# AN EXACT ALGORITHM FOR BIOBJECTIVE 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
Saliha Ferda Doğan
July 2019

An Exact Algorithm for Biobjective Integer Programming Problems
By Saliha Ferda Doğan
July 2019

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)

Özlem Çavuş İyigün

Esra Karasakal

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

# ABSTRACT

## AN EXACT ALGORITHM FOR BIOBJECTIVE INTEGER PROGRAMMING PROBLEMS

Saliha Ferda Doğan
M.S. in Industrial Engineering
Advisor: Firdevs Ulus
Co-Advisor: Özlem Karsu
July 2019

We propose an exact algorithm to find all nondominated points of biobjective integer programming problems, which arise in various applications of operations research. The algorithm is based on dividing objective space into regions (boxes) and searching them by solving Pascoletti-Serafini scalarizations with fixed direction vector. We develop variants of the algorithm, where the choice of the scalarization model parameters differ; and demonstrate their performance through computational experiments both as exact algorithms and as solution approaches under time restriction. The results of our experiments show the satisfactory behaviour of our algorithm, especially with respect to the number of mixed integer programming problems solved compared to an existing approach. The experiments also demonstrate that different variants have advantages in different aspects: while some variants are quicker in finding the whole set of nondominated solutions, other variants return good-quality solutions in terms of representativeness when run under time restriction.

*Keywords:* Biobjective integer programming; Pascoletti-Serafini scalarization; Algorithms.

# ÖZET

# TÜRKÇE BAŞLIK

Saliha Ferda Doğan
Endüstri Mühendisliği, Yüksek Lisans
Tez Danışmanı: Firdevs Ulus
İkinci Tez Danışmanı: Özlem Karsu
July 2019

Birçok yöneylem araştırması uygulamasında ortaya çıkan iki amaçlı tamsayılı programlama problemlerinin tüm baskın noktalarını bulabilmek için amaç fonksiyonu uzayında arama yapan kesin bir algoritma önerildi. Bu algoritma, arama uzayının kutulara bölünmesi ve yön vektörü sabitlenmiş bir Pascoletti-Serafini skalarizasyon modeli çözülerek kutuların aranmasına dayanmaktadır. Algoritmanın skalarizasyon probleminde kullanılan parametre seçimlerinde değişiklik yapılarak varyantları geliştirildi. Bu varyantların hem kesin birer algoritma olarak hem de zaman kısıtlaması altındaki çözüm yaklaşımları olarak performansları bilgisayımsal deneyler yapılarak ortaya konuldu. Deney sonuçları algoritmanın özellikle çözülen tamsayılı programlama problemleri bakımından mevcut yaklaşımlara göre daha iyi olduğunu göstermektedir. Deneyler ayrıca farklı varyantların farklı açılardan avantajlarını ortaya koymaktadır: bazı varyantlar tüm baskın çözüm kümesini daha kısa sürede bulurken, diğerleri zaman kısıtlaması altında temsil bakımından daha iyi kalitede çözümler döndürmektedir.

*Anahtar sözcükler*: İki Amaçlı Tam Sayılı Programlama, Pascoletti-Serafini skalarizasyonu, Algoritmalar.

# Acknowledgement

I would like to express my sincere gratitude to Asst. Prof. Firdevs Ulus and Asst. Prof. Özlem Karsu, for being such supporting, understanding, and open-hearted advisors, and also for their patient guidance and useful critiques for this research work. They have broadened my horizon with their knowledge and experience.

I would like to thank Asst. Prof. Özlem Çavuş İyigün and Prof. Esra Karasakal for reading and evaluating my thesis.

I am grateful to my parents, my father Sakıp, for his supports and self-sacrifice throughout my life, he is an inspiration of this journey, my mother Gülizar, for being such a warm-hearted and graceful. I feel her love at every second of my life, which helps me to overcome problems that I encountered. She is the pole star of my life. Furthermore, I would like to thank my siblings, Çağrı and Zeynep. They have brightened my life in many ways and made me who I am. They also have been always there to cheer me up in difficult times. I can not think of life without them. I also want to thank my brother's wife, Hatice, and my beautiful nieces, Elif and Beyza. They made my life better with their endless love. I learn true love and sacrifice. They are the sunshine of my life.

I also want to thank my friends in our department, Bashir Abdullahi Bashir, for all contributions to my master journey. He has witnessed my struggle since the very beginning of the master. Deniz Emre, for having such a beautiful soul and being lovely and sincere person. I feel lucky to find my soul twin. Furthermore, I am grateful to Merve Bolat, Gül Çulhan, Pelin Keşrit, Parinaz Toufani and Milad Malekipirbazari for the supportive and joyful environment they have provided during our graduate studies. I also would like to thank all the members of our department for creating this nice community.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Many real life problems in different areas such as scheduling, task assignment and transportation can be formulated as integer programming problems. Moreover, most real world problems include multiple criteria which are conflicting, so it is not possible to find a feasible solution that optimizes all objectives simultaneously. Therefore, generating the set of (or a subset of) nondominated points is important.

In this work, we focus on biobjective integer programming problems (BOIP) and propose an algorithm that returns the whole set of nondominated points of these problems. There are various algorithms that have been designed for BOIP in the literature. These algorithms can be divided into two according to the space they are searching, i.e., decision space search algorithms, which search in the space of feasible solutions, and objective (criterion) space search algorithms which search in the space of objective function values. The algorithms which explore the objective space solve single objective optimization problems related to the BOIP, called scalarization problems, repetitively. A scalarization is formulated by means of a real-valued scalarizing function of the objective functions of the BOIP, auxiliary scalar or vector variables and/or parameters [1].

There are several scalarizations proposed in the literature. The widely-used

ones are the weighted sum scalarization [2, 3], the $\epsilon$-constraint scalarization [4] and the (weighted) Chebyshev scalarization [5, 6]. Most of the current algorithms in the literature solve these scalarizations or their modifications repetitively to find the set of nondominated points. Commonly used ones are the perpendicular search and the $\epsilon$-constraint algorithm, which are based on weighted sum scalarization and $\epsilon$-constraint scalarization, respectively [7, 4, 8]. Examples of algorithms using weighted Chebychev scalarizations are proposed by [9] and [10], where a modified version of the scalarization is used. There are also two-phase algorithms, which generate supported nondominated points in the first phase and find the unsupported nondominated points by exploring the triangles defined by two consecutive supported nondominated points in the second phase [11, 12]. Recently, the balanced box algorithm is proposed by [13] and a two-stage algorithm which combines the balanced box and $\epsilon$-constraint algorithms is discussed by [14].

We propose an exact algorithm that finds the whole set of nondominated points to biobjective integer programming problems by searching predefined areas in the objective space. The algorithm is based on Pascoletti-Serafini scalarization [15], which has two parameters: a direction vector and a reference point. We adapt this scalarization model for biobjective integer programming settings and develop different variants of the algorithm by changing the selection rules of these parameters. In particular, we consider two different ways of selecting the reference point and three different ways of selecting the direction parameter. We compare these variants with respect to number of (mixed) integer programming problems solved and solution time. We also test the performances of the variants under time limit and report on the representativeness of the obtained solution sets using the (scaled) coverage error [16, 17]. Also, we compare the prominent variants with balanced box algorithm with respect to the number of integer programming problems solved.

The rest of this thesis is as follows. In Chapter 2, we give the preliminaries and the problem definition. In Chapter 3, we review the literature. In Chapters 4 and 5, we explain the base algorithm and its variants, respectively. We test the performances of the algorithm and report the results of our experiments in Chapter 6. We conclude our discussion in Chapter 7.

# Chapter 2

# Problem Definition and Preliminaries

In this chapter, we define biobjective integer programming problem (BOIP) and introduce some notations related to BOIP to facilitate presentation and discussion of other chapters.

A general biobjective integer programming problem is formulated as

$$\text{“min”}\big\{\, z = \big(z_1(x), z_2(x)\big) \mid x \in \mathcal{X} \subset \mathbb{Z}^n \,\big\}, \tag{$P$}$$

where $z_i(x)$, $i = 1, 2$ are integer-valued objective functions. The set $\mathcal{X}$ represents the feasible set in the decision space and the set $\mathcal{Y} := z(\mathcal{X})$ represents the feasible set in the objective/criterion space.

Throughout the thesis, the following notation is used, for $z^1, z^2 \in \mathbb{Z}^2$ :

$$\begin{aligned} z^1 \leq z^2 &\iff z_i^1 \leq z_i^2,\ \forall i, \\ z^1 \lneq z^2 &\iff z^1 \leq z^2 \text{ and } z^1 \neq z^2, \\ z^1 < z^2 &\iff z_i^1 < z_i^2,\ \forall i. \end{aligned}$$

Also, $\mathbb{R}^2_{\geq} := \big\{z \in \mathbb{R}^2 \mid z \geq 0\big\}$, $\mathbb{R}^2_{\gneq} := \big\{z \in \mathbb{R}^2 \mid z \gneq 0\big\}$.

A feasible outcome $z(x) \in \mathcal{Y}$ is dominated by $z(x^{'}) \in \mathcal{Y}$ or $z(x^{'})$ *dominates*

$z(x)$, if $z(x^{'}) \lneq z(x)$. If $z(x^{'}) < z(x)$, then $z(x^{'})$ *strictly dominates* $z(x)$. If there exists no $z(x^{'})$ that (strictly) dominates $z(x)$, then $z(x)$ is (*weakly*) *nondominated* and $x$ is (*weakly*) *efficient*.

An efficient solution $x'$ is a *supported efficient solution*, if it is an optimal solution of the following problem

$$\min\{\, \mu z_1(x) + (1-\mu) z_2(x) \mid x \in \mathcal{X} \},$$

where $0 < \mu < 1$. Then, $z(x^{'})$ is a *supported nondominated point*. Also, if an efficient solution $x^{'}$ is not supported, then it is *unsupported efficient solution* and $z(x^{'})$ is an *unsupported nondominated point*. The set of all weakly nondominated, nondominated and supported nondominated points are denoted by $\mathcal{N}_w$, $\mathcal{N}$ and $\mathcal{N}_s$, respectively.

In order to illustrate the definitions provided above, we consider an example where the feasible set in the objective space $z(\mathcal{X}) = \{a, b, c, d, e, f, g, h, i\}$ is given as in Figure 2.1.

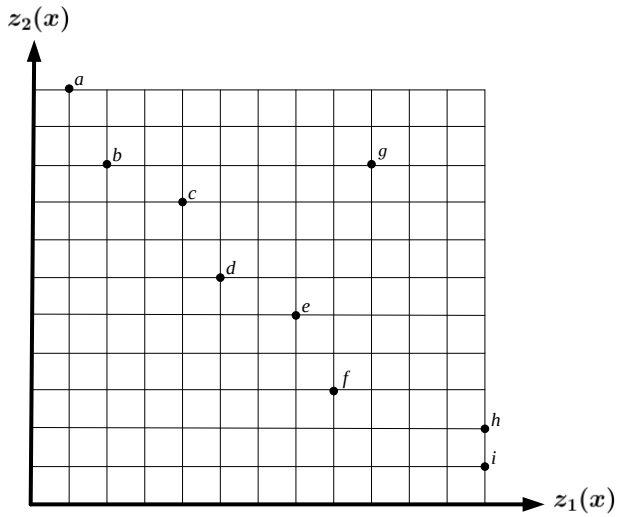


Figure 2.1: An example setting

For this example, $\mathcal{N}_w = \{a, b, c, d, e, f, h, i\}$, $\mathcal{N} = \{a, b, c, d, e, f, i\}$, $\mathcal{N}_s = \{a, b, d, f, i\}$.

Two specific points defined in the objective space are the ideal point and the nadir point. For $(P)$ the ideal point is

$$s^0 := \left( \min_{x \in \mathcal{X}} z_1(x),\ \min_{x \in \mathcal{X}} z_2(x) \right)$$

and the nadir point is

$$u^0 := \left( \max_{z \in \mathcal{N}} z_1, \max_{z \in \mathcal{N}} z_2 \right).$$

Another important concept that will be used throughout the thesis is the lexicographic optimization concept. It can be denoted as follows for $(P)$:

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

where $i, j \in \{1, 2\}$ and $i \neq j$.

Solving (2.1) requires solving two optimization problems in sequence. It starts with minimizing the $i^{th}$ objective, that is, one solves

$$\min\{\ z_i(x) \mid x \in \mathcal{X}\ \}.$$

Let an optimal solution of the problem be $x^{'}$. The procedure continues with solving the following problem

$$\min\{\ z_j(x) \mid x \in \mathcal{X},\ z_i(x) = z_i(x^{'})\ \}.$$

An optimal solution of the second model is an optimal solution to (2.1).

# Chapter 3

# Literature Review

The reduction of a biobjective optimization problem to a single objective optimization problem that is solved repeatedly to find the set of nondominated points is called scalarization. There are two important properties of a scalarization model: whether it allows you to find the whole set of nondominated points by changing the parameters of the model and whether the solution found by solving it is guaranteed to be efficient or only weakly efficient. Note that, these depend on the structure of the BOIP considered.

Many algorithms that are proposed to find the set of nondominated points of a BOIP utilize these scalarization models. In the following two sections, we briefly review the literature related to scalarization models and the algorithms that use these models to solve BOIP problems.

## 3.1 Scalarization Models

There are several scalarization models proposed in the literature. We discuss the mostly used ones in BOIP settings.

### 3.1.1 The Weighted Sum Scalarization

One of the well known scalarizations is the weighted sum scalarization [2, 3]. The model is given by

$$\min\{\, \mu z_1(x) + (1-\mu) z_2(x) \mid x \in \mathcal{X} \}, \tag{3.1}$$

where $0 < \mu < 1$.

**Lemma 1.** *[1] An optimal solution of* (3.1) *is an efficient solution. If* $x^{'}$ *is a supported efficient solution of* $(P)$*, then there exists* $\mu > 0$ *such that* $x^{'}$ *is an optimal solution of* (3.1)*.*

**Remark 1.** *The weighted sum scalarization can not find any unsupported nondominated point which follows from the definition of unsupported nondominated points.*

Consider point $e$ in Figure 2.1, which is an unsupported nondominated point, can not be an optimal solution to a weighted sum scalarization.

### 3.1.2 The $\epsilon$-Constraint Scalarization

The $\epsilon$-constraint scalarization is one of the prominent scalarizations used for BOIP [4]. While one of the two objectives is retained as objective, the other one is used in constraint of the model which is

$$\min\{\, z_i(x) \mid x \in \mathcal{X},\ z_j(x) \leq \epsilon,\ j \neq i \}, \tag{3.2}$$

where $i \in \{1, 2\}$. A detailed discussion can be found in [18].

**Lemma 2.** *[18] An optimal solution of* (3.2) *is weakly efficient. If* $x^{'}$ *is an efficient solution of* $(P)$*, then there exists* $\epsilon$ *such that* $x^{'}$ *is an optimal solution of* (3.2)*.*

The augmented $\epsilon$-constraint [19] can be considered to guarantee that the solutions obtained are efficient. It is given by

$$\min\{\, z_i(x) + \mu\ z_j(x) \mid x \in \mathcal{X},\ z_j(x) \leq \epsilon,\ j \neq i\}, \tag{3.3}$$

where $\mu > 0$ is a small number.

**Lemma 3.** *[19] An optimal solution of* (3.3) *is efficient. If* $x^{'}$ *is an efficient solution of* $(P)$*, then there exists* $\epsilon$ *and* $\mu$ *such that* $x^{'}$ *is an optimal solution of* (3.3).

### 3.1.3 The Weighted Chebychev Scalarization

The weighted Chebyshev scalarization [5] is as follows

$$\min\{\, \max_i \mu_i\big(z_i(x) - s_i\big) \mid x \in \mathcal{X}\}, \tag{3.4}$$

where $0 < \mu_1 < 1,\ \mu_2 = 1 - \mu_1$ and $s \in \mathbb{R}^2$ is a reference point such that $s_i < \min_{x\in\mathcal{X}} z_i,\ i \in \{1, 2\}$.

**Lemma 4.** *[20] An optimal solution of* (3.4) *is weakly efficient. If* $x^{'}$ *is an efficient solution of* $(P)$*, then there exists* $\mu > 0$ *such that* $x^{'}$ *is optimal for* (3.4).

In order to guarantee that the solutions obtained are efficient, the augmented weighted Chebychev scalarization [6] can be considered. It is given by

$$\min\{\, \max_i \mu_i \left(z_i(x) - s_i^0\right) + \lambda \sum_{i=1}^{2} (z_i(x) - s_i^0) \mid x \in \mathcal{X}\}, \tag{3.5}$$

where $0 < \mu_1 < 1,\ \mu_2 = 1 - \mu_1$ and ideal point $s^0$ is the reference point.

**Lemma 5.** *[6] If* $\lambda > 0$*, then an optimal solution of* (3.5) *is an efficient solution. If* $x^{'}$ *is an efficient solution of* $(P)$*, then there exists* $\lambda > 0$ *such that* $x^{'}$ *is optimal for* (3.5).

### 3.1.4 The Pascoletti and Serafini Scalarization

One of the prominent scalarization techniques is Pascoletti and Serafini scalarization [15]. It is given by the following scalar problem, which employs two parameters a reference point $s \in \mathbb{R}^2$ and direction $d \in \mathbb{R}^2_{\gtrsim}$

$$\min\{\ \rho \mid x \in \mathcal{X},\ z(x) \leq s + \rho d, \rho \in \mathbb{R}\}. \tag{3.6}$$

**Lemma 6.** *[15] An optimal solution of* (3.6) *is weakly efficient. If* $x^{'}$ *is an efficient solution of* $(P)$*, then there exists* $s$ *and* $d$ *such that* $x^{'}$ *is optimal for (3.6).*

One can modify the above model to ensure that the solution obtained is an efficient solution. One modification was proposed by Akbari et al. [21], namely the Modified Pascoletti-Serafini scalarization. The scalarization model is as follows

$$\min\{\ \rho - \sum_{i=1}^{2} \mu_i a_i \mid x \in \mathcal{X},\ z(x) \leq s + \rho d - a,\ \rho \in \mathbb{R},\ a \in \mathbb{R}^2_{\geq}\}, \tag{3.7}$$

where $\mu > 0$.

**Lemma 7.** *[21] If* $\mu > 0$*, an optimal solution of* (3.7) *is an efficient solution. If* $x^{'}$ *is an efficient solution of* $(P)$*, then there exists* $\mu \geq 0$*,* $s \in \mathbb{R}^2$ *and* $d \in \mathbb{R}^2_{\gtrsim}$ *such that* $x^{'}$ *is optimal for* (3.7)*.*

**Remark 2.** *Note that some of the scalarizations guarantee finding a weakly efficient solution rather than an efficient one. We see that there are modified versions of these models which guarantee finding an efficient solution. Instead of these modified models one can also solve a second stage model [10].*

## 3.2 Algorithms

The algorithms differ with respect to the scalarization employed and not all of them find the set of all nondominated points $\mathcal{N}$. In this section, we present some of the algorithms by categorizing according to the scalarization and indicate the what kind of set it provides.

### 3.2.1 Algorithms using the Weighted Sum Scalarization

#### 3.2.1.1 The Perpendicular Search

The perpendicular search algorithm [7] divides the objective space into boxes and explores them by solving a variation of the weighted sum scalarization. Each box is defined by two nondominated points and generic box is denoted as $[z^a, z^b] := \left\{z \in \mathbb{R}^2 \mid z_1^a \leq z_1 \leq z_1^b, z_2^b \leq z_2 \leq z_2^a\right\}$, where $z^a$ and $z^b$ are nondominated points. Each box is added to a set $\mathcal{B}$, wich is a collection of boxes to be explored in the following iterations.

The main steps of the algorithm can be seen below:

(S0) Solve $\text{lexmin}\left\{ z_1(x), z_2(x) \mid x \in \mathcal{X}\right\}$ and $\text{lexmin}\left\{z_2(x), z_1(x) \mid x \in \mathcal{X}\right\}$. Let optimal solutions be $x'$, $x''$ and define $z^1 := z\left(x'\right)$, $z^2 := z\left(x''\right)$, respectively. Initialize the set of nondominated points as $\mathcal{N} := \left\{z^1, z^2\right\}$ and the set of boxes to be explored as $\mathcal{B} := \left\{[z^1, z^2]\right\}$.
Set $\mu_1 > 0, \mu_2 > 0, 0 < \epsilon < 1$.

(S1) Consider $[z^a, z^b] \in \mathcal{B}$.
Solve $\min\left\{ \mu_1 z_1(x) + \mu_2 z_2(x) \mid x \in \mathcal{X},\ z_1(x) \leq\ z_1^b - \epsilon,\ z_2(x) \leq\ z_2^a - \epsilon \right\}$.

(S2) $\mathcal{B} \leftarrow \mathcal{B} \setminus \left\{[z^a, z^b]\right\}$.
If the problem is feasible, let an optimal solution be $x^{'}$ and $z^{'} := z(x^{'})$, $\mathcal{N} \leftarrow \mathcal{N} \cup \left\{z^{'}\right\}$, $\mathcal{B} \leftarrow \mathcal{B} \cup \left\{[z^a, z^{'}], [z^{'}, z^b]\right\}$.

(S3) If $\mathcal{B} = \emptyset$, stop. Otherwise, go to (S1).

Note that the constraints that are added to the weighted sum scalarization allow us to find unsupported nondominated points besides supported nondominated points, hence it is possible to find all nondominated points by this algorithm.

In the literature, there is a variation of the perpendicular search algorithm, which is called the binary search algorithm [22]. It solves the same scalarization

problem in (S1) by specifying the coefficients $\mu_1$ and $\mu_2$ as $\mu_1 = z_2^1 - z_2^2$, $\mu_2 = z_1^2 - z_1^1$ instead of fixing them at the beginning, and keeping the other steps the same.

#### 3.2.1.2 Two-Phase

Two-phase algorithms have been used various studies in the literature [11, 12]. In the first phase, supported nondominated points are generated by using weighted sum scalarization. The second phase is used to find the unsupported nondominated points by exploring the triangles defined by two consecutive supported nondominated points in the objective space. In this phase, lower bounds, upper bounds etc. are usually employed to not return the nondominated points found before.

### 3.2.2 The Algorithm using the $\epsilon$-Constraint Scalarization

The $\epsilon$-constraint algorithm is initialized by finding one of the corner points of the objective space, the best nondominated point according to the first or the second objective. Then, it finds the nondominated points iteratively, by moving from the already found corner point to the other corner point.

The main steps of the algorithm can be seen below:

(S0) Solve lexmin$\{ z_1(x), z_2(x) \mid x \in \mathcal{X} \}$, let an optimal solution be $x'$, and define $z^1 := z(x')$. $\mathcal{N} := \{z^1, z^2\}$.

(S1) $\epsilon = z_2(x^{'}) - 1$. Solve lexmin$\{ z_1(x), z_2(x) \mid x \in \mathcal{X},\ z_2(x) \leq \epsilon \}$.

(S2) If it is infeasible, stop.
Otherwise, let an optimal solution be $x^{'}$ and $z^{'} := z(x^{'})$, $\mathcal{N} \leftarrow \mathcal{N} \cup \{z^{'}\}$, go to (S1).

Note that, it is possible to find all nondominated points by this algorithm.

**Lemma 8.** *[8] The $\epsilon$-constraint algorithm solves at most 2 $|\mathcal{N}|$+1 problems.*

### 3.2.3 Algorithms using Weighted Chebychev Scalarization

One of the algorithms that uses the weighted Chebychev scalarization is proposed by Ralphs et al. [9]. This algorithm explores regions identified by two (weakly) nondominated points $z^a$, $z^b$ and denoted as $[z^a, z^b]$.

The main steps of the algorithm can be seen below:

(S0) Solve (3.4) for $\mu_1 = 1$ and $\mu_1 = 0$, let optimal solutions be $x'$, $x''$ and define $z^1 := z(x')$, $z^2 := z\left(x''\right)$, respectively. $\mathcal{N}_w := \left\{z^1, z^2\right\}$ $\mathcal{B} := \left\{[z^1, z^2]\right\}$.

(S1) Consider $[z^a, z^b] \in \mathcal{B}$, set $\mu_1 = (z_2^a - s_2^0)/(z_2^a - s_2^0 + z_1^b - s_1^0)$, $\mathcal{B} \leftarrow \mathcal{B} \backslash \left\{[z^a, z^b]\right\}$. Solve (3.4), let optimal solution be $x^{'}$ and define $z^{'} := z(x^{'})$. If $z^{'} \neq z^a$ or $z^{'} \neq z^b$, $\mathcal{N}_w \leftarrow \mathcal{N}_w \cup \left\{z^{'}\right\}$, $\mathcal{B} \leftarrow \mathcal{B} \cup \left\{[z^a, z^{'}], [z^{'}, z^b]\right\}$.

(S2) If $\mathcal{B} = \emptyset$, stop. Otherwise, go to (S1).

Note that, this algorithm returns a subset of $\mathcal{N}_w$ that contains $\mathcal{N}$, so a post processing step is required to determine $\mathcal{N}$.

### 3.2.4 The Balanced Box algorithm

The balanced box algorithm [13] divides the objective space into boxes which is defined by two points in the objective space and denoted as $\left[z^a, z^b\right] := \left\{z \in \mathbb{R}^2 \mid z_1^a \leq z_1 \leq z_1^b, z_2^b \leq z_2 \leq z_2^a\right\}$, where $z^a, z^b \in \mathcal{Y}$.

The main steps of the algorithm can be seen below:

(S0) Solve lexmin$\left\{ z_1(x), z_2(x) \mid x \in \mathcal{X}\right\}$ and lexmin$\left\{z_2(x), z_1(x) \mid x \in \mathcal{X}\right\}$ let

optimal solutions be $x'$, $x''$ and define $z^1 := z(x')$, $z^2 := z\left(x''\right)$, respectively. $\mathcal{N} := \left\{z^1, z^2\right\}$, $\mathcal{B} := \left\{[z^1, z^2]\right\}$, and set $0 < \epsilon < 1$.

(S1) Consider $[z^a, z^b] \in \mathcal{B}$.
Split $[z^a, z^b]$ horizontally into two boxes $\left[z^a, z^{a'}\right]$ and $\left[z^{b'}, z^b\right]$ where $z^{a'} = \left(z_1^b, \left(z_2^a + z_2^b\right)/2\right)$ and $z^{b'} = \left(z_1^a, \left(z_2^a + z_2^b\right)/2\right)$, $\mathcal{B} \leftarrow \mathcal{B} \cup \left(\left[z^a, z^{a'}\right], \left[z^{b'}, z^b\right]\right)$, $\mathcal{B} \leftarrow \mathcal{B} \setminus \left\{\left[z^a, z^b\right]\right\}$.

(S2) Solve lexmin$\left\{ z_1(x), z_2(x) \mid x \in \mathcal{X},\ z(x) \in \left[z^{b'}, z^b\right]\right\}$, let an optimal solution be $x'$ and $z' := z(x')$.
If $z' \neq z^b$, update $\left[z^{b'}, z^b\right]$ as $\left[z', z^b\right]$ and $\left[z^a, z^{a'}\right]$ by setting $z^{a'} = \left(z_1' - \epsilon, \left(z_2^a + z_2^b\right)/2\right)$, $\mathcal{N} \leftarrow \mathcal{N} \cup \left\{z'\right\}$.
Otherwise, $\mathcal{B} \leftarrow \mathcal{B} \setminus \left\{\left[z^{b'}, z^b\right]\right\}$.

(S3) Solve lexmin$\left\{ z_2(x), z_1(x) \mid x \in \mathcal{X},\ z(x) \in \left[z^a, z^{a'}\right]\right\}$, let an optimal solution be $x^*$ and $z^* := z(x^*)$.
If $z^* \neq z^a$, update $\left[z^a, z^{a'}\right]$ as $\left[z^a, z^*\right]$, $\mathcal{N} \leftarrow \mathcal{N} \cup \left\{z^*\right\}$.
Otherwise, $\mathcal{B} \leftarrow \mathcal{B} \setminus \left\{\left[z^a, z^{a'}\right]\right\}$.

(S4) If $\mathcal{B} = \emptyset$, stop. Otherwise, go to (S1).

Note that searching the box $\left[z^a, z^{a'}\right]$ $\left(\left[z^{b'}, z^b\right]\right)$ returns $z^a$ $(z^b)$ if and only if it is the only nondominated point in $\left[z^a, z^{a'}\right]$ $\left(\left[z^{b'}, z^b\right]\right)$. Also, note that it is possible to find all nondominated points by this algorithm.

**Lemma 9.** *[13] The balanced box algorithm solves at most 3 $|\mathcal{N}|$ problems.*

[14] propose a two-stage algorithm which combines the balanced box and $\epsilon$-constraint algorithms. In the first stage, it uses the balanced box and generates some nondominated points in different portions of the objective space. Then, at one point it switches to the second stage and uses $\epsilon$-constraint algorithm to find rest of the nondominated points. Combining $\epsilon$-constraint algorithm with balanced box algorithm reduces the number of problems solved.

Note that, the time of the switch has a significant impact on the performance of the algorithm, hence the authors determine an ideal switch time.

**Lemma 10.** *[14] The two-stage algorithm with ideal switch solves at most* $\lceil 2.5|\mathcal{N}| \rceil$ *problems.*

We propose another algorithm which using Pascoletti and Serafini scalarization in the next chapter.

# Chapter 4

# An Exact Algorithm for BOIP Problems

In this chapter, we explain the algorithm that is proposed for BOIP problems. Our aim is to generate all nondominated points of problem $(P)$.

The general idea of the algorithm can be described as follows. It divides the objective space into boxes and at each iteration it explores one of them by solving a Pascoletti-Serafini scalarization problem to find a (weakly) nondominated point. In order to ensure that the point is nondominated in the strict sense, an extra model(s) is(are) solved. Then, the explored box is discarded and two new boxes are defined to be explored in the next iterations. The algorithm continues until there are no boxes to explore.

In order to clarify the algorithm, first we explain initialization, and give some necessary details about the initial box. Then, we clarify the main loop.

The initialization step starts with defining the sets, $\mathcal{N}$ and $\mathcal{B}$ to represent the set of nondominated points and boxes to be investigated, respectively. Then, to determine the initial box that contains all nondominated points first, the following

lexicographic problem

$$\text{lexmin}\{\, z_1(x), z_2(x) \mid x \in \mathcal{X} \,\}$$

is solved and the nondominated point which is at the upper left-hand corner of the objective space of the problem $(P)$ is found. Let the nondominated point be $t^0$. Next, the following lexicographic problem

$$\text{lexmin}\{\, z_2(x), z_1(x) \mid x \in \mathcal{X} \,\}$$

is solved and the nondominated point at the bottom right-hand corner of the objective space is found. Let the nondominated point be $p^0$. Notice that, two outer corner nondominated points of the objective space $t^0$ and $p^0$ define the ideal point $s^0$, such that $s^0 = (t_1^0,\ p_2^0)$ and these three points define a sufficiently large box that includes all nondominated points, where the first component of $p^0$, $p_1^0$ is an upper bound for the first objective and the second component of $t^0$, $t_2^0$ is an upper bound for the second objective, see Figure 4.1 for the illustration of the initial box.

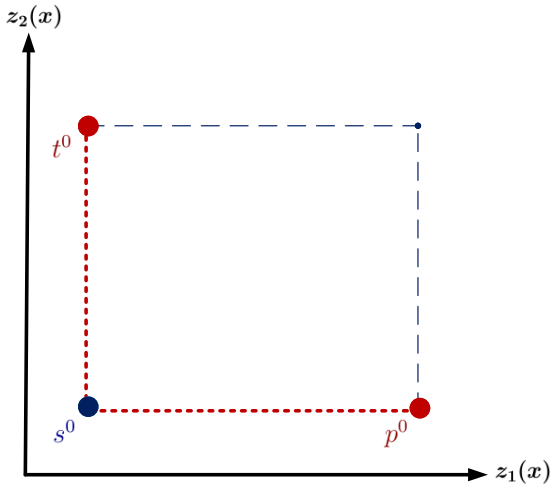


Figure 4.1: Initial box

To find all nondominated points, this box is need to be searched so it is added to set $\mathcal{B}$ to be searched in the first iteration. Throughout the algorithm, a box is defined by three points in the objective space, namely starting point $s$, the nondominated point $t$ which defines the first component of the starting point and

the nondominated point $p$ which defines the second component of the starting point, and denoted as follows

$$b = b(s, p, t) := \{\, y \in \mathcal{R}^2 \mid s_1 \leq y_1 \leq p_1,\ s_2 \leq y_2 \leq t_2 \,\}.$$

After the initialization step which defines the initial box, the algorithm continues with the main loop. At an arbitrary iteration, a box $b(s^b, p^b, t^b)$ from set $\mathcal{B}$ is selected. Then, the following optimization problem is solved to search the box,

$$\min\{\, \rho \mid x \in \mathcal{X},\ z(x) \leq s^b + \rho d,\ z_1(x) \leq\ p_1^b - \epsilon,\ z_2(x) \leq\ t_2^b - \epsilon \,\} \qquad (R(b,d))$$

where $d$ is a direction vector set to $d = [1, 1]$ and $\epsilon > 0$ is a sufficiently small number. The last two constraints are added to prevent finding the nondominated points $p^b$ and $t^b$, which are already found in the previous iterations. See Figure 4.2 for the illustration of the constraints. Thus, if this problem is infeasible, then there is no nondominated point other than $p^b$ and $t^b$ in the box. Otherwise, let the optimal solution of the $(R(b,d))$ be $(\rho^b, x^b)$ and the corresponding (weakly) nondominated point be $n^b := z(x^b)$. Note that $\rho^b$ is the step size and defines the point $y^b := s^b + \rho^b d$ which has at least one common component with $n^b$. Since the scalarization only guarantees that $n^b$ is weakly nondominated, the following problem(s) is(are) solved to ensure that a nondominated point is found. If the first components of $y^b$ and $n^b$ are equal $(n_1^b = y_1^b)$ then,

$$\min\{\, z_2(x) \mid x \in \mathcal{X},\ z_1(x) = z_1(x^b) \,\} \qquad (P_1(x^b))$$

is solved and, if second components are equal $(n_2^b = y_2^b)$ then,

$$\min\{\, z_1(x) \mid x \in \mathcal{X},\ z_2(x) = z_2(x^b) \,\} \qquad (P_2(x^b))$$

is solved. Let the solutions of $(P_1(x^b))$ and $(P_2(x^b))$ be $x^1$ and $x^2$, respectively and $n^1 := z(x^1)$ and $n^2 := z(x^2)$ be the corresponding points in the objective space. If only $(P_1(x^b))$ is solved, then $n^1$ is added to $\mathcal{N}$ and let $n^2$ is set to $n^b$. Symmetrically, if only $(P_2(x^b))$ is solved, then $n^2$ is added to $\mathcal{N}$ and let $n^1$ is set to $n^b$. Notice that if both $(P_1(x^b))$ and $(P_2(x^b))$ are solved, it is possible to find two nondominated points $n^1$ and $n^2$ in the same iteration. In this case, both $n^1$

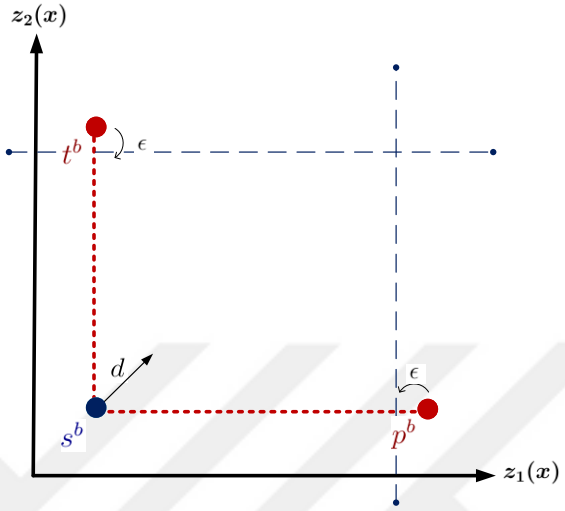


Figure 4.2: $R(b, d)$ for the box $b(s^b, p^b, t^b)$

and $n^2$ are added to $\mathcal{N}$. See Figure 4.3, 4.4, and 4.5 for illustrations of these cases.

After a nondominated point(s) is(are) found, it is known that its(their) dominated and dominating regions can not contain any nondominated points. Hence, these regions are excluded and the box $b(s^b, p^b, t^b)$ is split into two boxes using $n^1$ and $n^2$. Let the starting point for the first new box be $s^1$, given by $s^1 := (n^1, p^b_2)$. So, the first box is formed as $b(s^1, p^b, n^1)$ where $p^b$ is an upper bound for the first objective and $n^1$ is an upper bound for the second objective. Let the starting point for the second new box be $s^2$, given by $s^2 := (t^b_1, n^2_2)$. Then, the second box is formed as $b(s^2, n^2, t^b)$. See Figure 4.6, 4.7, 4.8, 4.9, and 4.10 for the illustrations of the newly formed boxes for different cases.

Finally, we avoid searching boxes which can not have any new nondominated points, by taking the advantage of the integrality of the problem $(P)$, and the structure of a box. That is, the boxes which do not satisfy $p^b_1 - s^b_1 > 1$ and $t^b_2 - s^b_2 > 1$ are eliminated since they can not include any nondominated points other than $p^b$ and $t^b$. After new boxes are defined and their sizes are checked to make sure that they can include nondominated points, they are added to set $\mathcal{B}$ to search in the next iterations. Then, the searched box $b(s^b, p^b, t^b)$ is removed

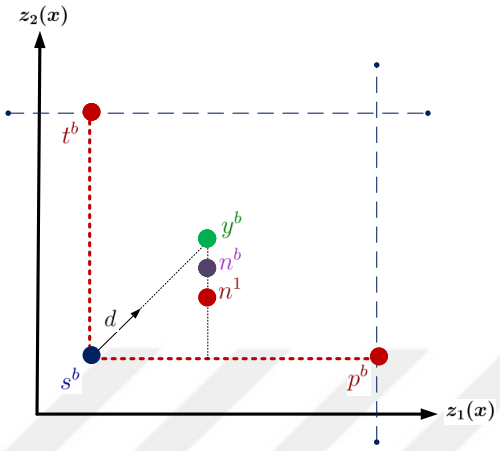


Figure 4.3: Only $P_1(x^b)$ is solved

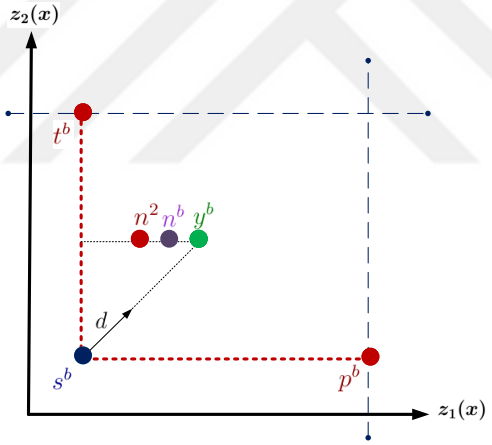


Figure 4.4: Only $P_2(x^b)$ is solved

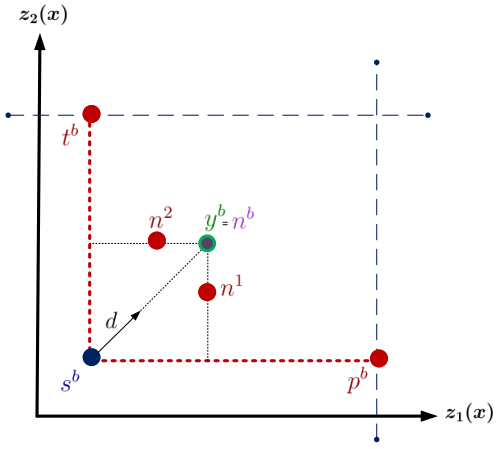


Figure 4.5: Both $P_1(x^b)$ and $P_2(x^b)$ are solved

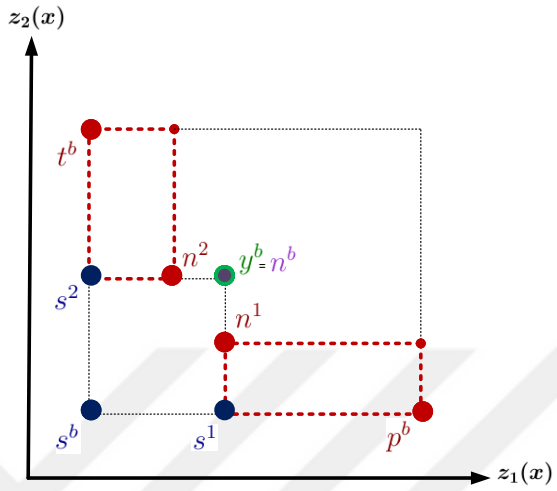


Figure 4.6: $(P_1(x^b))$ and $(P_2(x^b))$ are solved, $n^1$ and $n^2$ are found as nondominated points

from the set $\mathcal{B}$. The algorithm repeats the steps which are introduced above until there is no box in $\mathcal{B}$.

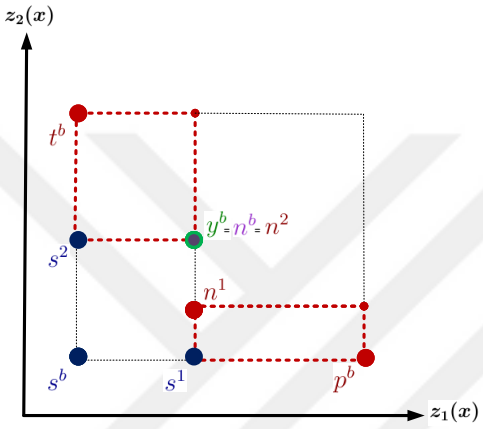


Figure 4.7: $(P_1(x^b))$ and $(P_2(x^b))$ are solved, $n^1$ is found as a nondominated point and $n^2$ is set to $n^b$.

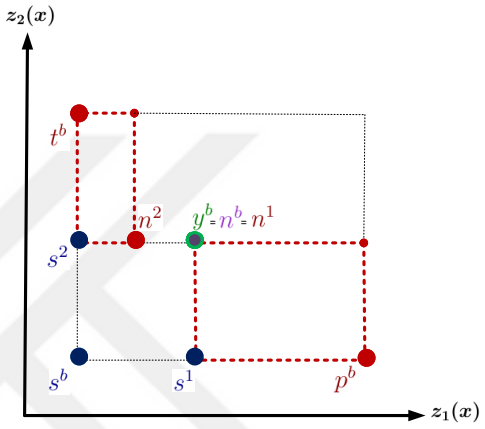


Figure 4.8: $(P_1(x^b))$ and $(P_2(x^b))$ are solved, $n^2$ is found as a nondominated point and $n^1$ is set to $n^b$.

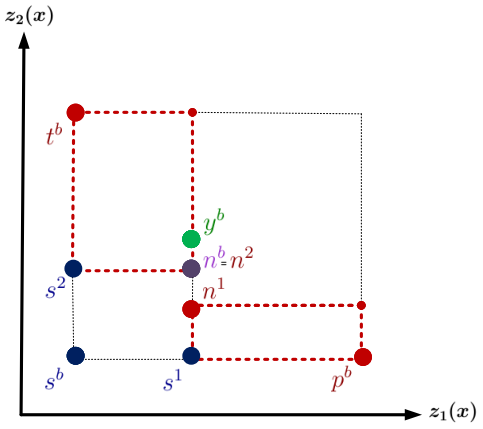


Figure 4.9: $(P_1(x^b))$ is solved, $n^1$ is found as a nondominated point and $n^2$ is set to $n^b$.

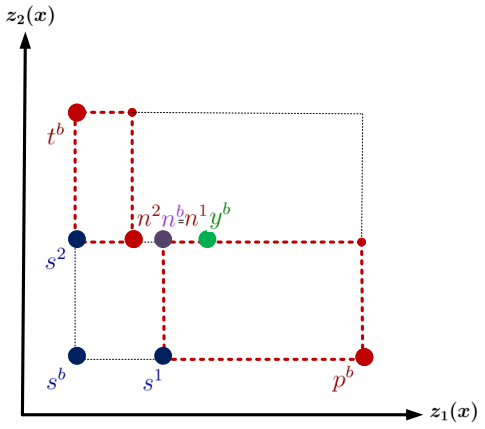


Figure 4.10: $(P_2(x^b))$ is solved, $n^2$ is found as nondominated point and $n^1$ is set to $n^b$.

**Algorithm 1:** The Proposed Algorithm for BOIP

**Input :** Image of feasible set, given by some problem formulation
**Output:** $\mathcal{N}$: set of nondominated solutions

**Initializations**

(I1) $d = [1, 1]$, $\epsilon < 1$ $\alpha = 1$, $k = 0$

(I2) Solve lexmin$\{z_1(x), z_2(x) | x \in \mathcal{X}\}$ ▷ `to find the nondominated point` $t^0$

(I3) Solve lexmin$\{z_2(x), z_1(x) | x \in \mathcal{X}\}$ ▷ `to find the nondominated point` $p^0$

(I4) $\mathcal{N} = \{t^0, p^0\}$, $s^0 = (t_1^0,\ p_2^0)$, $\mathcal{B} = \{b(s^0, p^0, t^0)\}$

**MainLoop**
  **while** $\mathcal{B}$ *is not empty* **do**
    Let $b(s^b, p^b, t^b) \in \mathcal{B}$ and solve $R(b, d)$
    **if** $R(b, d)$ *is feasible* **then**
      $y^b = s^b + \rho^b \cdot d$
      $n^b = z(x^b)$
      **if** $y_1^b = n_1^b$ **then**
        Solve $P_1(x^b)$
        $n^1 = z(x^1)$
      **else**
        $n^1 = n^b$
      **if** $y_2^b = n_2^b$ **then**
        Solve $P_2(x^b)$
        $n^2 = z(x^2)$
      **else**
        $n^2 = n^b$
      $s^1 = (n_1^1,\ p_2^b)$ ▷ `first box` $b(s^1, p^b, n^1)$
      $s^2 = (t_1^b,\ n_2^2)$ ▷ `second box` $b(s^2, n^2, t^b)$
      **if** $p_1^b - s_1^1 > \alpha$ *and* $n_2^1 - s_2^1 > \alpha$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(s^1, p^b, n^1)\}$
      **if** $n_1^2 - s_1^2 > \alpha$ *and* $t_2^b - s_2^2 > \alpha$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(s^2, n^2, t^b)\}$
      **if** $n_2^1 < n_2^b$ **then**
        $\mathcal{N} \leftarrow \mathcal{N} \cup \{n^1\}$
      **if** $n_1^2 < n_1^b$ **then**
        $\mathcal{N} \leftarrow \mathcal{N} \cup \{n^2\}$
      **if** $n_2^1 \geq n_2^b$ *and* $n_1^2 \geq n_1^b$ **then**
        $\mathcal{N} \leftarrow \mathcal{N} \cup \{n^b\}$
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \{b(s^b, p^b, t^b)\}$

**Proposition 4.0.1.** *Algorithm 1 solves* $(3N+C-3C_2-E-1)$ *integer programs, where* $N = |\mathcal{N}|$ *is the number of nondominated points,* $C$ *is the number of cases where* $(y^b = n^b)$*,* $C_2$ *is the number of sub-cases that two nondominated points are found and* $E$ *is the number of eliminated boxes using the elimination rule.*

*Proof.* The following expression, parts $(a) - (g)$ of which will be explained in detail, shows the number of models solved:

$$\underbrace{(4)}_{\textbf{(a)}} + \underbrace{(1)}_{\textbf{(b)}} + \underbrace{(2C_2)}_{\textbf{(c)}} + \underbrace{2(N-2-2C_2)}_{\textbf{(d)}} + \underbrace{(N-2)}_{\textbf{(e)}} + \underbrace{(C-C_2)}_{\textbf{(f)}} - \underbrace{E}_{\textbf{(g)}}$$

At the beginning of Algorithm 1, two lexicographical mininimization problems are solved to find $t^0$ and $p^0$ **(a)** and one $(R(b,d))$ problem is solved to search the initial box **(b)**. $2C_2$ points are found in $C_2$ number of cases ($n = y$ and two solutions are found), each of these points lead to a new box, hence a new $(R(b,d))$ model **(c)**. For the rest of the nondominated points, $(N - 2C_2 - 2)$, each point results in two new boxes (and hence two $(R(b,d))$ models to be solved) **(d)**. As for the $P$ models: $N$-2 points are found by solving a single second stage model (either $(P_1(x^b))$ or $(P_2(x^b))$) **(e)**. Moreover, when $y = n$ and only a single nondominated point is found (in $C - C_2$ number of iterations), we solve an extra $(P_1(x^b))$ or $(P_2(x^b))$, which does not yield a new point **(f)**. Finally, $E$ boxes are eliminated, avoiding the $(R(b,d))$ models that would otherwise have been solved **(g)**. □

# Chapter 5

# Variants of the Benson type Algorithm

In this section, we propose variants of the proposed algorithm (Algorithm 1), which are based on the same principals of the original algorithm however differ with respect to the direction and box definition. Table 5.1 summarizes the variants of the algorithm. The first letter of abbreviations refer to the direction option and the last letter refers to box definition. For example, the original algorithm uses a fixed direction vector and defines the boxes using nondominated points, hence will be referred as FDN (Fixed Direction Nondominated).

Recall that in the original algorithm we set $d = [1, 1]$ and use nondominated points to define boxes. The first set of variants (FDN, CDN, and NDN) keep the box definition the same but use different direction vectors. We then obtain variants (FDY, CDY, and NDY) by changing the box definition and use point $y^b$ instead of $n^b$. First, we introduce the ones that use different direction vectors, then explain the others that use different box definitions.

Table 5.1: The variants of the algorithm

| Variants | | Box Definition | |
|---|---|---|---|
| | | N | Y |
| Direction | Fixed | Algorithm 1 (FDN) | FDY |
| | Changing | CDN | CDY |
| | Nadir | NDN | NDY |

## 5.1 Direction Based Variants

The first variant which use different direction vectors is CDN, it is formed by customizing direction $d$ in each box $b$ as follows: $d^b = u^b - s^b$, where $u^b = (p_1^b, t_2^b)$, (see Figure 5.1). Also, line 4 in Algorithm 1 changes as follows:

$$[\text{Let } b(s^b, p^b, t^b) \in B \text{ and } d^b = u^b - s^b, \text{ solve } R(b, d^b)].$$

The rest of the algorithm is the same as FDN.

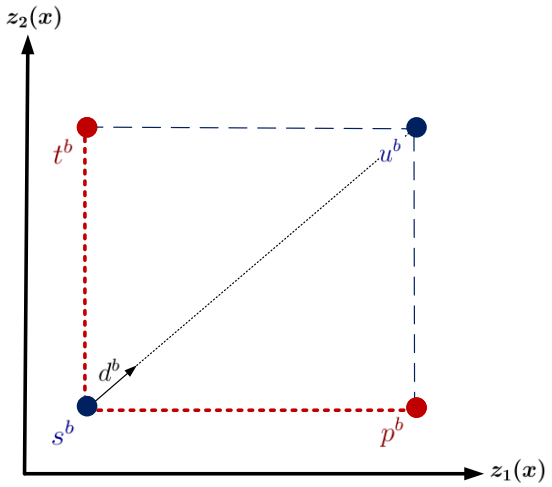


Figure 5.1: $u^b$ and $d^b$ for the box $b(s^b, p^b, t^b)$

In the second variant of the first set, NDN, we set direction $d$ in each box $b$ towards to the nadir point, $u^0$: $d^b = u^0 - s^b$. It only differs in line 4 in Algorithm 1 as follows :

$$[\text{Let } b(s^b, p^b, t^b) \in B \text{ and } d^b = u^0 - s^b, \text{ solve } R(b, d^b)].$$

## 5.2 Box Definition Based Variants

FDY differs from FDN in how the boxes are defined. The idea is to define the newly occured boxes in the smallest possible way. In order to do that, instead of using $n^b$, $y^b = s^b + \rho^b d$ is taken into account to define newly occurred boxes. The procedure is as follows. If only $(P_1(x^b))$ is solved (see lines 8-12), then $n^1$ is found as nondominated point and $n^2$ is set to $y^b$. Symmetrically, if only $(P_2(x^b))$ is solved (see lines 13-17), then $n^2$ is found as nondominated point and $n^1$ is set to $y^b$. Then, the first box is defined as $b(s^1, p^b, n^1)$ and second box is defined as $b(s^2, n^2, t^b)$ as before. To distinguish the variants compare Figures 4.9, 4.10 with Figures 5.2, 5.3.

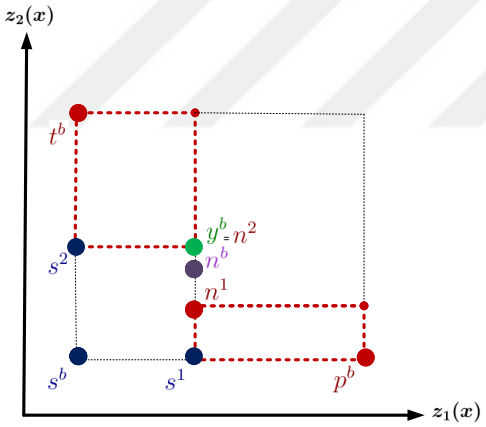


Figure 5.2: $P_1(x^b)$ solved, $n^1$ is found as nondominated point

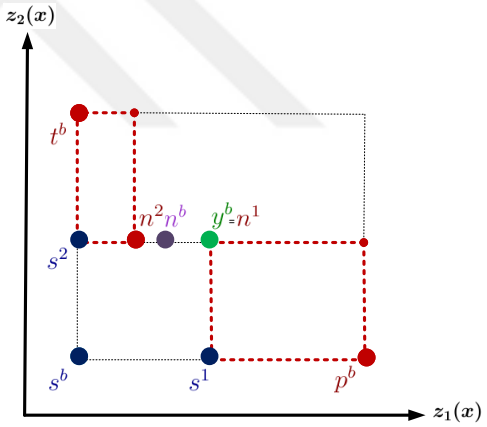


Figure 5.3: $P_2(x^b)$ solved, $n^2$ is found as nondominated point

Note that in this variant the elimination rule has to change since in a box $b(s^b, p^b, t^b)$, $p^b$ and $t^b$ are not necessarily nondominated points. Hence, different from FDN, even the boxes which satisfy $p_1^b - s_1^b = 1$ or $t_2^b - s_2^b = 1$ can contain new nondominated points. We modify the elimination rule accordingly: we eliminate the boxes which do not satisfy $p_1^b - s_1^b \geq 1$ and $t_2^b - s_2^b \geq 1$.

The variant CDY is formed by adopting customized direction definition of CDN (see line 4 of CDN) and the box definition and elimination rule of variant FDY (see lines 12, 17, 20, 22 of Algorithm 2).

**Algorithm 2:** FDY

**Input :** Image of feasible set, given by some problem formulation
**Output:** $\mathcal{N}$: set of nondominated solutions

1 **Initializations**

2 **MainLoop**
3 **while** $\mathcal{B}$ *is not empty* **do**
4 Let $b(s^b, p^b, t^b) \in \mathcal{B}$ and solve $R(b, d)$
5 **if** $R(b, d)$ *is feasible* **then**
6 $y^b = s^b + \rho^b \cdot d$
7 $n^b = z(x^b)$
8 **if** $y_1^b = n_1^b$ **then**
9 Solve $P_1(x^b)$
10 $n^1 = z(x^1)$
11 **else**
12 $n^1 = y^b$
13 **if** $y_2^b = n_2^b$ **then**
14 Solve $P_2(x^b)$
15 $n^2 = z(x^2)$
16 **else**
17 $n^2 = y^b$
18 $s^1 = (n_1^1,\ p_2^b)$ ▷ `first box` $b(s^1, p^b, n^1)$
19 $s^2 = (t_1^b,\ n_2^2)$ ▷ `second box` $b(s^2, n^2, t^b)$
20 **if** $p_1^b - s_1^1 \geq \alpha$ *and* $n_2^1 - s_2^1 \geq \alpha$ **then**
21 $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(s^1, p^b, n^1)\}$
22 **if** $n_1^2 - s_1^2 \geq \alpha$ *and* $t_2^b - s_2^2 \geq \alpha$ **then**
23 $\mathcal{B} \leftarrow \mathcal{B} \cup \{b(s^2, n^2, t^b)\}$
⋮

In NDY, we use customized direction definition of NDN (see line 4 of NDN) and the box definition and elimination rule of variant FDY (see lines 12, 17, 20, 22 of Algorithm 2).

# Chapter 6

# Computational Experiments

We examined the performances of the algorithms by testing them on two sets of problem instances that are used in previous studies in the literature [1]. Both sets contain four classes, (A, B, C, D), each with five instances. The first set consists of knapsack problem (KP) instances with 375 (A), 500 (B), 625 (C), and 750 (D) variables. The second set consists of biobjective assignment problem (AP) intances with $200 \times 200$ (A, B), and $300 \times 300$ (C, D) binary variables.

KP instances are 0-1 knapsack problem instances. In these problems we are given a set of items, each with a weight and a value. The problem is determining which items to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.
Let there be $n$ distinct items. Let $v_i$ and $w_i$ be the value and weight of item $i$, respectively and $W$ be the weight limit. The mathematical model is as follows:

$$\max \sum_{i=1}^{n} v_i x_i$$
$$\text{s.t} \quad \sum_{i=1}^{n} w_i x_i \leq W$$

[1] http:// hdl.handle.net/1959.13/1036183

where $x_i = \begin{cases} 1, & \text{if } i^{th} \text{ item is placed into knapsack} \\ 0, & \text{otherwise.} \end{cases}$

In a setting with a number of tasks that have to be performed by a set of agents, the assignment problem determines which task will be assigned to which agent such that the total assignment cost is minimum.
Let there be $n$ tasks and $n$ agents, and let $c_{ij}$ be the cost of assigning $j^{th}$ task to $i^{th}$ agent. The mathematical model is as follows:

$$\begin{aligned} \min & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} && i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n \\ \text{s.t} & \sum_{i=1}^{n} x_{ij} = 1 && j = 1, 2, \ldots, n \\ & \sum_{j=1}^{n} x_{ij} = 1 && i = 1, 2, \ldots, n \end{aligned}$$

where $x_{ij} = \begin{cases} 1, & \text{if } i^{th} \text{ agent is assigned } j^{th} \text{ task} \\ 0, & \text{otherwise.} \end{cases}$

The algorithms are coded in C++ and all mixed integer programming models are solved using CPLEX 12.6. All of the instances are run on a computer with Intel Xeon CPU E5-1650 3.6 GHz processor and 32 GB RAM. Computation times are given in central processing unit (CPU) seconds.

We first conduct preliminary experiments using class A of KP and AP sets, in which we compare all variants of the algorithms. The results are presented in Tables 6.1 and 6.2. In the tables, we show the average number of nondominated points in the first column. For each algorithm, the number of integer programming problems $\big(\#\ \text{IP}\big)$, $R(b,d)$ problems $\big(\#\ R(b,d)\big)$, and $P_1(x^b)$ or $P_2(x^b)$ problems $\big(\#\ P_i(x^b)\big)$ are reported. Besides, the overall time taken by the algorithms to find the set of all nondominated points $\big(\text{Run Time}\big)$, time spent on solving $R(b,d)$ problems $\big(\text{Time } R(b,d)\big)$, and $P_i(x^b)$ problems $\big(\text{Time } P_i(x^b)\big)$ are reported. Furthermore, we state the average time needed to solve a single $R(b,d)$ problem $\big(\text{Time } /\ \#\ R(b,d)\big)$, and $P_i(x^b)$ problem $\big(\text{Time } /\ \#\ P_i(x^b)\big)$. Also, we

report the number of feasible $\big(\#$ Feasible$\big)$ and infeasible $\big(\#$ Infeasible$\big)$ $R(b,d)$ problems and the number of cases where $(y^b = n^b)$ $\big(C\big)$, and the number of sub-cases that two nondominated points are found $\big(C_2\big)$. Lastly, we report the number of eliminated boxes $\big(E\big)$.

When we compare the algorithms which employ the same direction but differ in the box definition, i.e., $*$DN vs $*$DY, we see that for the majority of the cases, the algorithms which use nondominated points ($*$DN) outperform the algorithms which use $y^b$ ($*$DY) in both the total number of integer problems solved and in the total run time. Note that, an exception occurs for the algorithms NDN and NDY in Table 6.1. Besides, we observe that the number of $R(b,d)$ problems and eliminated boxes are lower for the algorithms using the nondominated point ($*$DN) than the ones that are using $y^b$ ($*$DY), except NDN and NDY in Table 6.1.

Furthermore, when we evaluate the algorithms which use the same box definition but differ in direction, i.e., FD$*$, CD$*$, and ND$*$, we deduce that there is a descending order in the total number of integer programming problems solved from FD$*$ to ND$*$ in both tables. The only exception occurs in AP when the boxes are defined using $y^b$. When we compare the run times, we see none of the variants is consistently the best across all settings, no order could be made.

As a result, for the algorithms which differ in box definition but employ the same direction, we observe that the algorithms using the nondominated point ($*$DN) yield better results than the ones that are using $y^b$ ($*$DY). Based on this observation, we can say that partitioning a box using nondominated points is a better box defining strategy. So, we decide to focus on the algorithms which use nondominated points in their box definition.

When we compare FDN, CDN, and NDN, we see that none outperforms the others, so a ranking can not be obtained. It is also observed that the run times are not directly related with the number of integer programming problems solved in the sense that the algorithm with the minimum number of integer programming problems solved is not necessarily the one with the minimum run time. For

example, in Table 6.2, NDN solves the minimum number of integer programming problems but has the maximum run time. This may be because, although it solves less integer programming problems overall, it solves more $R(b, d)$ problems and on the average $R(b, d)$ problems require more CPU time to be solved compared to the $P_i(x^b)$ problems.

In addition, if we consider the run times, for KP instances, FDN is the best performer and it is closely followed by CDN (7% slower than FDN). Moreover, NDN is 16% slower than FDN. For AP instances, the best performer is NDN and it is followed by FDN (5% slower than NDN). Note that, the performance of CDN is the worst for the AP instances but it is close to the performance of FDN (7% slower than NDN). Overall, we see that the run time performances of FDN and CDN are close to each other. NDN is the worst in KP and the best in AP instances. However, the difference in the KP instances seen to be more significant.

We perform further preliminary experiments to analyze the algorithms FDN, CDN, and NDN in more detail. We run the algorithms with time restriction and assess the quality of the solutions returned, i.e., we measure the representativeness of the generated subset. For this purpose, we use a measure called coverage error [16]. Similar measures are used in the literature to measure representativeness, an example is the coverage gap measure used recently in [17]. Here we provide the definition as in [16], where Chebyshev metric is used. We also use the scaled coverage error measure which is defined similar to the scalar coverage gap measure that is introduced in [17].

Table 6.1: Comparison of the Proposed Algorithms for class A of the set KP

| Class $\lvert\mathcal{N}\rvert$ | Algorithm | # IP | # $R(b,d)$ | # $P_i(x^b)$ | Run Time | Time $R(b,d)$ | Time $P_i(x^b)$ | Time / # $R(b,d)$ | Time / # $P_i(x^b)$ | # Feasible | # Infeasible | $C$ | $C_2$ | $E$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A:975.4 | FDN | 2541.20 | 1338.00 | 1199.20 | 838.06 | 663.60 | 173.68 | 0.50 | 0.14 | 966.00 | 372.00 | 233.20 | 7.40 | 595.00 |
| | FDY | 2762.80 | 1569.80 | 1189.00 | 947.70 | 770.33 | 176.48 | 0.49 | 0.15 | 965.80 | 604.00 | 223.20 | 7.60 | 362.80 |
| | CDN | 2398.60 | 1351.00 | 1043.60 | 894.72 | 787.13 | 106.83 | 0.58 | 0.10 | 971.00 | 380.00 | 72.60 | 2.40 | 592.00 |
| | CDY | 2512.60 | 1520.40 | 988.20 | 1098.23 | 955.42 | 141.70 | 0.62 | 0.14 | 973.00 | 547.40 | 15.20 | 0.40 | 426.60 |
| | NDN | 2325.20 | 1347.60 | 973.60 | 976.52 | 874.40 | 100.65 | 0.65 | 0.10 | 973.40 | 374.20 | 0.20 | 0.00 | 600.20 |
| | NDY | 2154.20 | 1176.80 | 973.40 | 932.05 | 789.01 | 141.74 | 0.67 | 0.14 | 973.40 | 203.40 | 0.00 | 0.00 | 771.00 |

Table 6.2: Comparison of the Proposed Algorithms for class A of the set AP

| Class $\lvert\mathcal{N}\rvert$ | Algorithm | # IP | # $R(b,d)$ | # $P_i(x^b)$ | Run Time | Time $R(b,d)$ | Time $P_i(x^b)$ | Time / # $R(b,d)$ | Time / # $P_i(x^b)$ | # Feasible | # Infeasible | $C$ | $C_2$ | $E$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A:708.4 | FDN | 1636.20 | 699.20 | 933.00 | 2150.36 | 1602.02 | 543.65 | 2.29 | 0.58 | 686.40 | 12.80 | 246.60 | 20.00 | 674.60 |
| | FDY | 1978.00 | 1056.60 | 917.40 | 2764.41 | 2212.31 | 546.09 | 2.09 | 0.60 | 685.60 | 371.00 | 231.80 | 20.80 | 315.60 |
| | CDN | 1553.40 | 712.00 | 837.40 | 2187.07 | 1728.28 | 453.86 | 2.43 | 0.54 | 697.20 | 14.80 | 140.20 | 9.20 | 683.40 |
| | CDY | 2206.40 | 1372.00 | 830.40 | 3924.58 | 3434.98 | 481.96 | 2.52 | 0.58 | 698.00 | 674.00 | 132.40 | 8.40 | 25.00 |
| | NDN | 1431.00 | 720.60 | 706.40 | 2043.09 | 1681.97 | 356.17 | 2.33 | 0.50 | 706.40 | 14.20 | 0.00 | 0.00 | 693.20 |
| | NDY | 1862.20 | 1151.80 | 706.40 | 2931.79 | 2552.17 | 372.86 | 2.22 | 0.53 | 706.40 | 445.40 | 0.00 | 0.00 | 262.40 |

**Definition 1.** Let $\bar{\mathcal{N}} \subseteq \mathcal{N}$ be a representative subset. The *coverage error of* $\bar{\mathcal{N}}$ *with respect to* $n \in \mathcal{N}$ is

$$CE_{\bar{\mathcal{N}}}(n) := \min_{\bar{n} \in \bar{\mathcal{N}}} \left( \max\{|n_1 - \bar{n}_1|, |n_2 - \bar{n}_2|\} \right),$$

the *coverage error* of $\bar{\mathcal{N}}$ is

$$CE(\bar{\mathcal{N}}) = \max_{n \in \mathcal{N}} CE_{\bar{\mathcal{N}}}(n)$$

and the *scaled coverage error* $SCE_{\bar{\mathcal{N}}}$ of $\bar{\mathcal{N}}$ is

$$SCE_{\bar{\mathcal{N}}} = \frac{CE(\bar{\mathcal{N}})}{\max\{u_1^0 - s_1^0, u_2^0 - s_2^0\}},$$

where $u^0$ and $s^0$ are the ideal and the nadir points, respectively.

We examine FDN, CDN, and NDN for class A instances of both problem sets. We run these algorithms for approximately one third of the average run times of FDN, CDN and NDN, i.e. 300 seconds and 700 seconds for set KP and set AP, respectively. As a result, we obtain representative subsets of $\mathcal{N}$, $\bar{\mathcal{N}}$, and report the following measurements for $\bar{\mathcal{N}}$: cardinality $\left(|\bar{\mathcal{N}}|\right)$, average cardinality $\left(\text{Average } |\bar{\mathcal{N}}|\right)$, coverage error $\left(CE\right)$, scaled coverage error $\left(SCE\right)$, and average scaled coverage error $\left(\text{Average } SCE\right)$ in Tables 6.3 and 6.4. In the tables, we observe that CDN is significantly better than FDN and NDN in terms of the number of nondominated points found. Furthermore, CDN performs over twenty five times better than its nearest opponent, which is FDN, in terms of coverage. Figures 6.1, 6.2, 6.3 show $\mathcal{N}$ and $\bar{\mathcal{N}}$ for FDN, CDN, and NDN for KP instances, respectively. It is clearly seen that the solution set returned by CDN includes solutions across the whole Pareto frontier and is more representative than the sets returned by the other two algorithms. Similar to Figures 6.1-6.3, we provide the figures corresponding to the AP set, see Appendix, A.1-A.15.

Based on these results, we conduct main experiments with FDN and CDN by considering the same measurements used in Tables 6.1 and 6.2. The results of our main experiments are given in Tables 6.5 and 6.6 for KP and AP, respectively.

We observe that CDN is the superior algorithm in terms of the total number of problems solved. However, when we look at the problem types that are solved,

Table 6.3: Comparison of the Coverage Errors of FDN, CDN and NDN for instances from class A of KP (in 300 seconds)

| Algorithm | Instance | $\lvert\bar{\mathcal{N}}\rvert$ | Average $\lvert\bar{\mathcal{N}}\rvert$ | $CE$ | $SCE$ | Average $SCE$ |
|---|---|---|---|---|---|---|
| FDN | 1 | 478 | 491 | 408 | 0.1278 | 0.1086 |
| | 2 | 528 | | 324 | 0.0964 | |
| | 3 | 438 | | 451 | 0.1167 | |
| | 4 | 559 | | 369 | 0.1010 | |
| | 5 | 452 | | 333 | 0.1013 | |
| CDN | 1 | 525 | 530 | 12 | 0.0038 | 0.0043 |
| | 2 | 562 | | 14 | 0.0042 | |
| | 3 | 458 | | 25 | 0.0065 | |
| | 4 | 605 | | 9 | 0.0025 | |
| | 5 | 500 | | 15 | 0.0046 | |
| NDN | 1 | 362 | 365.4 | 552 | 0.1729 | 0.1618 |
| | 2 | 418 | | 514 | 0.1529 | |
| | 3 | 311 | | 628 | 0.1625 | |
| | 4 | 432 | | 571 | 0.1563 | |
| | 5 | 304 | | 540 | 0.1642 | |

Table 6.4: Comparison of the Coverage Errors of FDN, CDN and NDN for instances from class A of AP (in 700 seconds)

| Algorithm | Instance | $\lvert\bar{\mathcal{N}}\rvert$ | Average $\lvert\bar{\mathcal{N}}\rvert$ | $CE$ | $SCE$ | Average $SCE$ |
|---|---|---|---|---|---|---|
| FDN | 1 | 230 | 227 | 727 | 0.2826 | 0.2727 |
| | 2 | 227 | | 724 | 0.2683 | |
| | 3 | 231 | | 663 | 0.2665 | |
| | 4 | 227 | | 711 | 0.2609 | |
| | 5 | 220 | | 771 | 0.2851 | |
| CDN | 1 | 262 | 269.8 | 23 | 0.0089 | 0.0086 |
| | 2 | 263 | | 23 | 0.0085 | |
| | 3 | 274 | | 20 | 0.0080 | |
| | 4 | 275 | | 24 | 0.0088 | |
| | 5 | 275 | | 24 | 0.0089 | |
| NDN | 1 | 229 | 230.6 | 757 | 0.2942 | 0.2879 |
| | 2 | 224 | | 784 | 0.2906 | |
| | 3 | 230 | | 705 | 0.2834 | |
| | 4 | 239 | | 757 | 0.2778 | |
| | 5 | 231 | | 794 | 0.2936 | |

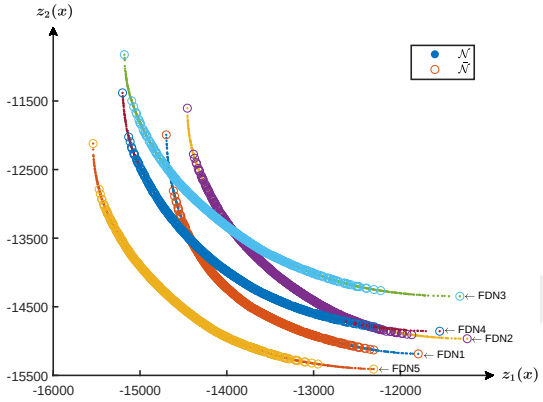


Figure 6.1: FDN for the problems in class A of KP

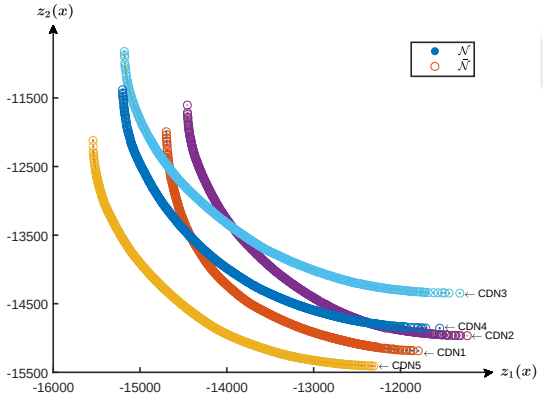


Figure 6.2: CDN for the problems in class A of KP

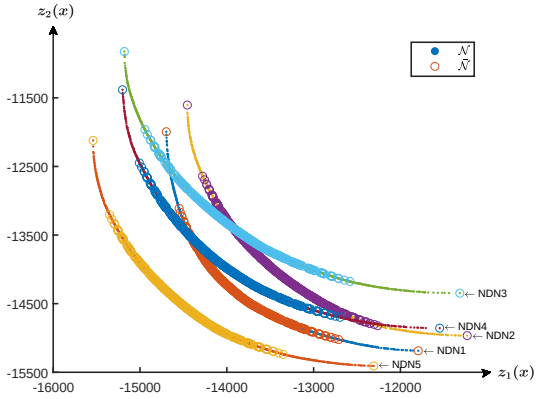


Figure 6.3: NDN for the problems in class A of KP

we see that CDN solves more $R(b,d)$ problems while FDN solves more $P_i(x^b)$ problems. Using a fixed direction significantly increases the number of cases where $(y^b = n^b)$, which leads to solving more integer programming problems.

On the other hand, we see that the original algorithm (FDN) outperforms its opponent in run time for all instances. This is because, FDN not only solves less of the more difficult $R(b,d)$ problems but also solves one $R(b,d)$ problem in notably less time (see class C for set AP, where the times are 5.22 and 12.04). So, it is conceivable that FDN is faster than CDN.

Furthermore, we compare the algorithms with the balanced box algorithm [13] and the two-stage algorithm [14]. Since the algorithms are coded and run on different platforms, we cannot compare the solution times. We, however report the difference in the number of integer problems solved for the balanced box algorithm as percentage in the last columns of the tables, calculated as follows:

$$\frac{(\text{\# of IP problems solved in BB - \# of IP problems solved in our algorithm })}{\text{\# of IP problems solved in our algorithm}} \times 100.$$

It is seen that the balanced box algorithm solves more integer programming problems for all of the problem instances considered. It solves 25.5%, 36.5% more problems than our best algorithm on average for KP and AP, respectively. In a similar way, when we calculate the difference for the two-stage algorithm, we see that the two-stage algorithm solves 4.62% and 13.76% more problems than CDN on average for KP and AP, respectively.

Lastly, we run these algorithms for the complete set of KP and AP instances with predetermined time limits as before and report the quality, coverage error measurements, of the representative subsets in Tables 6.7, 6.8. In the tables, we observe that there is no dominant algorithm in terms of the number of non-dominated points found. However, CDN performs at least ten times better than FDN.

Table 6.5: Comparison of FDN and CDN for the set KP

| Class $\lvert\mathcal{N}\rvert$ | Algorithm | # IP | # $R(b,d)$ | # $P_i(x^b)$ | Run Time | Time $R(b,d)$ | Time $P_i(x^b)$ | Time / # $R(b,d)$ | Time / # $P_i(x^b)$ | # Feasible | # Infeasible | $C$ | $C_2$ | $E$ | BB % Increase # IP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A:975.4 | FDN | 2541.20 | 1338.00 | 1199.20 | 838.06 | 663.60 | 173.68 | 0.50 | 0.14 | 966.00 | 372.00 | 233.20 | 7.40 | 595.00 | 15.15 |
| | CDN | 2398.60 | 1351.00 | 1043.60 | 894.72 | 787.13 | 106.83 | 0.58 | 0.10 | 971.00 | 380.00 | 72.60 | 2.40 | 592.00 | 22.00 |
| B:1539.4 | FDN | 3913.00 | 1984.20 | 1924.80 | 1546.16 | 1248.84 | 296.19 | 0.62 | 0.15 | 1515.00 | 469.20 | 409.80 | 22.40 | 1046.80 | 18.02 |
| | CDN | 3704.00 | 2027.80 | 1672.20 | 2711.98 | 2146.20 | 562.75 | 1.04 | 0.33 | 1532.20 | 495.60 | 140.00 | 5.20 | 1037.60 | 24.68 |
| C:2176.2 | FDN | 5453.60 | 2665.00 | 2784.60 | 2539.96 | 2024.57 | 513.34 | 0.76 | 0.18 | 2127.40 | 537.60 | 657.20 | 46.80 | 1590.80 | 19.71 |
| | CDN | 5152.20 | 2744.00 | 2404.20 | 3459.57 | 3077.83 | 379.94 | 1.12 | 0.16 | 2164.80 | 579.20 | 239.40 | 9.40 | 1586.60 | 26.71 |
| D:2791.8 | FDN | 6934.40 | 3231.40 | 3699.00 | 4605.52 | 3379.02 | 1224.07 | 1.06 | 0.34 | 2703.80 | 527.60 | 995.20 | 86.00 | 2177.20 | 20.78 |
| | CDN | 6503.20 | 3345.80 | 3153.40 | 5404.77 | 4801.35 | 600.78 | 1.43 | 0.19 | 2769.60 | 576.20 | 383.80 | 20.20 | 2194.40 | 28.79 |

Table 6.6: Comparison of FDN and CDN for the set AP

| Class $\lvert\mathcal{N}\rvert$ | Algorithm | # IP | # $R(b,d)$ | # $P_i(x^b)$ | Run Time | Time $R(b,d)$ | Time $P_i(x^b)$ | Time / # $R(b,d)$ | Time / # $P_i(x^b)$ | # Feasible | # Infeasible | $C$ | $C_2$ | $E$ | BB % Increase # IP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A:708.4 | FDN | 1636.20 | 699.20 | 933.00 | 2150.36 | 1602.02 | 543.65 | 2.29 | 0.58 | 686.40 | 12.80 | 246.60 | 20.00 | 674.60 | 29.89 |
| | CDN | 1553.40 | 712.00 | 837.40 | 2187.07 | 1728.28 | 453.86 | 2.43 | 0.54 | 697.20 | 14.80 | 140.20 | 9.20 | 683.40 | 36.81 |
| B:1416.2 | FDN | 3247.20 | 1475.80 | 1767.40 | 5354.08 | 4208.29 | 1136.63 | 2.85 | 0.64 | 1388.20 | 87.60 | 379.20 | 26.00 | 1301.60 | 30.84 |
| | CDN | 3096.20 | 1506.20 | 1586.00 | 5519.86 | 4552.19 | 957.71 | 3.02 | 0.60 | 1408.40 | 97.80 | 177.60 | 5.80 | 1311.60 | 37.22 |
| C:823.6 | FDN | 1895.00 | 803.60 | 1087.40 | 5644.20 | 4195.32 | 1437.21 | 5.22 | 1.32 | 798.60 | 5.00 | 288.80 | 23.00 | 794.60 | 30.39 |
| | CDN | 1839.40 | 815.80 | 1019.60 | 11212.29 | 9838.48 | 1360.20 | 12.04 | 1.33 | 809.00 | 6.80 | 210.60 | 12.60 | 803.20 | 34.33 |
| D:1827 | FDN | 4140.20 | 1808.20 | 2328.00 | 16403.48 | 12567.35 | 3811.83 | 6.95 | 1.64 | 1766.60 | 41.60 | 561.40 | 58.40 | 1726.00 | 32.38 |
| | CDN | 3980.40 | 1860.40 | 2116.00 | 17451.84 | 14172.12 | 3253.05 | 7.61 | 1.54 | 1812.00 | 48.40 | 304.00 | 13.00 | 1764.60 | 37.70 |

Table 6.7: Comparison of the Coverage Errors of FDN and CDN for the classes of KP

| Algorithm | Instance | Time | $\lvert\bar{\mathcal{N}}\rvert$ | Average $\lvert\bar{\mathcal{N}}\rvert$ | $CE$ | $SCE$ | Average $SCE$ | Algorithm | Instance | Time | $\lvert\bar{\mathcal{N}}\rvert$ | Average $\lvert\bar{\mathcal{N}}\rvert$ | $CE$ | $SCE$ | Average $SCE$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FDN | 1 | 300 | 478 | 491 | 408 | 0.1278 | 0.1086 | CDN | 1 | 300 | 525 | 530 | 12 | 0.0038 | 0.0043 |
| | 2 | | 528 | | 324 | 0.0964 | | | 2 | | 562 | | 14 | 0.0042 | |
| | 3 | | 438 | | 451 | 0.1167 | | | 3 | | 458 | | 25 | 0.0065 | |
| | 4 | | 559 | | 369 | 0.1010 | | | 4 | | 605 | | 9 | 0.0025 | |
| | 5 | | 452 | | 333 | 0.1013 | | | 5 | | 500 | | 15 | 0.0046 | |
| | 6 | 700 | 863 | 810.8 | 388 | 0.0904 | 0.0914 | | 6 | 700 | 847 | 789.6 | 12 | 0.0028 | 0.0036 |
| | 7 | | 748 | | 528 | 0.0986 | | | 7 | | 686 | | 20 | 0.0037 | |
| | 8 | | 728 | | 413 | 0.0860 | | | 8 | | 763 | | 15 | 0.0031 | |
| | 9 | | 821 | | 444 | 0.0986 | | | 9 | | 837 | | 18 | 0.0040 | |
| | 10 | | 894 | | 384 | 0.0836 | | | 10 | | 815 | | 19 | 0.0041 | |
| | 11 | 1000 | 1024 | 998 | 560 | 0.0938 | 0.0981 | | 11 | 1000 | 809 | 814.8 | 29 | 0.0049 | 0.0035 |
| | 12 | | 930 | | 597 | 0.1011 | | | 12 | | 712 | | 19 | 0.0032 | |
| | 13 | | 905 | | 714 | 0.1080 | | | 13 | | 754 | | 21 | 0.0032 | |
| | 14 | | 1156 | | 637 | 0.1033 | | | 14 | | 955 | | 22 | 0.0036 | |
| | 15 | | 975 | | 464 | 0.0844 | | | 15 | | 844 | | 16 | 0.0029 | |
| | 16 | 1670 | 1508 | 1345.8 | 665 | 0.0920 | 0.0987 | | 16 | 1670 | 1226 | 1074.4 | 16 | 0.0022 | 0.0026 |
| | 17 | | 1266 | | 650 | 0.0942 | | | 17 | | 963 | | 31 | 0.0045 | |
| | 18 | | 1370 | | 756 | 0.0986 | | | 18 | | 1066 | | 12 | 0.0016 | |
| | 19 | | 1240 | | 734 | 0.1044 | | | 19 | | 899 | | 22 | 0.0031 | |
| | 20 | | 1345 | | 721 | 0.1042 | | | 20 | | 1218 | | 12 | 0.0017 | |

Table 6.8: Comparison of the Coverage Errors of FDN and CDN for the classes of AP

| Algorithm | Instance | Time | $\lvert\bar{\mathcal{N}}\rvert$ | Average $\lvert\bar{\mathcal{N}}\rvert$ | $CE$ | $SCE$ | Average $SCE$ | Algorithm | Instance | Time | $\lvert\bar{\mathcal{N}}\rvert$ | Average $\lvert\bar{\mathcal{N}}\rvert$ | $CE$ | $SCE$ | Average $SCE$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FDN | 1 | 700 | 230 | 227 | 727 | 0,2826 | 0.2727 | CDN | 1 | 700 | 262 | 269.8 | 23 | 0.0089 | 0.0086 |
| | 2 | | 227 | | 724 | 0.2683 | | | 2 | | 263 | | 23 | 0.0085 | |
| | 3 | | 231 | | 663 | 0.2665 | | | 3 | | 274 | | 20 | 0.0080 | |
| | 4 | | 227 | | 711 | 0.2609 | | | 4 | | 275 | | 24 | 0.0088 | |
| | 5 | | 220 | | 771 | 0.2851 | | | 5 | | 275 | | 24 | 0.0089 | |
| | 6 | 1820 | 481 | 479.6 | 2566 | 0.3154 | 0.3131 | | 6 | 1820 | 634 | 621.4 | 37 | 0.0045 | 0.0046 |
| | 7 | | 478 | | 2420 | 0.3150 | | | 7 | | 631 | | 35 | 0.0046 | |
| | 8 | | 479 | | 2543 | 0.3198 | | | 8 | | 624 | | 38 | 0.0048 | |
| | 9 | | 476 | | 2361 | 0.3044 | | | 9 | | 604 | | 41 | 0.0053 | |
| | 10 | | 484 | | 2507 | 0.3108 | | | 10 | | 614 | | 32 | 0.0040 | |
| | 11 | 2810 | 360 | 380.2 | 548 | 0.2207 | 0.2254 | | 11 | 2810 | 361 | 246.2 | 29 | 0.0117 | 0.0157 |
| | 12 | | 365 | | 586 | 0.2224 | | | 12 | | 124 | | 66 | 0.0250 | |
| | 13 | | 388 | | 578 | 0.2284 | | | 13 | | 251 | | 37 | 0.0146 | |
| | 14 | | 382 | | 613 | 0.2348 | | | 14 | | 251 | | 35 | 0.0134 | |
| | 15 | | 406 | | 530 | 0.2206 | | | 15 | | 244 | | 33 | 0.0137 | |
| | 16 | 5650 | 611 | 617.2 | 3209 | 0.3235 | 0.3161 | | 16 | 5650 | 755 | 764 | 58 | 0.0058 | 0.0056 |
| | 17 | | 603 | | 3027 | 0.3102 | | | 17 | | 777 | | 57 | 0.0058 | |
| | 18 | | 643 | | 3038 | 0.3149 | | | 18 | | 766 | | 49 | 0.0051 | |
| | 19 | | 624 | | 3058 | 0.3157 | | | 19 | | 752 | | 54 | 0.0056 | |
| | 20 | | 605 | | 3053 | 0.3161 | | | 20 | | 770 | | 55 | 0.0057 | |

Overall, one can conclude that both variants are powerful in different aspects. When used to find the complete set of nondominated points, FDN works better since it runs faster. However, CDN is very promising variant when run with a time limit since it quickly provides a highly representative subset of solutions.

## 6.1 Further modifications of CDN

When we examine the results of average time spend for a $R(b,d)$ model, we observe that there is significant difference between FDN and CDN for class C of AP set. To understand the impact of the direction parameter on the time required to solve an $R(b,d)$ problem, we modified direction $d$ of $R(b,d)$ solved through CDN and obtained a modified CDN (Mod-CDN) as follows:

---

**if** $d_1 > d_2$ **then**
  $a$=round$\left(\frac{d_1}{d_2}\right)$
  $d = (a, 1)$
**else if** $d_1 < d_2$ **then**
  $a$=round$\left(\frac{d_2}{d_1}\right)$
  $d = (1, a)$

---

We examine the performance of Mod-CDN, for class C of AP, and compare the results of FDN, CDN and Mod-CDN in Table 6.9. We observe that CDN is the superior one in the number of integer programming problems solved while Mod-CDN is the worst one. However, in the number of $R(b,d)$ problems, FDN and Mod-CDN are quite close to each other and both are better than CDN. Also, in terms of run times, FDN is the predominant one and it is followed by Mod-CDN, except instance 14. When we analyze run times of Mod-CDN and CDN in detail, we see that Mod-CDN differs -1.43%, -36.46%, -5.62%, 20.55%, and -6.13% from CDN for each instances, respectively. $\left(\text{The difference is calculated as } \frac{\text{Run Time of Mod-CDN - Run Time of CDN}}{\text{Run Time of CDN}} \times 100.\right)$

When we analyze the times of each $R(b,d)$ model solved in CDN for the instances in class C of AP set, we see that the majority of the total time is occupied by only a few models. To overcome the extreme solution times of the models, we employ a different modification CDN. For this variant, we put a time limit to $R(b,d)$ models, and if the model is aborted due to time limit we modify the direction and solve the model with the new direction parameter. That is, we modify CDN starting with line 5 as follows:

---

Let $b(s^b, p^b, t^b) \in B$ and $d^b = u^b - s^b$, solve $R(b, d^b)$
**if** $R(b, d^b)$ *could not be solved within the time limit* **then**
    $d_2^b = d_2^b - 1$
    Solve $R(b, d^b)$

---

We examine CDN with time limited $R(b,d)$ models, setting time limit as 50 seconds, TL-CDN, in class C of AP, and compare it with FDN and CDN. Results are presented in Table 6.10. When we compare the number of integer problems solved by the algorithms, we observe that CDN is the predominant algorithm and it is closely followed by TL-CDN, as expected. When we analyze the table considering run times and $R(b,d)$ times of TL-CDN and CDN, we observe that there is a significant improvement, indicating that the modification is successful. However, FDN is still the superior algorithm in run times and $R(b,d)$ times.

To analyze TL-CDN further for representativeness, we run it with predetermined time limits and report coverage error and scaled coverage error for class C of AP. Table 6.11 shows the results for FDN, CDN, and TL-CDN. We deduce that this modification is successful in reducing the run time without sacrificing from performance in representativeness. TL-CDN performs even better than CDN as it returns more nondominated points.

Table 6.9: Comparison of the FDN, CDN and Mod-CDN for class C of the set AP

| Instance $\lvert\mathcal{N}\rvert$ | Algorithm | # IP | # $R(b,d)$ | # $P_i(x^b)$ | Run Time | Time $R(b,d)$ | Time $P_i(x^b)$ | Time / # $R(b,d)$ | Time / # $P_i(x^b)$ | # Feasible | # Infeasible | $C$ | $C_2$ | $E$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 11:813 | FDN | 1886 | 797 | 1085 | 5789.17 | 4290.68 | 1486.84 | 5.38 | 1.37 | 793 | 4 | 292 | 18 | 790 |
| | Mod-CDN | 1974 | 796 | 1174 | 7069.70 | 5434.01 | 1623.49 | 6.83 | 1.38 | 791 | 5 | 383 | 20 | 787 |
| | CDN | 1790 | 803 | 983 | 7171.94 | 5876.25 | 1283.30 | 7.32 | 1.31 | 797 | 6 | 186 | 14 | 792 |
| 12:827 | FDN | 1872 | 808 | 1060 | 5704.50 | 4282.00 | 1410.70 | 5.30 | 1.33 | 799 | 9 | 261 | 26 | 791 |
| | Mod-CDN | 1993 | 807 | 1182 | 7896.48 | 6275.74 | 1608.15 | 7.78 | 1.36 | 795 | 12 | 387 | 30 | 784 |
| | CDN | 1848 | 827 | 1017 | 12428.33 | 11047.07 | 1367.10 | 13.36 | 1.34 | 815 | 12 | 202 | 10 | 804 |
| 13:823 | FDN | 1915 | 805 | 1106 | 5655.16 | 4195.99 | 1447.53 | 5.21 | 1.31 | 801 | 4 | 305 | 20 | 798 |
| | Mod-CDN | 1999 | 808 | 1187 | 7723.02 | 6105.94 | 1604.46 | 7.56 | 1.35 | 805 | 3 | 382 | 16 | 803 |
| | CDN | 1860 | 817 | 1039 | 8182.69 | 6871.98 | 1297.93 | 8.41 | 1.25 | 814 | 3 | 225 | 7 | 812 |
| 14:841 | FDN | 1923 | 817 | 1102 | 5715.18 | 4225.55 | 1477.79 | 5.17 | 1.34 | 812 | 5 | 290 | 27 | 808 |
| | Mod-CDN | 2041 | 822 | 1215 | 17552.17 | 15766.31 | 1772.37 | 19.18 | 1.46 | 816 | 6 | 399 | 23 | 811 |
| | CDN | 1883 | 832 | 1047 | 14560.13 | 13066.53 | 1478.46 | 15.70 | 1.41 | 823 | 9 | 224 | 16 | 815 |
| 15:814 | FDN | 1879 | 791 | 1084 | 5356.99 | 3982.35 | 1363.21 | 5.03 | 1.26 | 788 | 3 | 296 | 24 | 786 |
| | Mod-CDN | 1980 | 797 | 1179 | 12877.11 | 11276.80 | 1587.27 | 14.15 | 1.35 | 794 | 3 | 385 | 18 | 792 |
| | CDN | 1816 | 800 | 1012 | 13718.39 | 12330.59 | 1374.22 | 15.41 | 1.36 | 796 | 4 | 216 | 16 | 793 |

Table 6.10: Comparison of the FDN, CDN and TL-CDN for class C of the set AP

| Instance $\lvert\mathcal{N}\rvert$ | Algorithm | # IP | # $R(b,d)$ | # $P_i(x^b)$ | Run Time | Time $R(b,d)$ | Time $P_i(x^b)$ | Time / # $R(b,d)$ | Time / # $P_i(x^b)$ | # Feasible | # Infeasible | $C$ | $C_2$ | $E$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 11:813 | FDN | 1886 | 797 | 1085 | 5789.17 | 4290.68 | 1486.84 | 5.38 | 1.37 | 793 | 4 | 292 | 18 | 790 |
| | TL-CDN | 1794 | 807 | 983 | 6888.21 | 5613.14 | 1262.15 | 6.96 | 1.28 | 797 | 6 | 186 | 14 | 792 |
| | CDN | 1790 | 803 | 983 | 7171.94 | 5876.25 | 1283.30 | 7.32 | 1.31 | 797 | 6 | 186 | 14 | 792 |
| 12:827 | FDN | 1872 | 808 | 1060 | 5704.50 | 4282.00 | 1410.70 | 5.30 | 1.33 | 799 | 9 | 261 | 26 | 791 |
| | TL-CDN | 1867 | 844 | 1019 | 7406.11 | 6107.61 | 1284.89 | 7.24 | 1.26 | 816 | 12 | 203 | 9 | 805 |
| | CDN | 1848 | 827 | 1017 | 12428.33 | 11047.07 | 1367.10 | 13.36 | 1.34 | 815 | 12 | 202 | 10 | 804 |
| 13:823 | FDN | 1915 | 805 | 1106 | 5655.16 | 4195.99 | 1447.53 | 5.21 | 1.31 | 801 | 4 | 305 | 20 | 798 |
| | TL-CDN | 1869 | 825 | 1040 | 6699.22 | 5420.68 | 1265.74 | 6.57 | 1.22 | 814 | 3 | 226 | 7 | 812 |
| | CDN | 1860 | 817 | 1039 | 8182.69 | 6871.98 | 1297.93 | 8.41 | 1.25 | 814 | 3 | 225 | 7 | 812 |
| 14:841 | FDN | 1923 | 817 | 1102 | 5715.18 | 4225.55 | 1477.79 | 5.17 | 1.34 | 812 | 5 | 290 | 27 | 808 |
| | TL-CDN | 1892 | 840 | 1048 | 6875.17 | 5530.81 | 1331.28 | 6.58 | 1.27 | 823 | 9 | 225 | 16 | 815 |
| | CDN | 1883 | 832 | 1047 | 14560.13 | 13066.53 | 1478.46 | 15.70 | 1.41 | 823 | 9 | 224 | 16 | 815 |
| 15:814 | FDN | 1879 | 791 | 1084 | 5356.99 | 3982.35 | 1363.21 | 5.03 | 1.26 | 788 | 3 | 296 | 24 | 786 |
| | TL-CDN | 1832 | 814 | 1014 | 7023.40 | 5767.34 | 1243.13 | 7.09 | 1.23 | 796 | 4 | 218 | 16 | 793 |
| | CDN | 1816 | 800 | 1012 | 13718.39 | 12330.59 | 1374.22 | 15.41 | 1.36 | 796 | 4 | 216 | 16 | 793 |

Table 6.11: Comparison of the Coverage Errors of FDN, CDN and TL-CDN for instances from class C of AP (in 2810 seconds)

| Algorithm | Instance | $\lvert\bar{\mathcal{N}}\rvert$ | Average $\lvert\bar{\mathcal{N}}\rvert$ | $CE$ | $SCE$ | Average $SCE$ |
|---|---|---|---|---|---|---|
| FDN | 1 | 360 | 380.2 | 548 | 0.2207 | 0.2254 |
| | 2 | 365 | | 586 | 0.2224 | |
| | 3 | 388 | | 578 | 0.2284 | |
| | 4 | 382 | | 613 | 0.2348 | |
| | 5 | 406 | | 530 | 0.2206 | |
| CDN | 1 | 361 | 246.2 | 29 | 0.0117 | 0.0157 |
| | 2 | 124 | | 66 | 0.0250 | |
| | 3 | 251 | | 37 | 0.0146 | |
| | 4 | 251 | | 35 | 0.0134 | |
| | 5 | 244 | | 33 | 0.0137 | |
| TL-CDN | 1 | 384 | 367.8 | 29 | 0.0117 | 0.0134 |
| | 2 | 348 | | 43 | 0.0163 | |
| | 3 | 372 | | 26 | 0.0103 | |
| | 4 | 392 | | 26 | 0.0100 | |
| | 5 | 343 | | 45 | 0.0187 | |

# Chapter 7

# Conclusion and Future Research

We propose an objective space search algorithm based on solving Pascoletti-Serafini scalarizations to return the whole nondominated set of bi-objective integer programming problems. We generate different variants by changing the box definition and direction vector in the scalarization problem.

We compare the performances of the algorithm variants via experiments, in which the algorithms are run with and without time limits and determine the variants that outperform the others. We conclude that the variants using nondominated points to define the boxes are better. Moreover, although the variant using a fixed direction leads to more integer programming problems solved, it requires less computational time since it solves less of the more difficult scalarization model. We, however observe that the variant utilizing a customized direction with respect to the box to be searched is powerful in terms of returning a highly representative subset (measured using coverage error) of the set of nondominated points when it is run with a time limit. We suggest an extension to this variant, which has lower solution times and better coverage error results.

We also compare the variants that are powerful in different aspects with balanced box algorithm, and show through computational experiments that these

variants solve significantly less problems than the balanced box method. Furthermore, when we compare these algorithms with two-stage algorithm, we see that our algorithms perform better in terms of the number of problems solved.

The proposed algorithms can be tested on different problem types. Moreover, further research can be performed in designing multi-objective extensions of the algorithm variants. Moreover, the algorithms could be modified for interactive settings and their performances could be analyzed.

# Bibliography


[1] M. Ehrgott, *Multicriteria Optimization (2. ed.)*. Springer, 2005.

[2] L. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE Transactions on Automatic Control*, vol. 8, pp. 59–60, January 1963.

[3] J. Koski, *Multicriteria Truss Optimization*, pp. 263–307. Boston, MA: Springer US, 1988.

[4] Y. Haimes, L. Lasdon, and D. Wismer, "On a bicriterion formulation of the problems of integrated system identification and system optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-1, no. 3, pp. 296–297, 1971.

[5] V. J. Bowman, "On the relationship of the tchebycheff norm and the efficient frontier of multiple-criteria objectives," in *Multiple Criteria Decision Making* (H. Thiriez and S. Zionts, eds.), (Berlin, Heidelberg), pp. 76–86, Springer Berlin Heidelberg, 1976.

[6] R. E. Steuer and E. Choo, "An interactive weighted tchebycheff procedure for multiple objective programming," *Mathematical Programming*, vol. 26, pp. 326–344, 10 1983.

[7] L. Chalmet, L. Lemonidis, and D. Elzinga, "An algorithm for the bi-criterion integer programming problem," *European Journal of Operational Research*, vol. 25, no. 2, pp. 292 – 300, 1986.

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

[9] T. Ralphs, M. Saltzman, and M. Wiecek, "An improved algorithm for solving biobjective integer programs," *Annals OR*, vol. 147, pp. 43–70, 2006.

[10] S. Sayın and P. Kouvelis, "The multiobjective discrete optimization problem: A weighted min-max two-stage optimization approach and a bicriteria algorithm.," *Management Science*, vol. 51, no. 10, pp. 1572 – 1581, 2005.

[11] E. Ulungu and J. Teghem, "The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems," *Foundations of Computing and Decision Sciences*, vol. 20, no. 2, pp. 149–165, 1995.

[12] A. Przybylski, X. Gandibleux, and M. Ehrgott, "Two phase algorithms for the bi-objective assignment problem," *European Journal of Operational Research*, vol. 185, no. 2, pp. 509 – 533, 2008.

[13] 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, no. 4, pp. 735–754, 2015.

[14] R. Dai and H. Charkhgard, "A two-stage approach for bi-objective integer linear programming," *Operations Research Letters*, vol. 46, no. 1, pp. 81 – 87, 2018.

[15] A. Pascoletti and P. Serafini, "Scalarizing vector optimization problems," *Journal of Optimization Theory and Applications*, vol. 42, no. 4, pp. 499–524, 1984.

[16] S. Sayın, "Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming," *Mathematical Programming*, vol. 87, pp. 543–560, 2000.

[17] G. Ceyhan, M. Kksalan, and B. Lokman, "Finding a representative nondominated set for multi-objective mixed integer programs," *European Journal of Operational Research*, vol. 272, no. 1, pp. 61 – 77, 2019.

[18] V. Chankong and Y. Haimes, "Optimization-based methods for multiobjective decision-making: An overview," *Large Scale Systems*, vol. 5, 08 1983.

[19] R. E. Steuer, *Multiple criteria optimization: theory, computation, and application*, vol. 233. Wiley New York, 1986.

[20] M. Ehrgott, *Multicriteria optimization*. Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, 2000.

[21] F. Akbari, M. Ghaznavi, and E. Khorram, "A revised pascoletti–serafini scalarization method for multiobjective optimization problems," *Journal of Optimization Theory and Applications*, vol. 178, no. 2, pp. 560–590, 2018.

[22] J. Riera-Ledesma and J. J. Salazar-González, "The biobjective travelling purchaser problem," *European Journal of Operational Research*, vol. 160, no. 3, pp. 599 – 613, 2005. Decision Analysis and Artificial Intelligence.

# Appendix A

# Detailed Results of the Test Instances

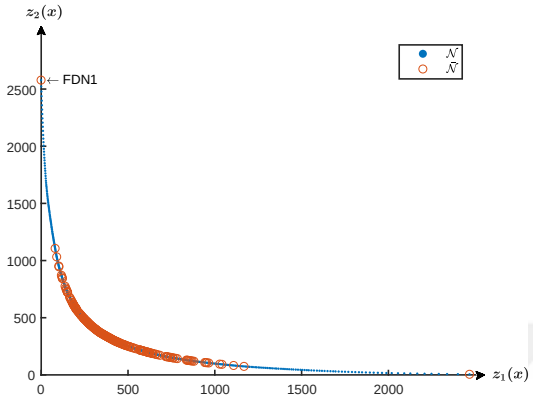


Figure A.1: FDN for the $1^{st}$ problem in class A of AP

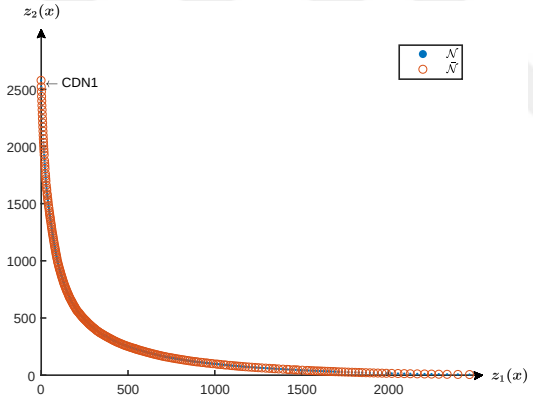


Figure A.2: CDN for the $1^{st}$ problem in class A of AP

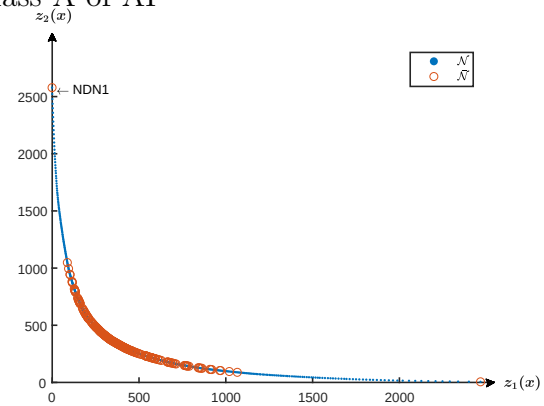


Figure A.3: NDN for the $1^{st}$ problem in class A of AP

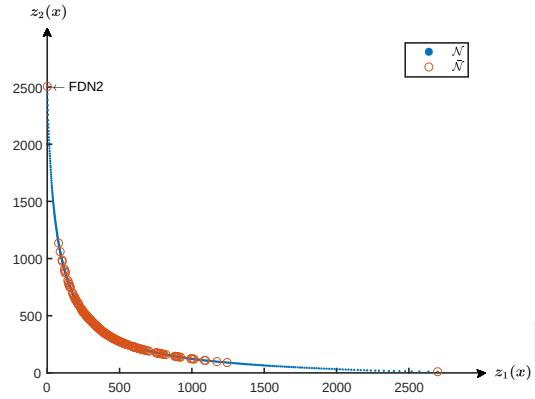


Figure A.4: FDN for the $2^{nd}$ problem in class A of AP

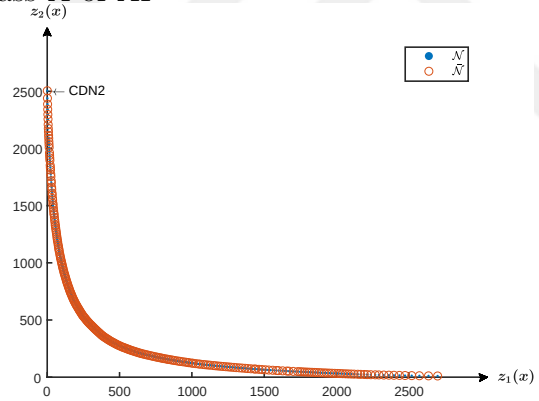


Figure A.5: CDN for the $2^{nd}$ problem in class A of AP

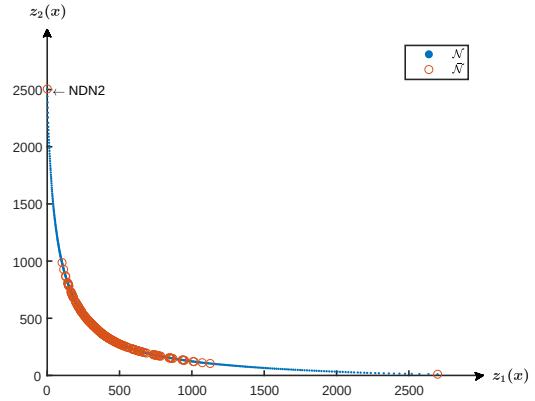


Figure A.6: NDN for the $2^{nd}$ problem in class A of AP

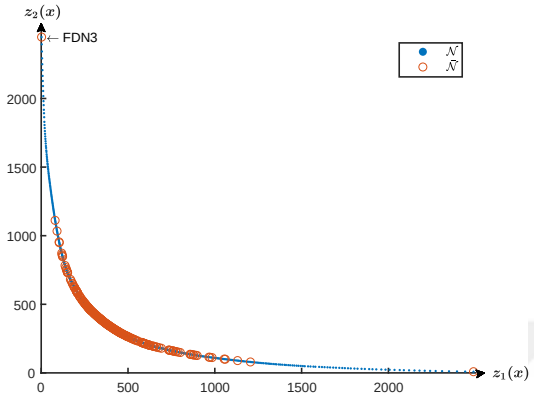


Figure A.7: FDN for the $3^{rd}$ problem in class A of AP

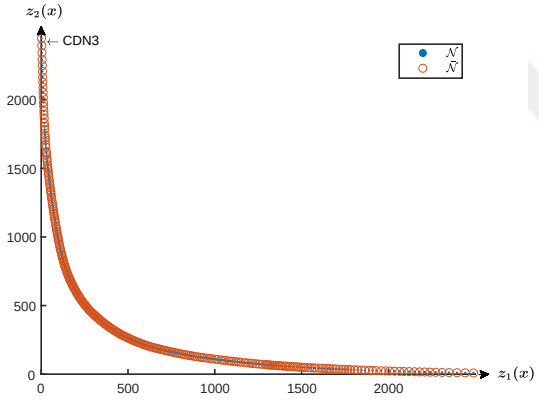


Figure A.8: CDN for the $3^{rd}$ problem in class A of AP

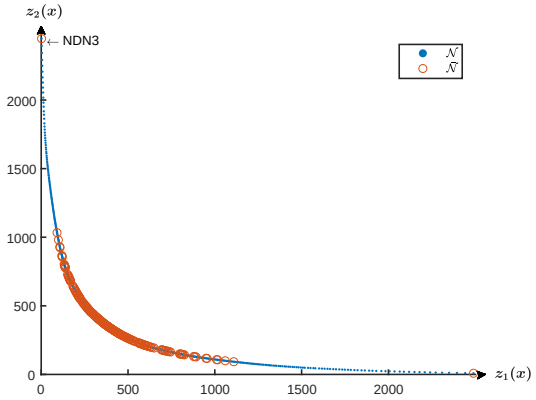


Figure A.9: NDN for the $3^{rd}$ problem in class A of AP

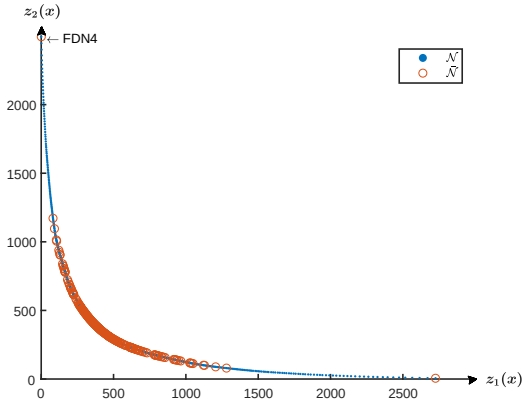


Figure A.10: FDN for the $4^{th}$ problem in class A of AP

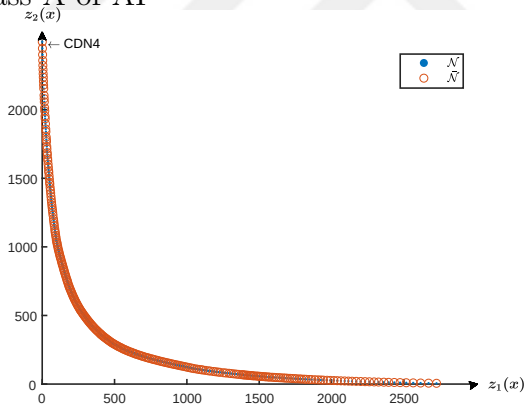


Figure A.11: CDN for the $4^{th}$ problem in class A of AP

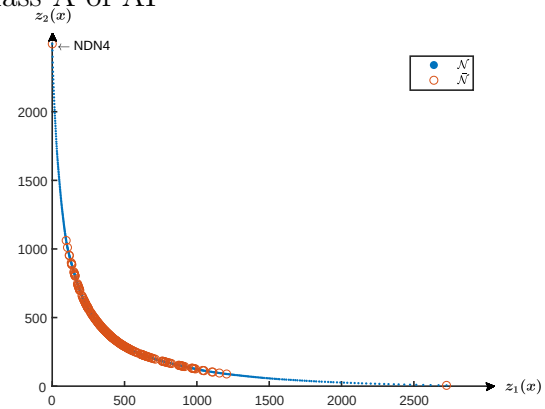


Figure A.12: NDN for the $4^{th}$ problem in class A of AP

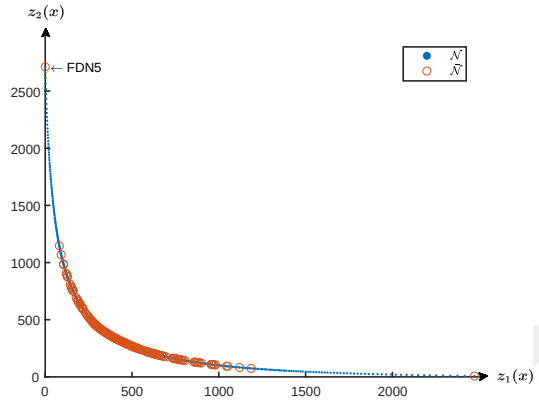


Figure A.13: FDN for the $5^{th}$ problem in class A of AP

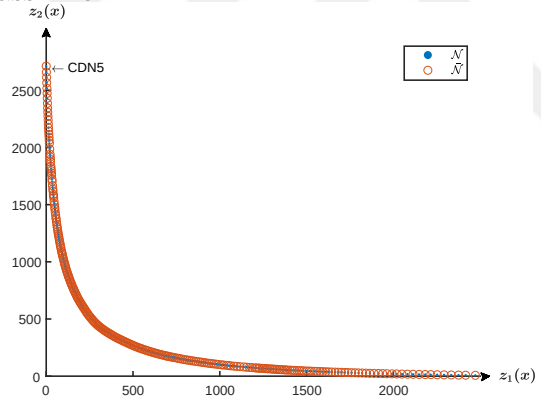


Figure A.14: CDN for the $5^{th}$ problem in class A of AP

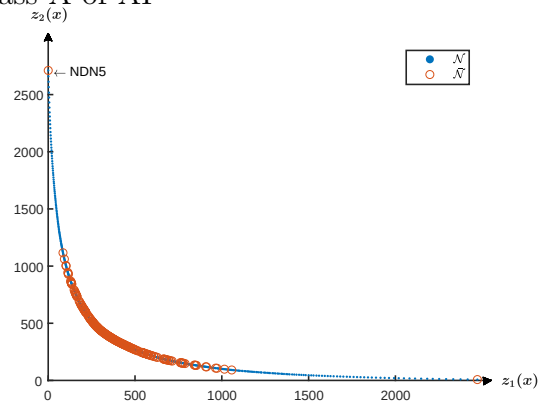


Figure A.15: NDN for the $5^{th}$ problem in class A of AP