]	[10.36,14.30]	[252,373]	[441;1450]	[51.1,109.2]
60	4.54	582.4	1688.9	325.0	8.65	450.9	1795.5	251.3	12.93	437.7	1887.6	216.4
	[4.42,4.68]	[498,644]	[1021;2630]	[227.5,452.4]	[7.79,9.45]	[359,499]	[972;4798]	[154.1,542.5]	[10.62,16.12]	[353,481]	[1017;4536]	[158.0,417.8]
70	4.55	703.9	2294.3	625.3	8.89	552.9	2214.3	474.6	12.19	515.4	1860.1	348.6
	[4.41,4.74]	[634,822]	[1073;3901]	[369.4,788.0]	[8.00,9.48]	[478,647]	[1433;3986]	[316.0,658.9]	[10.48,14.26]	[435,582]	[810;4048]	[219.2,486.8]
80	4.64	839.0	2734.2	1181.0	8.55	654.3	2640.6	864.3	12.78	631.6	2534.2	552.6
	[4.53,4.74]	[752,1003]	[1628;4481]	[777.0,1929.5]	[7.20,9.40]	[573,776]	[1350;4376]	[603.8,1404.1]	[9.88,15.05]	[554,730]	[1503;4218]	[346.1,874.2]
90	4.64	931.6	3583.1	1684.4	8.83	726.5	3220.3	1333.6	13.49	693.7	2623.7	741.8
	[4.53,4.71]	[830,1033]	[1520;5122]	[1261.2,2326.3]	[8.30,9.24]	[648,790]	[1293;6133]	[709.4,1905.5]	[9.16,18.14]	[608,754]	[1218;6160]	[517.7,1040.8]
100	4.63	1178.9	3438.8	2839.7	9.05	829.4	3471.3	2144.0	12.91	796.3	3736.2	1430.7
	[4.44,4.76]	[1030,1354]	[1843;6260]	[2021.3,3999.1]	[8.49,9.47]	[780,907]	[1470;6566]	[1568.7,3193.7]	[10.20,16.71]	[740,855]	[1863;5734]	[884.9,2473.4]

Table 5.4: RSGA-MODO results with L_∞-norm on the multiobjective assignment problem with $p = 3$.

	5%-Representation				**10%-Representation**				**20%-Representation**			
n	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)	z^*	$\lvert\mathcal{Y}_R\rvert$	#Model	CPU Time (s)
10	0.00	14.1	44.4	0.2	0.00	14.1	42.9	0.2	4.68	13.5	32.4	0.1
	[0.00,0.00]	[5,23]	[12;87]	[0.0,0.7]	[0.00,0.00]	[5,23]	[12;85]	[0.0,0.6]	[0.00,13.51]	[5,21]	[12;47]	[0.0,0.2]
20	3.56	157.5	416.0	9.0	8.23	106.0	213.1	4.9	13.68	77.1	155.8	2.2
	[0.00,4.76]	[65,255]	[180;595]	[3.4,13.6]	[7.25,9.84]	[61,145]	[127;272]	[2.0,9.2]	[10.68,16.38]	[39,121]	[81;247]	[1.1,3.7]
30	4.51	366.7	660.3	33.0	8.47	212.4	396.7	18.5	13.44	178.3	344.4	10.2
	[3.80,5.00]	[282,450]	[584;768]	[22.5,46.3]	[7.78,9.15]	[172,264]	[316;534]	[11.5,28.3]	[11.56,15.57]	[132,225]	[277;463]	[7.5,14.0]
40	4.57	571.8	910.3	115.9	8.93	369.0	632.7	68.3	13.62	309.4	555.1	31.9
	[4.25,5.00]	[373,690]	[697;1121]	[45.4,165.7]	[8.26,9.91]	[234,431]	[452;729]	[28.6,89.6]	[12.02,15.65]	[217,374]	[425;688]	[20.0,46.7]
50	4.72	743.8	1101.1	251.5	9.12	484.9	802.4	149.7	16.59	432.2	743.5	72.2
	[4.46,4.96]	[576,891]	[930;1406]	[136.8,377.6]	[8.68,9.56]	[356,644]	[657;1025]	[72.2,239.8]	[12.95,19.29]	[337,553]	[599;922]	[46.5,95.9]
60	4.83	1027.8	1489.0	643.1	9.22	669.7	1086.8	376.4	15.37	591.9	982.1	148.9
	[4.58,4.97]	[861,1211]	[1094;1723]	[463.9,836.0]	[8.59,9.69]	[464,831]	[708;1391]	[162.0,679.0]	[12.65,17.81]	[521,682]	[836;1206]	[88.8,200.1]
70	4.79	1208.5	1816.8	1173.4	9.28	857.5	1395.4	735.6	15.41	764.0	1246.1	251.6
	[4.58,5.00]	[1007,1391]	[1519;2331]	[621.2,1749.0]	[8.68,9.76]	[708,1015]	[1183;1856]	[353.5,1238.6]	[13.25,17.82]	[681,874]	[1142;1389]	[183.6,347.9]
80	4.83	1638.2	2339.0	2879.3	9.38	1168.3	1748.1	1828.8	15.11	937.3	1488.4	423.7
	[4.58,4.99]	[1297,1955]	[1958;2683]	[1416.4,5449.4]	[8.75,9.73]	[934,1426]	[1374;2055]	[722.7,2913.4]	[12.85,16.70]	[787,1028]	[1234;1663]	[349.6,491.9]
90	4.87	1755.0	2620.0	4278.7	9.56	1277.4	2018.6	2330.7	15.38	1058.9	1673.3	740.5
	[4.73,4.97]	[1602,1951]	[2334;3261]	[2417.7,5733.8]	[9.13,9.77]	[1091,1544]	[1685;2891]	[1358.3,4897.6]	[12.96,18.88]	[865,1322]	[1379;1884]	[356.2,1431.2]
100	4.90	2053.2	2953.8	6004.2	9.62	1509.5	2215.5	4523.5	16.25	1243.3	1893.3	1133.5
	[4.74,4.99]	[1813,2261]	[2498;4524]	[3805.3,7842.1]	[9.15,10.00]	[1326,1646]	[1928;2395]	[3372.4,6703.6]	[13.37,19.02]	[1050,1421]	[1688;2101]	[545.8,1767.3]

instances, RSGA-MODO generates a representation with exactly the same desired coverage error level.

5.5. Conclusion

We study generating representations with specified coverage errors for MODO problem with any number of objectives. Note that this is the first method in the literature that generates representations with a quality guarantee for multiobjective discrete optimization problems with any number of objectives.

We present an algorithm to search the outcome space with p-dimensional rectangles, and solve two-stage optimization problems to find nondominated solutions. During the search, any rectangle that satisfies the desired coverage error level is removed from the search list. We have shown that the proposed algorithm generates an α-representation of MODO nondominated set in a finite number of iterations. The algorithm is tested on multiobjective knapsack and multiobjective assignment problems. The proposed algorithm is able to generate representations efficiently. However, when we increase the desired coverage error level, the performance of the method diminishes due to navigation problem in the outcome space. In the following chapter, we model the problem of finding a nondominated solution inside a given rectangle as a bilevel optimization problem.

Chapter 6

BILEVEL PROGRAMMING FOR FINDING A NONDOMINATED SOLUTION IN A GIVEN SET: APPLICATIONS IN MULTIOBJECTIVE OPTIMIZATION

In the previous chapter, we present an algorithm to generate the representative sets with specified coverage errors for MODO problems. The proposed algorithm is limited to solve MODO problems. Additionally, when we increase the desired coverage error level, the performance of the method diminishes due to navigation problem in the outcome space. The cause of the navigation problem is the first-stage of the two-stage formulation, i.e. ε-constraint method itself. In ε-constraint method, we can only impose lower bounds on $p-1$ objective functions that are transformed into constraints. Additionally, the method has no control on the objective function that is taken into the objective function. Hence, we cannot target specific portion of the outcome space by using the two-stage formulation. To eliminate this problem, we model the problem of finding a nondominated solution in a given rectangle as a bilevel programming problem. By using this result, we propose an algorithm to generate representative sets with specified coverage errors for multiobjective optimization problems. We test the method on multiobjective linear programming problems. We solve the linear bilevel programming subproblems by using penalty approach and integer linear programming formulation. Finally, the representation algorithm is applied to obtain a better accuracy on support vector machine classification problem with imbalanced data sets.

6.1. Introduction

The bilevel programming problem (BPP) is an optimization problem whose constraints are determined by an another optimization problem. In other words it is a hierarchical optimization problem consisting of two levels where the first and second level is called upper level problem and lower level problem, respectively.

Bilevel optimization problems are commonly found in a number of real-world problems [14]. This includes problems in the domain of transportation [110], economics [168], engineering design [118], energy sector [16]. Other typical applications of BPP are toll setting problem, structural optimization and defense applications. In toll setting problem, a highway authority sets tolls on a subset of arcs of the network, while the users aim to find the shortest path route from origin to destination on the network. The goal of the upper level program (highway authority) is to maximize toll revenue, it is not in its interest to set tolls at very high values, in which case the users will be discouraged from using the tolled subnetwork. The problem aims to find the right balance between tolls that generate high revenues and tolls that attract customers [58]. In structure optimization, the upper level problem figures out the shape of the structure, choice of materials, amount of material to minimize the cost subject to bounds on displacements, stresses and contact forces. Values of the upper level can only be determined by solving the potential energy minimization problem which is the lower level problem of the bilevel formulation [106, 117]. Defense applications can be categorized as strategic offense and defense models. In the strategic offense model, the upper level problem chooses minimum-cost offensive forces capable of achieving specified destruction of various resources in which lower level problem allocates the specified defensive forces to minimize the destruction. In the strategic defense model, the upper level problem chooses minimum-cost defensive forces capable of assuring specified survival of various resources in which lower level problem allocates the specified offensive forces to minimize the surviving resources [35].

Bilevel programming problems are typically challenging [13]. Hence, most of the studies have focused on the BPP with nice properties such as linear, quadratic or convex [44]. In particular, the bilevel linear programming problem (BLPP), which has linear objective functions and constraints, is the most studied one [187]. Still, over the years, more complex bilevel programs were studied such as BLPP with discrete variables [15, 184], BPP with a nonconvex inner problem [142], BPP with multiple objectives [75].

Several solution methods have been proposed to solve different types of BPPs. Descent approach [183], bundle algorithm [53], penalty approach [132], trust-region method [45], smooth approximation of the KKT transformation [86] and transforming BPP into binary integer programming problem by using KKT optimality conditions for the lower level problem [84] have been proposed to solve BPP with convex lower level problem. Additionally, complementary pivots [105] method have been proposed to solve BLPPs. This method is also based on the reformulation of BLPP using KKT conditions. For discrete BPPs, branch-and-bound [143], cutting plane [54], and branch-and-cut methods [56] have been used so far.

In this study, we use bilevel optimization problem to obtain a nondominated solution in a rectangle. Many studies have been proposed to obtain nondominated solutions for MOP [65]. Most of these studies utilize scalarization methods to deal with the multiple objectives in which the MOP is turned into a single objective optimization problem [72]. However, in these methods it is difficult to target a specific part of the nondominated set which may contain the preferred nondominated solution for the decision maker. In this study, we model the problem of finding a nondominated solution in a given rectangle as a bilevel optimization problem. This result has several applications in multiobjective optimization.

We propose an algorithm to generate representative sets for MOP by using the bilevel programming formulation. The algorithm is tested on multiobjective linear programming problem instances. Bilevel programming problems are solved by using

penalty approach and integer linear programming (ILP) reformulation. Finally, we apply the method to obtain a better accuracy on support vector machine classification problem with imbalanced data sets [3]. In the following section, we present the general formulation for the bilevel programming problem. In Section 6.3, theoretical findings related to a bilevel programming formulation that aims to determine a nondominated solution in a given set are given. In Section 6.4, the proposed representative algorithm is presented. In Section 6.5, the results of the proposed algorithm are shown on a sample multiobjective linear programming problem. In Section 6.6, computational results are presented on multiobjective linear programming problem instances. In Section 6.7, the results of the proposed algorithm on support vector machine classification problem on imbalanced data sets are given. Finally, conclusions are presented in Section 6.8.

6.2. Bilevel Programming Problem

A general formulation of bilevel programming problem can be written as follows:

$$\begin{aligned} \textbf{(BPP)} \quad & \min_{x} \quad f^U(x,z) \\ & \text{s.t.} \quad g(x,z) \leq 0 \\ & \qquad \min_{z} \quad f^L(x,z) \\ & \qquad \text{s.t.} \quad h(x,z) \leq 0. \end{aligned}$$

In the above formulation, the decision vectors are divided into two classes, the upper level variables $x \in \mathbb{R}^{n_U}$ and the lower level variables $z \in \mathbb{R}^{n_L}$. Similarly, $f^U : \mathbb{R}^{n_U} \times \mathbb{R}^{n_L} \to \mathbb{R}$ and $f^L : \mathbb{R}^{n_U} \times \mathbb{R}^{n_L} \to \mathbb{R}$ are the upper level and the lower level objective functions. Without loss of generality, the set of constraints for the upper and lower level problems, $g : \mathbb{R}^{n_U} \times \mathbb{R}^{n_L} \to \mathbb{R}^{m_U}$ and $h : \mathbb{R}^{n_U} \times \mathbb{R}^{n_L} \to \mathbb{R}^{m_L}$, are given in the inequality form. Definitions related to BPP are given below [14].

1. Feasible set of the bilevel program is

$$M = \{(x, z) : g(x, z) \leq 0, h(x, z) \leq 0\}.$$

2. For a given x, the feasible set for the lower level program is

$$M(x) = \{z \in \mathbb{R}^{n_L} : h(x, z) \leq 0\}.$$

3. Projection of M onto the upper level formulation's decision space is

$$M(\mathcal{X}) = \{x \in \mathbb{R}^{n_U} : g(x, z) \leq 0, h(x, z) \leq 0 \text{ for some } z \in \mathbb{R}^{n_L}\}.$$

4. For any $x \in M(\mathcal{X})$, rational reaction set of the lower level problem is

$$\Omega(x) = \{z \in \mathbb{R}^{n_L} : z \in \underset{\hat{z} \in \mathbb{R}^{n_L}}{\arg\min}\{f^L(x, \hat{z}) : \hat{z} \in M(x)\}\}.$$

5. Inducible region regroups the feasible points of the BLPP, corresponds to the feasible set of the upper level program.

$$\mathcal{IR} = \{(x, z) \in M : z \in \Omega(x)\}.$$

6.3. Finding a Nondominated Solution in a Given Rectangle

We model the problem of finding a nondominated solution in a rectangle as a bilevel optimization problem. The upper and the lower decision variables are $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^n$. The upper level objective function is $f^U(x, z) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, and the lower level objective function is $f^L(z) : \mathbb{R}^n \to \mathbb{R}$. Both levels share a common set of constraints defined by $g : \mathbb{R}^n \to \mathbb{R}$. These constraints form a set $\mathcal{X} \subseteq \mathbb{R}^n$ such that $\mathcal{X} = \{\hat{x} \in \mathbb{R}^n : g(\hat{x}) \leq 0\}$. Additionally, the upper level problem includes constraints

related to the given rectangle, and the lower level problem includes set of constraints related to domination property. Given a rectangle $R \subseteq \mathbb{R}^p$, a metric defined on $\mathbb{R}^p$, and a positive weight vector $w \in \mathbb{R}^p$, the bilevel optimization formulation of the problem can be defined as follows,

$$\begin{aligned}
(\boldsymbol{P}) \quad \min_{x} \quad & f^U(x,z) = d(f(x), f(z)) \\
\text{s.t.} \quad & x \in \mathcal{X} \\
& f(x) \in R \\
& \min_{z} \quad f^L(z) = w^T f(z) \\
& \text{s.t.} \quad z \in \mathcal{X} \\
& \qquad f(z) \leq f(x).
\end{aligned}$$

For any $x \in \mathcal{X}$ and $f(x) \in R$, the rational reaction set for the lower level formulation is

$$\Omega(x) = \{z \in \mathcal{X} : z \in \arg\min_{\hat{z} \in \mathcal{X}} \{w^T f(\hat{z}) : \hat{z} \in \mathcal{X}, f(\hat{z}) - f(x) \leq 0\}\}.$$

In bilevel formulation, for any $x \in \mathcal{X}$, the lower level aims to find a (efficient) solution $z \in \mathcal{X}$ such that all components of $f(z) \in \mathbb{R}^p$ are less than or equal to $f(x) \in \mathbb{R}^p$. The upper level program finds a solution x which is mapped into rectangle $R \subseteq \mathbb{R}^p$. The rectangle R is defined by lower vertex $l \in \mathbb{R}^p$ and upper vertex $u \in \mathbb{R}^p$. Goal of the bilevel program is to minimize the distance between the mapping of two solutions $x \in \mathcal{X}$ and $z \in \mathcal{X}$ in the outcome space over the inducible region of P.

At first, we need to show that for any $x \in \mathcal{X}$ the optimal solution of the lower level formulation is efficient and any efficient solution is optimal to lower level formulation.

Theorem 17. *For any $x \in \mathcal{X}$ and $w > 0$, any $z^* \in \Omega(x)$ is efficient.*

Proof. For any $x \in \mathcal{X}$, let $z^* \in \Omega(x)$. Assume for a contradiction that there exists

$z' \in \mathcal{X}$ that dominates z^*. Then $f_j(z') \leq f_j(z^*) \leq f_j(x)$ for all $j \in \{1, \ldots, p\}$ and there exists $\hat{j} \in \{1, \ldots, p\}$ such that $f_j(z') < f_j(z^*) \leq f_j(x)$. This implies that

$$\sum_{j=1}^{p} w_j f_j(z') < \sum_{j=1}^{p} w_j f_j(z^*) \leq \sum_{j=1}^{p} w_j f_j(x).$$

Hence, $z^* \notin \Omega(x)$. Thus z^* is efficient. □

Theorem 18. *For any efficient solution z^*, there exists $x' \in \mathcal{X}$ such that $z^* \in \Omega(x')$.*

Proof. Let $z^* \in \mathcal{X}_E$. We claim $z^* \in \Omega(x')$ and $f(z^*) = f(x')$. Suppose that $z^* \notin \Omega(x')$. Since $z^* \in \mathcal{X}_E \subseteq \mathcal{X}$ and $f(x^*) = f(x')$, z^* is a feasible solution for the lower level problem. Let $z' \in \Omega(x')$. Note that $f_j(z') \leq f_j(x') = f_j(z^*)$ for all $j \in \{1, \ldots, p\}$. Since $z' \in \Omega(x')$ and $z^* \notin \Omega(x')$,

$$\sum_{j=1}^{p} w_j f_j(z') < \sum_{j=1}^{p} w_j f_j(z^*).$$

The equation above implies that $f_{\hat{j}}(z') < f_{\hat{j}}(z^*)$ for some $\hat{j} \in \{1, \ldots, p\}$. Then z' dominates z^* which contradicts that z^* is an efficient solution. □

We have shown that the lower level formulation of the bilevel program is efficient and any efficient solution can be obtained with the lower level formulation. In Theorems 17 and 18, we do not have any assumptions for the problem (decision variables, upper and lower level objective functions and constraints). Hence, for any $x \in X$ any solution z^* from rational reaction set, $z^* \in \Omega(x)$, is efficient. Additionally, for any $x \in \mathcal{X}$, the rational reaction set contains all efficient solutions that satisfy the inequality $f(z) \leq f(x)$.

We expect the bilevel formulation to determine a nondominated solution in the rectangle R if such a solution exists. Otherwise, the formulation should identify that $R \cap \mathcal{Y}_N = \emptyset$.

Theorem 19. *For a given rectangle $R \subseteq \mathbb{R}^p$, if there exists (x^*, z^*) that solves P and $d(f(x^*), f(z^*)) = 0$, then $f(x^*) \in R \cap \mathcal{Y}_N$.*

Proof. Let a rectangle $R \subseteq \mathbb{R}^p$ and $w > 0$ be given. Let (x^*, z^*) be the optimal solution of P such that $d(f(x^*), f(z^*)) = 0$. Since d is a metric, $d(f(x^*), f(z^*)) = 0$ implies that $f(x^*) = f(z^*)$. Note that $z^* \in \Omega(x^*)$ and by Theorem 17, and z^* is efficient. Since $f(x^*) = f(z^*)$, $x^* \in \mathcal{X}_E$ and $f(x^*) \in \mathcal{Y}_N$. Since x^* is feasible to P, $f(x^*) \in R$. These imply that $f(x^*) \in R \cap \mathcal{Y}_N$. □

Theorem 20. *For a given rectangle $R \subseteq \mathbb{R}^p$, $R \cap \mathcal{Y}_N = \emptyset$ if and only if either P is infeasible or there exists (x^*, z^*) that solves P and $d(f(x^*), f(z^*)) > 0$.*

Proof. ($\Rightarrow$) For a given rectangle $R \subseteq \mathbb{R}^p$, let P have no feasible solution. This implies that $R \cap \mathcal{Y} = \emptyset$. Since $\mathcal{Y}_N \subseteq \mathcal{Y}$, $R \cap \mathcal{Y}_N = \emptyset$. Suppose P is feasible and let (x^*, z^*) solve P with $d(f(x^*), f(z^*)) > 0$. Since $f(z^*) \leq f(x^*)$ and $d(f(x^*), f(z^*)) > 0$, z^* dominates x^*. This implies that $f(x^*) \notin \mathcal{Y}_N$. Assume that there exists $x' \in \mathcal{X}_E$ such that $f(x') \in R$. Then there exists $z' \in \Omega(x')$ such that $z' \in \mathcal{X}_E$. This implies that (x', z') is feasible to P. Since $f(z') \leq f(x')$ and $x' \in \mathcal{X}_E$, $f(x') = f(z')$. Then $d(f(x'), f(z')) = 0$ which contradicts optimality of (x^*, z^*) to P. Hence $R \cap \mathcal{Y}_N = \emptyset$. ($\Leftarrow$) For a given rectangle $R \subseteq \mathbb{R}^p$, let $R \cap \mathcal{Y}_N = \emptyset$. Since $\mathcal{Y}_N \subseteq \mathcal{Y}$, $R \cap \mathcal{Y}_N \subseteq R \cap \mathcal{Y}$. i) $R \cap \mathcal{Y} = \emptyset$. This implies that there exists no feasible solution that maps into R. Hence, inducible region of the bilevel formulation is empty set, and the bilevel formulation is infeasible. ii) $R \cap \mathcal{Y} \neq \emptyset$. Then for any $x' \in \mathcal{X}$ with $f(x') \in R \cap \mathcal{Y}$ there exist $z' \in \mathcal{X}_E$ such that z' dominates x'. Hence, (x', z') is feasible to P and $d(f(x'), f(z')) > 0$. □

In Theorem 20, we use the domination property. This property assumes that if a feasible solution x is dominated, then there exists an efficient solution z such that z dominates x [99].

In Theorems 19 and 20, we have concluded the following. For a given set $R \subseteq \mathbb{R}^p$, let P be feasible and (x^*, z^*) be an optimal solution. If $f(x^*) = f(z^*)$, than $x^* \in \mathcal{X}$

is mapped into R and x^* is an efficient solution, $f(x^*) \in R$ and $x^* \in \mathcal{X}_E$. If P is infeasible or $f(x^*) \neq f(z^*)$, than there exists no nondominated solution in R. These results are applicable to any kind of multiobjective optimization problem that satisfies the domination property. Note that this is not a restrictive assumption, since most of the multiobjective optimization problems satisfy this condition.

Remark: In Theorems 17-20, we do not use the properties of a rectangle R. Hence, the rectangle can be replaced with an arbitrary set S including a nonconvex set. In other words, the bilevel formulation can be used to find a nondominated solution in a given set S. In this case, the bilevel formulation P can be used in different problems of multiobjective optimization. Most typical way is to utilize P in interactive methods. In interactive methods, the decision maker is involved in the solution process and continuously interacts with the method to determine the most preferred solution [90]. The decision maker expresses preferences at each iteration in order to get efficient solutions that are of interest to him/her and learn what kind of solutions are attainable [140]. If the preferences of the decision maker can be expressed by using the set S in the upper level formulation of P, the bilevel program is able to determine whether the preferences of the decision maker is attainable or not. If there exist a feasible solution in the set in line with the decision maker's preferences, there may or may not be a nondominated solution in set S. If there exists no nondominated solution in S, than the resulting solution from S is the closest solution to the efficient set. The decision maker can either use this solution or update the preferences to obtain a nondominated solution. The proposed bilevel formulation to obtain a nondominated solution in set S can speedup the procedure and reduce the required involvement of the decision maker compared to conventional methods. Another application area to use proposed bilevel formulation is finding the nadir point for MOP. Nadir point is constructed from the worst objective function values over the efficient set, and determination of the nadir point is generally a hard problem [71]. Each component of the nadir point can be obtained by using bisection method

[191] with the bilevel formulation. Finally, bilevel formulation can be used to generate representative sets. In the following section, we propose an algorithm to generate representative sets with quality guarantees which uses P to obtain a nondominated solution in a partition. These imply that the bilevel programming formulation P is applicable to a wide range of problems in multiobjective optimization. However, even linear bilevel programming problems have been shown to be NP-hard and inapproximable within any constant factor in polynomial time [54]. Therefore solving P may be computationally demanding.

In this chapter, we scrutinize bilevel programs with linear objectives and constraints. We use penalty approach [132] and integer programming reformulation [84] to solve the bilevel linear programming problems. Both methods utilize KKT optimality conditions to turn the bilevel linear programming problem into a single level optimization problem [55] which is referred to as a mathematical program with complemantarity constraints [129]. In penalty approach, complemantarity constraints of KKT optimality conditions are moved into the objective function with a penalty parameter. In integer programming reformulation, complemantarity constraints are turned into linear constraints by using binary variables and a sufficiently large coefficient [14].

When we consider multiobjective linear programming problem, i.e. finding a nondominated solution in a rectangle R for MOLP, P turns into a bilevel linear programming problem. In bilevel formulation P. We use L_1-norm as a metric, because it can be linearized. The bilevel problem can be formulated with L_1-norm as follows,

$$
\begin{aligned}
(\boldsymbol{P'}) \quad \min_{x\in\mathcal{X}} \quad & f^U(x,z) = \sum_{j=1}^{p} |f_j(x) - f_j(z)| \\
\text{s.t.} \quad & g(x) \leq 0 \\
& f(x) \in R \\
& \min_{z\in\mathcal{X}} \quad f^L(z) = \sum_{j=1}^{p} w_j f_j(z) \\
& \text{s.t.} \quad g(z) \leq 0 \\
& \qquad f(z) \leq f(x).
\end{aligned}
$$

In MOLP, $c \in \mathbb{R}^{p\times n}$ represents the objective coefficients, $A \in \mathbb{R}^{m\times n}$ is technical coefficients, $b \in \mathbb{R}^m$ is the right hand side vector. $x, z \in \mathbb{R}^n$ are the decision vectors of the problem. The weights of the lower level problem, $w \in \mathbb{R}^p$, is set to $\mathbf{1} \in \mathbb{R}^{\mathbf{P}}$, i.e. $w = \mathbf{1}$. We define $c' \in \mathbb{R}^p$ such that $c' = \mathbf{1}^T c$. In upper level objective function of P', since $f(z) \leq f(x)$, we can eliminate the absolute value. Then the problem turns into a bilevel linear programming problem. The bilevel linear formulation of the problem with L_1-norm can be formulated as follows,

$$
\begin{aligned}
(\boldsymbol{P_L}) \quad \min_{x\in\mathcal{X}} \quad & f^U(x,z) = c'(x-z) \\
\text{s.t.} \quad & Ax \leq b \\
& l \leq cx \leq u \\
& \min_{z\in\mathcal{X}} \quad f^L(z) = c'z \\
& \text{s.t.} \quad Az \leq b \\
& \qquad cz \leq cx
\end{aligned}
$$

For the lower level formulation of P_L, we can write KKT optimality conditions to transform the problem into single level formulation. Let $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^p$ be the dual variables associated with the two sets of constraints. Then KKT conditions for

the lower level problem can be defined as follows,

- Stationarity

$$-c' + u^T A + v^T c = 0 \tag{6.1}$$

- Primal feasibility

$$\begin{aligned} Az - b &\leq 0 \\ cz - cx &\leq 0 \end{aligned} \tag{6.2}$$

- Dual feasibility

$$\begin{aligned} u &\geq 0 \\ v &\geq 0 \end{aligned} \tag{6.3}$$

- Complementary slackness

$$\begin{aligned} u(Az - b) &= 0 \\ v(cz - cx) &= 0 \end{aligned} \tag{6.4}$$

Among KKT optimality conditions stationarity and feasibility are linear in x, z, u and v. However, complementary slackness condition is bilinear. In integer programming reformulation, this nonlinerity is removed by using binary variables and a sufficiently large coefficient M. We define two binary vectors, $\psi \in \{0,1\}^m$ and $\omega \in \{0,1\}^p$, for two different constraint sets. Integer linear programming formulation of P_{IP} is as follows,

$$
\begin{aligned}
P_{IP} \quad \min \quad & c'(x-z) \\
\text{s.t.} \quad & Ax \leq b \\
& cx \geq l \\
& cx \leq u \\
& c' + u^T A + v^T c = 0 \\
& Az - b \leq 0 \\
& cz - cx \leq 0 \\
& Az - b \geq -M(\mathbf{1} - \psi) \\
& cz - cx \geq -M(\mathbf{1} - \omega) \\
& u \leq Mt \\
& v \leq Mz \\
& u, v \geq 0 \\
& \psi \in \{0,1\}^m \\
& \omega \in \{0,1\}^p
\end{aligned}
$$

The integer programming formulation includes additional $n+2m+2p$ constraints, $m+p$ non-negative, and binary variables. In the following section, we give an algorithm to generate representative sets for MOP problems. This algorithm needs to determine whether there exists a nondominated solution in a partition. Hence, we use bilevel programming formulation P in this method. Later on, this algorithm is used to generate representative sets for MOLP. BLP subproblems are solved by using penalty approach and P_{IP}.

6.4. Proposed Algorithm

In this chapter, we intend to generate representative sets for MOP problems with prespecified quality guarantees. For this problem, we are using rectangular bisection method which is widely used in global optimization problems [100]. The method starts with an initial rectangle that covers $\mathcal{Y}$. In each iteration, the algorithm picks a rectangle from the search list, and solves the bilevel formulation. If there exists an efficient solution that maps into the rectangle, then the algorithm subdivides the rectangle along the longest edge. If there exists no nondominated solution in the selected rectangle, algorithm removes the rectangle from the list. Algorithm is terminated when there exists no element in the list. We call this algorithm Representative Set Generation Algorithm (RSGA). The steps of RSGA is given below.

Steps of the RSGA

Input: Desired coverage error α, MOP, i.e. objective functions $f_j(x)$ for $j = 1, \ldots, p$, and feasible set $\mathcal{X}$.

Output: α-representation of the nondominated set ($\mathcal{Y}_R$) and α-approximation of $\mathcal{Y}_N$.

Step-0: $\mathcal{Y}_R = \emptyset$. Initialize the list $L = \{R\}$ such that $\mathcal{Y} \subseteq R$. Initialize approximation of nondominated set $L_R = \emptyset$.

Step-1: If L is empty, go to Step-4. Otherwise, pick the worst representative rectangle R_i from the list L. Solve bilevel formulation with $R_i \subseteq \mathbb{R}^p$ and $w > 0$. If there exists $y \in \mathcal{Y}_R$ such that $y \in R_i$, then go to Step-3.

Step-2:

- If bilevel program is feasible, let (x^*, z^*) be an optimal solution of P.
  - For all $S_t \in L$, if $S_t \subseteq R(y^I, f(z^*)) \cup R(f(z^*), y^U)$, than $L = L \setminus \{S_t\}$. Go to Step 3.
  - If $d(f(x^*), f(z^*)) = 0$, then $\mathcal{Y}_R = \mathcal{Y}_R \cup \{f(x^*)\}$. Go to Step 3.
  - If $d(f(x^*), f(z^*)) > 0$, then $L = L \setminus S_i$. Go to Step-1.
- Else (If bilevel program is infeasible), then $L = L \setminus S_i$. Go to Step-1.

Step-3: - Apply rectangular bisection for R_i and insert the refinement elements to L. Let $\hat{j} = \arg\max\{j \in \{1, \ldots, p\} : u_j^i - l_j^i\}$, $R^1 = \{y \in R_i : y_{\hat{j}} \leq (u_{\hat{j}}^i - l_{\hat{j}}^i)/2\}$, $R^2 = \{y \in R_i : y_{\hat{j}} \geq (u_{\hat{j}}^i - l_{\hat{j}}^i)/2\}$. $L = L \setminus R_i$. $L = L \cup \{R^1\} \cup \{R^2\}$.

- Check the coverage error. For each $R_t \in L$, if $\max\limits_{\bar{y} \in R_t} \min\limits_{y \in \mathcal{Y}_R} d(\bar{y}, y) \leq \alpha$, than $L = L \setminus R_t$. $L_R = L_R \cup R_t$. Go to Step-1.

Step-4: Return representative set $\mathcal{Y}_R$ and approximation of the nondominated set L_R.

RSGA starts to search the outcome space with a sufficiently large rectangle, and iteratively subdivides the initial rectangle by obtaining new nondominated solutions. Some of the subsets of the initial search space are removed if either there exists no nondominated solution in the rectangle or the rectangle satisfies the acceptable coverage error level α. In Theorem 21, we show that RSGA generates an α-representation of $\mathcal{Y}_N$. Additionally, we combine the rectangles that are eliminated with the coverage error to obtain an α-approximation of the nondominated set.

Theorem 21. *RSGA generates an α-representation of $\mathcal{Y}_N$.*

Proof. Any solution for the rational reaction set of the bilevel formulation is efficient. Therefore, all elements of $\mathcal{Y}_R$ are nondominated solutions. RSGA initializes the search with rectangle R such that the image of the feasible set in the outcome space is a subset of R, $\mathcal{Y} \subseteq R$. Hence, R includes all nondominated solutions of given MOP. In each iteration, bilevel program either finds a nondominated solution or proves that there exists no nondominated solution in a given partition of R. Hence, removed partitions by using bilevel formulation do not contain any nondominated solution. Another possibility of removing a partition is that if the partition is a subset of rectangle $R(y^I, f(z^*))$ or $R(f(z^*), y^U)$. By Theorem 17, z^* is an efficient solution. If there exist a solution $y \in R(y^I, f(z^*))$, than y dominates $f(z^*)$. This contradicts with the efficiency of z^*. Any solution $y \in R(f(z^*), y^U)$ is dominated by $f(z^*)$. Hence,

there exist no nondominated solution in $R(y^I, f(z^*))$ and $R(f(z^*), y^U)$. Finally, any partition of R can be removed from the list L, if the coverage error of the partition is less than or equal to α. Hence, at termination, RSGA generates an α-representation of $\mathcal{Y}_N$. □

6.5. Illustrative Example

RSGA is tested on a sample MOLP. In this example, we use P_{IP} formulation to find a solution to bilevel formulation P. We solve integer linear programming formulation by using CPLEX [47]. The MOLP test problem [26] is given in the following format.

$$\max\{Ix : Ax \leq b, x \geq 0\}$$

where I is $n \times n$ identity matrix,

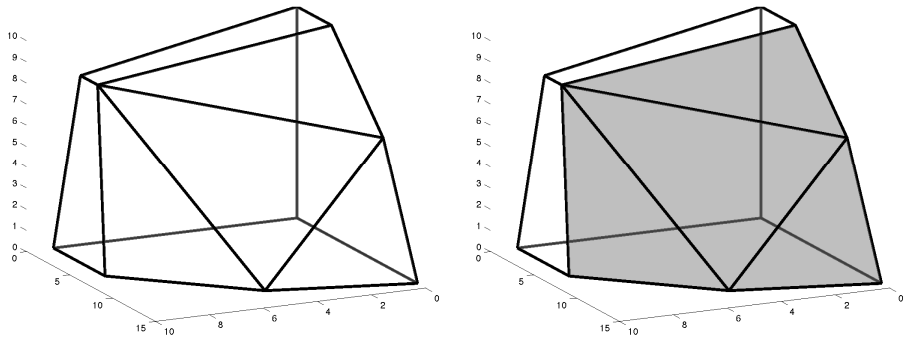
$$A = \begin{bmatrix} 6 & 15 & 10 \\ 5 & 8 & 12 \\ 22 & 29 & 28 \\ 24 & 16 & 11 \\ 1 & 0 & 4 \\ 8 & 0 & 1 \end{bmatrix}, \; b = \begin{bmatrix} 210 \\ 152 \\ 458 \\ 312 \\ 40 \\ 72 \end{bmatrix}.$$

In this example, since the objective vectors form an identity matrix, the outcome space and the decision space are the same. Hence, the image of the feasible set in the outcome space can be expressed as follows,

$$\mathcal{Y} = \{y \in \mathbb{R}^3 : Ay \leq b, y \geq 0\}.$$

The feasible set ($\mathcal{X}$) of the sample MOLP is given in Figure 6.1a. Since the outcome set is same as the feasible set, Figure 6.1a also shows the outcome space

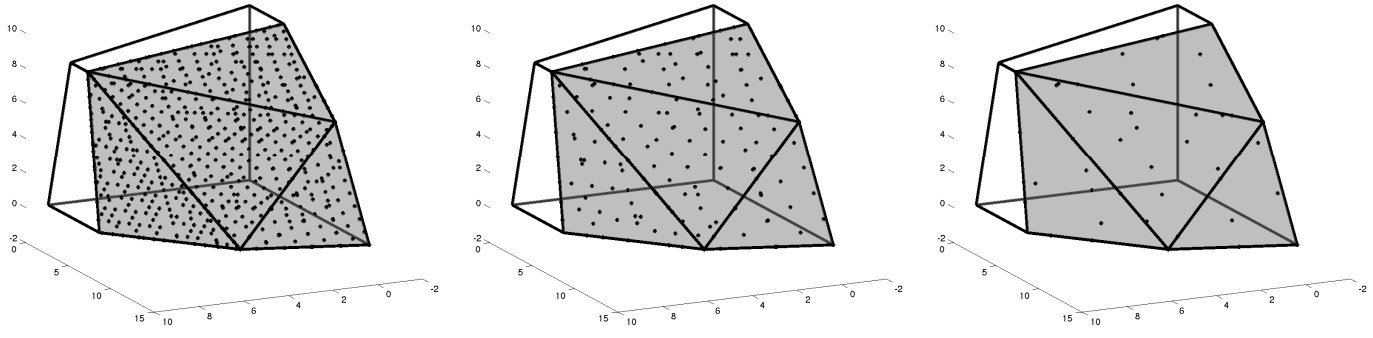
($\mathcal{Y}$). In MOLP, the union of efficient faces forms the nondominated set $\mathcal{Y}_N$. By using the algorithm in [159], we generate nondominated set of the sample MOLP which is shown in Figure 6.1b.



(a) Feasible set of the sample MOLP. (b) Nondominated set of the sample MOLP.

Figure 6.1: Feasible set (outcome space) and nondominated set of the sample MOLP.

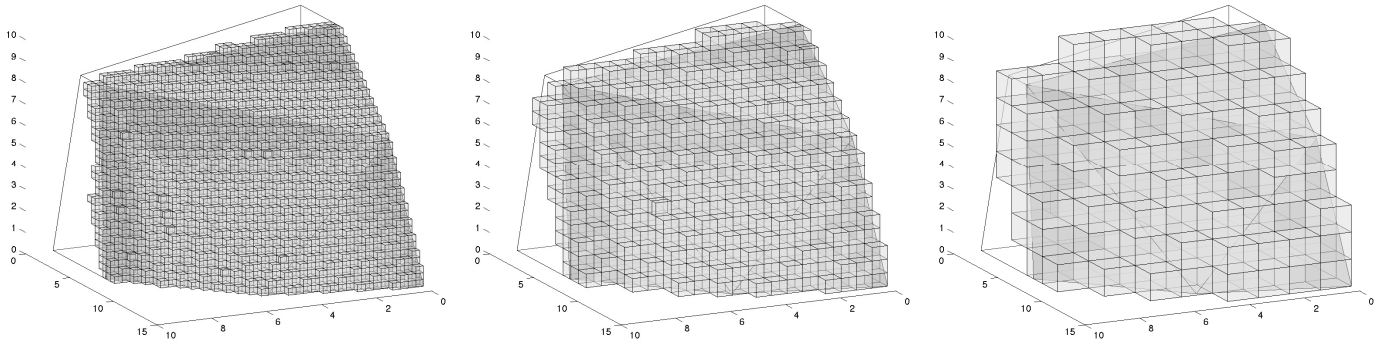
We test the RSGA with three different representation factors that are 5%, 10% and 20%. In Figure 6.2, representative sets for three different representation factors are given.



(a) $\alpha = 5\%$ (b) $\alpha = 10\%$ (c) $\alpha = 20\%$

Figure 6.2: Representative sets of the sample MOLP.

The cardinality of representative sets are 603, 163, and 43, respectively. RSGA solves 1017, 278, and 83 subproblems until termination. Proposed algorithm also generates approximation of the nondominated set. Here, we search over rectangles, so that the resulting approximation is union of rectangles. In Figure 6.3, approximation of the nondominated set with different factors are given.



(a) $\alpha = 5\%$ (b) $\alpha = 10\%$ (c) $\alpha = 20\%$

Figure 6.3: Approximation of the nondominated set with union of rectangles.

The proposed algorithm generates a representative set $\mathcal{Y}_R$ that satisfies the specified coverage error level α. Still, it is interesting to compute the actual representation factor of $\mathcal{Y}_R$. We obtained the representative set and nondominated set of the sample MOLP, so we can calculate the coverage error level of the representative sets.

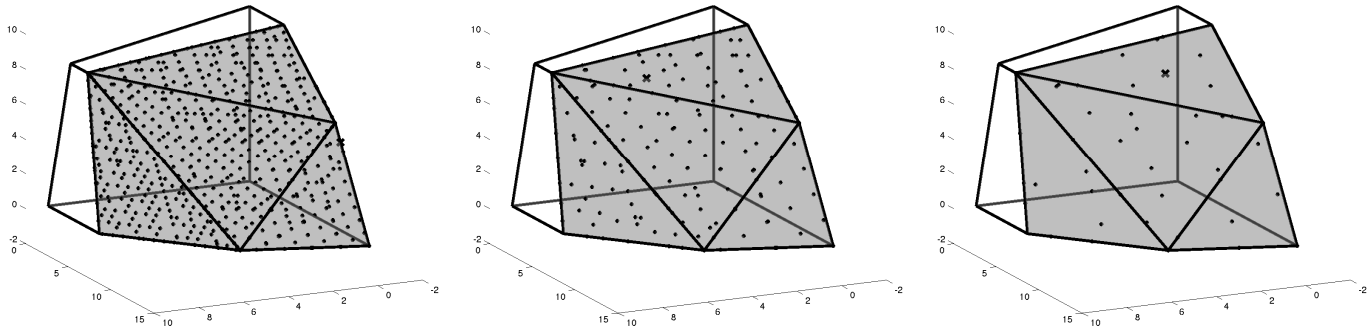
Given nondominated set $\mathcal{Y}_N$ and representative set $\mathcal{Y}_R$ of a sample MOLP, coverage error level of the representative set can be computed with the following integer linear programming problem [160]. In this formulation, N represents the cardinality of the representative set.

$$
\begin{aligned}
P_D(L_\infty) \quad \max \quad & \gamma \\
\text{s.t.} \quad & \gamma - d^i \le 0 && i = 1, \ldots, N \\
& d^i = (z_j + u_j^i - \bar{y}_j^i)/(y_j^N - y_j^I) && j = 1, \ldots, p;\ i = 1, \ldots, N \\
& d^i = (-z_j + o_j^i + \bar{y}_j^i)/(y_j^N - y_j^I) && j = 1, \ldots, p;\ i = 1, \ldots, N \\
& u_j^i - M t_j^i \le 0 && j = 1, \ldots, p;\ i = 1, \ldots, N \\
& o_j^i - M s_j^i \le 0 && j = 1, \ldots, p;\ i = 1, \ldots, N \\
& \sum_{j=1}^{p} (t_j^i + s_j^i) \le 2p - 1 && i = 1, \ldots, N \\
& Ax \le b \\
& A^{I_k} x \ge b^{I_k} - M(1 - v_k) && k = 1, \ldots, K \\
& \sum_{k=1}^{K} v_k \ge 1 \\
& z_j = c_j x && j = 1, \ldots, p \\
& \gamma \ge 0 \\
& u_j^i, o_j^i \ge 0 && j = 1, \ldots, p;\ i = 1, \ldots, N \\
& v_k \in \{0, 1\} && k = 1, \ldots, K \\
& t_j^i, s_j^i \in \{0, 1\} && j = 1, \ldots, p;\ i = 1, \ldots, N \\
& d_i \ge 0 && i = 1, \ldots, N
\end{aligned}
$$

In this problem, we include K binary variables where K represents the number of efficient faces. For the efficient face F_k, we define an index set I_k that has the indices of the binding constraints. F_k can be formulated as follows,

$$F_k = \{x \in \mathbb{R}^n : Ax \leq b, A^{I_k}x = b^{I_k}\}.$$

This formulation is solved by using CPLEX [47] to calculate the coverage error level of the three different representations obtained before. The optimal objective values of the formulation, i.e. actual coverage error levels, are 4.63% for $\alpha = 5\%$, 7.22% for $\alpha = 10\%$, and 15.56% for $\alpha = 20\%$. In Figure 6.4, we show the worst represented point of the nondominated set.



(a) $\alpha = \%5$ (b) $\alpha = \%10$ (c) $\alpha = \%20$

Figure 6.4: Worst representative points of the nondominated sets.

On sample MOLP, we observe that representative set is well-dispersed over the nondominated set (uniformity), and the number of nondominated solutions (cardinality) in the representative set decreases significantly when we double the coverage error level. Additionally, coverage error level of the resulting representative set is close to the desired coverage level α. In the following section, we conduct further tests on MOLP instances.

6.6. Computational Results

RSGA is used to generate representative sets of randomly generated MOLP instances. Here, we test the effects of the problem size and number of objectives on the proposed algorithm. In these tests, bilevel programs are solved by using penalty approach and integer programming reformulation. In penalty approach, complemantarity constraints are taken into the objective function. We use GAMS/NLPEC [82] solver to reformulate the problem with a penalty approach. After the reformulation, there is a bilinear term in the objective function, so the resulting formulation is a global optimization problem [100]. In RSGA, the solution of the bilevel program should be globally optimal. Hence, we use GAMS/BARON [177] to obtain a global optimal solution for the penalty approach reformulation. We also use integer programming reformulation P_{IP} to deal with the bilevel program. Integer programming formulations are solved by using IBM CPLEX 12.4 [47].

RSGA is implemented in C++. All tests were conducted on a shared cluster with Intel Xeon 2.3 GHz CPU and 4 GB memory limit with Linux operating system. Different MOLP categories are generated based on problem size, and 10 instances are generated randomly for each problem category. The average over 10 instances is reported. Computation of each instance is interrupted after 25,000 CPU seconds. A blank cell in a table indicates that none of the 10 instances could be completed within the time limit.

In MOLP, m and n represent the number of constraints and number of variables, respectively, and x is the decision vector of the problem. Given coefficients of the objective functions c_l^j, the technical coefficients a_{rl}, and right-hand side values b_r where $r \in \{1, \ldots, m\}$, $l \in \{1, \ldots, n\}$, and $j \in \{1, \ldots, p\}$, MOLP problem is defined as follows.

$$
\begin{aligned}
(MOLP) \quad \max \quad & \sum_{l=1}^{n} c_l^j x_l \quad j = 1, \ldots, p \\
\text{s.t.} \quad & \sum_{l=1}^{n} a_{rl} x_l \leq b_r \quad r = 1, \ldots, m \\
& x_l \geq 0 \quad l = 1, \ldots, n.
\end{aligned}
$$

We generate two sets of MOLP instances. In the first set instances, we want to test the effect of the problem size on RSGA. Hence, for this set of instances, the number of objectives is 3, and number of variables is $n \in \{5, 10, 15, 20, 25\}$ and $m = n$. In the second set of instances, we test the effects of the number of objectives on RSGA. Therefore, for a fixed problem size, $n = 15$ and $m = 5$, the instances are generated with $p = \{2, 3, 4, 5\}$. The parameters of the model are randomly generated integer numbers with ranges similar to used in [5]. The coefficients of the objective functions (c_l^j) are generated in the ranges $[-100, -1]$ and $[0, 100]$ with probability 0.2 and 0.8, respectively. The technical coefficients (a_{rl}) are generated in the ranges $[-100, -1]$ with probability 0.1, $[1, 100]$ with probability 0.8, and $a_{rl} = 0$ with probability 0.1. Finally, right-hand side value (b_r) of each constraint is also generated randomly in the range of 100 and $\sum_{l=1}^{n} a_{rl}$. In these tests, the coverage error level is defined as 10%, i.e. $\alpha = 10\%$. Test results of RSGA on the first set of instances are given in Table 6.1.

As seen in Table 6.1, while RSGA with ILP formulation is able to generate all representative sets in a given time limit, penalty approach cannot handle test problems with 15 variables or more. Increase in problem size does not change the number of solved models and size of the representative set statistics significantly. On the other hand, CPU time increases with increase in the problems size. This occurs due to subproblem solution time. In Table 6.2, effects of number objectives on RSGA is tested. Since ILP reformulation perform better than penalty approach, we only test

Table 6.1: RSGA with penalty approach and integer programming reformulation tests on MOLP.

	RSGA with Penalty Approach			**RSGA with ILP Reformulation**		
$m \times n$	$\lvert\mathcal{Y}_R\rvert$	#Models	CPU Time (s)	$\lvert\mathcal{Y}_R\rvert$	#Models	CPU Time (s)
5×5	104.9	245.2	49.2	107.6	250.8	6.1
10×10	84.6	198.9	1251.9	88.0	206.9	27.3
15×15	116.5	227.4	10652.1	124.2	244.3	151.2
20×20				116.0	232.4	919.1
25×25				122.1	241.6	1581.5
30×30				122.0	247.9	2874.8

the RSGA with ILP formulation.

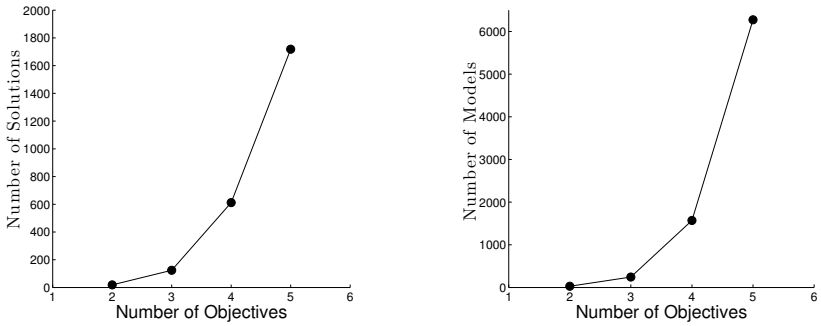
Table 6.2: Testing the number of objectives of MOLP on RSGA with integer programming reformulation.

	RSGA with ILP Reformulation		
p	$\lvert\mathcal{Y}_R\rvert$	#Models	CPU Time (s)
2	19.0	29.7	12.9
3	124.2	244.3	136.9
4	612.0	1571.4	1004.8
5	1718.5	6274.5	8778.3

Increase in the number of objectives effects the solution time of RSGA significantly. The number of solutions in $\mathcal{Y}_R$, number of models solved and solution time statistics increase exponentially with increase in the number of objectives as seen in Figure 6.5. The reason of exponential escalation is complexity of MOP.

6.7. Application to SVM Classification for Imbalanced Data Sets

RSGA can be used to generate a representative set for any MOLP with any number of objective functions. In this section, we apply RSGA to obtain better accuracy



(a) Number of models solved. (b) CPU Time (s).

Figure 6.5: Effect of number objectives on RSGA with ILP formulation.

for SVM classification on imbalanced data sets problem. Askan and Sayin model the SVM classification problem for imbalanced data sets as a three-objective linear programming problem [10]. They then solve this problem heuristically and report findings on some sample problems. Hence, we can apply RSGA to obtain the representative set for three-objective SVM formulation. Since representative method generates a set which is well-dispersed over the nondominated set, we may expect better accuracy than their classifiers.

Classification is the process of assigning data to one of a set of predetermined class labels. In imbalanced data sets, negative instances outnumber the positive instances. SVM constructs a hyperplane that separates the data into two classes [173]. On a collection of examples $\mathcal{D} = \{(x_i, y_i) : x_i \in R^n, y_i \in \{1, 1\}\}$, classical L_2-norm SVM is formulated as follows.

$$\begin{aligned} \min \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i(w \cdot x_i - b) \geq 1 - \xi_i \quad 1, \ldots, N \\ & \xi_i \geq 0 \quad i = 1, \ldots, N \end{aligned}$$

In this formulation, the goal of the L_2-norm minimization is to maximize the margin of the separating hyperplane, C is a positive coefficient, and slack variables ξ_i measure the degree of misclassification of the data x_i.

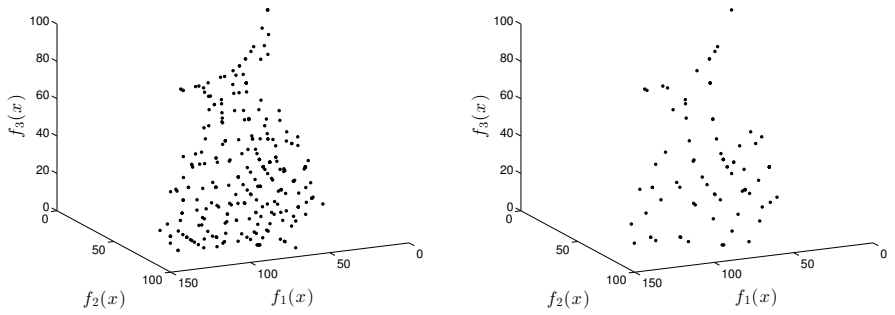
When there is an imbalanced distribution in the data set a typical classifier would be biased towards one class because it has the goal of maximizing overall accuracy. Unlike classical SVM formulation, they consider positive and negative error sums in different objective functions to eliminate the bias [10]. Their formulation incorporates L_1-norm minimization with the error sums for the two classes independently. Three-objective linear programming formulation is given below.

$$
\begin{aligned}
(\textbf{SVM-3C}) \quad \min \quad & \mathbf{1}^T(w^+ - w^-),\ \sum_i \xi_i^-,\ \sum_i \xi_i^+ \\
\text{s.t.} \quad & y_i((w^+ - w^-)^T x_i + b) \geq 1 - \xi_i^- \quad i = 1, \ldots, N^- \\
& y_i((w^+ - w^-)^T x_i + b) \geq 1 - \xi_i^+ \quad i = N^- + 1, \ldots, N \\
& \xi_i^- \geq 0 \quad i = 1, \ldots, N^- \\
& \xi_i^+ \geq 0 \quad i = N^- + 1, \ldots, N \\
& w^+,\ w^- \geq 0
\end{aligned}
$$

Above, $\mathbf{1} \in \mathbb{R}^{\mathbf{n}}$ is the vector of 1s, N^- is the number of instances that belong to the majority class, N^+ is the number of instances that belong to the minority class and $N = N^- + N^+$ is the number of all instances in the data set. The separating hyperplane is defined by the vector $w = w^+ - w$, and therefore the first objective function is minimization of L_1-norm of the normal vector.

We utilize RSGA to determine the classifiers for SVM-3C problem. We test the method on the data set named Yeast. The training set includes 250 negative instances ($N^- = 250$) and 75 positive instances ($N^+ = 75$). The classifiers are tested on a test set which includes 213 negative instances and 88 positive instances. We test

RSGA with two different representation factors that are 5% and 10%. In Figure 6.6, representative sets of three objective linear programming problems are given.



(a) $\alpha = 5\%$ (b) $\alpha = 10\%$

Figure 6.6: Representative sets of SVM-3C.

For coverage error levels of $\alpha = 5\%$ and $\alpha = 10\%$, the numbers of nondominated solutions are $|\mathcal{Y}_R| = 279$ and $|\mathcal{Y}_R| = 83$, and the number of solved models are 558 and 187, respectively. Each nondominated solution in the representative set is a classifier. To compute the performance of each classifier, we need to compute the number of true positive (TP), false negative (FN), true negative (TN) and false positive (FP) instances in the test set. By using these statistics, we can compute sensitivity and specificity of a classifier as follows.

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

The performance of the classifier is computed by the geometric mean of sensitivity and specificity values. RSGA generates α-representation of SVM-3C's nondominated set. Each solution in the representative set is a classifier. Hence, we have tens of

classifiers. On the other hand, traditional methods in SVM generally finds only one classifier that separates the positive and negative instances. We use validation set to decrease the number of classifiers to one. The test set is randomly split into two equal parts to obtain a validation set and test set. For Askan and Sayin's method and RSGA, the best performing classifier with respect to geometric means is determined over the validation set. Then, this single classifier is evaluated over the test set. In Table 6.3, accuracy results of Askan and Sayin's procedure and RSGA are reported on validations and test sets of Yeast data.

Table 6.3: Comparing the accuracy results of RSGA and Askan and Sayin's procedure on Yeast data set.

	RSGA				**Askan and Sayin**	
	Validation Set		Test Set		Test Set	
	Gmeans	Sensitivity	Gmeans	Sensitivity	Gmeans	Sensitivity
$\alpha = 5\%$	1	1	1	1	0.924926	0.954545
$\alpha = 10\%$	0.977008	0.954545	0.916075	0.863636	0.924926	0.954545

As seen in Table 6.3, RSGA generates the best performing possible classifier with $\alpha = 5\%$ which outperforms Askan and Sayin's accuracy result. For $\alpha = 10\%$, the performance of Askan and Sayin's method is better than RSGA. Still, the accuracy result of RSGA is close to Askan and Sayin's result.

6.8. Conclusion

We present a bilevel programming formulation to obtain a nondominated solution in a given rectangle. This result is applicable to different problems in multiobjective optimization literature. By using this result, we give a representative algorithm for multiobjective optimization problems. The representative algorithm can be applied to any MOP with any number of objective functions. The proposed algorithm is tested

on MOLP instances with various sizes. Finally, we apply this method to obtain better accuracy for SVM on imbalanced data sets. Solving bilevel optimization problems is challenging, and computational results validate this statement. As a future work, new exact or heuristic approaches can be developed for bilevel optimization to enhance the performance of the proposed algorithm.

Chapter 7

CONCLUSION

In this thesis, we develop exact and representation methods for multiobjective discrete optimization problems and general multiobjective optimization problems with any number of objectives. In Chapter 3, we proposed an algorithm to solve multiobjective discrete optimization problems with any number of objective functions. We showed that the algorithm can generate the entire nondominated set in a finite number of steps. The proposed method uses the ε-constraint scalarization and is based on a partitioning mechanism that searches the $(p-1)$-dimensional space exhaustively. The proposed method is compared with previous algorithms, and is seen to outperform all of them on the experimented problem instances. The number of models solved per nondominated solution may be a better comparison criterion for the scalarization methods than the computation time statistics. The proposed algorithm solved at most 1.99 subproblems per nondominated solution on the test problems. This ratio is highly competitive compared to previous studies. Nevertheless, as problem size increases and the number of nondominated solutions grows, the requirements of the approach become unrealistically high. We modify the algorithm to generate representations of the nondominated set with desired quality guarantees. This algorithm searches the outcome space with p-dimensional rectangles, and solve two-stage optimization problems to find nondominated solutions. During the search, any rectangle that satisfies the desired coverage error level is removed. We have shown that the proposed algorithm generates an α-representation of MODO nondominated set in finite number of iterations. The algorithm is tested on multiobjective knapsack and multiobjective assignment problem instances. The proposed algorithm is able to

generate representation efficiently. However, when we increase the desired coverage error level, the performance of the method diminishes due to navigation problem in the outcome space. To overcome this problem, we model the problem of finding a nondominated solution in a given set as a bilevel programming problem. We have shown that the bilevel formulation finds a nondominated solution in the rectangle if such a solution exists. By using this result, we give a representative algorithm for multiobjective optimization problems. The proposed algorithm is tested on multiobjective linear programming problem with various sizes. Finally, we apply this method to obtain better accuracy for support vector machine on imbalanced data sets. Solving bilevel optimization problems is challenging, and computational results validate this statement. As future work, new exact or heuristic approaches can be developed for bilevel optimization to enhance the performance of the proposed algorithm.

We also study the problem of finding the nadir point of multiobjective discrete optimization problems. Along with the ideal point these points define bounds of the efficient set. They can be used in representation algorithms instead of heuristic estimates. The proposed nadir point determination algorithm is based on an exhaustive search of the $(p-2)$-dimensional space. The proposed algorithm guarantees to find the nadir point exactly in a finite number of steps. The proposed algorithm is compared with previous algorithms. In computational results, we see that the algorithm outperforms former methods except for the multiobjective discrete optimization problems with five objectives instances.

BIBLIOGRAPHY

[1] M. Abbas and D. Chaabane. Optimizing a linear function over an integer efficient set. *European Journal of Operational Research*, 174(2):1140–1161, 2006.

[2] M. Abbas, M. E.-A. Chergui, and M. A. Mehdi. Efficient cuts for generating the non–dominated vectors for multiple objective integer linear programming. *International Journal of Mathematics in Operational Research*, 4(3):302–316, 2012.

[3] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *Machine Learning: ECML 2004*, pages 39–50. Springer, 2004.

[4] M. Alves and J. Climaco. A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, 180(1):99–115, 2007.

[5] M. J. Alves and J. P. Costa. An exact method for computing the nadir values in multiple objective linear programming. *European Journal of Operational Research*, 198(2):637–646, 2009.

[6] L. T. H. An, P. D. Tao, and L. D. Muu. Numerical solution for optimization over the efficient set by dc optimization algorithms. *Operations Research Letters*, 19(3):117–128, 1996.

[7] K. Anagnostopoulos and G. Mamanis. A portfolio optimization model with three objectives and discrete variables. *Computers & Operations Research*, 37(7):1285–1297, 2010.

[8] Y. P. Aneja and K. P. K. Nair. Bicriteria transportation problem. *Management Science*, 25(1):73–78, 1979.

[9] R. Armananzas and J. A. Lozano. A multiobjective approach to the portfolio optimization problem. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1388–1395. IEEE, 2005.

[10] A. Askan and S. Sayin. Svm classification for imbalanced data sets using a multiobjective optimization framework. *Annals of Operations Research*, 216(1):191–203, 2014.

[11] H. Aytug and S. Sayın. Exploring the trade-off between generalization and empirical errors in a one-norm svm. *European Journal of Operational Research*, 218(3):667–675, 2012.

[12] T. P. Bagchi. *Multiobjective Scheduling by Genetic Algorithms*. Springer, 1999.

[13] J. Bard. Some properties of the bilevel programming problem. *Journal of Optimization Theory and Applications*, 68(2):371–378, 1991.

[14] J. F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*, volume 30 of *Nonconvex Optimization and Its Applications*. Springer, 1998.

[15] J. F. Bard and J. T. Moore. An algorithm for the discrete bilevel programming problem. *Naval Research Logistics (NRL)*, 39(3):419–435, 1992.

[16] J. F. Bard, J. Plummer, and J. Claude Sourie. A bilevel programming approach to determining tax credits for biofuel production. *European Journal of Operational Research*, 120(1):30–46, 2000.

[17] C. Bazgan, H. Hugot, and D. Vanderpooten. Solving efficiently the $0-1$ multi-objective knapsack problem. *Computers & Operations Research*, 36:260–279, 2009.

[18] A. Ben-Tal. Characterization of pareto and lexicographic optimal solutions. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making Theory and Application*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 1–11. Springer Berlin Heidelberg, 1980.

[19] A. Ben-Tal and S. Zlobe. Convex programming and the lexicographic multicriteria problem. *Mathematische Operationsforschung und Statistik. Series Optimization*, 8(1):61–73, 1977.

[20] H. Benson. Vector maximization with two objective functions. *Journal of Optimization Theory and Applications*, 28(2):253–257, 1979.

[21] H. P. Benson. Optimization over the efficient set. *Journal of Mathematical Analysis and Applications*, 98(2):562–580, 1984.

[22] H. P. Benson. An all-linear programming relaxation algorithm for optimizing over the efficient set. *Journal of Global Optimization*, 1(1):83–104, 1991.

[23] H. P. Benson. A finite, nonadjacent extreme-point search algorithm for optimization over the efficient set. *Journal of Optimization Theory and Applications*, 73(1):47–64, 1992.

[24] H. P. Benson. A bisection-extreme point search algorithm for optimizing over the efficient set in the linear dependence case. *Journal of Global Optimization*, 3(1):95–111, 1993.

[25] H. P. Benson, D. Lee, and J. P. McClure. Global optimization in practice: An application to interactive multiple objective linear programming. *Journal of Global Optimization*, 12(4):353–372, 1998.

[26] H. P. Benson and S. Sayin. Towards finding global representations of the efficient set in multiple objective mathematical programming. *Naval Research Logistics*, 44(1):47–67, 1997.

[27] H. P. Benson and E. Sun. A weight set decomposition algorithm for finding all efficient extreme points in the outcome set of a multiple objective linear program. *European Journal of Operational Research*, 139(1):26–41, 2002.

[28] D. Bertsimas, V. Cacchiani, D. Craft, and O. Nohadani. A hybrid approach to beam angle optimization in intensity-modulated radiation therapy. *Computers & Operations Research*, 40(9):2187–2197, 2013.

[29] J. F. Bérubé, M. Gendreau, and J. Y. Potvin. An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *European Journal of Operational Research*, 194(1):39–50, 2009.

[30] G. R. Bitran and J. M. Rivera. A combined approach to solve binary multicriteria problems. *Naval Research Logistics Quarterly*, 29(2):181–201, 1982.

[31] H. L. Bodlaender, P. Gritzmann, V. Klee, and J. Leeuwen. Computational complexity of norm-maximization. *Combinatorica*, 10(2):203–225, 1990.

[32] S. Bolintineanu. Minimization of a quasi-concave function over an efficient set. *Mathematical Programming*, 61(1):89–110, 1993.

[33] H. Bonnel and C. Yalçın Kaya. Optimization over the efficient set of multiobjective convex optimal control problems. *Journal of Optimization Theory and Applications*, 147(1):93–112, 2010.

[34] V. J. Bowman. On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In H. Thiriez and S. Zionts, editors, *Multiple Criteria Decision Making*, volume 130 of *Lecture Notes in Economics and Mathematical Systems*, chapter 53, pages 76–85. Springer-Verlag, Berlin, Germany, 1976.

[35] J. Bracken and J. T. McGill. Defense applications of mathematical programs with optimization problems in the constraints. *Operations Research*, 22(5):1086–1096, 1974.

[36] J. Branke, B. Scheckenbach, M. Stein, K. Deb, and H. Schmeck. Portfolio optimization with an envelope-based multi-objective evolutionary algorithm. *European Journal of Operational Research*, 199(3):684–693, 2009.

[37] R. Caballero and M. Hernández. The controlled estimation method in the multiobjective linear fractional problem. *Computers & Operations Research*, 31(11):1821–1832, 2004.

[38] M. E. Captivo, J. Clímaco, J. R. Figueira, E. Martins, and J. L. Santos. Solving bicriteria $0-1$ knapsack problems using a labeling algorithm. *Computers & Operations Research*, 30(12):1865–1886, 2003.

[39] V. Chankong and Y. Y. Haimes. On the characterization of noninferior solutions of the vector optimization problem. *Automatica*, 18(6):697–707, 1982.

[40] V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making Theory and Methodology*. Elsevier Science, New York, 1983.

[41] C.-W. Chen* and D. Sha. Heuristic approach for solving the multi-objective facility layout problem. *International Journal of Production Research*, 43(21):4493–4507, 2005.

[42] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-objective Problems*, volume 5. Springer-Verlag New York Inc, 2007.

[43] J. Cohon, R. Church, and D. Sheer. Generating multiobjective trade-offs: an algorithm for bicriterion problems. *Water Resources Research*, 15(5):1001–1010, 1979.

[44] B. Colson, P. Marcotte, and G. Savard. Bilevel programming: A survey. *4OR*, 3(2):87–107, 2005.

[45] B. Colson, P. Marcotte, and G. Savard. A trust-region method for nonlinear bilevel programming: algorithm and computational experience. *Computational Optimization and Applications*, 30(3):211–227, 2005.

[46] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[47] Cplex. IBM ILOG Cplex Optimization Studio 12.4, 2012.

[48] K. Daechert and K. Klamroth. A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems. *Journal of Global Optimization*, pages 1–34, 2014.

[49] I. Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[50] J. P. Dauer. Optimization over the efficient set using an active constraint approach. *Mathematical Methods of Operations Research*, 35(3):185–195, 1991.

[51] J. P. Dauer and T. A. Fosnaugh. Optimization over the efficient set. *Journal of Global Optimization*, 7(3):261–277, 1995.

[52] K. Deb and K. Miettinen. Nadir point estimation using evolutionary approaches: Better accuracy and computational speed through focused search. In M. Ehrgott, B. Naujoks, T. J. Stewart, and J. Wallenius, editors, *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, volume 634 of *Lecture Notes in Economics and Mathematical Systems*, pages 339–354. Springer Berlin Heidelberg, 2010.

[53] S. Dempe. A bundle algorithm applied to bilevel programming problems with non-unique lower level solutions. *Computational Optimization and Applications*, 15(2):145–166, 2000.

[54] S. Dempe. *Foundations of Bilevel Programming*. Springer, 2002.

[55] S. Dempe and A. B. Zemkoho. On the karush–kuhn–tucker reformulation of the bilevel optimization problem. *Nonlinear Analysis: Theory, Methods & Applications*, 75(3):1202–1218, 2012.

[56] S. DeNegre and T. K. Ralphs. A branch-and-cut algorithm for integer bilevel linear programs. In *Operations Research and Cyber-Infrastructure*, pages 65–78. Springer, 2009.

[57] C. Dhaenens, J. Lemesre, and E. G. Talbi. K-ppm: A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200(1):45–53, 2010.

[58] M. Didi-Biha, P. Marcotte, and G. Savard. Path-based formulations of a bilevel toll setting problem. In S. Dempe and V. Kalashnikov, editors, *Optimization with Multivalued Mappings*, volume 2 of *Springer Optimization and Its Applications*, pages 29–50. Springer US, 2006.

[59] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer. Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, 131(1-4):79–99, 2004.

[60] A. Durso. An interactive combined branch-and-bound/tchebycheff algorithm for multiple criteria optimization. In A. Goicoechea, L. Duckstein, and S. Zionts, editors, *Multiple Criteria Decision Making: Theory and Applications in Business, Industry and Government*, pages 107–122. Springer-Verlag, New York, 1992.

[61] J. G. Ecker and I. A. Kouada. Finding all efficient extreme points for multiple objective linear programs. *Mathematical Programming*, 14(1):249–261, 1978.

[62] J. G. Ecker and J. H. Song. Optimizing a linear function over an efficient set. *Journal of Optimization Theory and Applications*, 83(3):541–563, 1994.

[63] H. G. Eggleston. *Convexity*. Cambridge University Press, 1958.

[64] M. Ehrgott. Approximation algorithms for combinatorial multicriteria optimization problems. *International Transactions in Operational Research*, 7(1):5–31, 2000.

[65] M. Ehrgott. *Multicriteria Optimization*. Springer, Berlin, 2005.

[66] M. Ehrgott. A discussion of scalarization techniques for multiple objective integer programming. *Annals of Operations Research*, 147(1):343–360, 2006.

[67] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22:425–460, 2000.

[68] M. Ehrgott and X. Gandibleux. Multiobjective combinatorial optimization – theory, methodology, and applications. In M. Ehrgott and X. Gandibleux, editors, *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, volume 52, chapter 8, pages 369–444. Kluwer Academic Publishers, 2003.

[69] M. Ehrgott and X. Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *Sociedad de Estadística e Investigación Operativa*, 12(1):1–89, 2004.

[70] M. Ehrgott and X. Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *Top*, 12(1):1–63, 2004.

[71] M. Ehrgott and D. Tenfelde-Podehl. Computation of ideal and nadir values and implications for their use in MCDM methods. *European Journal of Operational Research*, 151(1):119–139, 2003.

[72] G. Eichfelder. *Adaptive Scalarization Methods in Multiobjective Optimization*. Springer, 2008.

[73] G. Eichfelder. An adaptive scalarization method in multiobjective optimization. *SIAM Journal on Optimization*, 19(4):1694–1718, 2009.

[74] G. Eichfelder. A constraint method in nonlinear multi-objective optimization. In *Multiobjective Programming and Goal Programming*, pages 3–12. Springer, 2009.

[75] G. Eichfelder. Multiobjective bilevel optimization. *Mathematical Programming*, 123(2):419–449, 2010.

[76] P. K. Eswaran, A. Ravindran, and H. Moskowitz. Algorithms for nonlinear integer bicriterion problems. *Journal of Optimization Theory and Applications*, 63:261–279, 1989.

[77] A. Eusébio, J. R. Figueira, and M. Ehrgott. On finding representative non-dominated points for bi-objective integer network flow problems. *Computers & Operations Research*, 48:1–10, 2014.

[78] G. W. Evans. An overview of techniques for solving multiobjective mathematical programs. *Management Science*, 30(11):1268–1282, 1984.

[79] S. L. Faulkenberg. *Quality representation in multiobjective programming*. PhD thesis, Clemson University, 2009.

[80] S. L. Faulkenberg and M. M. Wiecek. On the quality of discrete representations

in multiple objective programming. *Optimization and Engineering*, 11(3):423–440, 2010.

[81] S. L. Faulkenberg and M. M. Wiecek. Generating equidistant representations in biobjective programming. *Computational Optimization and Applications*, 51(3):1173–1210, 2012.

[82] M. C. Ferris and T. S. Munson. Complementarity problems in GAMS and the PATH solver. *Journal of Economic Dynamics and Control*, 24:165–188, 2000.

[83] J. R. Figueira, G. Tavares, and M. M. Wiecek. Labeling algorithms for multiple objective integer knapsack problems. *Computers & Operations Research*, 37(4):700–711, 2010.

[84] J. Fortuny-Amat and B. McCarl. A representation and economic interpretation of a two-level programming problem. *The Journal of the Operational Research Society*, 32(9):783–792, 1981.

[85] Y. Fu and U. M. Diwekar. An efficient sampling approach to multiobjective optimization. *Annals of Operations Research*, 132(1-4):109–134, 2004.

[86] M. Fukushima and J.-S. Pang. Convergence of a smoothing continuation method for mathematical progams with complementarity constraints. In *Ill-posed Variational Problems and Regularization Techniques*, pages 99–110. Springer, 1999.

[87] X. Gandibleux, F. Beugnies, and S. Randriamasy. Martins' algorithm revisited for multi-objective shortest path problems with a maxmin cost function. *4OR*, 4(1):47–59, 2006.

[88] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman San Francisco, CA, 1979.

[89] A. M. Geoffrion. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22:618–630, 1968.

[90] A. M. Geoffrion, J. S. Dyer, and A. Feinberg. An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Management Science*, 19(4):357–368, 1972.

[91] J. Granat and F. Guerriero. The interactive analysis of the multicriteria shortest path problem by the reference point method. *European Journal of Operational Research*, 151(1):103–118, 2003.

[92] J. Guddat, F. Guerra Vasquez, K. Tammer, and K. Wendler. *Multiobjective and Stochastic Optimization Based on Parametric Optimization*. Mathematical Research. Berlin: Akademie-Verlag., 1985.

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

[94] H. W. Hamacher. A note on K best network flows. *Annals of Operations Research*, 57(1):65–72, 1995.

[95] H. W. Hamacher, C. R. Pedersen, and S. Ruzika. Finding representative systems for discrete bicriterion optimization problems. *Operations Research Letters*, 35(3):336–344, 2007.

[96] P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multiple criteria decision making: Theory and applications*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 109–127, Berlin, 1980. Springer, Heidelberg, Germany.

[97] F. Hausdorff. *Set Theory*. Chelsea Publ. Co., New York, 1962.

[98] S. Helbig. On a constructive approximation of the efficient outcomes in bicriterion vector optimization. *Journal of Global Optimization*, 5(1):35–48, 1994.

[99] M. I. Henig. The domination property in multicriteria optimization. *Journal of Mathematical Analysis and Applications*, 114(1):7–16, 1986.

[100] R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches*. Springer, 1996.

[101] H. Isermann. The enumeration of the set of all efficient solutions for a linear multiple objective program. *Operational Research Quarterly*, pages 711–725, 1977.

[102] H. Isermann and R. E. Steuer. Computational experience concerning payoff tables and minimum criterion values over the efficient set. *European Journal of Operational Research*, 33(1):91–97, 1988.

[103] J. M. Jorge. An algorithm for optimizing a linear function over an integer efficient set. *European Journal of Operational Research*, 195(1):98–103, 2009.

[104] N. Jozefowiez, F. Semet, and E. G. Talbi. Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189(2):293–309, 2008.

[105] J. Júdice and A. Faustino. A sequential lcp method for bilevel linear programming. *Annals of Operations Research*, 34(1):89–106, 1992.

[106] J. J. Júdice, A. M. Faustino, I. M. Ribeiro, and A. S. Neves. On the use of bilevel programming for solving a structural optimization problem with discrete variables. In *Optimization with Multivalued Mappings*, pages 123–142. Springer, 2006.

[107] J. Karaivanova, S. Narula, and V. Vassilev. An interactive procedure for multiple objective integer linear programming problems. *European Journal of Operational Research*, 68(3):344–351, 1993.

[108] E. Karasakal and M. Köksalan. Generating a representative subset of the nondominated frontier in multiple criteria decision making. *Operations research*, 57(1):187–199, 2009.

[109] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer Verlag, 2004.

[110] A. Khandelwal, M. Puri, et al. Bilevel time minimizing transportation problem. *Discrete Optimization*, 5(4):714–723, 2008.

[111] I. Kim and O. De Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29(2):149–158, 2005.

[112] I. Y. Kim and O. L. de Weck. Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation. *Structural and Multidisciplinary Optimization*, 31(2):105–116, 2006.

[113] G. Kirlik and S. Sayın. Computing the nadir point for multiobjective discrete optimization problems. *Journal of Global Optimization*, pages 1–21, 2014.

[114] G. Kirlik and S. Sayın. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232(3):479–488, 2014.

[115] G. Kiziltan and E. Yucaoğlu. An algorithm for multiobjective zero-one linear programming. *Management Science*, pages 1444–1453, 1983.

[116] K. Klamroth and M. Wiecek. Dynamic programming approaches to the multiple criteria knapsack problem. *Naval Research Logistics*, 47(1):57–76, 2000.

[117] A. Klarbring. *An Introduction to Structural Optimization*, volume 153. Springer, 2008.

[118] M. Kočvara and J. V. Outrata. Effective reformulations of the truss topology design problem. *Optimization and Engineering*, 7(2):201–219, 2006.

[119] M. M. Köksalan and P. N. S. Sagala. Interactive approaches for discrete alternative multiple criteria decision making with monotone utility functions. *Management Science*, 41(7):1158–1171, 1995.

[120] P. Kouvelis and S. Sayın. Algorithm robust for the bicriteria discrete optimization problem. *Annals of Operations Research*, 147(1):71–85, 2006.

[121] M. Laumanns, L. Thiele, and E. Zitzler. An adaptive scheme to generate the pareto front based on the epsilon-constraint method. In J. Branke, K. Deb, K. Miettinen, and R. E. Steuer, editors, *Practical Approaches to Multi-Objective Optimization*, number 04461 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl, Germany, 2005.

[122] M. Laumanns, L. Thiele, and E. Zitzler. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942, 2006.

[123] A. Le Thi Hoai and P. Dinh Tao. Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *Journal of Global Optimization*, 11(3):253–285, 1997.

[124] J. Lemesre, C. Dhaenens, and E. G. Talbi. Parallel partitioning method (ppm): A new exact method to solve bi-objective problems. *Computers & Operations Research*, 34(8):2450–2462, 2007.

[125] S. Leyffer. A complementarity constraint formulation of convex multiobjective optimization problems. *INFORMS Journal on Computing*, 21(2):257–267, 2009.

[126] X. Li, P. Beullens, D. Jones, and M. Tamiz. An integrated queuing and multi-objective bed allocation model with application to a hospital in china. *Journal of the Operational Research Society*, 60(3):330–338, 2009.

[127] B. Lokman and M. Köksalan. Finding all nondominated points of multi-objective integer programs. *Journal of Global Optimization*, 57(2):347–365, 2013.

[128] P. Lučic and D. Teodorovic. Simulated annealing for the multi-objective aircrew rostering problem. *Transportation Research Part A: Policy and Practice*, 33(1):19–45, 1999.

[129] Z.-Q. Luo, J.-S. Pang, and D. Ralph. *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press, 1996.

[130] R. Malhotra, H. L. Bhatia, and M. C. Puri. Bi-criteria assignment problem. *Journal of the Operational Research Society of India*, 19(2):84–96, 1982.

[131] O. Marcotte and R. M. Soland. An interactive branch-and-bound algorithm for multiple criteria optimization. *Management Science*, pages 61–75, 1986.

[132] P. Marcotte and D. L. Zhu. Exact and inexact penalty methods for the generalized bilevel programming problem. *Mathematical Programming*, 74(2):141–157, 1996.

[133] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[134] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[135] M. Masin and Y. Bukchin. Diversity maximization approach for multiobjective optimization. *Operations Research*, 56(2):411–424, 2008.

[136] G. Mavrotas. Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465, 2009.

[137] G. Mavrotas and D. Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, 107(3):530–541, 1998.

[138] G. Mavrotas and D. Diakoulaki. Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming. *Applied Mathematics and Computation*, 171(1):53–71, 2005.

[139] G. Mavrotas and K. Florios. An improved version of the augmented ε-constraint method (augmecon2) for finding the exact pareto set in multi-objective integer programming problems. *Applied Mathematics and Computation*, 219(18):9652–9669, 2013.

[140] K. Miettinen. *Nonlinear Multiobjective Optimization*, volume 12. Springer, 1999.

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

[142] A. Mitsos, P. Lemonidis, and P. I. Barton. Global solution of bilevel programs with a nonconvex inner program. *Journal of Global Optimization*, 42(4):475–513, 2008.

[143] J. T. Moore and J. F. Bard. The mixed integer linear bilevel programming problem. *Operations Research*, 38(5):911–921, 1990.

[144] M. Özlen and M. Azizoğlu. Multi-objective integer programming: A general approach for generating all non-dominated solutions. *European Journal of Operational Research*, 199(1):25–35, 2009.

[145] M. Ozlen, B. A. Burton, and C. A. MacRae. Multi-objective integer programming: An improved recursive algorithm. *Journal of Optimization Theory and Applications*, 160(2):470–482, 2014.

[146] O. Özpeynirci and M. Köksalan. An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs. *Management Science*, 56(12):2302–2315, 2010.

[147] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings of 41st Annual Symposium on Foundations of Computer Science*, pages 86–92. IEEE, 2000.

[148] A. Pascoletti and P. Serafini. Scalarizing vector optimization problems. *Journal of Optimization Theory and Applications*, 42(4):499–524, 1984.

[149] V. Pereyra, M. Saunders, and J. Castillo. Equispaced pareto front construction for constrained bi-objective optimization. *Mathematical and Computer Modelling*, 57(9):2122–2131, 2013.

[150] A. Przybylski, X. Gandibleux, and M. Ehrgott. Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185(2):509–533, 2008.

[151] A. Przybylski, X. Gandibleux, and M. Ehrgott. A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 22(3):371–386, 2010.

[152] A. Przybylski, X. Gandibleux, and M. Ehrgott. A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3):149–165, 2010.

[153] A. Pugachev, A. Boyer, and L. Xing. Beam orientation optimization in intensity-modulated radiation treatment planning. *Medical Physics*, 27(6):1238–1245, 2000.

[154] T. Ralphs, M. Saltzman, and M. Wiecek. An improved algorithm for solving biobjective integer programs. *Annals of Operations Research*, 147(1):43–70, 2006.

[155] G. R. Reeves and R. C. Reid. Minimum values over the efficient set in multiple objective decision making. *European Journal of Operational Research*, 36(3):334–338, 1988.

[156] M. J. Rosenblatt and Z. Sinuany-Stern. Generating the discrete efficient frontier to the capital budgeting problem. *Operations Research*, pages 384–394, 1989.

[157] S. Ruzika. *On multiple objective combinatorial optimization*. PhD thesis, University of Kaiserslautern, 2007.

[158] S. Ruzika and M. M. Wiecek. Approximation methods in multiobjective programming. *Journal of Optimization Theory and Applications*, 126(3):473–501, 2005.

[159] S. Sayın. An algorithm based on facial decomposition for finding the efficient set in multiple objective linear programming. *Operations Research Letters*, 19(2):87–94, 1996.

[160] S. Sayın. Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Mathematical Programming*, 87(3):543–560, 2000.

[161] S. Sayin. Optimizing over the efficient set using a top-down search of faces. *Operations Research*, 48(1):65–72, 2000.

[162] S. Sayın. A procedure to find discrete representations of the efficient set with specified coverage errors. *Operations Research*, pages 427–436, 2003.

[163] S. Sayın and P. Kouvelis. The multiobjective discrete optimization problem: A weighted min-max two-stage optimization approach and a bicriteria algorithm. *Management Science*, 51(10):1572–1581, 2005.

[164] B. Schandl, K. Klamroth, and M. Wiecek. Norm-based approximation in bicriteria programming. *Computational Optimization and Applications*, 20(1):23–42, 2001.

[165] B. Schandl, K. Klamroth, and M. Wiecek. Norm-based approximation in multicriteria programming. *Computers & Mathematics with Applications*, 44(7):925–942, 2002.

[166] P. Serafini. Some considerations about computational complexity for multiobjective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent advances and historical development of vector optimization*, volume 294 of *Lecture Notes in Economics and Mathematical Systems*, Berlin, 1986. Springer-Verlag.

[167] L. Shao and M. Ehrgott. Finding representative nondominated points in multiobjective linear programming. In *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on*, pages 245–252. IEEE, 2007.

[168] H. D. Sherali, A. L. Soyster, and F. H. Murphy. Stackelberg-nash-cournot equilibria: characterizations and computations. *Operations Research*, 31(2):253–276, 1983.

[169] R. Solanki. Generating the noninferior set in mixed integer biobjective linear programs: an application to a location problem. *Computers & Operations Research*, 18:1–15, January 1991.

[170] S. P. E. C. SPEC. Spec cpu results, 2012.

[171] R. Steuer. *Multiple Criteria Optimization: Theory, Computation and Applications*. John Wiley & Sons, New-York, 1986.

[172] R. E. Steuer and E. U. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26(3):326–344, 1983.

[173] J. A. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[174] J. Sylva and A. Crema. A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research*, 158(1):46–55, 2004.

[175] J. Sylva and A. Crema. A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs. *European Journal of Operational Research*, 180(3):1011–1027, 2007.

[176] J. Sylva and A. Crema. Enumerating the set of non-dominated vectors in multiple objective integer linear programming. *RAIRO - Operations Research*, 42(3):371–387, 2008.

[177] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.

[178] J. Teghem and P. Kunsch. A survey of techniques for finding efficient solutions to

multi-objective integer linear programming. *Asia-Pacific Journal of Operational Research*, 3(2):95–108, 1986.

[179] D. Tenfelde-Podehl. A recursive algorithm for multiobjective combinatorial optimization problems with q criteria. Technical report, Institut für Mathematik, Technische Universität Graz, Austria, August 2003.

[180] E. L. Ulungu and J. Teghem. Multi-objective combinatorial optimization problems: A survey. *Journal of Multi-Criteria Decision Analysis*, 3:83–104, 1994.

[181] E. L. Ulungu and J. Teghem. The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20(2):149–165, 1995.

[182] D. Vaz, L. Paquete, C. M. Fonseca, K. Klamroth, and M. Stiglmayr. Representation of the non-dominated set in biobjective combinatorial optimization. *BUW-IMACM 14/06*, 2014.

[183] L. Vicente, G. Savard, and J. Júdice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81(2):379–399, 1994.

[184] L. Vicente, G. Savard, and J. Judice. Discrete linear bilevel programming problem. *Journal of Optimization Theory and Applications*, 89(3):597–614, 1996.

[185] T. Vincent, F. Seipp, S. Ruzika, A. Przybylski, and X. Gandibleux. Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers & Operations Research*, 40(1):498–509, 2013.

[186] M. Visée, J. Teghem, M. Pirlot, and E. L. Ulungu. Two-phases method and branch and bound procedures to solve the bi–objective knapsack problem. *Journal of Global Optimization*, 12(2):139–155, 1998.

[187] U.-P. Wen and S.-T. Hsu. Linear bi-level programming problems - a review. *Journal of the Operational Research Society*, 42(2):125–133, 1991.

[188] D. J. White. A branch and bound method for multi-objective boolean problems. *European Journal of Operational Research*, 15(1):126–130, 1984.

[189] A. P. Wierzbicki. The use of reference objectives in multiobjective optimization. In *Multiple Criteria Decision Making Theory and Application*, pages 468–486. Springer, 1980.

[190] A. B. Wijeratne, M. A. Turnquist, and P. B. Mirchandani. Multiobjective routing of hazardous materials in stochastic networks. *European Journal of Operational Research*, 65(1):33–43, 1993.

[191] J. H. Wilkinson, J. H. Wilkinson, and J. H. Wilkinson. *The Algebraic Eigenvalue Problem*, volume 87. Clarendon Press Oxford, 1965.

[192] Y. Yamamoto. Optimization over the efficient set: Overview. *Journal of Global Optimization*, 22(1):285–317, 2002.

[193] F. You, L. Tao, D. J. Graziano, and S. W. Snyder. Optimal design of sustainable cellulosic biofuel supply chains: Multiobjective optimization coupled with life cycle assessment and input–output analysis. *AIChE Journal*, 58(4):1157–1180, 2012.

[194] P. L. Yu and M. Zeleny. The set of all nondominated solutions in linear cases and a multicriteria simplex method. *Journal of Mathematical Analysis and Applications*, 49(2):430–468, 1975.

[195] L. Zadeh. Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, 8(1):59 – 60, 1963.

[196] M. Zelany. A concept of compromise solutions and the method of the displaced ideal. *Computers & Operations Research*, 1(3-4):479–496, 1974.

[197] W. Zhang and M. Reimann. A simple augmented ϵ-constraint method for multi-objective mathematical integer programming problems. *European Journal of Operational Research*, 234(1):15–24, 2014.

VITA

GÖKHAN KİRLİK was born in Eskişehir, Turkey in 1986. He graduated with a B.S. degree from Eskişehir Osmangazi University Department of Industrial Engineering in 2007, with a 3^{rd} rank. In a double major program, he received another B.S. degree from Department of Computer Engineering in 2008. He started the M.Sc. program in Industrial Engineering at Eskişehir Osmangazi University in 2008. During his M.Sc. studies, he worked on a Scientific & Technical Research Council of Turkey's research project which was about mobile robot route planning for complete coverage. He received his M.Sc. degree in 2009. Meanwhile, he joined the Ph.D. program in Industrial Engineering and Operations Management at Koç University in 2010. He worked as a teaching and research assistant at Koç University.

His current research interests include multiobjective optimization, integer programming, scheduling, and heuristic methods.

He was also a recipient of the IBM Ph.D. Fellowship Award in 2013.