EXACT SOLUTION ALGORITHMS FOR BIOBJECTIVE MIXED INTEGER PROGRAMMING PROBLEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

By
Deniz Emre
August 2020

Exact solution algorithms for biobjective mixed integer programming problems
By Deniz Emre
August 2020

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Firdevs Ulus(Advisor)

Özlem Karsu(Co-Advisor)

Çağın Ararat

Diclehan Tezcaner Öztürk

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

EXACT SOLUTION ALGORITHMS FOR BIOBJECTIVE MIXED INTEGER PROGRAMMING PROBLEMS

Deniz Emre
M.S. in Industrial Engineering
Advisor: Firdevs Ulus
Co-Advisor: Özlem Karsu
August 2020

In this thesis, objective space based exact solution algorithms for biobjective mixed integer programming problems are proposed. The algorithms solve scalarization models in order to explore predetermined regions of the objective space called boxes, defined by two nondominated points. The initial box is defined by the two extreme nondominated points of the Pareto frontier, which includes all nondominated points. At each iteration of the algorithms, a box is explored either by a weighted sum or a Pascoletti-Serafini scalarization to determine nondominated line segments and points. The first algorithm creates new boxes immediately when it finds a nondominated point by solving Pascoletti-Serafini scalarization, whereas the second algorithm conducts additional operations after obtaining a nondominated point by this scalarization. Our computational experiments demonstrate the computational feasibility of the algorithms.

Keywords: Biobjective mixed integer linear programming, exact solution algorithm, Pascoletti-Serafini scalarization.

ÖZET

İKİ AMAÇLI KARMA DOĞRUSAL PROGRAMLAMA PROBLEMLERİ İÇİN TAM SONUÇ VEREN ALGORİTMALAR

Deniz Emre
Endüstri Mühendisliği, Yüksek Lisans
Tez Danışmanı: Firdevs Ulus
İkinci Tez Danışmanı: Özlem Karsu
August 2020

Bu çalışmada iki amaçlı karma doğrusal programlama problemleri için amaç fonksiyonu uzayında çalışan iki algoritma önerilmiştir. İki algoritma da problemin baskın noktalarını tam bir şekilde bulmaktadır. Algoritmalar arama bölgesini önceden tanımlanan kutulara bölmekte ve bu kutuların özelliklerine göre ağırlıklı ortalama ya da Pascoletti-Serafini skalarizasyon modellerini hiç kutu kalmayana kadar çözmektedir. İlk algoritma Pascoletti-Serafini skalarizasyonu çözdükten hemen sonra yeni kutular tanımlarken ikinci algoritma bu skalarizasyondan elde edilen noktayı kullanarak ek modeller çözmektedir. Sayısal analizler algoritmaların fizibilitesini göstermektedir.

Anahtar sözcükler: İki Amaçlı Karışık Tam Sayılı Programlama, kesin çözüm algoritması, Pascoletti-Serafini skalarizasyonu.

Acknowledgement

I would like to express my sincere appreciation to my advisors, Dr. Firdevs Ulus and Dr. Özlem Karsu, for their supervision, time and patience. Their guidance and support deeply contributed to my academic journey. Without their inspiration and support it would have been impossible to finish this thesis.

I would like to thank the members of my thesis committee Dr. Çağın Ararat and Dr. Diclehan Tezcaner Öztürk for their valuable comments.

I would like to thank Heyecan Utke Koyuncuoğlu for always cheering me up. I am deeply grateful to my parents, Yasemin Emre and Ömer Emre, for their endless love and support. My greatest appreciation goes to my brother Volkan Emre, who constantly encourages and inspires me. Without his mentorship, I would not be the person I am today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multiobjective optimization problems (MOPs) arise in many fields such as engineering applications, economics and health care. These problems involve more than one conflicting objectives, which means there is no single solution that optimizes all of the objective functions simultaneously. Hence, there are trade-offs among objectives in MOP, and the optimality term is replaced by nondominance. Solution algorithms designed for MOP aim to find the set of nondominated solutions.

The algorithms designed for solving MOPs can be categorized into two with respect to the space the algorithm works: Decision space based algorithms and criterion space based algorithms. In this study, we propose two criterion space algorithms, Algorithm 1 and Algorithm 2, to find the exact set of nondominated points of a biobjective mixed integer linear programming problem (BOMILP).

Criterion space algorithms benefit from single objective optimization solvers since they solve such problems consecutively. These single objective problems are the so called scalarization problems, which will be discussed in detail in Chapter 2.

To the best our knowledge, there are five criterion space algorithms for BOMILPs [1–5] and the first such algorithm is introduced in 2015 [1]. Mainly,

two types of scalarization methods are used in these works: weighted sum scalarization and ϵ-constraint scalarization. Our motivation is introducing alternative solution approaches to the existing BOMILP literature using Pascoletti- Serafini scalarization [6], which has the potential to obtain a diverse subset of nondominated points under time limit [7]. We also benefit from the common scalarization techniques for BOMILPs for the two algorithms we propose: The algorithms search for nondominated points within predefined regions (boxes) using either weighted sum scalarization or Pascoletti-Serafini scalarization. The choice of the scalarization problem is based on the properties of the box to be searched.

The algorithms are implemented and the performances are observed through computational experiments. We compare Algorithms 1 and 2 both with each other also with the existing algorithms [1–5] from the literature.

The rest of this thesis is organized as follows. In Chapter 2, we introduce key definitions and models. In Chapter 3, we review the relevant studies in the literature. In Chapters 4 and 5, the details of the proposed algorithms, Algorithm 1 and Algorithm 2 are presented, respectively. In Chapter 6, we provide the summary of our computational experiments. Finally, we conclude our discussion and mention some future research directions in Chapter 7.

Chapter 2

Problem Definition and Preliminaries

In this chapter, we define biobjective mixed integer linear programming problem and provide other basic definitions and models that will be used throughout the thesis.

The general formulation of a BOMILP can be seen below:

$$\min\{\, z(x) := (z_1(x), z_2(x)) : x \in \mathcal{X} \,\}, \tag{P}$$

where $\mathcal{X} = \{x \in \mathbb{R}^k \times \mathbb{Z}^q : Ax \leq b, x \geq 0\}$ and $z(x) : \mathbb{R}^k \times \mathbb{Z}^q \to \mathbb{R}^2$. The objective functions are linear in x. The sets $\mathcal{X}$ and $\mathcal{Y} := \{z(x) : x \in \mathcal{X}\}$ represent the feasible set in the decision space and the feasible set in the criterion space, respectively. The vector $x \in \mathcal{X}$ consists of both real valued and integer parts, hence the notation $x = (x_C, x_I)$ will be used throughout the thesis.

The nonnegative orthant is denoted by $\mathbb{R}^2_+ := \{z \in \mathbb{R}^2 : z \geq 0\}$.

Definition 1. *For x, $x' \in \mathcal{X}$, $z(x')$ dominates $z(x)$ if $z_i(x') \leqslant z_i(x)$ for $i = 1, 2$ and $z_i(x') < z_i(x)$ for at least one i. $z(x')$ strictly dominates $z(x)$ if $z_i(x') < z_i(x)$ for all i. A feasible solution $x' \in \mathcal{X}$ is (weakly) efficient if there is no other $x \in \mathcal{X}$ such that $z(x)$ (strictly) dominates $z(x')$.*

$\mathcal{X}_E$ and $\mathcal{Y}_N$ denote the set of all efficient solutions and the set of all nondominated points, respectively. Note that the frontier $\mathcal{Y}_N$ can contain isolated points and open, closed and half-open line segments as in Figure 2.1.

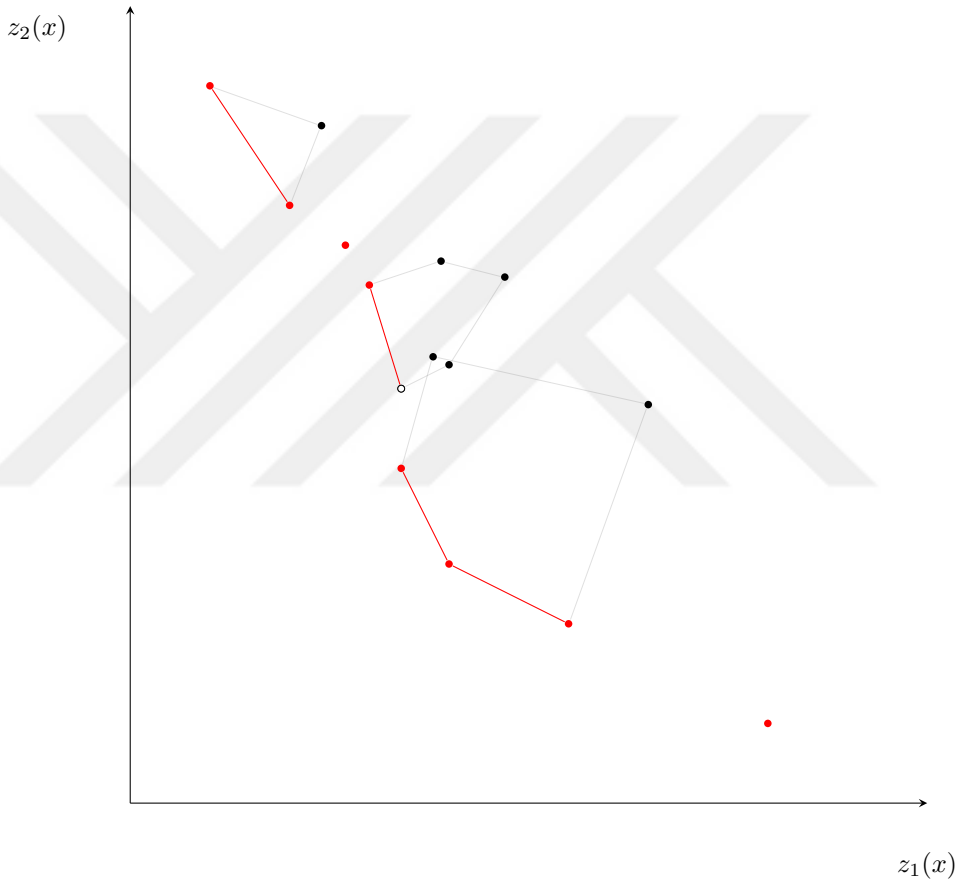

Figure 2.1: $\mathcal{Y}_N$ is represented by red color

Lexicographic optimization is used to obtain a single nondominated point, especially in the initial steps of the solution algorithms for MOPs. It is denoted as follows:

$$\text{lexmin}\{\, z_i(x), z_j(x) : x \in \mathcal{X} \,\}, \tag{2.1}$$

where $i, j \in \{1, 2\}$ and $i \neq j$. The lexicographic optimization requires solving two single objective optimization problems consecutively: First it minimizes the

i^{th} objective by solving

$$\min\{\, z_i(x) : x \in \mathcal{X} \,\}. \tag{2.2}$$

Then, for x^* being an optimal solution of (2.2), one solves the following problem for the j^{th} objective:

$$\min\, \{\, z_j(x) : x \in \mathcal{X},\ z_i(x) = z_i(x^*) \,\}. \tag{2.3}$$

A BOMILP can be converted into a biobjective linear programming problem by fixing all integer decision variables of $x = (x_C, x_I)$ to a specific integer vector. The resulting problem is called *slice problem* and represented as follows:

$$\min\{\, z(x) : x \in \mathcal{X},\ x_I = \bar{x}_I \,\} \tag{$SP(\bar{x})$}$$

for some $\bar{x}_I \in \mathbb{Z}^q$.

Let z^1 and z^2 be two points such that $z_1^1 < z_1^2$ and $z_2^2 < z_2^1$. The closed line segment between these points is denoted by

$$[z^1, z^2] = \{\alpha z^1 + (1-\alpha) z^2 : \alpha \in [0,1]\} \tag{2.4}$$

Similarly, open or half open line segments are denoted by (z^1, z^2), $(z^1, z^2]$ and $[z^1, z^2)$, where we replace $\alpha \in [0,1]$ in (2.4) by $\alpha \in (0,1)$, $\alpha \in (0,1]$ and $\alpha \in [0,1)$, respectively. We denote a line segment by $L(z^1, z^2)$ when we do not specify if the endpoints are open or closed.

2.1 Scalarization Models

Criterion space based solution algorithms for multiobjective programming problems rely on iteratively solving single objective scalarization problems. A scalarization problem is a single objective programming problem the solution of which provides an (weakly) efficient solution to the original multiobjective problem. We now give the scalarization models, variants of which will be used in our solution algorithm.

2.1.1 The Weighted Sum Scalarization

A single objective weighted sum scalarization problem can be obtained by aggregating the two objective functions using a weight vector $\lambda \in \mathbb{R}^2$ with $\lambda_i > 0$ for $i = 1, 2$.

$$\min\{\, \lambda_1 z_1(x) + \lambda_2 z_2(x) : x \in \mathcal{X}\} \qquad (WS(\lambda))$$

Lemma 1. *[8] An optimal solution of $(WS(\lambda))$ is an efficient solution.*

If there exists $\lambda \in \mathbb{R}^2$ with $\lambda > 0$ such that x is an optimal solution to $(WS(\lambda))$, then $z(x)$ is a supported nondominated point for (P). By definition, only supported nondominated points can be obtained by $(WS(\lambda))$.

2.1.2 Pascoletti-Serafini Scalarization

Pascoletti and Serafini [6] propose the following scalarization problem for vector optimization problems

$$\min\{\, \rho :\ z(x) \leq r + \rho d,\ x \in \mathcal{X},\ \rho \in \mathbb{R}\}, \qquad (PS(r,d))$$

where $r \in \mathbb{R}^2$ is a reference point and $d \in \mathbb{R}^2 \setminus \{0\}$ is a direction vector.

Lemma 2. *[6] For an optimal solution (ρ^*, x^*) of $(PS(r,d))$, x^* is weakly efficient.*

In Chapter 4, we solve second stage models to ensure that a nondominated point is obtained as follows. Let (ρ^*, x^*) be an optimal solution of $(PS(r,d))$ and let $y = r + \rho^* d$. It is known that y and $z^* = z(x^*)$ have at least one common component [7]. If the first components of z^* and y are equal ($z_1^* = y_1$), then we solve

$$\min\{z_2(x)\ \ : x \in \mathcal{X},\ \ z_1 = z_1(x^*)\}. \qquad (S_1(x^*))$$

If second components are equal ($z_2^* = y_2$), then

$$\min\{z_1(x) \ : x \in \mathcal{X}, z_2 = z_2(x^*)\} \qquad (S_2(x^*))$$

is solved. Note that z^* and y can be equal in both components, in which case both $(S_1(x^*))$ and $(S_2(x^*))$ are solved consecutively.

2.1.3 The ϵ-constraint Scalarization

ϵ-constraint scalarization is a well-known method used for solving biobjective integer programming problems. It is also commonly used in criterion space BOMILP algorithms. In this scalarization, one objective function is selected to be optimized while the other is used in constraints:

$$\min\{z_i(x) : \ x \in \mathcal{X}, \ z_j(x) \leq \epsilon, \ i \neq j, \ \ j \in 1, 2\}, \qquad (2.5)$$

where $i \in \ \{1, 2\}$.

Lemma 3. *[9] An optimal solution of* (2.5) *is weakly efficent.*

To find efficient solutions, one can solve second stage problems similar to the ones solved after a Pascoletti-Serafini scalarization.

Chapter 3

Literature Review

The algorithms focusing on BOMILPs can be categorized with respect to their working space as stated in Chapter 1. The algorithm that is proposed in this work is a criterion space based algorithm. Therefore, we briefly review the criterion based solution algorithms.

The first criterion space algorithm for BOMILP was introduced by Boland et al. [1]. It mainly explores the criterion space subregions that are defined as triangles and rectangles, and is called Triangle Splitting Algorithm (TSA). The algorithm starts with an initial rectangle whose corner points are nondominated and entire Pareto frontier locates within this rectangle. When exploring a rectangle, weighted sum scalarization problems are solved and local extreme supported points within the rectangle are found. Then, right-angled triangles are created using these local extreme supported points as in Figure 3.1. Each triangle is explored to determine whether all points on the hypotenuse are nondominated or not. If the entire hypotenuse is nondominated, the corner points are updated as connected. Otherwise, the triangles are split vertically or horizontally. Two cases can occur after this splitting: a single new point or two new points can be found by lexicographic optimization. Within the triangles two new rectangles are defined between the corner points of the triangle and the new points. Then, the triangle is removed from the list and two new rectangles are added to the list

of rectangles to be explored. This process repeats recursively until the lists of triangles and rectangles to be searched are both empty.

We exemplify the process in Figures 3.1-3.3. In Figure 3.1 a rectangle with corner points u^0 and l^0 is explored and its (local) extreme points are found as u^0, z^e and l^0. In Figure 3.2 the triangle defined by u^0 and z^e (denoted by $t(u^0, z^e)$) is split horizontally and a single z^* is found by solving lexicographic optimization, i.e. the first objective function is minimized where the search region is under the horizontal split line (the line is also included). There are two cases when this first lexicographic optimization problem is solved: z^* can be strictly under the split line or on the split line. If it is under the line, then a second lexicographic optimization problem is solved, which minimizes the second objective function and defining the search region as on the left side of z^*. Otherwise, no additional model is solved. The case that z^* is on the split line is illustrated in Figure 3.2. In Figure 3.3, $t(u^0, z^e)$ is split into two rectangles with corner points u^0, z^* and z^*, z^e, respectively. Note that in Figure 3.2, TSA splits a nondominated line segment. This splitting requires postprocessing to define the line segment that z^* belongs to.

Another criterion space algorithm for BOMILP is proposed by Soylu and Yildiz [2]. The algorithm uses $\epsilon-$constraint method and Tabu constraints to determine the Pareto frontier, hence it is called $\epsilon, Tabu-constraint$ (ϵ-TC) algorithm. The algorithm starts with an initial nondominated point of the search region that contains the whole frontier. Then it finds the corresponding slice and searches for Pareto line segments of the slice problem (whose feasible region is the polyhedron consisting of $x \in \mathcal{X}$, that have the same integer variables as the initial solution). Each line segment is checked to detect whether all points on the line segment are nondominated or not. For this purpose, it uses Tabu constraints and searches a point belonging to another slice that dominates the line segment. If there is no such point, the line segment is labeled as nondominated. Otherwise, it finds this point and computes the dominated part of the line segment by projection. The algorithm then shifts to line segments of the new slice, and sequentially finds the whole Pareto frontier.

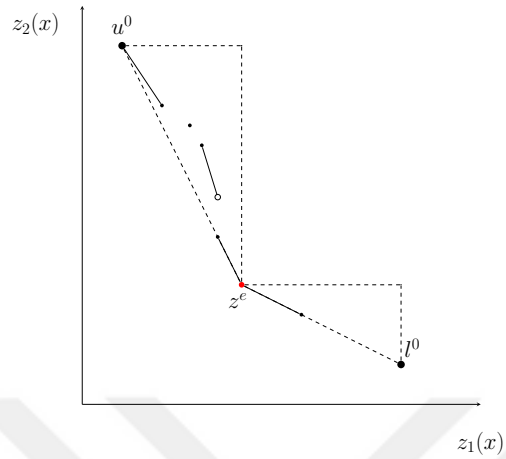

Figure 3.1: Find extreme points (z^e) inside the rectangle using WS(λ) and create triangles

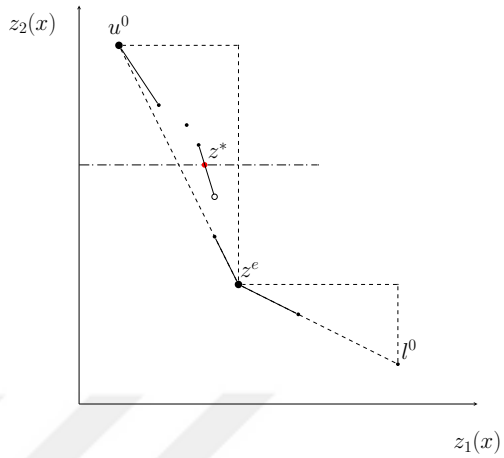

Figure 3.2: Split $t(u^0, z^e)$ horizontally and find z^* by lexicographic optimization

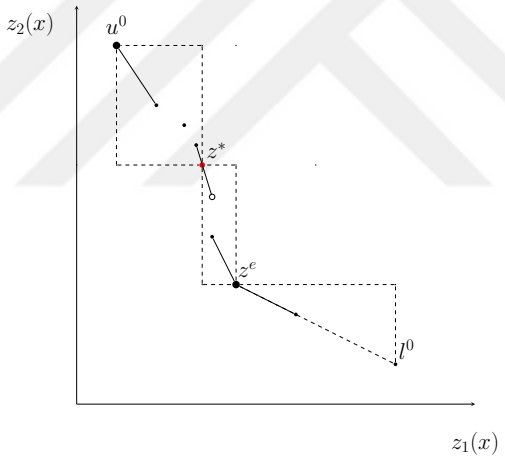

Figure 3.3: Create two new rectangles with corner points u^0, z^* and z^*, z^e.

Fattahi and Turkay presented one direction search (ODS) method [3]. Similar to the $\epsilon, Tabu - constraint$ algorithm, it also moves along a direction within the search region. The algorithm starts from the lower-right corner and finds the Pareto points towards the upper-left corner. After finding an initial point, the algorithm checks for extreme points of the corresponding slice problem. Since the algorithm starts from the lower-right corner, these points are located to the left side of the initial point. There are two cases: the neighbor extreme point on the same slice exists (Case 1) or not (Case 2). If Case 2 holds, then it looks for a solution that has different integer components. For example, the lower-right point (l^0) is a singleton in Figure 3.4 hence it does not have a neighbor extreme

point, which is located on its left side. In this case, the algorithm switches to the next integer set by using Tabu constraints which exclude the current slice from the search region and it minimizes the second objective function, $z_2(x)$, then finds a new point z^* as in Figure 3.4. Then, the same procedure is repeated for z^*: the algorithm seeks a neighbor extreme point belonging to the same slice and on the left side of z^*. This time Case 1 holds and neighbor extreme point is found as z^e. Next, it checks whether the line segment between those two points is nondominated. Note that in the example below $[z^e, z^*]$ is nondominated. If this is not the case, it finds the dominated part of the line segment by searching a point belonging to another slice. The process for finding nearest extreme point repeats for z^e.

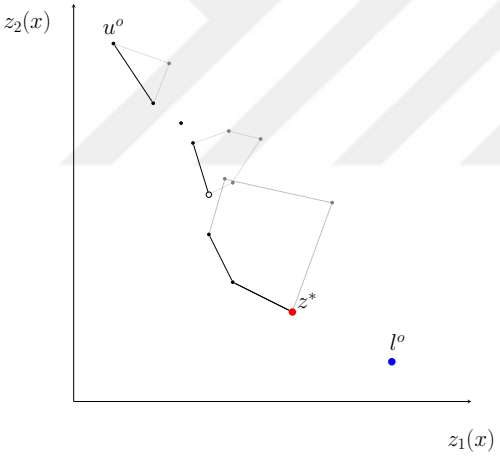

Figure 3.4: Case 1: l^0 is a singleton, find a point z^* that has different integer solution

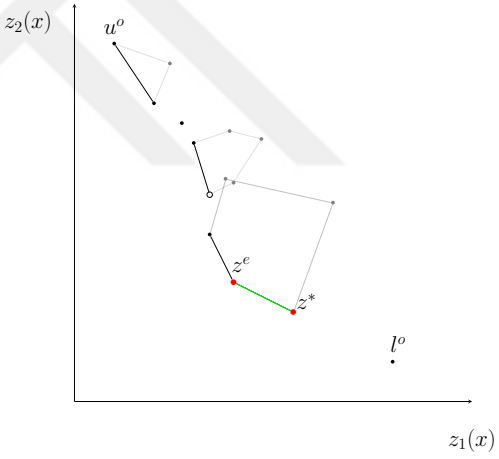

Figure 3.5: Case 2: find nearest extreme point z^e, update $L(z^e, z^*)$ as nondominated

Perini et al. [5] recently introduced another algorithm called Boxed Line Method (BLM). The algorithm extends the Balanced Boxed method [10], which is designed to solve biobjective integer programming problems. Both algorithms split the initial box horizontally and search for a nondominated point below or on the split line μ (i.e. $z_2(x) = \mu$) by solving lexicographic optimization . The first lexicographic optimization model prioritizes the first objective function and finds a nondominated point z^*. If $z_2^* < \mu$ (Case 1), then both algorithms perform the same steps: They solve a second lexicographic optimization problem which prioritizes the second objective function and finds $z^{'}$ on the left side of z^* as in

Figure 3.6, then they create two new boxes with corners u^0, $z^{'}$ and z^*, l^0 as in Figure 3.7.

The difference occurs when $z_2^* = \mu$ (Case 2) indicating the possibility that z^* is a point on a line segment. In this case, BLM seeks a nondominated line segment such that $z^* \in L(z^1, z^2)$ and defines new boxes accordingly as illustrated in Figures 3.8 and 3.9, respectively. We also benefit from a similar approach while we design our Algorithm 2 as we will explain in Chapter 5. The authors also provide an upper bound on the number of single objective mixed integer linear programs solved to find the Pareto frontier by BLM.

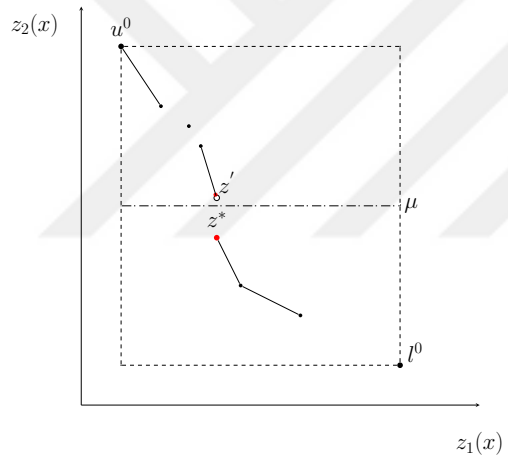

Figure 3.6: Case 1: $z_2^* < \mu$, find $z^{'}$

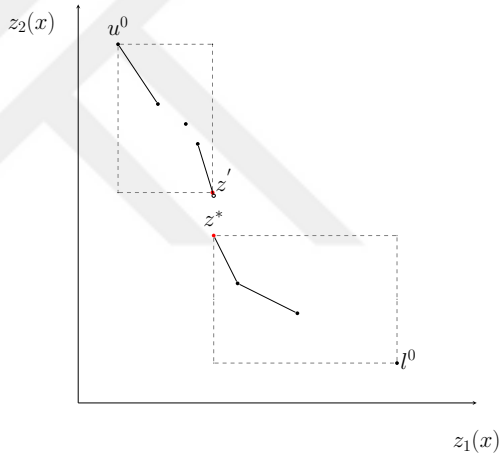

Figure 3.7: Update boxes: Remove box with corners u^0, l^0 and add boxes with corners u^0, $z^{'}$ and z^*, l^0, respectively

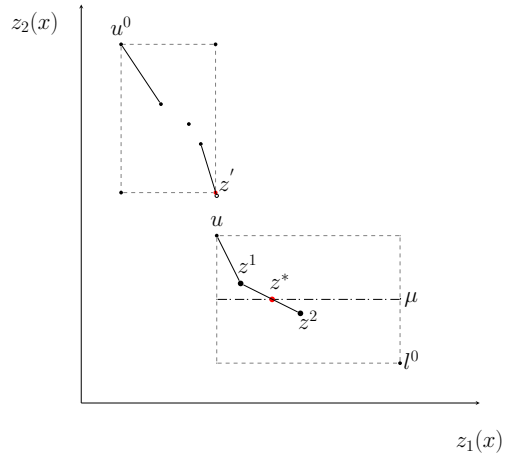

Figure 3.8: Update z^* as new u for the new box and search box with corners u, l^0. Then, find new z^* by lexicographic optimization. Case 2 holds: ${z_2}^* = \mu$, find $[z^1, z^2]$

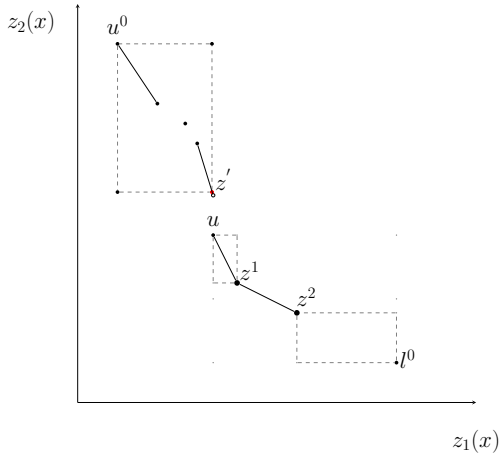

Figure 3.9: Update boxes: Remove box with corners u, l^0 and add boxes with corners u, z^1 and z^2, l^0, respectively

The last algorithm that we consider in this category is Search-and-remove (SR) algorithm introduced by Soylu [4]. The algorithm searches slices of the problem by dichotomic search, then removes the current slice using Tabu constraints enables to search for nondominated points on other slices.

The algorithms which are proposed in this work provide a new approach to the existing criterion space literature for BOMILP using Pascoletti-Serafini scalarization.

As mentioned in Chapter 1, criterion space BOMILP algorithms rely on solving single objective scalarization problems. These scalarization methods are summarized in Table 3.1. WS(λ)(MILP) and WS(λ)(LP) stand for the weighted sum scalarizations solved for mixed integer linear programming and linear programming problems, respectively. The proposed algorithms are presented as Alg. 1 and Alg. 2 which will be detailed in Chapter 4 and 5. Moreover, the table shows the algorithms that benefit from Tabu constraints. To the best of our knowledge, we present the first criterion space BOMILP algorithm that benefit from Pascoletti- Serafini scalarization, hence we provide an alternative approach to the existing literature.

Table 3.1: Type of Scalarizations

	WS(λ)(MILP)	WS(λ)(LP)	ϵ-constraint	PS(r, d)	Tabu constraints
TSA	✓	✓	✓		
ϵ-TC		✓	✓		✓
ODS			✓		✓
BLM	✓	✓	✓		✓
SR	✓	✓			✓
Alg.1	✓			✓	✓
Alg 2.	✓	✓	✓	✓	✓

Although criterion space BOMILP algorithms use similar scalarization methods, they differ in their designs. For instance, ϵ-TC algorithm and ODS start from one corner and move towards the other corner. Hence, at any given time they represent a specific part of the Pareto frontier whereas TSA and BLM may provide solutions from diverse parts of the Pareto frontier. If a decision maker wants to get an overview of the Pareto frontier under time limit, then the algorithms which maintain a diverse set of solutions would fit better for the decision

maker. We set the parameters of the Pascoletti-Serafini scalarization as to create a diverse set of points at any time through the algorithms.

Chapter 4

Algorithm 1

We design an exact solution algorithm for BOMILP given by (P) in Chapter 2. The algorithm works in the criterion space, and it solves weighted sum and Pascoletti-Serafini scalarization problems iteratively.

Throughout the algorithm, the set of nondominated points and the nondominated line segments are denoted by $\mathcal{N}$ and $\mathcal{L}$, respectively. $\mathcal{B}$ represents the set of regions that have to be explored. A search region is also called a box and it is defined by two nondominated points u and l, satisfying $u_1 < l_1$, as $b(u,l) = \{z : (u_1, l_2) \leq z \leq (l_1, u_2)\}$. For notational simplicity, we let u_I be the vector of integer variables for x where $z(x) = u$. Similarly l_I is the vector of integer variables for x, where $z(x) = l$. The algorithm categorizes the boxes according to the integer variable values of the corner points as follows: Boxes whose corner points have the same integer variable values ($u_I = l_I$), and boxes whose corner points differ in the integer components ($u_I \neq l_I$). These two types are explored by solving different scalarization models.

At each iteration, the algorithm explores one box from $\mathcal{B}$. The explored boxes are removed from $\mathcal{B}$ and the algorithm terminates if there is no box to be explored. The pseudocode of the algorithm is given in Algorithm 1, and the details of it is explained next.

The algorithm starts with initialization as follows: First,

$$\text{lexmin}\{\, z_1(x), z_2(x) : x \in \mathcal{X} \,\} \tag{4.1}$$

is solved and upper-left corner (u^0) of $\mathcal{Y}_N$ is obtained. Next, we solve

$$\text{lexmin}\{\, z_2(x), z_1(x) : x \in \mathcal{X} \,\} \tag{4.2}$$

and find the nondominated point (l^0) in the lower-right corner of $\mathcal{Y}_N$. We then initialize $\mathcal{L}$, $\mathcal{N}$ and $\mathcal{B}$ as $\emptyset$, $\{u^0, l^0\}$ and $\{b(u^0, l^0)\}$, respectively (line 3). Note that $b(u^0, l^0)$ defines the initial box that contains all nondominated points of (P), see Figure 4.1. Note also that if $u^0 = l^0$ the problem has a single solution, hence the algorithm will terminate at the first stage. Moreover, if $u^0 \neq l^0$, then $u_i^0 \neq l_i^0$ for any i since both are nondominated points. In this case, $u_2^0 > l_2^0$ and $u_1^0 < l_1^0$ hold.

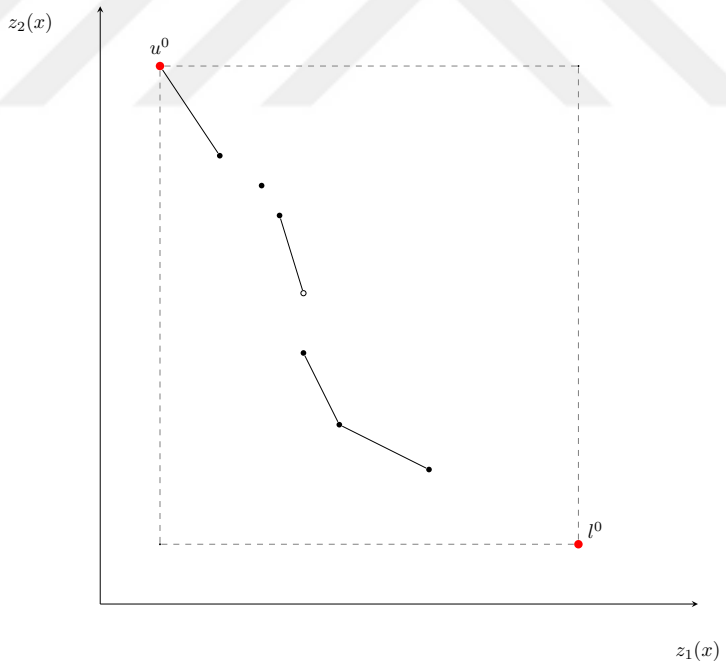

Figure 4.1: Initial Box

At an arbitrary iteration, in which we explore a box $b(u, l)$, we first remove $b(u, l)$ from $\mathcal{B}$ (line 6). Then, we check whether the box is large enough to be explored (line 7). If it is, we check whether $u_I = l_I$ (line 8). In this case, we solve weighted sum scalarization with a box constraint given by

$$\min\{\, \lambda_1 z_1(x) + \lambda_2 z_2(x) : x \in \mathcal{X}, z_1(x) \leq l_1, z_2(x) \leq u_2\}, \tag{$WS(b)$}$$

where $\lambda_1 = u_2 - l_2 > 0$ and $\lambda_2 = l_1 - u_1 > 0$. Let x^* an the optimal solution to $(WS(b))$. Two cases are possible:

Case 1: $\lambda_1 z_1(x^*) + \lambda_2 z_2(x^*) = \lambda_1 u_1 + \lambda_2 u_2$. That is, $z(x^*) = z^*$ lies on the same line segment with u and l. Then all points on the line segment are nondominated and $[u, l]$ is added to $\mathcal{L}$ (line 11). See Proposition 1.

Case 2: $\lambda_1 z_1(x^*) + \lambda_2 z_2(x^*) < \lambda_1 u_1 + \lambda_2 u_2$. Then $z(x^*) = z^*$ is a new nondominated point and added to $\mathcal{N}$. Two new boxes, $b(u, z^*)$ and $b(z^*, l)$, are added to $\mathcal{B}$ (lines 13-14).

Cases 1 and 2 are illustrated in Figures 4.3 and 4.4, respectively.

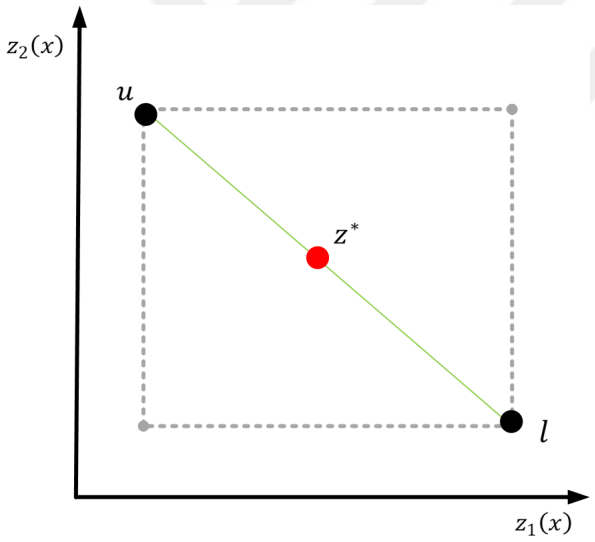

Figure 4.2: $\lambda_1 z_1(x^*) + \lambda_2 z_2(x^*) = \lambda_1 u_1 + \lambda_2 u_2$. $\mathcal{L} \leftarrow \mathcal{L} \cup \{[u, l]\}$

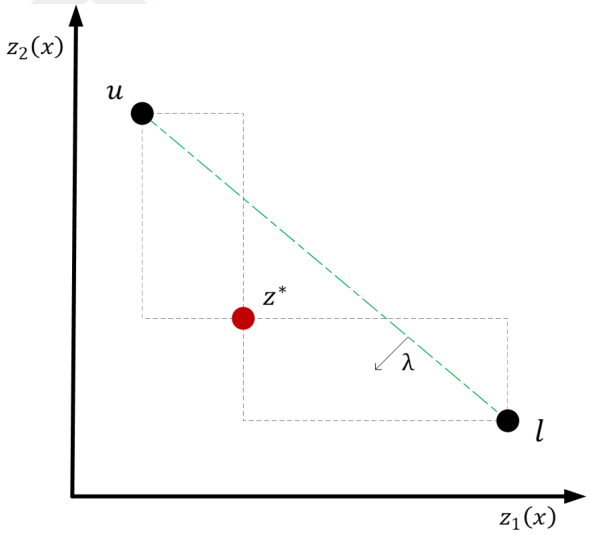

Figure 4.3: $\lambda_1 z_1(x^*) + \lambda_2 z_2(x^*) < \lambda_1 u_1 + \lambda_2 u_2$. $\mathcal{N} \leftarrow \mathcal{N} \cup \{z^*\}$. $\mathcal{B} \leftarrow \{b(u, z^*), b(z^*, l)\}$

Proposition 1. *Let u and l be two nondominated points with $u_I = l_I$, and x^* be an optimal solution to $(WS(b))$ where b represents a box defined by corner points u, l and $\lambda_1 = u_2 - l_2$, $\lambda_2 = l_1 - u_1$. If $\lambda_1 z_1(x^*) + \lambda_2 z_2(x^*) = \lambda_1 u_1 + \lambda_2 u_2$, then $[u, l]$ is a nondominated line segment.*

Proof. The corner points u, l are nondominated, hence there are no other nondominated points in $\{u\} - \mathbb{R}^2_+$ and $\{l\} - \mathbb{R}^2_+$, see Figure 4.4. Consider the slice problem $(\mathrm{SP}(u_I))$, where $u_I = l_I$ holds. As it is a biobjective linear programming problem with convex feasible regions in the decision and criterion spaces, line segment $[u, l]$ is feasible for $(\mathrm{SP}(u_I))$. Then, it is also feasible for (P). If $\lambda_1 z_1(x^*) + \lambda_2 z_2(x^*) = \lambda_1 u_1 + \lambda_2 u_2$, then there is no feasible point in $[u, l] - \mathbb{R}^2_+$ as this would contradict to the optimality of x^*. Hence $[u, l]$ is nondominated.

□

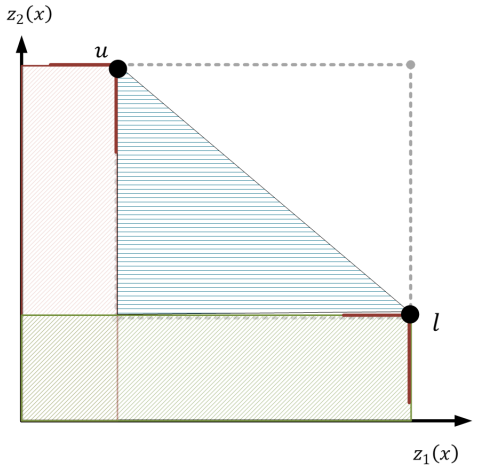

Figure 4.4: Shaded triangle may contain nondominated points

If $(u_I \neq l_I)$ for $b(u, l)$, we solve Pascoletti-Serafini scalarization (Algorithm 1, line 17) given by

$$\min\{\, \rho : \; z \leq r + \rho d, z_1(x) \leq l_1 - \epsilon, z_2(x) \leq u_2 - \epsilon, x \in \mathcal{X}, \; \rho \in \mathbb{R}\}, \qquad (PS(b))$$

where $r = (u_1, l_2)$ is the reference point, $d = [(l_1 - u_1), (u_2 - l_2)]$ is the direction vector and ϵ is a sufficiently small approximation error. $\epsilon > 0$ is used in order to check equivalence of two points and this will be detailed in Section 6. If the model is infeasible, it means that u and l are the only nondominated points in $b(u, l)$. Otherwise, let (x^*, ρ^*) be an optimal solution to $(PS(b))$ and let $y = r + \rho^* d$. From Chapter 2, we know that y and z^* has at least one common component. In order to ensure the obtained solution is efficient in each case we solve second stage models as follows:

Case 1: If only the first components of z^* and y are equal ($z_1^* = y_1$), solve $(S_1(x^*))$ (Procedure 1, lines 21-24). Let x^1 be an the optimal solution to $(S_1(x^*))$, then $z(x^1) = n^1$ is added to $\mathcal{N}$ and $b(u, n^1)$ and $b(n^1, l)$ are added to $\mathcal{B}$. See Figures 4.5 and 4.6.

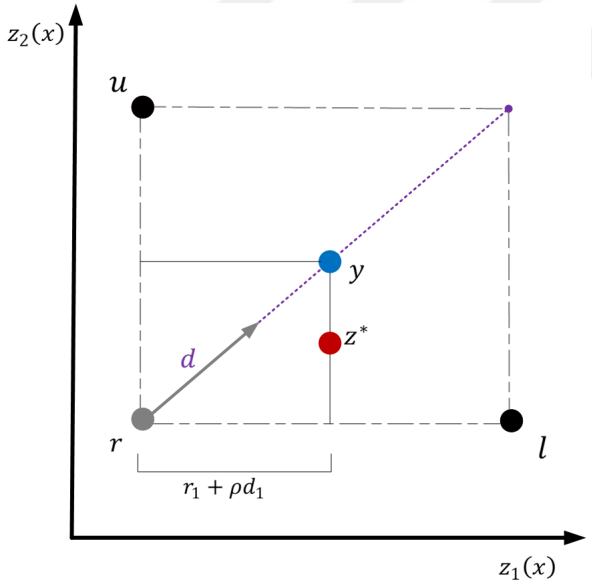

Figure 4.5: $(z_1^* = y_1)$

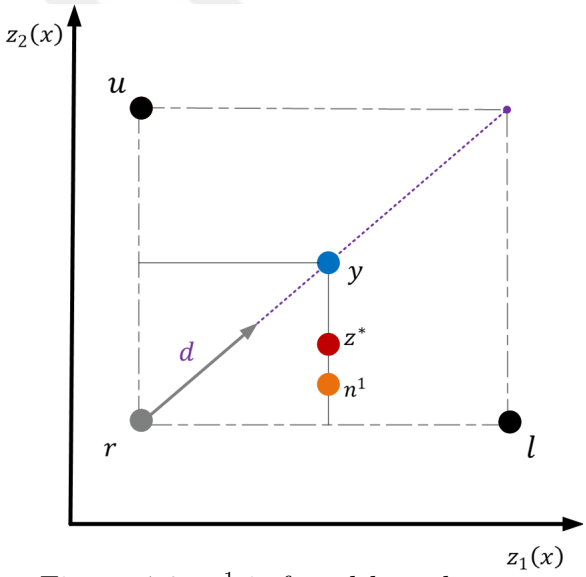

Figure 4.6: n^1 is found by solving $S_1(x^*)$. $\mathcal{N} \leftarrow \mathcal{N} \cup \{n^1\}$

Case 2: If only the second components are equal ($z_2^* = y_2$), solve $(S_2(x^*))$ (Procedure 1, lines 25-28). Let x^2 be an optimal solution, then $z(x^2) = n^2$ is added to $\mathcal{N}$, and $b(u, n^2)$ and $b(n^2, l)$ are added to $\mathcal{B}$.

Case 3: If both components are equal, that is $y = z^*$. Then, both $(S_1(x^*))$ and $(S_2(x^*))$ are solved (Procedure 1, lines 8-9) . Now four possible cases, which are illustrated in Figures 4.7- 4.10, can occur.

- **Case 3.1:** $n^1 = n^2$ (Procedure 1, line 11)
- **Case 3.2:** $n^1 \neq n^2$ and $n^i \neq y$ for $i = 1, 2$ (Procedure 1, line 17)
- **Case 3.3:** $n^1 \neq n^2$ and $n^2 = y$ (Procedure 1, line 13).
- **Case 3.4:** $n^1 \neq n^2$ and $n^1 = y$ (Procedure 1, line 15).

Figure 4.7 shows Case 3.1, in which $(S_1(x^*))$ and $(S_2(x^*))$ return the same solution. Figure 4.8 shows Case 3.2 where two new nondominated points are obtained. Finally, Figures 4.9 and 4.10 illustrate Cases 3.3 and 3.4, in which a single nondominated solution is returned by either $(S_1(x^*))$ or $(S_2(x^*))$.

The algorithm continues until there is no box to explore in set $\mathcal{B}$.

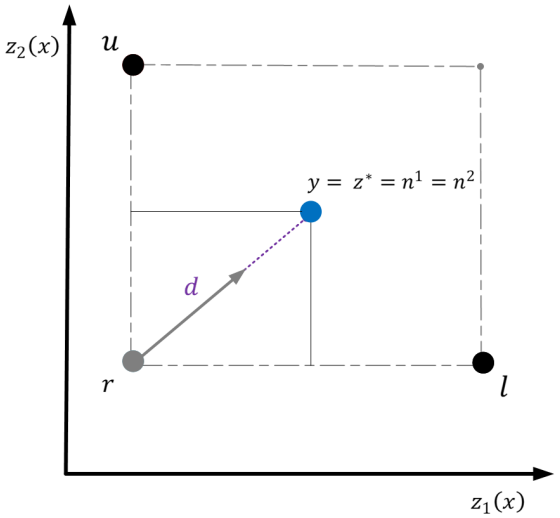

Figure 4.7: Case 3.1: $\mathcal{N} \leftarrow \mathcal{N} \cup \{n^1\}$. $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, n^1), b(n^1, l)\}$

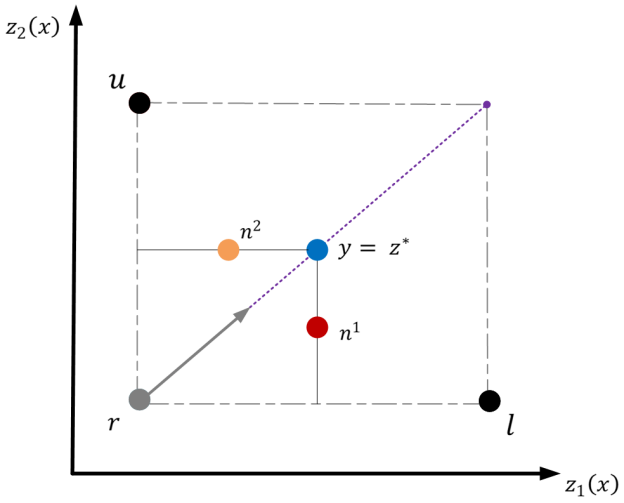

Figure 4.8: Case 3.2: $\mathcal{N} \leftarrow \mathcal{N} \cup \{n^1, n^2\}$. $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, n^2), b(n^1, l)\}$

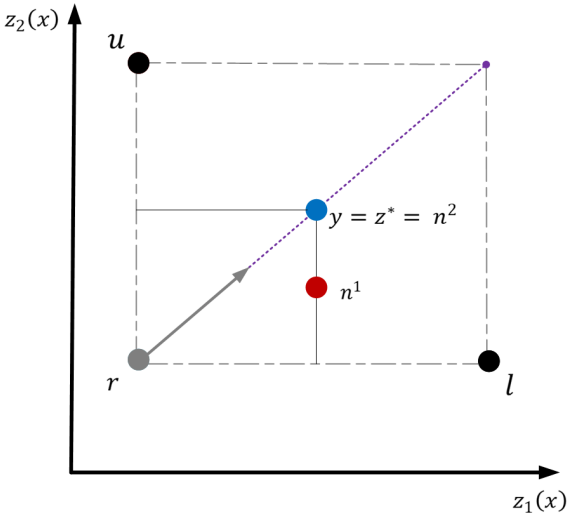

Figure 4.9: Case 3.3: $\mathcal{N} \leftarrow \mathcal{N} \cup \{n^1\}$ $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, n^1), b(n^1, l)\}$

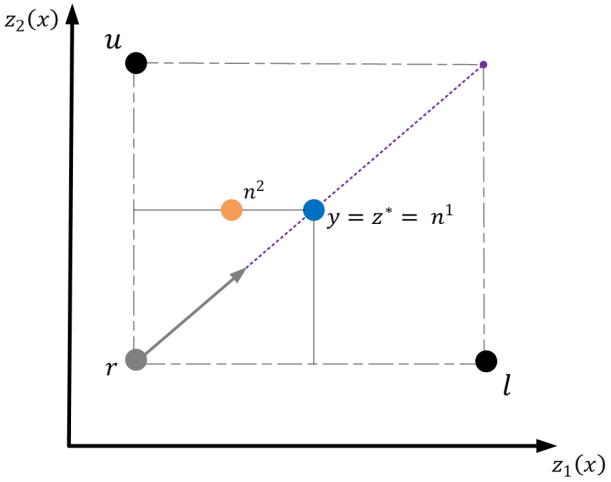

Figure 4.10: Case 3.4: $\mathcal{N} \leftarrow \mathcal{N} \cup \{n^2\}$. $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, n^2), b(n^2, l)\}$

Algorithm 1: Algorithm 1

Input : (P)
Output : $(\mathcal{N}, \mathcal{L})$

$x^u \leftarrow \text{lexmin}\{\, z_1(x), z_2(x) : x \in \mathcal{X}\,\}$; $u^o = z(x^u)$
$x^l \leftarrow \text{lexmin}\{\, z_2(x), z_1(x) : x \in \mathcal{X}\,\}$; $l^o = z(x^l)$
$\mathcal{N} = \{u^o, l^o\}$, $\mathcal{B} = \{b(u^o, l^o)\}$, $\mathcal{L} = \emptyset$
while $\mid \mathcal{B} \mid > 1$ **do**
    Take the first box $b(u, l)$ in $\mathcal{B}$
    $\mathcal{B} \leftarrow \mathcal{B} \setminus b(u, l)$
    if $\mid u_1 - l_1 \mid > \epsilon$ *or* $\mid u_2 - l_2 \mid > \epsilon$ **then**
        if $u_I = l_I$ **then**
            $z^* \leftarrow WS(b)$ $(w = (u_2 - l_2, l_1 - u_1))$
            if $w^T z^* = w^T u$ **then**
                $\mathcal{L} \leftarrow \mathcal{L} \cup [u, l]$
            else
                $\mathcal{N} \leftarrow \mathcal{N} \cup \{z^*\}$
                $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, z^*), b(z^*, l)\}$
            end
        else if then
            $Z \leftarrow$ **PSprocedure1**(u, l)
            if $|Z| \neq \emptyset$ **then**
                if $|Z| = 1$ $(Z = \{z^*\})$ **then**
                    $\mathcal{N} \leftarrow \mathcal{N} \cup \{z^*\}$
                    $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, z^*), b(z^*, l)\}$
                if $|Z| = 2$ $(Z = \{n^1, n^2\})$ **then**
                    $\mathcal{N} \leftarrow \mathcal{N} \cup \{n^1, n^2\}$
                    $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, n^2), b(n^1, l)\}$
                end
            end
        end
    end
end
return $(\mathcal{N}, \mathcal{L})$

Procedure 1: PSprocedure1(u, l)

Input : (u, l)
Output: Z

$Z = \emptyset$, $flag = 0$;
Solve $(PS(b))$,
if $(PS(b))$ *is infeasible* **then**
  there is no other point in $b(u, l)$, return
end
let (ρ^*, x^*) be an optimal solution and $y = r + \rho^* d\,,\ z^* = z(x^*)$
if $\mid y_1 - z_1^* \mid < \epsilon$ *and* $\mid y_2 - z_2^* \mid < \epsilon$ **then**
  $x^2 \leftarrow \min\{z_1(x) : z_2(x) = y_2, x \in \mathcal{X}\};\ n^2 = z(x^2)$
  $x^1 \leftarrow \min\{z_2(x) : z_1(x) = y_1, x \in \mathcal{X}\};\ n^1 = z(x^1)$
  $flag = 1$;
  if $\mid n_1^1 - n_1^2 \mid < \epsilon$ *and* $\mid n_2^1 - n_2^2 \mid < \epsilon$ **then**
    $Z = \{n^1\}$
  else if $n^2 = y$ **then**
    $Z = \{n^1\}$
  else if $n^1 = y$ **then**
    $Z = \{n^2\}$
  else
    $Z = \{n^1, n^2\}$
  end
end
if $y_1 = z_1^*$ *and* $flag = 0$ **then**
  $x^1 \leftarrow \min\{z_2(x) : z_1(x) = y_1, x \in \mathcal{X}\};\ n^1 = z(x^1)$
  $Z = \{n^1\}$
end
if $y_2 = z_2^*$ *and* $flag = 0$ **then**
  $x^2 \leftarrow \min\{z_1(x) : z_2(x) = y_2, x \in \mathcal{X}\};\ n^2 = z(x^2)$
  $Z = \{n^2\}$
end

Chapter 5

Algorithm 2

Recall that in Algorithm 1 when $u_I \neq l_I$ for a box $b(u,l)$ we solve $(PS(b))$ and the second stage models depending on the Cases 1, 2 and 3. Algorithm 2 differs from Algorithm 1 in Case 3.1 as follows: When $y = z^*$, both second stage models solved. If these models return a single nondominated point, i.e. $y = z^* = n^1 = n^2$, then it is highly likely that z^* lies on a nondominated line segment. Algorithm 2 aims to find the line segment, if there is any, and updates sets $\mathcal{L}$ and $\mathcal{B}$, accordingly.

The difference between Algorithms 1 and 2 when case 3.1 occurs is exemplified in Figures 5.1-5.3. In 5.1, $b(u,l)$ is explored and z^* is found (Case 3.1). Algorithm 1 updates $\mathcal{B}$ as in Figure 5.2 while in Algorithm 2, the end points of the line segment containing z^* are found, which are z^1 and z^2 in Figure 5.3. The endpoints are closed if they are nondominated, otherwise they are open. In this example, one of the endpoints (z^2) is open as seen in Figure 5.3. In this case, the nondominated point dominating this end point is also found (z^{newR}) and the new boxes are generated accordingly.

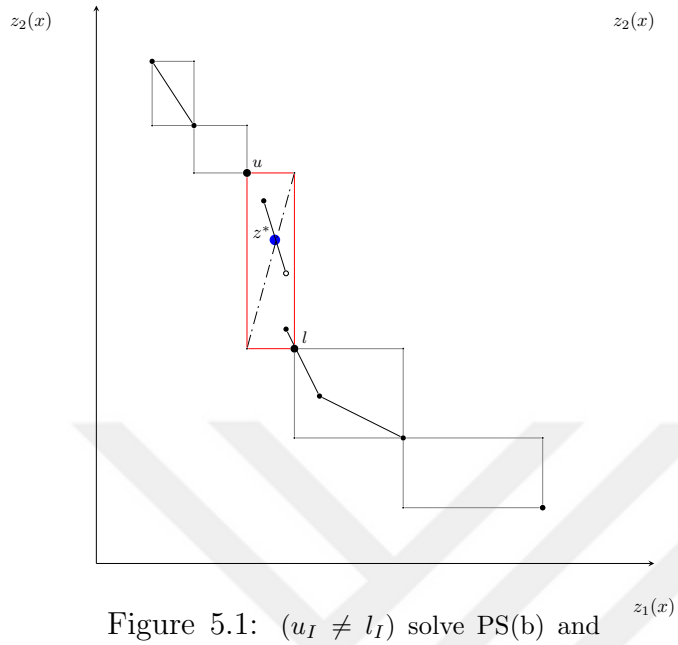

Figure 5.1: $(u_I \neq l_I)$ solve PS(b) and find z^*

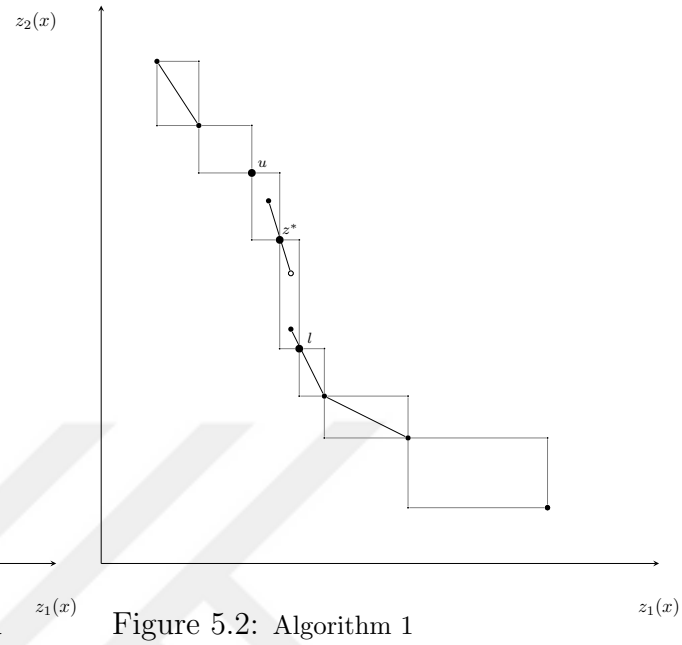

Figure 5.2: Algorithm 1
$\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, z^*), b(z^*, l)\}$

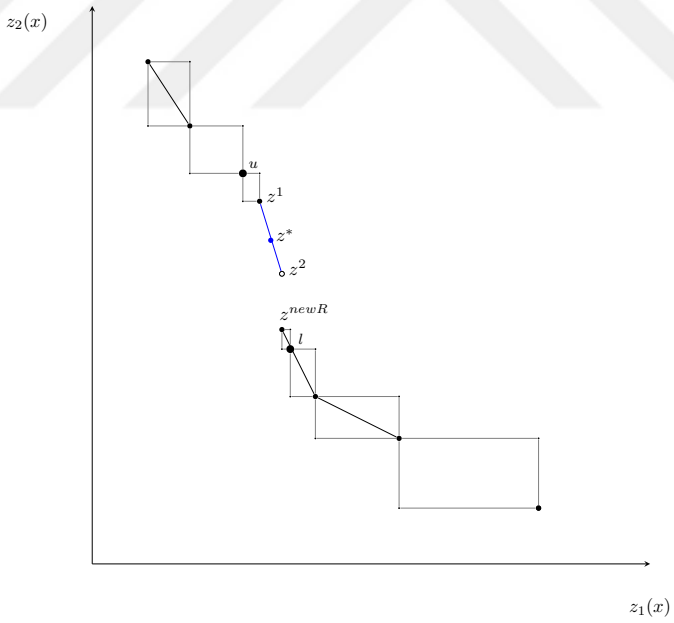

Figure 5.3: Algorithm 2
$\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, z^1), b(z^{newR}, l)\}$, $\mathcal{L} \leftarrow \mathcal{L} \cup \{[z^1, z^2)\}$

The pseudocode of Algorithm 2 is given in Algorithm 2. As seen, the algorithm explores the boxes in the same way as Algorithm 1: If the corner points of a box have the same integer variable values, then weighted sum scalarization is solved and $\mathcal{B}$ and $\mathcal{L}$ are updated accordingly (lines 7-13). Otherwise, the Pascoletti-Serafini scalarization is solved. If Cases 1, 2, 3.2, 3.3 and 3.4 are observed, the algorithm works the same as Algorithm 1 (see lines 17-21, 29 of Algorithm 2 and lines 13-30 of Procedure 2). However, in Case 3.1 a new function, Line Extension, in which the nondominated line segment containing z^* found is called (Procedure 2, line 12).

We now explain the line extension function, which is given in Procedure 3. As a first step we figure out whether z^* is an isolated point or not. To determine this, we solve the following lexicographic optimization problem for x^* satisfying $z^* = z(x^*)$:

$$\text{lexmin}\{(z_2, z_1) : z_1(x) \leq z_1^* + \epsilon_p,\ x_I = x_I^*\}. \tag{5.1}$$

(5.1) gives us a point z^r that has the same integer solution with x^*. Note that we extend the feasible region towards right of z^* by tolerance value $\epsilon_p > 0$. If the model returns the same point as z^*, that is if the distance between z^* and z^r is less than $\epsilon > 0$, then we say z^* is right isolated and update flagR as 1 to keep this information (lines 4-7 of Procedure 3).

Similarly, we also solve the following model to determine whether z^* is left isolated or not (lines 33-36 of Procedure 3).

$$\text{lexmin}\{(z_1, z_2) : z_2(x) \leq z_2^* + \epsilon_p,\ x_I = x_I^*\} \tag{5.2}$$

If z^* is both right and left isolated, then it is isolated. Algorithms 1 and 2 follow the same steps when z^* is isolated.

If z^* is not isolated, the aim is to find the nondominated line segment $L(z^1, z^2)$ that contains z^*. First, we calculate the weight vector w which supports both z^* and the point z^r obtained by solving (5.1) with $w = (z_2^* - z_2^r, z_1^r - z_1^*)$. Then, we

solve the following model.

$$\min\{w^T z(x) : \ x_I = x_I^*, \ z_1(x) \leq l_1, \ z_2(x) \leq u_2\} \tag{5.3}$$

Let x^r be an optimal solution. We update z^r as $z^r = z(x^r)$. If z^* and z^r are on the same line segment supported by w, i.e, $w^T z^* = w^T z^r$ (Procedure 3, line 9), then the algorithm seeks a point z^R as the right endpoint of the line segment within the box. Otherwise, the process repeats updating z^r until $w^T z^* = w^T z^r$ holds. Hence, the algorithm guarantees that there is no other point belonging the same slice with z^* such that $L(z^*, z^r)$ is dominated by this point. The decision mechanism for z^R is iteration number (i). If $w^T z^* = w^T z^r$ is not obtained in the first iteration, then the algorithm solves (5.3) at least once. In this case, the point z^r is considered as an endpoint since (5.3) finds extreme supported points as it is a weighted sum scalarization model with additional box constraints. Then, z^R is updated as z^r (line 12). However, if $w^T z^* = w^T z^r$ occurs in the first iteration, it is needed to guarantee that z^r is the right most endpoint. For this purpose we solve (Procedure 3, line 14)

$$\min\{z_2(x) : w^T z(x) = w^T z^*, \ x_I = x_I^*, \ z_1(x) \leq l_1, \ z_2(x) \leq u_2\}. \tag{5.4}$$

Let x^R be an optimal solution and $z^R = z(x^R)$. $L(z^*, z^R)$ is found to be feasible but we do not yet know whether the whole line segment is nondominated or not. In order to check this, we search a point that belongs another slice and dominates $L(z^*, z^R)$. In this searching process, the slice created by x^* must be excluded from the search region. For this purpose, we benefit from Tabu constraint, as in [2]. Tabu constraint uses Hamming Distance which calculates the absolute difference between two integer valued vectors and can be formulated as follows:

$$H(x_I, x_I^*) = \sum_{i=1, \, x_{Ii}^*=0}^{q} x_{Ii} + \sum_{i=1, \, x_{Ii}^*=1}^{q} (1 - x_{Ii}) \tag{5.5}$$

Here we search a solution x which is not on the same slice with x^*. Hence, we calculate the Hamming distance between their integer parts as in (5.5). In order to guarantee x is belonged to a different slice, we add $H(x_I, x_I^*) \geq 1$ as a constraint and it is shown as $x_I \neq x_I^*$ in (5.6).

$$\min\{z_1(x) : w^T z(x) \leq w^T z^*,\ x_I \neq x_I^*,\ z_2(x) \leq z_2^*,\ z_1(x) \leq l_1, z_2(x) \leq u_2\}. \quad (5.6)$$

(5.6) returns the left most point from another slice that dominates parts of $L(z^*, z^R)$. If (5.6) is infeasible, then z^2 is updated as z^R and added to $\mathcal{N}$ since there is no point that dominates $[z^*, z^R]$ (Procedure 3, line 22). Otherwise, let x^S be the optimal solution to (5.6) and $z^S = z(x^S)$. Then we solve a second stage model to guarantee finding a nondominated point (line 25).

$$\min\{z_2(x) : z_1(x) = z_1^S,\ x \in \mathcal{X}\} \quad (5.7)$$

Let x^{newR} be an optimal solution to (5.7). Next, we find z^2 and decide whether it is open or not. For this purpose we compare the first components of z^R and z^{newR}. There are two cases:

Case 1(R): $z_1^R < z_1^{newR}$. Then, z^2 is set as z^R (Procedure 3, line 27). Later in Algorithm 2, both z^{newR} and z^2 are added to $\mathcal{N}$, $b(z^2, z^{newR})$ and $b(z^{newR}, l)$ are added to $\mathcal{B}$ and $\mathcal{L}$ is updated accordingly as well. (see Figure 5.6).

Case 2(R): $z_1^{newR} \leq z_1^R$. In this case some parts of the line segment $L(z^*, z^R)$ is dominated by z^{newR}, hence we find z^2 by projecting z^{newR} onto $L(z^*, z^R)$ (Procedure 3, line 30). In this case, the nondominated line segment is $[z^*, z^2)$. Later in Algorithm 2, z^{newR} and $b(z^{newR}, l)$ are added to $\mathcal{N}$ and $\mathcal{B}$, respectively. Then $\mathcal{L}$ is a updated (see Figure 5.7).

Remark 1. *Note that updating $\mathcal{N}$, $\mathcal{L}$ and $\mathcal{B}$ in Algorithm 2 also depends on the results obtained after extending the line towards the left side, see Cases 1(L), 2(L) below.*

Cases 1(R) and Case 2(R) can be seen in Figures 5.4 and 5.5, respectively.

Extension for the left side of z^* is symmetric. For computational efficiency we use the information, which is indicating that z^* is right isolated or not. If z^* is not right isolated, then we already know the weight vector w, hence we directly solve the following model when z^* is also not left isolated (line 39).

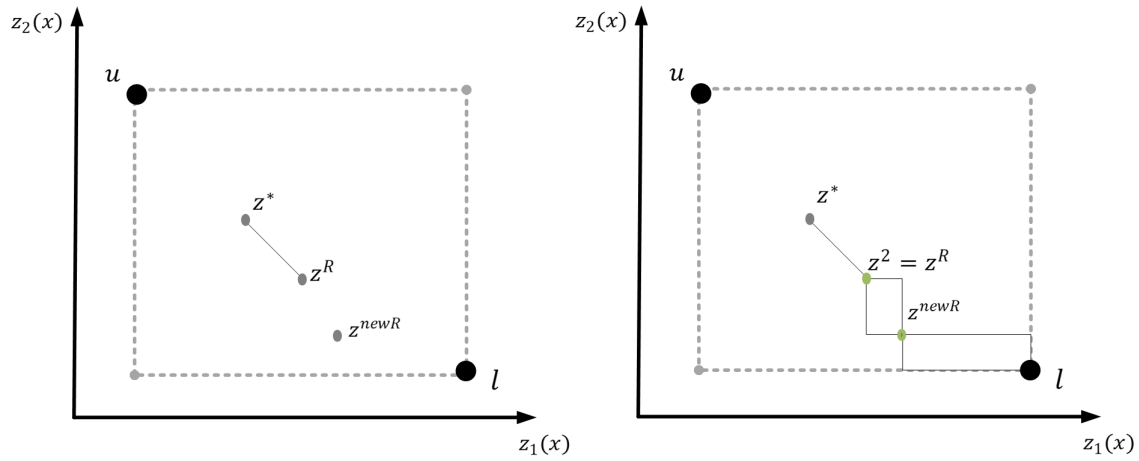

Figure 5.4: Case 1(R): $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(z^R, z^{newR}), b(z^{newR}, l)\}$

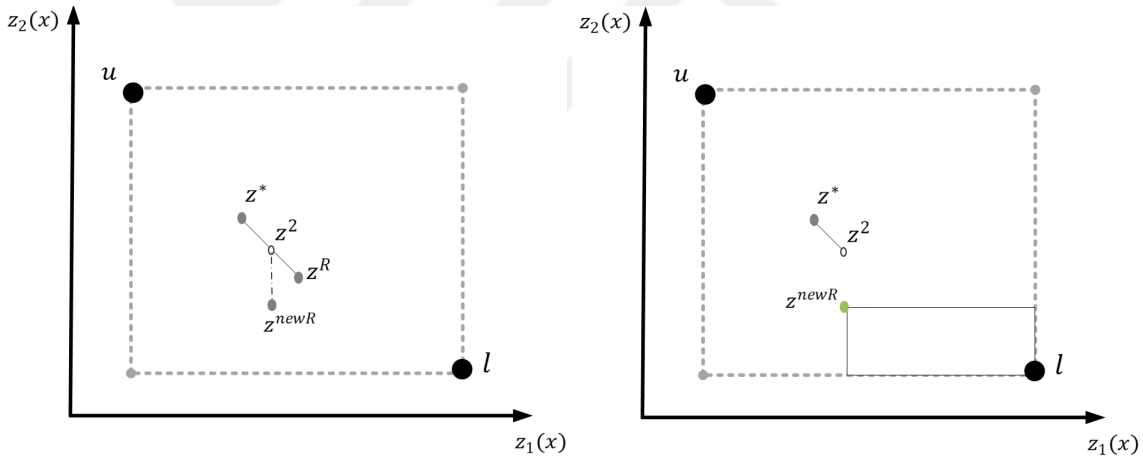

Figure 5.5: Case 2(R): $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(z^{newR}, l)\}$

$$\min\{z_1(x) : w^T z(x) = w^T z^*,\ x_I = x_I^*,\ z_1(x) \leq l_1,\ z_2(x) \leq u_2\} \tag{5.8}$$

Otherwise, we follow similar steps to the ones we follow for finding z^R (lines 41-52). Let x^L be an optimal solution to (5.8) and $z^L = z(x^L)$. Then we search for the points belonging to different slices by solving the following problem,

$$\min\{z_2(x) : w^T z(x) \leq w^T z^*,\ x_I \neq x_I^*,\ z_1^* \leq z_1(x),\ z_1(x) \leq l_1,\ z_2(x) \leq u_2\}. \tag{5.9}$$

For an optimal solution x^P, we let $z^P = z(x^P)$. Then if (5.9) is feasible,

$$\min\{z_1(x) : z_2(x) = z_2^P,\ x \in \mathcal{X}\} \tag{5.10}$$

is solved and x^{newL} is found (line 59). We have again two cases for $z^{newL} = z(x^{newL})$.

Case 1(L): $z_2^L < z_2^{newL}$ (Procedure 3, line 60). Then, z^1 is updated as z^L. Later in Algorithm 2, both z^{newL} and z^L are added to $\mathcal{N}$. $b(u, z^{newL})$ and $b(z^{newL}, z^1)$ are added to $\mathcal{B}$ and $\mathcal{L}$ is updated (see Figure 5.6).

Case 2(L): $z_2^{newL} \leq z_2^L$. That is $L(z^L, z^*)$ is dominated by z^{newL}, hence we find z^1 by projecting z^{newL} onto $L(z^L, z^*)$ where z^1 is open (Procedure 3, line 64). z^{newL} and $b(u, z^{newL})$ are added to $\mathcal{N}$ and $\mathcal{B}$, accordingly in Algorithm 2 and $\mathcal{L}$ is updated (see Figure 5.7).

Figures 5.6 and 5.7 show two cases, where both endpoints z^1 and z^2 are closed and open, respectively.

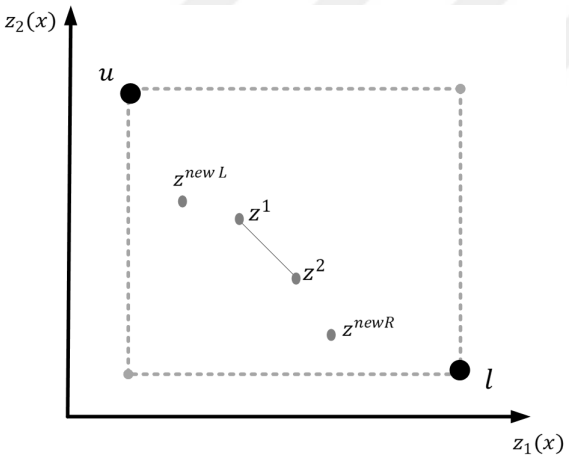

Figure 5.6: Case 1(R) and Case 1(L) hold: lines 26,60 of Procedure 3 and 21-23, 31-36 of Algorithm 2

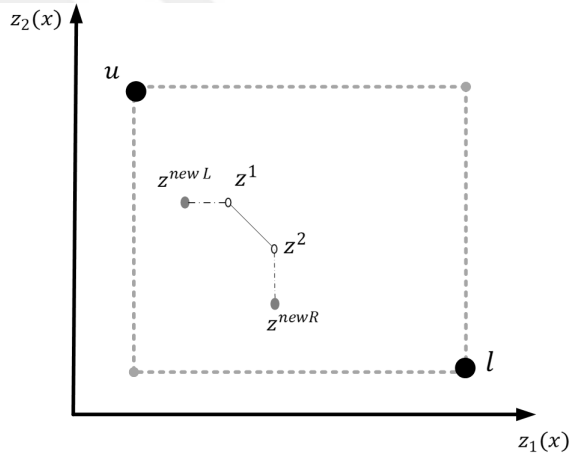

Figure 5.7: Case 2(R) and Case 2(L) hold: lines 29 and 63 of Procedure 3 and 21, 29, 46 of Algorithm 2

Algorithm 2: Algorithm 2

Input : (P)
Output : $(\mathcal{N}, \mathcal{L})$

$x^u \leftarrow \text{lexmin}\{\, z_1(x), z_2(x) : x \in \mathcal{X} \,\};\ u^o = z(x^u)$
$x^l \leftarrow \text{lexmin}\{\, z_2(x), z_1(x) : x \in \mathcal{X} \,\};\ l^o = z(x^l)$
$\mathcal{N} = \{u^o, l^o\},\ \mathcal{B} = \{b(u^o, l^o)\},\ \mathcal{L} = \emptyset$
while $|\ \mathcal{B}\ | > 1$ **do**
  Take the first box $b(u, l)$ in $\mathcal{B}$, $\quad \mathcal{B} \leftarrow \mathcal{B} \setminus b(u, l)$
  if $|\ u_1 - l_1\ | > \epsilon$ *or* $|\ u_2 - l_2\ | > \epsilon$ **then**
    if $u_I = l_I$ **then**
      $z^* \leftarrow WS(b)\ \ (w = (u_2 - l_2, l_1 - u_1))$
      if $w^T z^* = w^T u$ **then**
        $\mathcal{L} \leftarrow \mathcal{L} \cup [u, l]$
      else
        $\mathcal{N} \leftarrow \mathcal{N} \cup \{z^*\}$
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, z^*), b(z^*, l)\}$
    else
      $[z^{newL}, z^{newR}, z^1, z^2, Lopen, Ropen, lineflag, flagboxL, flagboxR] \leftarrow$ **PSprocedure2**(u, l)
      if $z^1 \neq \emptyset$ *or* $z^2 \neq \emptyset$ **then**
        if $z^1 = z^2$ **then**
          $\mathcal{N} \leftarrow \mathcal{N} \cup \{z^1\}$
          $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, z^1), b(z^1, l)\}$
        else
          $N \leftarrow \{z^{newL}, z^{newR}\}$
          if $flagboxL = 1$ *and* $flagboxR = 1$ **then**
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, z^{newL}), b(z^{newL}, z^1), b(z^2, z^{newR}), b(z^{newR}, l)\}$
          else if $flagboxL = 1$ *and* $flagboxR = 0$ **then**
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, z^{newL}), b(z^{newL}, z^1), b(z^{newR}, l)\}$
          else if $flagboxL = 0$ *and* $flagboxR = 1$ **then**
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, z^{newL}), b(z^2, z^{newR}), b(z^{newR}, l)\}$
          else
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(u, z^{newL}), b(z^{newR}, l)\}$
          if $lineflag = 1$ **then**
            if $Lopen = 0$ *and* $Ropen = 0$ **then**
              $\mathcal{L} \leftarrow \mathcal{L} \cup [z^1, z^2]$
              if $z^1 \neq z^{newL}$ **then**
                $\mathcal{N} \leftarrow \mathcal{N} \cup \{z^1\}$
              if $z^2 \neq z^{newR}$ **then**
                $\mathcal{N} \leftarrow \mathcal{N} \cup \{z^2\}$
            if $Lopen = 0$ *and* $Ropen = 1$ **then**
              $\mathcal{L} \leftarrow \mathcal{L} \cup [z^1, z^2)$
              if $z^1 \neq z^{newL}$ **then**
                $\mathcal{N} \leftarrow \mathcal{N} \cup \{z^1\}$
            if $Lopen = 1$ *and* $Ropen = 0$ **then**
              $\mathcal{L} \leftarrow \mathcal{L} \cup (z^1, z^2]$
              if $z^2 \neq z^{newR}$ **then**
                $\mathcal{N} \leftarrow \mathcal{N} \cup \{z^2\}$
            if $Lopen = 1$ *and* $Ropen = 1$ **then**
              $\mathcal{L} \leftarrow \mathcal{L} \cup (z^1, z^2)$
return $(\mathcal{N}, \mathcal{L})$

Procedure 2: PSprocedure2(u, l)

Input : (u, l)
Output : $(z^{newL}, z^{newR}, z^1, z^2, Lopen, Ropen, lineflag, flagboxL, flagboxR)$

Solve $(PS(b))$,
if $(PS(b))$ *is infeasible* **then**
  $(z^{newL}, z^{newR}, z^1, z^2, Lopen, Ropen, lineflag, flagboxL, flagboxR) = (\emptyset, \emptyset, \emptyset, \emptyset, 0, 0, 0, 0, 0)$
  return
end
let (ρ^*, x^*) be an optimal solution and $y = r + \rho^* d\,,\ z^* = z(x^*)$
if $\mid y_1 - z_1^* \mid < \epsilon$ *and* $\mid y_2 - z_2^* \mid < \epsilon$ **then**
  $x^1 \leftarrow \min\{z_1(x) : z_2(x) = y_2, x \in \mathcal{X}\};\ n^1 = z(x^1)$
  $x^2 \leftarrow \min\{z_2(x) : z_1(x) = y_1, x \in \mathcal{X}\};\ n^2 = z(x^2)$
  if $\mid n_1^1 - n_1^2 \mid < \epsilon$ *and* $\mid n_2^1 - n_2^2 \mid < \epsilon$ **then**
    $z^* = n^1$, $lineflag = 1$
    $[z^{newL}, z^{newR}, z^1, z^2, Lopen, Ropen, flagboxL, flagboxR)] \leftarrow$ **LineExtension**(u, l, z^*, x^*)
  else if $n^1 = y$ **then**
    $z^1 = z^2 = n^2$, $lineflag = 0$
    $z^{newL} = z^{newR} = z^2$, $Lopen = Ropen = 0$
  else if $n^2 = y$ **then**
    $z^1 = z^2 = n^1$, $lineflag = 0$
    $z^{newL} = z^{newR} = z^1$, $Lopen = Ropen = 0$
  else
    $z^1 = n^1$; $z^2 = n^2$; $lineflag = 0$
    $z^{newL} = z^1,\ z^{newR} = z^2$, $Lopen = Ropen = 0$
  end
end
if $y_1 = z_1^*$ **then**
  $x^2 \leftarrow \min\{z_2(x) : z_1(x) = y_1, x \in \mathcal{X}\};\ z^1 = z^2 = z(x^2),\ lineflag = 0$
  $z^{newL} = z^1,\ z^{newR} = z^{newL}$, $Lopen = Ropen = 0$
end
if $y_2 = z_2^*$ **then**
  $x^1 \leftarrow \min\{z_1(x) : z_2(x) = y_2, x \in \mathcal{X}\};\ z^1 = z^2 = z(x^1),\ lineflag = 0$
  $z^{newL} = z^1,\ z^{newR} = z^{newL}$, $Lopen = Ropen = 0$
end

Procedure 3: Line Extension(u, l, z^*, x^*)

Input : (u, l, z^*, x^*)

Output: $(z^{newL}, z^{newR}, z^1, z^2, Lopen, Ropen, flagboxL, flagboxR)$

Fix ϵ, ϵ_p, $flagw = 0$, $flagR = 0$, $flagL$=0, $Lopen$=0, $Ropen$=0, $flagboxR$=0, $flagboxL$=0

$i = 0$, $x^r \leftarrow \text{lexmin}\{(z_2, z_1) : z_1(x) \leq z_1^* + \epsilon_p,\ x_I = x_I^*\}$; $z^r = z(x^r)$

while *flagw=0 and* $flagR = 0$ **do**

  if $|z_1^* - z_1^r| < \epsilon$ *and* $|z_2^* - z_2^r| < \epsilon_d$ **then**

    $z^2 = z^*$, $z^{newR} = z^2$, flagR=1

    BREAK

  end

  $w = [z_2^* - z_2^r, z_1^r - z_1^*]$, $x^r \leftarrow \min\{w^T z(x) : x_I = x_I^*,\ z(x) \in b(u, l)\}$; $z^r = z(x^r)$

  if $w^T z^r = w^T z^*$ **then**

    $flagw$ =1

    if $i \geq 1$ **then**

      $z^R = z^r$

    else

      $x^R \leftarrow \min\{z_2(x) : w^T z(x) = w^T z^*,\ x_I = x_I^*,\ z(x) \in b(u, l)\ \}$; $z^R = z(x^R)$

    end

  end

  $i \leftarrow i + 1$

end

if *flagR=0* **then**

  Solve $\min\{z_1(x) : w^T z(x) \leq w^T z^*,\ x_I \neq x_I^*,\ z_2(x) \leq z_2^*,\ z(x) \in b(u, l)\ \}$

  if *not feasible* **then**

    $z^{newR} = z^R$, $z^2 = z^{newR}$

  else

    Let x^S be an optimal solution and $z^S = z(x^S)$

    $x^{newR} \leftarrow \min\{z_2(x) : z_1(x) = z_1^S\ \}$; $z^{newR} = z(x^{newR})$

    if $(z_1^R < z_1^{newR})$ **then**

      $z^2 = z^R$

      $flagboxR = 1$

    else

      $z_1^2 = z_1^{newR}$, $z_2^2 = \frac{z_2^* - z_2^R}{z_1^* - z_1^R}(z_1^{newR} - z_1^*) + z_2^*$, $Ropen = 1$

    end

  end

end

$i = 0,\ x^l \leftarrow \text{lexmin}\{(z_1, z_2) : z_2(x) \leq z_2^* + \epsilon_p,\ x_I = x_I^*\};\ z^l = z(x^l)$

if $|z^l - z^*| < \epsilon$ **then**

  $z^1 = z^*\ ,\ z^{newL} = z^1,\ flagL = 1$

end

if *flagL=0* **then**

  if $flagw = 1$ **then**

    find $x^L \leftarrow \min\{z_1(x) : w^T z(x) = w^T z^*,\ x_I = x_I^*,\ z(x) \in b(u, l)\ \};\ z^L = z(x^L)$

  end

  while *flagw=0* **do**

    $w = [(z_2^l - z_2^*, z_1^* - z_1^l)],\ x^l \leftarrow \min\{w^T z(x) : x_I = x_I^*,\ z(x) \in b(u, l)\};\ z^l = z(x^l)$

    if $(w^T z^l = w^T z^*)$ *and* $|z_1^* - z_1^l| > \epsilon$ *and* $|z_2^* - z_2^l| > \epsilon$ **then**

      $flagw = 1$

      if $i \geq 1$ **then**

        $z^L = z^l$

      else

        $x^L \leftarrow \min\{z_1(x) : w^T z(x) = w^T z^*, x_I = x_I^*, z(x) \in b(u, l)\ \}; z^L = z(x^L)$

      end

    end

    $i \leftarrow i + 1$

  end

end

Solve $\min\{z_2(x) : w^T z(x) \leq w^T z^*,\ x_I \neq x_I^*,\ z_1^* \leq z_1(x),\ z(x) \in b(u, l)\ \}$

if *not feasible* **then**

  $z^{newL} = z^1, z^1 = z^L$

else

  Let x^P be an optimal solution and $z^P = z(x^P)$

  $x^{newL} \leftarrow \min\{z_1(x) : z_2(x) = z_2^P\ \};\ z^{newL} = z(x^{newL})$

  if $(z_2^L < z_2^{newL})$ **then**

    $z^1 = z^L$

    $flagboxL = 1$

  else

    $z_2^1 = z_2^{newL},\ z_1^1 = \frac{(z_2^{newL} - z_2^*)(z_1^L - z_1^*)}{z_2^L - z_2^*} + z_1^*,\ Lopen = 1$

  end

end

Chapter 6

Computational Experiments

The computational results of the proposed algorithms and comparison with the existing algorithms are presented in this section. The performances of the algorithms are tested using BOMILP instances provided in [1]. There are five sets of problems named as C20, C40, C80, C160 and C320, each with five instances. In each set, the number corresponds to the number of variables of the respective problems. In each problem instance, half of the decision variables are binary and the rest are continuous. The mathematical model of the problems is as follows:

$$
\begin{aligned}
\min \quad & z_1(y,x) = \sum_{i=1}^{n_c} c_i^1 y_i + \sum_{i=1}^{n_b} f_j^1 x_j \\
\min \quad & z_2(y,x) = \sum_{i=1}^{n_c} c_i^2 y_i + \sum_{i=1}^{n_b} f_j^2 x_j \\
\text{s.t.} \quad & \sum_{i=1}^{n_c} a_{ij} y_i + a_j^{'} x_j \leq b_j \quad \forall j \in \{1,...,n_b\} \\
& \sum_{i=1}^{n_c} a_{ij} y_i \leq b_j \quad \forall j \in \{n_b+1,...,m-1\} \\
& \sum_{i=1}^{n_b} x_j \leq \frac{n_b}{3} \\
& y_i \in \mathbb{R}_+ \quad \forall i \in \{1,...,n_c\} \\
& x_j \in \{0,1\} \quad \forall j \in \{1,...,n_b\}
\end{aligned}
\tag{6.1}
$$

Here, n_b and n_c denote the number of binary and continuous variables, respectively; c_i^1 and c_i^2 are the associated costs of continuous variables in objective functions whereas f_j^1 and f_j^2 denote the costs of binary decision variables in objective functions; a_{ij} and $a_j^{'}$ are constraint matrix coefficients for continuous variables and binary variables, respectively; and m denotes the number of constraints.

All algorithms are coded in MATLAB using CPLEX 12.7 and run on a computer with Intel Core M-5Y10c 0.8 GHz and 8 GB RAM.

Note that through the algorithms, we use the following tolerances:

- ϵ: It is used when we compare two numbers to determine whether they are equal or not. Note that throughout the algorithms, we check the equivalence of two points by checking the Tchebychev distance between them. If $\max_{i=1,2}|x_i - y_i| < \epsilon$, then x and y are close enough, and treated as they are equal. ϵ is set to 10^{-5} throughout the Algorithms 1 and 2.

- ϵ_p: It is used in lines 2 and 33 of Procedure 3 and set to 10^{-5}

The tolerance values affect the outputs of the algorithms, which we mainly report in terms of the number of nondominated points (N) and the number of nondominated line segments (L) found. We also keep track of the number of linear and mixed integer programming problems (LP and MILP, respectively) as well as the number of ($PS(b)$) problems solved (PS). We illustrate the effect of tolerance parameters with some computational tests on Algorithm 2. We change the value of ϵ_p in (5.1) and report the results in Table 6.1 for C40 instances, which are numbered from 11 to 15.

It is observed that as ϵ_p decreases, both PS and MILP increase. This is because, in this case, more points are considered as isolated, which leads to immediately creating new boxes and solving additional models.

We obtain line segments and nondominated points by solving different scalarization models at different iterations. Hence, a line segment of the nondominated frontier can be represented by small line segments which are found at different times. To rectify this, we perform postprocessing as follows: We take consecutive points which define line segments, say, z^1, z^2 and z^3 from the list of nondominated line segments. Then, we

Table 6.1: Effects of ϵ_p on results of Algorithm 2

Instance	ϵ_p	N	MILP	LP	PS
11	10^{-4}	1106	2570	742	152
	10^{-5}	1158	3035	1309	276
12	10^{-4}	734	1682	511	123
	10^{-5}	756	1832	611	151
13	10^{-4}	989	2402	889	165
	10^{-5}	999	2844	1398	269
14	10^{-4}	990	2360	812	149
	10^{-5}	1000	2677	1140	228
15	10^{-4}	791	1874	637	116
	10^{-5}	784	2071	792	163

compute the slopes of these consecutive line segments as $s_1 = \frac{z_2^1 - z_2^2}{z_1^2 - z_1^1}$ and $s_2 = \frac{z_2^2 - z_2^3}{z_1^3 - z_1^2}$. If $s_1 = s_2$, then we remove z^2 from the list since it is a part of $L(z^1, z^3)$. Table 6.2 shows the change in N and L before (B) and after postprocessing when $\epsilon = 10^{-5}$ (A(10^{-5})) and $\epsilon = 10^{-3}$ (A(10^{-3})) for Algorithm 1 for the same set of instances used in Table 6.1. Here, $\epsilon > 0$ is used to decide whether slope values are equal or not.

Table 6.2: Postprocessing

Instance		N	L
11	B	1521	1475
	A(10^{-5})	1100	1055
	A(10^{-3})	1093	1048
12	B	921	879
	A(10^{-5})	726	684
	A(10^{-3})	708	666
13	B	1795	1724
	A(10^{-5})	1011	940
	A(10^{-3})	998	928
14	B	1578	1526
	A(10^{-5})	991	939
	A(10^{-3})	989	938
15	B	1253	1213
	A(10^{-5})	792	752
	A(10^{-3})	792	752

As ϵ increases the number of nondominated points and line segments are decreasing. Since instances have small line segments with close slope values, we do not want to lose the details of Pareto frontier. Therefore, we use 10^{-5} for rest of the comparison results,

which are given in sections 6.1 and 6.2. CPLEX tolerances are set to 10^{-7}. We choose these values in line with the other studies in the literature.

6.1 Comparison of Algorithms 1 and 2

We now provide the results of our computational experiments, in which we compare the two algorithms we propose. For both algorithms, the number of nondominated points (N) and the number of nondominated line segments (L), the number of mixed integer linear programming problems solved (MILP), the total time (T), the total number of $(PS(b))$ models solved (PS), the time spent per $(PS(b))$ model (tPS), total number of $(WS(\lambda))$ models solved (WS), the time spent per $(WS(\lambda))$ model (tWS), the number of cases that $(PS(b))$ model is infeasible (INF), total number of cases when $y = z^*$ and $n^1 = n^2$ (C3.1), $y_1 = z_1^*$ (C1) and $y_2 = z_2^*$ (C2) are provided.

Table 6.3 reports the average results of each instance set where each set have five instances. Alg. represents the method that we use; A1 and A2 stand for Algorithm 1 and Algorithm 2, respectively. Note that algorithms produce slightly different N values due to their designs: they create different boxes and solve different models. In Chapter 5, it is stated that Algorithm 2 differs from Algorithm 1 when $n^1 = n^2$ (C3.1) and reduces the search region whenever possible by obtaining a line segment whereas Algorithm 1 creates new boxes, which are illustrated in Figures 5.2 and 5.3, respectively. Each new box requires solving either $(WS(b))$ or $(PS(b))$. The average reductions in PS and WS are 65.08% and 16.93%, respectively in Algorithm 2, which directly results in solving fewer MILPs. The improvement of Algorithm 2 over Algorithm 1 in terms of the MILPs solved is 24.53% on average, while the total time is reduced by 19.38% on average.

Table 6.3: Comparison of Alg. 1 and Alg. 2

Instance	Alg.	N	L	MILP	T	PS	tPS	WS	tWS	INF	C3.1	C1	C2
C20	A1	46.80	36.8	256.6	14.23	64.00	0.05	100.6	0.030	9	37	10.4	7.6
	A2	57.4	45.8	161.2	7.57	32.6	0.039	54.8	0.017	6.4	15.8	5.2	5.2
C40	A1	189.6	170.2	922.4	41.68	181.8	0.038	448.4	0.029	18.4	128.8	16	18.6
	A2	203.4	183.2	576.6	31.56	77	0.038	284.6	0.028	8.2	53.4	8	7.4
C80	A1	924	874	3913.8	322.00	636.2	0.079	2188	0.068	49	502.4	39.4	45.4
	A2	939.4	888.8	2491.8	221.78	217.4	0.075	1582.8	0.062	15.8	174.6	12.8	14.2
C160	A1	3677.2	3555.6	13605.4	4359.13	1850.6	0.303	8478.8	0.290	120.6	1546	35	25.6
	A2	3699.2	3577.4	9014.6	3152.84	494	0.291	6609.2	0.270	20	440.2	17.6	16.2
C320	A1	17863	17488.2	55462.6	119173.46	5827.4	2.688	39284.4	1.52	382.8	4906.2	237.6	300.8
	A2	18141.8	17754	43724.4	96069.33	2184.2	2.911	33602.6	1.841	81.2	1581	57	56

We observe that the nondominated line segments of the Pareto frontier are relatively small for the given instances, see Figures A.1 and A.2 in Appendix A. Moreover, there are line segments that can be considered as almost vertical or horizontal, and this structure affects the performance of Algorithm 2 adversely. For example, if z^* obtained by $(PS(b))$ is on an almost vertical line segment, then the result returned by solving (5.1) declares z^* as right isolated for given ϵ. We believe that the total improvement of Algorithm 2 could be higher if different instances are used. This issue is discussed in Chapter 7 as a future work.

6.2 Comparison with Existing Algorithms

It is difficult to directly compare the proposed algorithms to the existing algorithms for the following reasons:

1. All algorithms work as approximation algorithms using different tolerance values, hence provide results with different levels of precision. This difference renders a comparison of solution times difficult. Using high values for the tolerance parameters makes the algorithms work faster, yet it leads to obtaining a less precise approximation of the frontier.

2. The existing algorithms report different types of outputs to evaluate their performances. For instance, SR reports only the solution times and the number of MILP solved whereas ODS reports the number of nondominated line segments. Evaluating the performances by such different measures makes the comparison process harder.

3. The computational environment that results are obtained are different.

For those reasons, we do not compare the solutions times in this section but provide information on other performance measures. Table 6.4 shows the results of the existing algorithms and proposed algorithms. Note that some measures are not returned by some algorithms, hence they are marked as (-). BLM only reports the outputs for instance sets C160 and C320. For further computational tests instead of using C20-C80, new sets of instances were created in [5] to better demonstrate the advantages

of the proposed algorithm. In [5], the authors propose a variant for BLM that solves a weighted sum scalarization problem and slice problem instead of lexicographic optimization problems when the corners of the box have the same integer solutions. In this variant (SIS-BLM), the number of MILPs solved decreases dramatically as seen in Table 6.4. The MILP values of ϵ-TC are based on [1]. ϵ-TC(*) in Table 6.4 indicates the values taken from [5].

Table 6.4: Comparison with Existing Algorithms

Instance	Alg.	N	L	MILP	LP
C20	TSA	54.4	-	133.2	4.4
	ϵ-TC	-	-	56.8	-
	ODS	-	34.8-36.6	58.2	107
	SR	-	-	104.2	-
	A1	46.8	36.8	256.6	-
	A2	57.4	45.8	161.2	93.2
C40	TSA	204	-	428.6	13.2
	ϵ-TC	-	-	201.2	-
	ODS	-	149.4-170.4	195-211.6	408.8-444.4
	SR	-	-	298.4	-
	A1	198.6	170.2	922.4	-
	A2	203.4	183.2	576.6	319.6
C80	TSA	796.6	-	1609.6	101.8
	ϵ-TC	-	-	940.6	-
	ODS	-	650.2-862.2	804.6-978.6	1841.6-2241.8
	SR	-	-	968.6	-
	A1	924	874	3913.8	-
	A2	939.4	888.8	2491.8	1050
C160	TSA	1518.8	-	2839.6	356.8
	ϵ-TC(*)	3545.2	-	3658.6	14029.6
	ODS	-	1301.6-3018.4	1782.4-3341.4	4375.2-8084.4
	SR	-	-	3071.2	-
	BLM	3584	-	10873	30136
	(SIS-BLM)	3548.4	2271.2	10456.8	408.6
	A1	3677.2	3555.6	13605.4	-
	A2	3699.2	3577.4	9014.6	2795.4
C320	TSA	3139.4	-	5388.2	1313.2
	ϵ-TC(*)	16513	-	16873.6	93570.6
	ODS	-	2374-8116.6	3803.2-9185	8256-26239.8
	SR	-	-	10152.2	-
	BLM	17505.4	-	54735.6	173209.6
	(SIS-BLM)	3584	-	10873	30136
	A1	17863	17488.2	55462.6	-
	A2	16554.6	7863.6	47372.4	1432.2

The results demonstrate the differences in the precision of the returned frontiers (shown by N and L). For example, in set C160, TSA finds 57.15%, 57.62% and 58.94% less nondominated points than ϵ-TC, BLM and Algorithm 2, respectively, indicating that it is closer to an approximation algorithm.

ODS uses the number of nondominated line segments found as a performance measure. They report two values obtained from different tolerance levels, hence we provide both of these values in Table 6.4. As mentioned at the beginning of this chapter, since different parts of a line segment may be obtained by different scalarizations at different execution times of our algorithms, we perform postprocessing and report the new line segments after postprocessing. ODS does not require postprocessing.

Recall that in Algorithm 2, when $n^1 = n^2$, we call the line extension procedure to find the line that z^* belongs to. Line extension process follows the same structure with BLM's line extension part and finds two points z^L and z^R by extending to the left and right, respectively. Then, both algorithms search for a nondominated point that dominates $L(z^L, z^R)$. During this search process BLM solves many LPs iteratively, whereas Algorithm 2 does not solve any LP, instead it solves at most four MILPs. In Table 6.4, it is shown that Algorithm 2 solves 91.91% less LPs on average in comparison with BLM. When we compare BLM and Algorithm 2 in terms of MILP, we see that Algorithm 2 solves 18.55% less MILPs. However, the reason of this reduction is not as clear as in LP. There are two possibilities for this reduction: (1) If $u_I = l_I$ for Algorithm 2 and $z_2 < \mu$ for BLM, then they do not require to extend a line. Algorithm 2 solves one weighted sum scalarization (MILP=1) whereas BLM solves two lexicographic optimization problems (MILP=4). (2) When algorithms search a point that dominates $L(z^L, z^R)$ Algorithm 2 defines a search region in $b(u, l)$, therefore we can obtain Case 1(R) and Case 1(L). However BLM conducts a search in a region bounded by z^L and z^R such that $z_1(x) \leq z_1^R,\ z_2(x) \leq z_2^L$. Hence it requires solving additional MILPs in the next iterations to find the point obtained by Case 1(R) which may result in solving additional MILPs, see Figure 6.1.

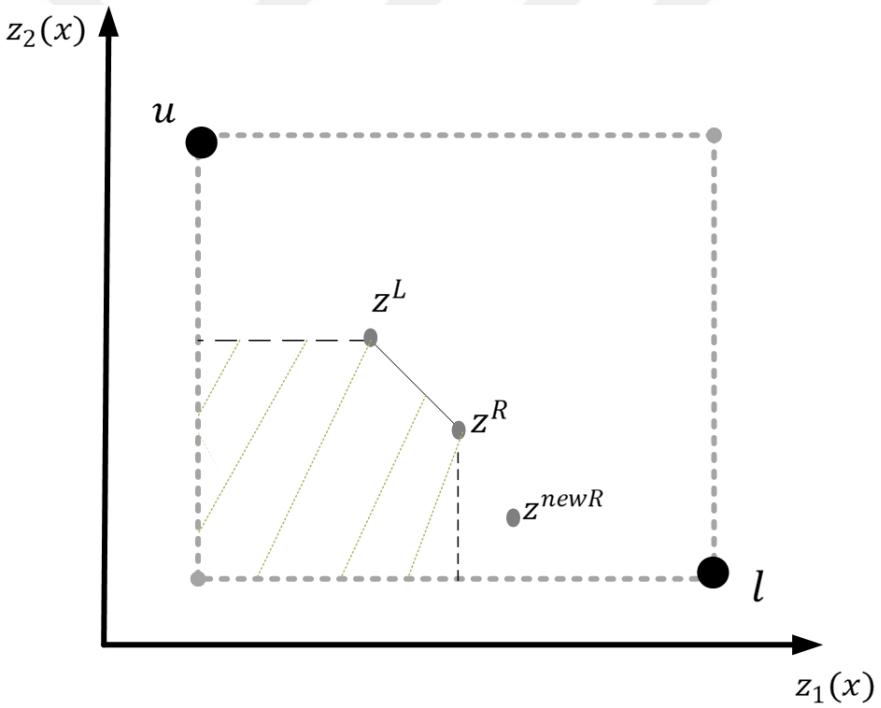

Figure 6.1: BLM misses z^{newR} since its search region is defined by $z_1(x) \leq z_1^R,\ \ z_2(x) \leq z_2^L$

Chapter 7

Conclusion and Future Research

We propose two criterion space solution algorithms for BOMILP, which are based on solving weighted sum scalarizations and Pascoletti-Serafini scalarizations, sequentially. Both algorithms obtain the whole Pareto frontier. The major contribution of both algorithms is using Pascoletti-Serafini scalarization for BOMILP, and obtaining diverse set of solutions at any given time. After solving Pascoletti-Serafini scalarization, Algorithm 1 immediately creates two boxes with the obtained point (z^*). However, Algorithm 2 takes the obtained point as an input and solves additional models to find the nondominated line segment that z^* belongs to if there is any. Detecting this line reduces the search region to be searched, hence decreases the number of mixted integer linear programming problems solved. Therefore, when we compare the performances of the proposed algorithms, Algorithm 2 outperforms in terms of the solution time and the number of mixed integer linear programming problems solved.

As mentioned in Chapter 6, the test instances have small line segments which are almost horizontal or vertical in some parts of the nondominated frontier. These cases may affect the performance of Algorithm 2. We believe that the improvement of Algorithm 2 over Algorithm 1 would be higher if we use different instances with less number of such almost horizontal and vertical line segments. In [5], this concern is also mentioned by stating that the instances may bias the performance comparison. Hence new instances are introduced in [5]. The performance of Algorithm 2 can be evaluated using these instances for further research.

When the proposed algorithms and the existing algorithms are compared, it is difficult to say that one outperforms the others, as stated in 6.2. This comparison mainly depends on the performance measure, for example ϵ-TC, BLM and Algorithm 2 provide more detail with respect to N when they are compared to TSA. However, if the performance measure is choosen as MILP, then TSA outperforms all others with its approximation feature, which demonstrates a trade-off between accuracy and computational effort.

Bibliography

[1] N. Boland, H. Charkhgard, and M. Savelsbergh, "A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method," *INFORMS Journal on Computing*, vol. 27, pp. 597–618, 2015.

[2] B. Soylu and G. Yildiz, "An exact algorithm for biobjective mixed integer linear programming problems," *Computers and Operations Research*, vol. 72, pp. 204–213, 2016.

[3] A. Fattahi and M. Turkay, "A one direction search method to find the exact nondominated frontier of biobjective mixed-binary linear programming problems," *European Journal of Operational Research*, vol. 266(2), pp. 415–425, 2018.

[4] B. Soylu, "The search-and-remove algorithm for biobjective mixed-integer linear programming problems," *European Journal of Operational Research*, vol. 268, pp. 281–299, 2018.

[5] T. Perini, N. Boland, D. Pecin, and M. Savelsbergh, "Criterion space method for biobjective mixed integer programming: The boxed line method," *INFORMS Journal on Computing*, vol. 32(1), pp. 16–39, 2019.

[6] A. Pascoletti and P. Serafini, "Scalarizing vector optimization problems," *Journal of Optimization Theory and Applications*, vol. 42, pp. 499–524, 1984.

[7] S. Dogan, "An exact algorithm for biobjective integer programming problems," *Bilkent University Institutional Repository*, 2019.

[8] M. Ehrgott, "Multicriteria optimization," *Springer*, 2005.

[9] V. Chankong and Y. Haimes, "Optimization-based methods for multiobjective decision-making: An overview," *Large Scale Systems In Information And Decision Technologies*, vol. 5(1, pp. 1–33, 1983.

[10] N. Boland, H. Charkhgard, and M. Savelsbergh, "A criterion space search algorithm for biobjective integer programming: The balanced box method," *INFORMS Journal on Computing*, vol. 27(4), pp. 735–754, 2015.

Appendix A

Nondominated Line Segments which are almost horizontal or vertical

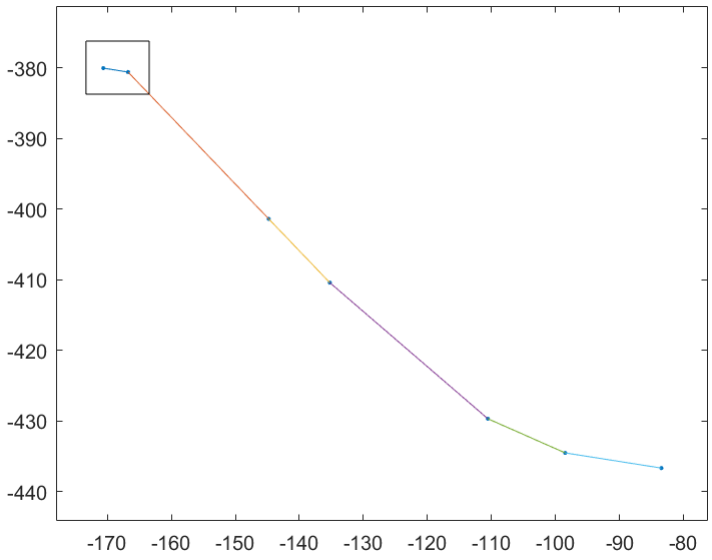

Figure A.1: Almost horizontal nondominated line segment: z^1=(−170.6334, −380.0483), z^2=(−166.7753, −380.6011)

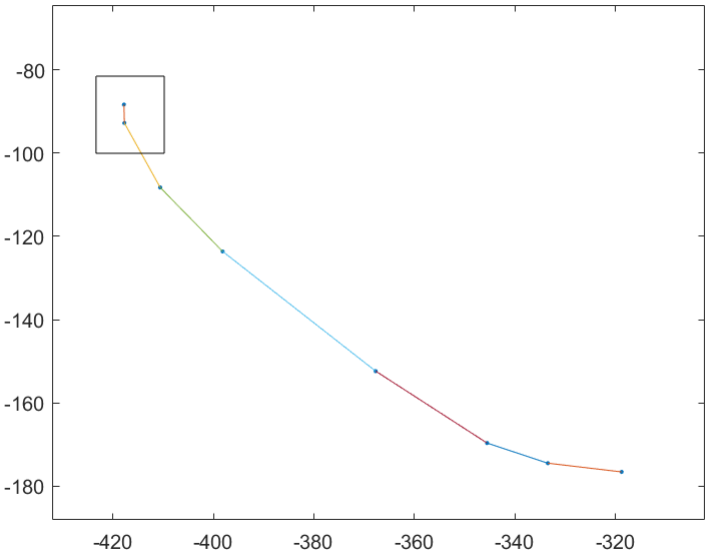

Figure A.2: Almost vertical nondominated line segment: z^1=(−417.7815, −88.3073), z^2=(−417.7027, −92.7430)