FINDING ALL EQUITABLY NON-DOMINATED POINTS OF MULTIOBJECTIVE 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

Seyit Ulutaş

September 2023

FINDING ALL EQUITABLY NON-DOMINATED POINTS OF MULTIOBJECTIVE INTEGER PROGRAMMING PROBLEMS

By Seyit Ulutaş

September 2023

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.

Özlem Karsu(Advisor)

Gülşah Karakaya

Firdevs Ulus

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

FINDING ALL EQUITABLY NON-DOMINATED POINTS OF MULTIOBJECTIVE INTEGER PROGRAMMING PROBLEMS

Seyit Ulutaş
M.S. in Industrial Engineering
Advisor: Özlem Karsu
September 2023

Equitable multiobjective programming (E-MOP) problems are multiobjective programming problems of a special type. In E-MOP, the decision-maker has equity concerns and hence has an equitable rational preference model. In line with this, our aim is to find all equitably non-dominated points (EN) of the multiobjective integer problems. There are different approaches to solving E-MOP problems. We use equitable aggregation functions and develop two different algorithms; one for equitable biobjective integer programming (E-BOIP) problems and one for equitable multiobjective integer programming (E-MOIP) problems with more than two objectives. In the first algorithm, we solve Pascoletti Serafini (PS) scalarization models iteratively while ensuring getting a weakly equitably non-dominated point in each iteration. In the second algorithm, we use cumulative ordered weighted average in the ExA algorithm of Özpeynirci and Köksalan [1] to find all extreme supported equitably non-dominated points (ESN) first. After finding all ESNs, we use them to define the regions that could contain EN. Then we use split algorithm and find all the remaining ENs. We also provide a split only version of the algorithm since the process of finding all ESNs could be time consuming. We compare two versions in multiobjective assignment and knapsack problem instances. Although the split only version is quicker, the original version of the algorithm is useful since it gives information about the weight space decomposition of ESNs. The weight space decomposition discussion is also provided.

Keywords: equitable optimization, multiobjective integer programming, Pascoletti Serafini scalarization, split algorithm, weight space decomposition.

ÖZET

ÇOK AMAÇLI TAM SAYILI PROGRAMLAMA PROBLEMLERİNİN TÜM EŞİTLİKÇİ BASKIN NOKTALARINI BULMA

Seyit Ulutaş
Endüstri Mühendisliği, Yüksek Lisans
Tez Danışmanı: Özlem Karsu
Eylül 2023

Eşitlikçi çok amaçlı programlama (E-MOP) problemleri, çok amaçlı programlama problemlerinin özel bir türüdür. E-MOP problemlerinde karar vericinin eşitlik kaygısı vardır ve bu yüzden eşitlikçi rasyonel seçim modeline sahiptir. Bunun yanında bizim amacımız da verilen çok amaçlı tam sayılı programlama probleminin tüm eşitlikçi baskın noktalarını (EN) bulmaktır. E-MOP problemlerini çözmek için farklı yaklaşımlar mevcuttur. Bu çalışmada eşitlikçi toplam fonksiyonları kullanılarak iki farklı algoritma geliştirilmiştir. İlki eşitlikçi iki amaçlı tam sayılı programlama (E-BOIP) problemleri için, ikincisi ikiden fazla amacı olan eşitlikçi çok amaçlı tam sayılı programlama (E-MOIP) problemlerini ele almaktadır. İlk algoritmada, her aşamada zayıf eşitlikçi baskın nokta bulacak şekilde art arda Pascoletti Serafini (PS) skalarizasyon modelleri çözülür. İkinci algoritmada ise ilk olarak kümülatif sıralı ağırlıklı toplamı Özpeynirci ve Köksalan'ın [1] ExA algoritmasında kullanarak tüm uç destekli eşitlikçi baskın noktalar (ESN) bulunmaktadır. Tüm ESN'ler bulunduktan sonra, onları kullanarak EN bulundurabilecek bölgeler tanımlanmakta ve ayrık algoritma ile kalan tüm EN'ler bulunmaktadır. ESN'leri bulmak zaman aldığından bu algoritmanın sadece ayrık algoritma kullanan bir versiyonu da verilmiştir. Bu iki versiyon çok amaçlı atama ve sırt çantası problemlerinde karşılaştırılmıştır. Sadece ayrık algoritma kullanan versiyonu hızlı olsa da, algoritmanın orijinal versiyonunun ESN'lerin ağırlık uzayı analizi hakkında bilgi vermesi ayrı bir fayda sağlamaktadır. Ağırlık uzayı analizi de ayrıca verilmiştir.

Anahtar sözcükler: eşitlikçi eniyileme, çok amaçlı tam sayılı programlama, Pascoletti Serafini skalarizasyonu, bölme algoritması, ağırlık uzayı analizi.

Acknowledgement

While completing my thesis and during my undergraduate and graduate studies, my advisor Assoc. Prof. Özlem Karsu provided me with great guidance and support. Her wisdom and encouragement were crucial in shaping my thesis and overcoming the challenges I faced along the way. I sincerely thank her for this valuable contribution. I am also happy that I had the chance to working with her.

My family, Aynur Ulutaş, Metin Ulutaş and Burak Ulutaş were always there for me. Their love, support and understanding played a great role in completing my thesis and getting through this important period. Without their support, I could not have been successful in this journey.

I would also like to thank my friends Göktuğ Dinçer and Soner Gül. You have always been by my side with our game sessions and meetings throughout the master's degree. I am happy that we have formed such a strong friendship.

Atahan Bayır, Onur Kılınç, Burak Taş, Şebnem Manolya Demir, Büşra Bayrak, Başak Fiş, Egemen Elver, Çağla Ergül, Buse Şen and Ömer Ekmekcioğlu, thanks to you, the master's period was easier and more enjoyable. I am very happy to studying with you.

Finally, I would like to express my deepest gratitiude to Zülal. When I felt lost, you have helped me to find my self. You always find a way to cheer me up. You never ceased to amaze me. You were always there for me and I will always be there for you too.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mathematical programming is used widely in various fields. Although some problems have a single objective to be optimized, in many problems there are more than one objective. We call these kind of problems as multiobjective programming (MOP) problems. In MOP problems, the objectives are generally conflicting. For instance, the min cost and min CO_2 emission objectives of an energy planning model are conflicting since reducing CO_2 emission requires investment which incurs cost [2]. Therefore, in MOP problems it is not possible to find a single solution that gives the best value for each objective. If one of the objectives of a solution cannot be improved without worsening one of the other objectives, the solution is said to be efficient and the corresponding objective function vector is non-dominated.

In classical MOP problems, objectives are of different types and not directly comparable. We are interested in problems that are called equitable multiobjective programming (E-MOP) problems. In general E-MOP problems, each objective is of the same type and comparable. For instance, let us consider a project selection problem. In this problem we need to choose which projects to invest by considering the given budget. Moreover, assume that there are several stakeholders. Our aim is to maximize each stakeholder's profits simultaneously and we are indifferent to each stakeholder. Hence, for the three stakeholders case, the profits

of (1,3,2) is equivalent to (3,1,2) in the equitable preference sense. Our aim is to find all equitably non-dominated points that are non-dominated with respect to such equitable preferences for multiobjective integer programming (MOIP) problems.

In this thesis, we first explain the general structure of E-MOP problems and their properties and give the problem definition in Chapter 2. After that, in Chapter 3 we review the literature about general MOP (MOIP) and E-MOP (E-MOIP) problems and their solution methods. Then, we propose a novel algorithm for E-MOIP problems that have two objectives in Chapter 4. In Chapter 5, we propose an algorithm and its variant for E-MOIP problems that have more than two objectives. The computational results of these two algorithms and the comparison of them are provided in Chapter 6, alongside a discussion on weight space decomposition. Finally the conclusion is given in Chapter 7.

Chapter 2

Problem Definition and Preliminaries

In a multiobjective programming (MOP) problem, there are more than one objective functions. The model below is a general model for the MOP problem.

$$\begin{aligned} &max\ \text{“}z_1(x), z_2(x), \ldots, z_p(x)\text{”} \\ &s.t.\ x \in \mathcal{X} \end{aligned}$$

where x denotes the decision variable vector of size n and $\mathcal{X}$ is the feasible set. $z_i(x)$ is the i^{th} objective function where $i = 1, \ldots, p$ and $z = (z_1(x), z_2(x), \ldots, z_p(x))$. Let $\mathcal{Z}$ be the image of set $\mathcal{X}$ in the criterion (objective) space. For MOIP and E-MOIP, $\mathcal{X} \subseteq \mathbb{Z}^n$.

The decision maker (DM) is typically assumed to have a rational preference model, which implies the following: For two image vectors z^1 and z^2, z^1 *dominates* z^2 if $z_i^1 \geq z_i^2$ for all $i = 1, \ldots, p$ and $z_i^1 > z_i^2$ for at least one i where $i = 1, \ldots, p$. $z^1 \in \mathcal{Z}$ is called a non-dominated (Pareto) point if there is no $z^2 \in \mathcal{Z}$ such that $z_i^1 \leq z_i^2$ for all $i = 1, \ldots, p$ and $z_i^1 \neq z_i^2$ for at least one i where $i = 1, \ldots, p$. $x \in \mathcal{X}$ is called an efficient solution if $z(x)$ is a non-dominated point. The DM would not be interested in a dominated point. However, there is usually a set of points,

the elements of which are not dominated by any other feasible image vector, called the non-dominated set. In a priori solution approaches for MOP problems, the aim is to find the best solutions according to the DM's preference. For the interactive solution approaches, in each iteration a preference is asked from the DM [3], [4]. The aim of a posteriori solution approaches to MOP problems is to find all non-dominated points.

The optimal solutions of the weighted sum problem $\max\{\lambda_1 z_1(x) + \lambda_2 z_2(x) + \cdots + \lambda_p z_p(x) : x \in \mathcal{X}\}$ for some $\lambda \in \mathbb{R}^p_>$, $\sum_{i=1}^p \lambda_i = 1$ are *supported efficient solutions* [5]. For a supported efficient solution x^* if there exists no supported efficient solutions x^1 and x^2 ($x^1 \neq x^2$) and $\alpha \in (0,1)$ such that $z(x^*) = \alpha z(x^1) + (1-\alpha) z(x^2)$, then it is called an *extreme supported efficient solution*. Otherwise it is called *non-extreme supported efficient solution*. The remaining efficient solutions that are not optimal solutions of $\max\{\lambda_1 z_1(x) + \lambda_2 z_2(x) + \cdots + \lambda_p z_p(x) : x \in \mathcal{X}\}$ for any $\lambda \in \mathbb{R}^p_>$, $\sum_{i=1}^p \lambda_i = 1$ are called *unsupported efficient solutions*. The image of an extreme supported efficient solution is an extreme supported non-dominated point.

Equitable multiobjective programming (E-MOP) problems are similar to MOP problems. In these problems, the underlying motivation is finding equitable allocations of a certain good across multiple entities. In such problems, each objective function shows the amount of good allocated to one entity. Therefore, as opposed to classical MOP, in E-MOP the objectives are of the same type and measured in same units.

In E-MOP, since the DM has equity concerns, her preference model is assumed to be an equitable rational preference model. Therefore, the dominance concept that is relevant in such settings is equitable dominance, which requires the following two additional properties to be satisfied for the underlying preference model:

- The DM is indifferent between z and any permutation of z since there is *symmetry* in E-MOP problems. For instance, DM is indifferent between (2, 4, 7) and (4, 7, 2).

- The DM prefers $z - \epsilon e_i + \epsilon e_j$ to z for $0 < \epsilon < z_i - z_j$ if $z_i > z_j$, which is called the *principle of transfers*. For instance, DM prefers (5, 6, 7) to (5, 4, 9).

The cumulative ordering map of z is defined as follows: $\theta(z) = (\theta_1(z), \theta_2(z), \ldots, \theta_p(z))$ where $\theta_j(z) = \sum_{i=1}^{j} \overrightarrow{z_i}$ for all $j = 1, \ldots, p$ and $\overrightarrow{z_i}$ is the i^{th} minimum objective.

Cumulative ordering map can be used to check equitable dominance as follows: For two image vectors z^1 and z^2, z^1 *equitably dominates* z^2 if $\theta(z^1) \geq \theta(z^2)$ and $\theta_i(z^1) > \theta_i(z^2)$ for at least one i where $i = 1, \ldots, p$.

Definition 1. *$z^1 \in \mathcal{Z}$ is called equitably non-dominated if and only if there is no $z^2 \in \mathcal{Z}$ such that $\theta(z_i^1) \leq \theta(z_i^2)$ for all $i = 1, \ldots, p$ and $\theta(z_i^1) \neq \theta(z_i^2)$ for at least one i where $i = 1, \ldots, p$. $x \in \mathcal{X}$ is called an equitably efficient solution if $z(x)$ is an equitably non-dominated point.*

Definition 2. *$z^1 \in \mathcal{Z}$ is called weakly (equitably) non-dominated if and only if there is no $z^2 \in \mathcal{Z}$ such that $z_i^1 < z_i^2$ ($\theta(z_i^1) < \theta(z_i^2)$) for all $i = 1, \ldots, p$. $x \in \mathcal{X}$ is called an weakly (equitably) efficient solution if $z(x)$ is an weakly (equitably) non-dominated point.*

The optimal solutions of the weighted cumulative ordered sum problem $\max\{\lambda_1\theta_1(z(x)) + \lambda_2\theta_2(z(x)) + \cdots + \lambda_p\theta_p(z(x)) : x \in \mathcal{X}\}$ for some $\lambda \in \mathbb{R}^p_>$, $\sum_{i=1}^{p} \lambda_i = 1$ are *supported equitably efficient solutions*. For a supported equitably efficient solution x^* if there exists no supported equitably efficient solutions x^1 and x^2 ($x^1 \neq x^2$) and $\alpha \in (0, 1)$ such that $\theta(z(x^*)) = \alpha\theta(z(x^1)) + (1-\alpha)\theta(z(x^2))$, then it is called an *extreme supported equitably efficient solution*. Otherwise it is called *non-extreme supported equitably efficient solution*. The remaining equitably efficient solutions that are not optimal solutions of $\max\{\lambda_1\theta_1(z(x)) + \lambda_2\theta_2(z(x)) + \cdots + \lambda_p\theta_p(z(x)) : x \in \mathcal{X}\}$ for any $\lambda \in \mathbb{R}^p_>$, $\sum_{i=1}^{p} \lambda_i = 1$ are called *unsupported equitably efficient solutions*. The image of an extreme supported equitably efficient solution is an extreme supported equitably non-dominated point.

For instance, let (5,6,7) and (5,4,9) be our non-dominated points. Both of

them are extreme supported non-dominated points. Notice that their cumulative ordering maps are (5,11,18) and (4,9,18) respectively. (5,6,7) equitably dominates (5,4,9) since $(5, 11, 18) \geq (4, 9, 18)$ and $(5, 11, 18) \neq (4, 9, 18)$. Therefore, (5,6,7) is the only extreme supported equitably non-dominated point.

Throughout the text, we denote the set of equitably non-dominated points by EN and set of extreme supported equitably non-dominated points by ESN.

Thus, the aim of a posteriori solution approach to E-MOP problems is to find all equitably non-dominated points of a given MOP problem. Our aim is to develop such generic algorithms that find the whole set of equitably non-dominated points for equitable multiobjective integer programming (E-MOIP) problems with two objectives and E-MOIP problems with more than two objectives.

Proposition 1. *A feasible solution $x \in \mathcal{X}$ is an equitably efficient solution of the given MOIP problem if and only if it is an efficient solution of the following nonlinear programming model [6].*

$$\max \text{“}\overrightarrow{z_1}(x), \overrightarrow{z_1}(x) + \overrightarrow{z_2}(x), \ldots, \sum_{i=1}^{p} \overrightarrow{z_i}(x)\text{”}$$
$$s.t.\ x \in \mathcal{X}$$

which is equivalent to

$$\max \text{“}\theta_1(z(x)), \theta_2(z(x)), \ldots, \theta_p(z(x))\text{”}$$
$$s.t.\ x \in \mathcal{X}$$

where n is the number of decision variables. We call this model as E-MOIP model.

We use E-MOIP model for the split based algorithm in Chapter 5 with algorithms developed for MOIP since finding an efficient solution of E-MOIP model gives us an equitably efficient solution of the given MOIP problem by Proposition 1.

Chapter 3

Literature Review

There are various solution methods for MOP problems in the literature. A posteriori solution approaches for MOP problems can be classified according to the space they work on.

Some of them work on the decision space. For instance, Mavrotas and Diakoulaki [7] use a branch and bound algorithm for the mixed 0-1 multiple objective linear programming problem and De Santis and Eichfelder [8] present a branch and bound algorithm for MOIP problems with convex quadratic objectives. Moreover, Bazgan et al. [9] use dynamic programming for the 0-1 multiobjective knapsack problem. Gausemeier et al. [10] apply dynamic programming on the multiobjective optimization of vehicle velocity profile. Mahmoudimehr and Sebghati [11] develop a dynamic programming method for MOP and applies it to an energy management system.

Some of them work on the objective function space. For instance, Özlen and Azizoğlu [12] use ϵ-constraint method for MOIP problems. Javadi et al. [13] model self-scheduling of home energy management system as MOP problem and uses ϵ-constraint method. Tamby and Vanderpooten [14] propose a generic algorithm by extending the ϵ-constraint method and compute non-dominated set

of MOIP problems. Kirlik and Sayın [15] develops an algorithm for MOIP problems motivated by the ϵ-constraint method and the algorithm works on $p-1$ dimensional rectangles of the criteria space. Sylva and Crema [16] use a weighted sum method for multiobjective integer programming problems. Rehman and Khan [17] apply weighted sum method for multicriteria wind turbine selection problem. Hua and Abdullah [18] combine weighted sum method with Dijkstra's Algorithm for identification of the best path according to the multiple criteria.

We can also classify these solution approaches for MOP problems according to the problem type as linear programming (MOLP), integer programming (MOIP) and mixed-integer programming (MOMIP). [10], [11], [13] and [17] study MOP problems. [8], [9], [12], [14], [15], [16] and [18] work on MOIP problems. [7], [19], [20] and [21] focuses on MOMIP problems.

Equity is considered in many studies of optimization. For scheduling problems, equity can be considered as the fair distributions of work load among employees [22], while in allocation problems it can be considered as the allocation of resource among the entities [23]. Lorenz dominance also can be adapted for fair optimization [24]. Moreover, for portfolio optimization minimizing the maximum regret is a way to address equity concerns [25]. Karsu and Morton [26] discusses two equity related concerns which are equitability and balance. We have equitability concern when we are not interested in the identities of entities. Kaynar and Karsu [27] develop an interactive algorithm that guides the DM to the best choice among a set of allocation alternatives according to her preference. Karsu and Erkan [28] proposes an approach for resource allocation problems that maximizes total benefit while minimizing the imbalance defined according to a reference distribution. Yavuz [29] proposes two approaches for fair resource allocation problem. In the first approach she uses ordered weighted average for wide range of weights. In the second approach, she gets a reference point from the DM and maximizes efficiency by controlling fairness in the constraints according to the reference point. In this thesis, we focus on equitability since we focus on the distribution and there is symmetry. As Karsu and Morton [26] discusses there are three main approaches to incorporate the equitability concerns in mathematical programming.

First one is the Rawlsian approach. In general the Rawlsian approach focuses on the worst-off entity and the aim is to get the best possible value for this worst-off entity [30]. Tavana et al. [20] aim to maximize the minimum vaccine delivery-to-demand ratio for equitable COVID-19 vaccine distribution by presenting a mixed-integer linear program. Chen and Hooker [21] propose an optimization procedure for balancing Rawlsian approach and Utilitarian approach which focuses on maximizing the total utility without considering the distribution of the utility over entities. In our problem, the worst-off entity is the value of the minimum objective since we consider a maximization problem.

Second one is the inequality index based approach. The inequality index based approaches use an inequality index which incorporates equitability concerns. This index can be used either as an objective or constraint. Some commonly used inequality indices are as follows [26],

- The range between the minimum and maximim levels of outcomes $(\max_{i=1,\dots,p} z_i - \min_{i=1,\dots,p} z_i)$
- Mean deviation
- Variance $(\frac{\sum_{i=1}^{p}(z_i-\bar{z})^2}{p})$ where $\bar{z} = \frac{\sum_{i=1}^{p} z_i}{p}$
- Gini coefficient $(\frac{\sum_{i=1}^{p}\sum_{j=1}^{p}|z_i-z_j|}{2p\sum_{i=1}^{p} z_i})$
- Sum of pairwise (absolute) differences $(\sum_{i=1}^{p}\sum_{j=1}^{p}|z_i - z_j|)$

The last one is inequity-averse aggregation function based approach. This approach uses inequity-averse aggregation functions in order to incorporate equitability concerns. Those aggregation functions should be symmetric and address the concerns related to inequity-aversion.

Kostreva and Ogryczak [31] defines equitable efficiency and equitable rational preference relation. Baatar and Wiecek [32] derives the complete preference structure of equitability. Equitable rational preference relation has reflexivity, transitivity, strict monotonicity, anonymity and satisfies principle of transfers.

An aggregation function is called equitable aggregation function if it follows equitable rational preference relation. In other words, if it is strictly increasing, symmetric and satisfies the principle of transfers, the aggregation function is equitable aggregation function [6]. Moreover, if a solution maximizes an equitable aggregation function, it is also equitably efficient.

Kostreva et al. [6] uses ordered weighted averaging (OWA) as an equitable aggregation function. They first order the objectives from the worst to the best. With strictly decreasing and positive weights for these ordered objectives, they get an equitably non-dominated points from the OWA model. In order to find the order among the objectives, they introduce a linearization. We customize OWA for the cumulative ordered weighted averaging and use it in our algorithm in Chapter 5.

Bashir and Karsu [33] develop two algorithms for E-MOIP problems. The first algorithm is an interactive algorithm and asks the DM's preference among the proposed two points in each iteration. They used cumulative OWA model for finding equitably non-dominated points. This algorithm finds a subset of the set of equitably non-dominated points. In the second algorithm, they try to fit hyperplanes to the equitably non-dominated frontier. After fitting hyperplanes they select points on them and define regions around these points. They find a subset of the set of equitably non-dominated points that is well spread by searching equitably non-dominated points in these regions.

Singh [34] presents two scalarization for minimization problem and proves they are equitable scalarizations. The first one is exponential schur scalarization which is $\sum_{i=1}^{p} e^{az_i}$ for $a \geq 0$. The other one is squared-sum schur scalarization which is $\sum_{i=1}^{p}(z_i - a)^2$ for $a < \min_{i=1,\ldots,p} a_i$ where $a_i = \min_{x \in \mathcal{X}} z_i(x)$ for $i = 1, \ldots, p$.

In this thesis, for E-MOIP problems with two objectives, we developed a novel algorithm by customizing Pascoletti Serafini scalarization and ensuring it gives a weakly equitably non-dominated point. Then, for E-MOIP problems having more than two objectives, we developed an algorithm using OWA scalarization for cumulative ordered objectives. Both algorithms work on objective function

space. Hence, they iteratively solve respective scalarization problems.

Chapter 4

An Algorithm for E-MOIP Problems With $p = 2$

There is a vast amount of approaches for finding the non-dominated points of MOIP problems. Tchebychev and Pascoletti Serafini (PS) scalarizations are commonly used in MOIP problems [35] [36]. Both scalarizations try to find the "closest" solution to a reference point with respect to a given weight or distance. We first discuss the models for the classical MOP.

4.1 Scalarization Models

Let z^R be the reference point from which the distance is minimized.

Tchebychev Scalarization: Tchebychev scalarization finds the solution that is closest to a reference vector z^R in terms of (weighted) Tchebychev distance by solving the model below for weight vector $w > 0$. We take $z_i^R = max_{x \in \mathcal{X}} z_i(x) \quad \forall i = 1, \ldots, p$ [35].

$$
\begin{aligned}
&min\ \alpha \\
&s.t.\ \alpha \geq w_i(z_i^R - z_i(x)) \quad \forall i = 1, \ldots, p \\
&\quad x \in \mathcal{X}
\end{aligned}
$$

Proposition 2. *Let (α^*, x^*) be the optimal solution of Tchebychev scalarization problem. Then, $z(x^*)$ is a weakly non-dominated point [37].*

Proof. See Appendix A. □

Pascoletti Serafini (PS) Scalarization: PS scalarization is similar to the Tchebychev scalarization. However, instead of a weight vector, PS uses a direction vector $d \geq 0$ $(d \neq 0)$ while finding the solution [36].

$$
\begin{aligned}
&min\ \rho \\
&s.t.\ \rho \geq \frac{z_i^R - z_i(x)}{d_i} \quad \forall i = 1, \ldots, p \\
&\quad x \in \mathcal{X}
\end{aligned}
$$

Proposition 3. *Let (ρ^*, x^*) be the optimal solution of PS scalarization problem with reference point z^R and direction vector d. Then, $z(x^*)$ is a weakly non-dominated point [36].*

Proof. See Appendix B. □

Proposition 4. *Assume $z_i^R = c \forall i = 1, \ldots, p$ and $c \geq \max_{i=1,\ldots,p, x \in \mathcal{X}} z_i(x)$. For a given solution x, the minimum value of ρ in the Pascoletti Serafini scalarization problem with $d : d_i = 1\ \forall i = 1, \ldots, p$ ($d = \mathbf{1}$) is determined by the minimum component of the $z(x)$. The higher the minimum component of the $z(x)$, the lower is the ρ value.*

Proof. In Pascoletti Serafini scalarization problem with $d = 1$, the aim is to solve the model below.

$$\begin{aligned} &\min \rho \\ &s.t.\ \rho \geq z_i^R - z_i(x) \quad \forall i = 1, \ldots, p \\ &\quad x \in \mathcal{X} \end{aligned}$$

This ensures that $\rho^* = \max_{i=1,\ldots,p,x\in\mathcal{X}}\{z_i^R - z_i(x)\} = \max_{i=1,\ldots,p,x\in\mathcal{X}}\{c - z_i(x)\} = c - \min_{i=1,\ldots,p,x\in\mathcal{X}}\{z_i(x)\}$.

Hence, higher value of the minimum component of $z(x)$ yields lower ρ^* value in Pascoletti Serafini scalarization problem. □

Hence solving the PS problem is equivalent to $\min\{c - \min_{i=1,\ldots,p,x\in\mathcal{X}} z_i(x)\} = \max \min_{i=1,\ldots,p,x\in\mathcal{X}} z_i(x)$.

Proposition 5. *Let (ρ^*, x^*) be the optimal solution of Pascoletti Serafini scalarization problem with $z_i^R = c\ \forall i = 1, \ldots, p$ where $c \geq \max_{i=1,\ldots,p,x\in\mathcal{X}} z_i(x)$ and $d = \mathbf{1}$. Then, $z(x^*)$ is a weakly equitably non-dominated point. Moreover, it has the highest $\theta(z_1) = \overrightarrow{z}_1$ level.*

Proof. To the contrary assume that $z(x^*)$ is not weakly equitably non-dominated. Then $\exists\, x' \in \mathcal{X}$, $z(x')$ strictly equitably dominates $z(x^*)$ such that $\theta(z(x')) > \theta(z(x^*))$. Then, the following holds:

$$\begin{aligned} \theta_1(z(x')) &> \theta_1(z(x^*)) \\ \min_{i=1,\ldots,p} z_i(x') &> \min_{i=1,\ldots,p} z_i(x^*) \\ \rho' &< \rho^* \end{aligned}$$

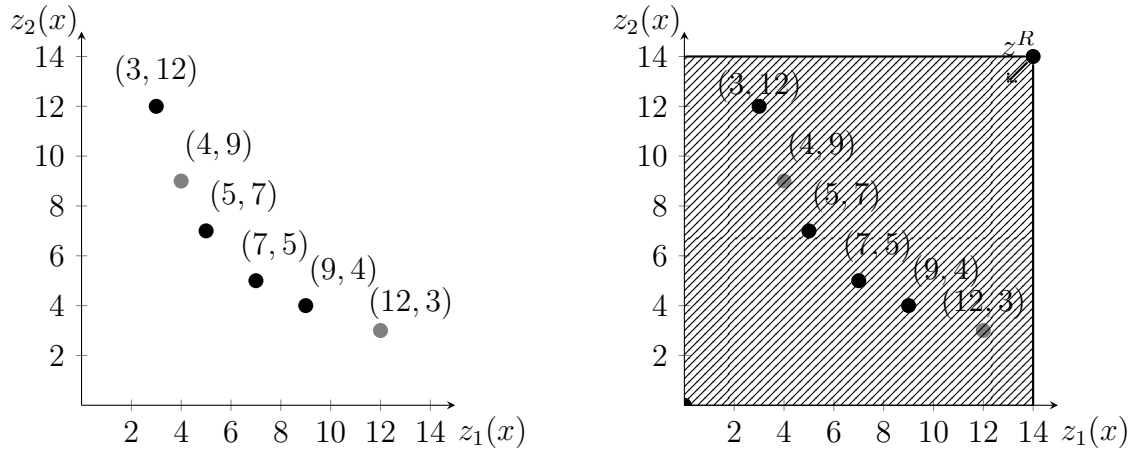
This **contradicts** the optimality of (ρ^*, x^*) for the Pascoletti Serafini scalarization problem. □

By using these properties, we customize PS scalarizations for E-MOIP and develop an algorithm for equitable biobjective integer programming (E-BOIP) problems. The proposed algorithm for E-BOIP problem will use PS scalarization.

4.2 The Algorithm

This algorithm utilizes PS scalarization by iteratively changing the reference and direction vectors and finds all equitably non-dominated points of a given E-BOIP problem. As it finds new points, it removes the regions that are dominated by and dominating these points and searches for new points in the objective function space.

An illustrative example is provided with the explanation of the algorithm. In Figure 4.1a, all equitably non-dominated points of an example E-BOIP problem is shown. The black points denote the feasible equitably non-dominated points and the gray ones show the infeasible permutations of them. For instance, although (9,4) is feasible (4,9) is not. We consider the gray points as they are feasible during the algorithm since there is symmetry in our problem.



(a) Set EN

(b) Reference point z^R and feasible region

Figure 4.1: Algorithm steps

The algorithm starts with an initialization part, in which each z_i is maximized individually for all $i = 1, 2$ (line 2 of Algorithm 1). Let $z_i^* = \max_{x \in \mathcal{X}}\{z_i(x)\}$ be the maximum values obtained and $z^* = \max\{z_1^*, z_2^*\}$. Then, the reference point is set as $z^R = (c, c)$ where $c \geq z^*$ (see Figure 4.1b) (line 3 of Algorithm 1). Since z^R is created by using such c which is greater than or equal to the maximum of the maximum values of two objectives, we know that our feasible region in the objective space is bounded by it (see Figure 4.1b).

After finding z^R, we solve model $P_1(z^R, d)$ for $d = (1, 1)$.

$$\mathbf{P_1(z^R, d)}$$

$$\begin{aligned} \min\ & \rho \\ s.t.\ & z(x) \geq z^R - \rho d \\ & x \in \mathcal{X} \end{aligned}$$

Let the solution be x'. From Proposition 5, it is known that $z(x')$ is weakly equitably non-dominated. It has the highest $\theta_1(z)$, i.e. $\overrightarrow{z}_1(x)$ level. However, there may still be points having the same $\theta_1(z)$ level but with better values for $\theta_2(z)$, i.e. $\sum_{i=1}^{2} z_i(x)$. Therefore, we solve $Q_1(x')$ to ensure that we have an equitably non-dominated point.

$$\mathbf{Q_1(x')}$$

$$\begin{aligned} \max\ & z_1(x) + z_2(x) \\ s.t.\ & z_1(x) \geq \overrightarrow{z_1}(x') \\ & z_2(x) \geq \overrightarrow{z_1}(x') \\ & x \in \mathcal{X} \end{aligned}$$

Let the solution be x^* (line 4 of Algorithm 1). $z(x^*)$ is the equitably non-dominated point closest to the reference point with respect to the Tchebychev distance.

For simplicity, we will call these two consecutive models as problem $M(z^R, d)$ which is as follows: Solve model $P_1(z^R, d)$ and let its solution be x'. If x' exists, solve $Q_1(x')$. Let the solution of $M(z^R, d)$ be x^*. $z(x^*)$ is added to EN (the set of equitably non-dominated points found).

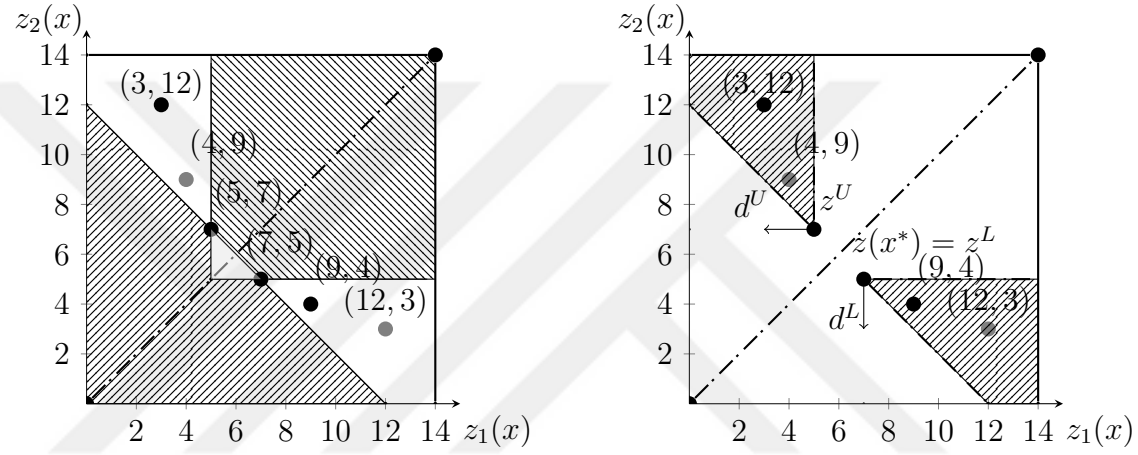
Based on the newly found point $z(x^*)$, two new reference points, z^L and z^U are defined as $(\overrightarrow{z_2}(x^*), \overrightarrow{z_1}(x^*))$ and $(\overrightarrow{z_1}(x^*), \overrightarrow{z_2}(x^*))$ respectively (see Figure 4.2a) (line 8 of Algorithm 1). These two vectors are permutations of $z(x^*)$ and are used to search for equitably non-dominated points in each side of the perfect equality line $(z_1(x) = z_2(x))$ since E-MOIP problem has symmetry. Let (5,7) be the point we found by solving $M(z^R, d)$ (see Figure 4.2a). Notice that, (5,7) is also feasible. Hence, after solving $M(z^R, d)$, we also check whether the permutation of this point lying on the other side is also feasible. In the case of feasibility, we add both points to the equitably non-dominated points list. To check the feasibility we solve the below model:

$$\mathbf{S}(\mathbf{x}^*)$$

$$\begin{aligned} &\max 0 \\ &s.t.\ z_1(x) = z_2(x^*) \\ &\qquad z_2(x) = z_1(x^*) \\ &\qquad x \in \mathcal{X} \end{aligned}$$

The upper right shaded region in Figure 4.2a is the region dominating z^L and z^U. However, we know that this region has to be empty since if there is an equitably non-dominated point in this region, it should be found before $z(x^*)$ as it would be closer to z^R in terms of Tchebychev distance. Similarly, the points in the gray region in the middle are not dominated by the new reference points;

their minimum objective is greater than the minimum objective of the reference points so this region can not contain any equitably non-dominated points. Finally the lower right shaded region is the region dominated by z^L and z^U. Therefore, the remaining regions that could contain equitably non-dominated points are the shaded regions shown in Figure 4.2b.



(a) Found solution and resulting areas

(b) Ref. points, directions and feasible regions

Figure 4.2: Algorithm steps continued

We set $T = z_1(x^*) + z_2(x^*)$ (line 8 of Algorithm 1). The algorithm solves $P_2(T, z^L, d^L)$ and $P_2(T, z^U, d^U)$ for $d^L = (0, 1)$ and $d^U = (1, 0)$. We set a lower bound on the sum of objectives as $T + \epsilon$ (for a small $\epsilon \geq 0$) in order to avoid returning the respective reference points found before.

$$\mathbf{P_2}(\mathbf{T}, \mathbf{z^R}, \mathbf{d})$$

$$\begin{aligned} &\min \rho \\ &s.t.\ z(x) \geq z^R - \rho d \\ &\quad z_1(x) + z_2(x) \geq T + \epsilon \\ &\quad x \in \mathcal{X} \end{aligned}$$

Proposition 6. *Let x' and x'' be the solutions obtained by solving $P_2(T, z^L, d^L)$*

and $P_2(T, z^U, d^U)$, respectively. $z(x')$ and $z(x'')$ are also weakly equitably non-dominated.

Proof. In Figure 4.3a, we can see the feasible regions in an arbitrary iteration of the algorithm. Assume that $x^{'}$ is the optimal solution of $P_2(T, z^L, d^L)$, $d_L = (0, 1)$ and $z(x^{'})$ is not weakly equitably non-dominated. Then, there should be at least one $z(x^*)$ that strictly equitably dominates $z(x^{'})$ which means $\theta(z(x^{'})) < \theta(z(x^*))$. In Figure 4.3b, the shaded region is the set of points strictly equitably dominating $z(x^{'})$. Therefore, $z(x^*)$ should be in this shaded region. In this region, we know that $z_1(x^*) \geq z_2(x^*)$. We also know that both $z_1(x^{'})$ and $z_1(x^*)$ are greater than z_1^L. We can write $P_2(T, z^L, d^L)$ as follows:

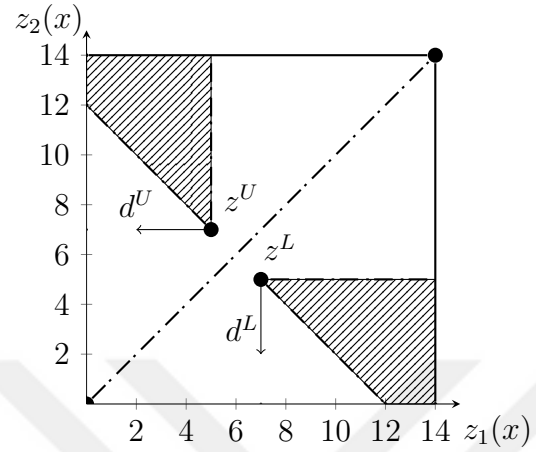
$$\mathbf{P_2(T, z^L, d^L)}$$

$$\begin{aligned} \min\ & \rho \\ s.t.\ & z_1(x) \geq z_1^L \\ & z_2(x) \geq z_2^L - \rho \\ & z_1(x) + z_2(x) \geq T + \epsilon \\ & x \in \mathcal{X} \end{aligned}$$

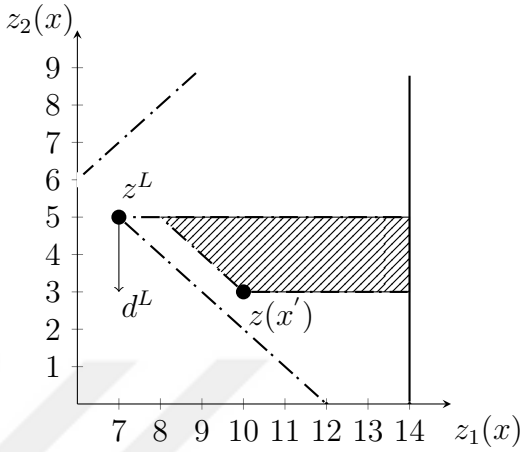
We know that both $z(x^{'})$ and $z(x^*)$ satisfy the first and third constraint since they are over the $z_1(x) + z_2(x) = T$ line and have greater $z_1(x)$ values. Then, the solution with a greater $z_2(x)$ value yields a lower ρ value in the second constraint. Therefore, the optimal solution of $P_2(T, z^L, d^L)$ should be x^*. However, it contradicts our initial assumption. The proof for the upper part for $P_2(T, z^U, d^U)$ model is similar.

□

Let x' and x'' be the solutions obtained by solving $P_2(T, z^L, d^L)$ and $P_2(T, z^U, d^U)$, respectively. $z(x')$ and $z(x'')$ are also weakly equitably non-dominated by Proposition 6. The algorithm solves $Q_L(x')$ and $Q_U(x'')$ in order to ensure that we have equitably non-dominated points (line 9 of Algorithm 1).



(a) Feasible regions in an arbitrary iteration



(b) Solving $P_2(T, z^L, d^L)$

Figure 4.3: Illustration for proof of Proposition 6

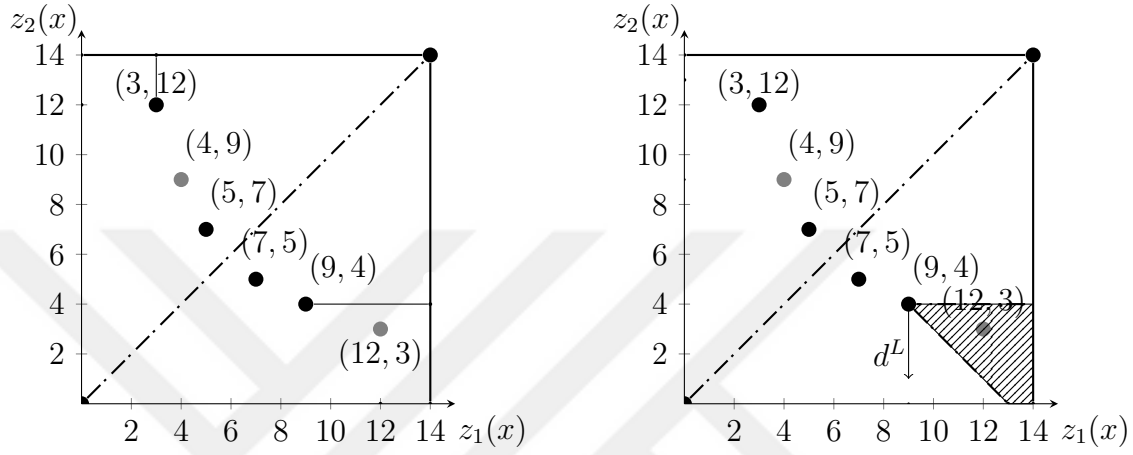
$$\mathbf{Q_L(x')} \qquad\qquad \mathbf{Q_U(x')}$$

$$\begin{aligned} \max\ & z_1(x) \\ s.t.\ & z_2(x) = z_2(x') \\ & x \in \mathcal{X} \end{aligned} \qquad\qquad \begin{aligned} \max\ & z_2(x) \\ s.t.\ & z_1(x) = z_1(x') \\ & x \in \mathcal{X} \end{aligned}$$

P models can be referred as stage one and Q models can be referred as stage two for this algorithm. For simplicity, we will refer to the two consecutive models $P_2(T, z^L, d^L)$ and $Q_L(x')$ as $M_L(T, z^L, d^L)$. Similarly, we will refer $P_2(T, z^U, d^U)$ and $Q_U(x')$ as $M_U(T, z^U, d^U)$.

After finding points with $M_L(T, z^L, d^L)$ and $M_U(T, z^U, d^U)$, the algorithm compares these two points and picks the point that equitably dominates the other (line 10 of Algorithm 1). If these two points are symmetric, we add both points to EN and update z^L and z^U as these points according to their sides and solve $M_L(T, z^L, d^L)$ and $M_U(T, z^U, d^U)$ again. If both of them are equitably non-dominated, it picks the point with the lower $z_1(x)+z_2(x)$ value in order to narrow down the feasible region without eliminating any equitably non-dominated points (line 17 of Algorithm 1). We provide an example in Figure 4.4. We see that among

the two points (9,4) and (3,12), (9,4) is picked as it has a lower $z_1(x)+z_2(x)$ value. Let x^* be the solution picked. $(z(x^*))$ is added into EN (lines 18 of Algorithm 1).

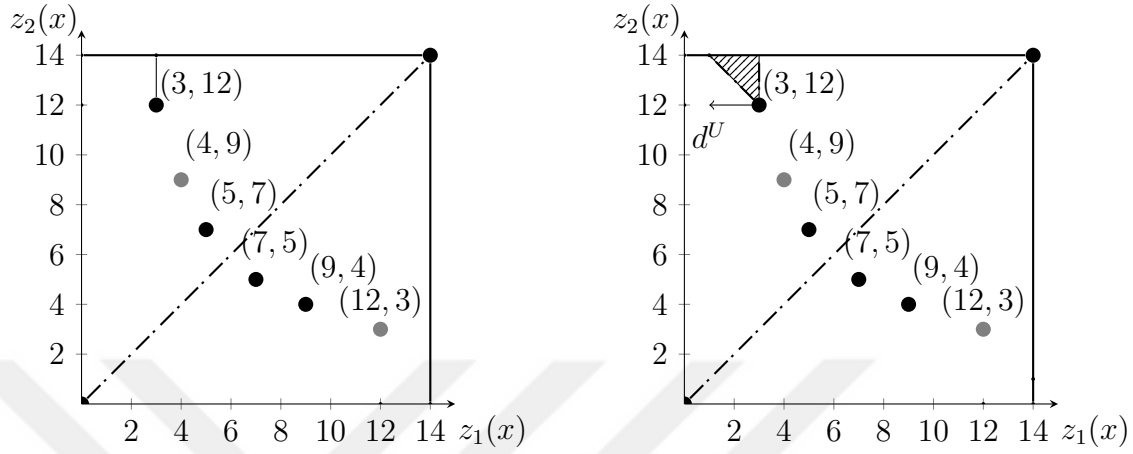


(a) Solutions found (b) New ref. points and feasible regions

Figure 4.4: Algorithm steps continued

After that, it updates T as $z_1(x^*) + z_2(x^*)$, z^L as $(\overrightarrow{z_2}(x^*), \overrightarrow{z_1}(x^*))$ and z^U as $(\overrightarrow{z_1}(x^*), \overrightarrow{z_2}(x^*))$ (line 14 of Algorithm 1). Then, it solves $M_L(T, z^L, d^L)$ and $M_U(T, z^U, d^U)$ and repeats the same steps until both models become infeasible, indicating that there are no other equitably non-dominated points (see Figure 4.5a).

To increase the efficiency of the algorithm, if any $M_L(T, z^L, d^L)$ or $M_U(T, z^U, d^U)$ becomes infeasible, the algorithm stops exploring that region since any subset would also be empty in terms of the equitably non-dominated points (see Figure 4.5b) (lines 28-34 of Algorithm 1). Moreover, for the case where there are two different equitably non-dominated points and we picked the one that has the lower sum, the algorithm only solves $M_L(T, z^L, d^L)$ or $M_U(T, z^U, d^U)$ depending on the side where the picked point is in and compares the new point with the point that is not picked in the previous step (see Figure 4.4b) (lines 19-25 of Algorithm 1).



(a) New solution found

(b) New ref. point and feasible region

Figure 4.5: Algorithm steps continued

The pseudo code of this algorithm can be found in Algorithm 1. A computational experiment for this algorithm can be found in Chapter 6.4.

Algorithm 1

Let EN $= \emptyset$, $T = 0$, $z^L = 0$, $z^U = 0$, $d = (1, 1), d^L = (0, 1), d^U = (1, 0)$;
Let $z_i^* = \max_{x \in \mathcal{X}}\{z_i(x)\}$ for all $i = 1, \ldots, p$ and $z^* = \max\{z_1^*, z_2^*\}$.
Define $z^R = (c, c)$, $c \geq z^*$;
Solve $M(z^R, d)$ and let x^* be the solution;
if $S(x^*)$ is feasible **then**
    Let the solution of the model be x^{**} and EN $=$ EN $\cup \{z(x^{**})\}$
end if
EN $=$ EN $\cup \{z(x^*)\}$, $T = z_1(x^*) + z_2(x^*)$, $z^L = (\overrightarrow{z_2}(x^*), \overrightarrow{z_1}(x^*))$, $z^U = (\overrightarrow{z_1}(x^*), \overrightarrow{z_2}(x^*))$;
Solve $M_L(T, z^L, d^L)$ and $M_U(T, z^U, d^U)$ and let x' and x'' be the solutions respectively;
while x' or x'' exist **do**
    if both exist **then**
        if one of them equitably dominates the other **then**
            Let the equitably dominating one be x^*;
            EN $=$ EN $\cup \{z(x^*)\}$, $T = z_1(x^*) + z_2(x^*)$, $z^L = (\overrightarrow{z_2}(x^*), \overrightarrow{z_1}(x^*))$, $z^U = (\overrightarrow{z_1}(x^*), \overrightarrow{z_2}(x^*))$;
            Solve $M_L(T, z^L, d^L)$ and $M_U(T, z^U, d^U)$ and let x' and x'' be the solutions respectively;
        else
            Select the solution with lower $\overrightarrow{z_1}(x) + \overrightarrow{z_2}(x)$ or choose one point arbitrarily in the case of equal sums. Let this solution be x^*;
            EN $=$ EN $\cup \{z(x^*)\}$, $T = z_1(x^*) + z_2(x^*)$;
            if x' is the selected solution **then**
                $z^L = (\overrightarrow{z_2}(x^*), \overrightarrow{z_1}(x^*))$;
                Solve $M_L(T, z^L, d^L)$ and let x' be the solution;
            else
                $z^U = (\overrightarrow{z_1}(x^*), \overrightarrow{z_2}(x^*))$;
                Solve $M_U(T, z^U, d^U)$ and let x'' be the solution;
            end if
        end if
    else
        if x' exists **then**
            $z^L = (\overrightarrow{z_2}(x^*), \overrightarrow{z_1}(x^*))$;
            Solve $M_L(T, z^L, d^L)$ and let x' be the solution;
        else
            $z^U = (\overrightarrow{z_1}(x^*), \overrightarrow{z_2}(x^*))$;
            Solve $M_U(T, z^U, d^U)$ and let x'' be the solution;
        end if
    end if
end while
return EN.

Chapter 5

Split Based Algorithm for E-MOIP Problems With $p \geq 3$

A Split Based Algorithm (SBA) is developed in order to find all ENs of general E-MOIP problems ($p \geq 3$) efficiently.

The Split Based Algorithm works in the objective function space and has two parts. In the first part, it uses the modified version of the ExA algorithm developed by Özpeynirci and Köksalan [1]. ExA finds the extreme supported non-dominated points of classical multiobjective (mixed) integer programming (MOMIP) problems. We use this algorithm to find the extreme supported non-dominated points of the problem with objective functions corresponding to the elements of the cumulative ordered vector. We know that these non-dominated points are also equitably non-dominated points of the original E-MOIP problem by Proposition 1. Therefore, SBA finds all extreme supported equitably non-dominated points (ESN) of the problem in the first part.

In the second part, SBA uses a modified version of the split algorithm discussed in Karsu and Ulus [38]. The original algorithm is designed to find all non-dominated points of a classical multiobjective integer programming (MOIP) problem. It relies on searching subregions in the objective function space, called

boxes. We adapt it to the equitable setting as we will explain later in detail.

In its original form, the split algorithm starts with exploring an initial box, defined by the ideal point and lower bound for the nadir point. In our algorithm, we define a set of initial boxes based on the ESNs found in the first stage. We then search each box to find the remaining ENs. Whenever a new point is found, new boxes are generated and the algorithm stops when there are no new boxes to be explored. The algorithm only discards part of the search space that are guaranteed not to include any non-dominated points. Hence, when SBA stops, it returns all ENs.

The weight space for the ESNs can be generated by using the weight information gathered in the first part of SBA. Hence, the decision maker can easily figure out which preference among the objectives corresponds to which point. An example of it can be found in the Chapter 6.

Both parts of the SBA rely on existing algorithms designed for the classical MOIP. To be able to adopt these for E-MOIP, SBA needs to identify the cumulative ordered objectives within the scalarization model. For that purpose, it uses the linearization introduced by Kostreva et al. [6]. By using this linearization, we can get the linearized scalarization model of the E-MOIP model introduced in Chapter 2 below. Hence, we know that any solution of this model is equitably efficient by Proposition 1.

$$
\begin{aligned}
\max \ & \sum_{k=1}^{p} \lambda_k \theta_k \\
s.t. \ & r_k - d_{ik} \leq z_i(x) \quad i,k = 1,\dots,p \\
& \theta_k = k r_k - \sum_{i=1}^{p} d_{ik} \quad k = 1,\dots,p \\
& d_{ik} \geq 0 \quad i,k = 1,\dots,p \\
& x \in \mathcal{X}
\end{aligned}
$$

5.1 Split Based Algorithm

5.1.1 Finding ESNs

Split Based Algorithm solves the following model in order to find the maximum sum of the objectives (line 1 of Algorithm 2).

$$\max \sum_{k=1}^{p} z_k(x)$$
$$s.t.\ x \in \mathcal{X}$$

Let x^* be the optimal solution of this model and $T^* = \sum_{k=1}^{p} z_k(x^*)$ (line 1 of Algorithm 2). Then, SBA proceeds to the modified ExA part and finds ESNs.

Özpeynirci and Köksalan [1] developed ExA algorithm for MOMIP problems assuming that objective functions are of minimization type and all objective values are non-negative. For a MOMIP problem with $p \geq 3$, at first they create p dummy points as $m^i = M \times e_i, i = 1, \ldots, p$ where e_i is the i^{th} unit vector and M is a sufficiently large number that is greater than the lower bound specified by Özpeynirci [39]. The dummy points are also included in the extreme supported non-dominated points set since they are not dominated by the other extreme supported non-dominated points and do not dominate each other. Let Y_E and Y_M denote the set of extreme supported non-dominated points and the set of dummy points respectively, and $Y_{EM} = Y_E \cup Y_M$.

Moreover, Özpeynirci and Köksalan [1] prove that if $Y_E \neq \emptyset$ and $p \geq 3$, then every point in Y_{EM} is adjacent to at least p points in Y_{EM}. By using this fact, the ExA algorithm finds all extreme supported non-dominated points of the given MOIP problem. ExA algorithm can be found in Appendix C.

We explain the modified version of this algorithm that we created for E-MOIP. For relevance, we keep the notation used in [1]. SBA initializes $Y_E^{'} = \emptyset$, $k = 0$,

$\mathcal{V} = \emptyset$, $\mathcal{F} = \emptyset$, $\mathcal{L} = \emptyset$ and $M = p * (T^* + 1)^2$ (lines 2,3 of Algorithm 2). The p dummy points are created accordingly and set of the dummy points are added to $\mathcal{L}$ as $\mathcal{L} = \{\{m^1, m^2, \ldots, m^p\}\}$ (lines 3,4 of Algorithm 2). Let $Y_E^{'}$ denote the set of ESNs found so far, $\mathcal{L}$ be the list of stages to be searched, $\mathcal{V}$ be the list of stages already searched and $\mathcal{F}$ be the list of facet-defining stages. $\mathcal{L}$

At any iteration the algorithm chooses p of the already-found points and checks whether the set of points is facet defining or whether new points can be found below the hyperplane passing through them. For that purpose it keeps the list of p-point-sets (called stages), $\mathcal{L}$, which is initialized by the set of p dummy points.

At any iteration of the algorithm, a stage $R = \{r^1, r^2, \ldots, r^p\}$ is selected from $\mathcal{L}$ to be explored (line 5 of Algorithm 2). Then the normal (λ) of the hyperplane passing through these vectors is calculated by finding λ: $\lambda r^1 = \lambda r^2 = \cdots = \lambda r^p$. If $\lambda \geq 0$, then SBA solves E-MOIP(λ) for minimization problems and modified E-MOIP(λ) for maximization problems (lines 6-8 of Algorithm 2), which are as follows:

E-MOIP(λ):

$$\min \sum_{k=1}^{p} \lambda_k \theta_k \tag{5.1}$$

$$s.t.\ r_k + d_{ik} \geq z_i(x) \quad i, k = 1, \ldots, p \tag{5.2}$$

$$\theta_k = k r_k + \sum_{i=1}^{p} d_{ik} \quad k = 1, \ldots, p \tag{5.3}$$

$$d_{ik} \geq 0 \quad i, k = 1, \ldots, p \tag{5.4}$$

$$x \in \mathcal{X} \tag{5.5}$$

Modified E-MOIP(λ):

$$\min \sum_{k=1}^{p} \lambda_k s_k \tag{5.6}$$

$$s.t.\ r_k - d_{ik} \leq z_i(x) \quad i,k = 1,\ldots,p \tag{5.7}$$

$$\theta_k = kr_k - \sum_{i=1}^{p} d_{ik} \quad k = 1,\ldots,p \tag{5.8}$$

$$T^* + 1 - \theta_k = s_k \quad k = 1,\ldots,p \tag{5.9}$$

$$d_{ik} \geq 0 \quad i,k = 1,\ldots,p \tag{5.10}$$

$$x \in \mathcal{X} \tag{5.11}$$

Notice that SBA tries to minimize the negated objectives in modified E-MOIP(λ) instead of maximization, since ExA algorithm is designed for the minimization problems.

Let r^* be the image of the optimal solution of modified E-MOIP(λ). If r^* is not a new point, i.e. it is already in R, then R is added to $\mathcal{F}$ as a facet-defining stage. If not, the algorithm defines p new stages, updates $\mathcal{L}$ as $\mathcal{L} = \mathcal{L} \cup \{\{r^1,\ldots,r^{p-1},r^*\},\{r^1,\ldots,r^*,r^p\},\ldots,\{r^*,\ldots,r^{p-1},r^p\}\}$ and adds r^* to the $Y_E^{'}$ (lines 9-16 of Algorithm 2).

If λ has a negative component, the algorithm finds a $y \in (Y_E^{'} \cup Y_M) \backslash R$ and $\lambda^{'} \geq 0$ such that $\lambda^{'} y \leq \lambda^{'} r$ for all $r \in R$. Then it sets $r^* = y$ and defines the p new stages accordingly (line 18 of Algorithm 2).

At the end of each iteration, the algorithm updates $\mathcal{L}$ as $\mathcal{L} = \mathcal{L} \setminus R$ and stops when $\mathcal{L} = \emptyset$ (lines 20-23 of Algorithm 2). Then, SBA proceeds to the split part by using the $Y_E^{'}$.

5.1.2 Finding the Remaining ENs

After finding all ESNs of the problem as $Y_E^{'}$, SBA algorithm utilizes a split algorithm to find the remaining ENs in the second part. The split algorithm divides

objective function space into p dimensional regions called boxes and searches non-dominated points in these boxes [38]. As it finds new points, it generates new boxes according to the point found and remove the boxes that are already searched or redundant.

Let $\mathcal{B}$ be the set of boxes. For a box B_i, B_i^l and B_i^u denote the lower and upper corner points of the box respectively. At first $\mathcal{B} = \emptyset$. Then, SBA constructs an upper reference point as $z_i^U = \max_{y \in Y_E^{'}} y_i \ \forall i \in \{1, ..., p\}$ is the maximum value of the i^{th} objective among the all points $Y_E^{'}$ and the lower reference point as $z^L = (0, 0, \ldots, 0)$ (line 24 of Algorithm 2). SBA creates the first box as B which has $B^l = z^L$ and $B^u = z^U$ (line 25 of Algorithm 2). B simply denotes the whole feasible region. For the following steps, SBA uses "isredundant", "CheckSolve" and "NewBoxes" procedures introduced by Karsu and Ulus [38]. First, SBA updates $\mathcal{B} = \{B\}$ (line 25 of Algorithm 2). After that, it generates new boxes according to the ESNs found in the first part(lines 26-29 of Algorithm 2). For a given point z and set of boxes $\mathcal{B}$, $NewBoxes(\mathcal{B}, z)$ (Algorithm 5) generates new boxes and the boxes to be deleted such that after the deletion the remaining boxes are distinct and not dominated by the point z. SBA uses this procedure for all the ESNs in $Y_E^{'}$. It generates new boxes using the ESNs found and removes any dominated boxes such that each box corresponds a distinct feasible region that is not dominated by ESNs or not dominate the ESNs.

After that, SBA selects the first box from the list $\mathcal{B}$ and checks whether we need to solve Modified Split(B) model for that box (lines 31-32 of Algorithm 2). Let $PrevBox$ be the list of the boxes searched and their respective non-dominated points (if there is one) such as (B_i, z_i). For a selected box B, $CheckSolve(B, PrevBox)$ (Algorithm 4) checks for a box $B^{'}$ in $PrevBox$ such that $B^{'} \supseteq B$. If there exists such box and it contains no non-dominated points, there is no need to search B since it is subset of $B^{'}$. On the other hand, if there exists such a box and contains a non-dominated point that is also in $B^{'}$. There is also no need for searching B. If non of these cases hold, then for B, SBA solves the modified split model below (lines 33-41 of Algorithm 2). This model also maximizes the cumulative ordered weighted average. Therefore, it gives an EN by Proposition 1.

Modified Split(B):

$$\max \sum_{k=1}^{p} \lambda_k \theta_k \tag{5.12}$$

$$s.t.\ r_k - d_{ik} \leq z_i(x) \quad i,k = 1,\ldots,p \tag{5.13}$$

$$\theta_k = k r_k - \sum_{i=1}^{p} d_{ik} \quad k = 1,\ldots,p \tag{5.14}$$

$$B^l \leq \theta_k \leq B^u \quad k = 1,\ldots,p \tag{5.15}$$

$$d_{ik} \geq 0 \quad i,k = 1,\ldots,p \tag{5.16}$$

$$x \in \mathcal{X} \tag{5.17}$$

Let $x^{B'}$ the solution of the modified Split(B') and $n^{B'}$ be the respective cumulative ordered objectives vector. $n^{B'} = \emptyset$ if the model is infeasible. Then SBA updates $PrevBox$ as $PrevBox = PrevBox \cup \{B', n^{B'}\}$. After that, for a feasible model, SBA adds $n^{B'}$ to the $Y_E^{'}$ and generate new boxes using the $NewBoxes(n^{B'}, \mathcal{B})$ procedure (lines 42-45 of Algorithm 2). Finally, it deletes the box searched from the box list $\mathcal{B}$ (line 46 of Algorithm 2). SBA selects another box from $\mathcal{B}$, solves the Modified Split model for that box and follows the same steps. Finally, when there is no box left in $\mathcal{B}$, SBA stops and returns all equitably non-dominated points found as $Y_E = Y_E^{'}$ (line 48 of Algorithm 2).

5.2 Split Only Version of SBA

Split Only Version of SBA (SOV) is also developed for E-MOIP problems with $p \geq 3$. To perform faster, SOV does not use the ExA module and hence does not find ESNs. Instead, it calculates the upper and lower reference points as in SBA and defines the first box using these two points as upper and lower corner points respectively. After that, it follows lines 30-48 of Algorithm 2 and returns the set of ENs.

Algorithm 2 Split Based Algorithm

1: Solve $\max_{x\in\mathcal{X}} \sum_{i=1}^{p} z_i(x)$ and let the optimum value be T^*.
2: Let $Y_E' = \emptyset$, $k = 0$, $\mathcal{V} = \emptyset$, $\mathcal{F} = \emptyset$, $\mathcal{L} = \emptyset$.
3: Define p dummy variables as $m^i = M \times e_i, i = 1, \ldots, p$ where e_i is the i^{th} unit vector.
4: Update $\mathcal{L}$, $\mathcal{L} = \{\{m^1, \ldots, m^p\}\}$.
5: Select a stage $R = \{r^1, r^2, \ldots, r^p\} \in \mathcal{L}$ and set $\mathcal{V} = \mathcal{V} \cup \{R\}$.
6: Solve λ for $\lambda r^1 = \lambda r^2 = \cdots = \lambda r^p$.
7: **if** $\lambda \in \mathbb{R}^p_>$ **then**
8: Solve E-MOIP(λ) and let $r^* = (r_1^*, r_2^*, \ldots, r_p^*)$ be the optimal point.
9: **if** $r^* \in R$ **then**
10: Set $\mathcal{F} = \mathcal{F} \cup \{R\}$.
11: **else**
12: $\mathcal{L} = \mathcal{L} \cup \{\{r^1, \ldots, r^{p-1}, r^*\}, \{r^1, \ldots, r^*, r^p\}, \ldots, \{r^*, \ldots, r^{p-1}, r^p\}\}$ and $\mathcal{L} = \mathcal{L} - (\mathcal{L} \cap \mathcal{V})$.
13: **if** $r^* \notin Y_E^{'}$ **then**
14: $k = k + 1$, $y^k = r^*$ and $Y_E' = Y_E' \cup \{y^k\}$.
15: **end if**
16: **end if**
17: **else**
18: Find $y \in Y_{EM}^{'} \setminus R$ and $\lambda^{'} \in \mathbb{R}^p_>$ such that $\lambda^{'} y \leq \lambda^{'} r \ \forall r \in R$. Set $r^* = y$ and go to line 12.
19: **end if**
20: $\mathcal{L} = \mathcal{L} - R$.
21: **if** $\mathcal{L} \neq \emptyset$ **then**
22: Go to line 5.
23: **end if**
24: Let the upper reference point be $z^U = \{\max_{y\in Y_E^{'}} y_1, \max_{y\in Y_E^{'}} y_2, \ldots, \max_{y\in Y_E^{'}} y_p\}$ and the lower reference point be $z^L = (0, 0, \ldots, 0)$.
25: Define the first box as B with $B^l = z^L$ and $B^u = z^U$ and let $\mathcal{B} = B$, $PrevBox = \emptyset$
26: **for** $y \in Y_E^{'}$ **do**
27: [$\mathcal{B}$,D] = NewBoxes($\mathcal{B}, y$);
28: $\mathcal{B} = \mathcal{B} \setminus D$;
29: **end for**
30: **while** $\mathcal{B} \neq \emptyset$ **do**
31: Let B be the first box in $\mathcal{B}$ and $D = B$.
32: [n^B, *solve*] = CheckSolve(B,$PrevBox$)
33: **if** solve = 1 **then**
34: Solve Modified Split(B);
35: **if** The problem is feasible **then**
36: Let x^* be an optimal solution;
37: $n^B = z(x^*)$, PrevBox = PrevBox $\cup\{(B, n^B)\}$;
38: **else**
39: PrevBox = PrevBox $\cup\{B, \emptyset\}$;
40: **end if**
41: **end if**
42: **if** $n^B \neq \emptyset$ **then**
43: $Y_E^{'} = Y_E^{'} \cup \{n^B\}$
44: [$\mathcal{B}$,D] = NewBoxes($\mathcal{B}, n^B$);
45: **end if**
46: $\mathcal{B} = \mathcal{B} \setminus D$;
47: **end while**
48: $Y_E = Y_E^{'}$
49: **return** Y_E: The set of all equitably non-dominated points

Algorithm 3 isredundant($\bar{B}, \mathcal{B}, B$)

redundant = 0;
for $\hat{B} \in \mathcal{B} \setminus \{\bar{B}, B\}$ **do**
    if $(\bar{B}^l \geq \hat{B}^l)$ **then**
        ElimRed = ElimRed + 1, redundant = 1;
        break;
    end if
end for
if redundant = 1 **then**
    return TRUE;
else
    return FALSE;
end if

Algorithm 4 CheckSolve(B,PrevBox)

$n^B = \emptyset$, solve = 1;
for $(\hat{B}, \hat{n}) \in$ PrevBox **do**
    if $B^l \geq \hat{B}^l$ and $\hat{n} = \emptyset$ **then**
        solve = 0, ElimInf = ElimInf + 1;
        break;
    end if
    if $B^l \geq \hat{B}^l, \hat{n} \neq \emptyset$ and $\hat{n} \geq B^l$ **then**
        $n^B = \hat{n}$, solve = 0, ElimOpt = ElimOpt + 1;
        break;
    end if
end for
return [n^B, solve].

Algorithm 5 NewBoxes($\mathcal{B}$,n^B)

$D = \emptyset$, Btemp $= \mathcal{B}$;
for $\mathbf{B} \in \mathcal{B}$ **do**
  if $n^B \geq \mathbf{B}^l$ **then**
    $D = D \cup \{\mathbf{B}\}$;
    for $j \in \{1, \ldots, p\}$ **do**
      if $z_j^I - n_j^B \geq 1$ **then**
        $\bar{\mathbf{B}}_i^l = \mathbf{B}_i^l$ for $i \neq j$ and $\bar{\mathbf{B}}_j^l = n_j^B - \epsilon$
        Let $\bar{\mathbf{B}}$ be a box with lower corner $\bar{\mathbf{B}}^l$.
        if not isredundant($\bar{\mathbf{B}}, \mathcal{B}, \mathbf{B}$) **then**
          Btemp = Btemp $\cup \{\bar{\mathbf{B}}\}$ ($\bar{\mathbf{B}}$ is added to the end of Btemp);
        end if
      end if
    end for
  end if
end for
$\mathcal{B}$ = Btemp;
return $[\mathcal{B}, D]$: Updated set of boxes to be searched and boxes to be deleted.

Chapter 6

Computational Experiments

In this section, we observe the performance of the SBA algorithm on multiobjective binary knapsack and multiobjective assignment problems. Moreover, we compare SBA with SOV. Finally, we discuss the weight space decomposition for the ESNs found.

For all computational experiments, the algorithms are coded in MATLAB R2022a and models are solved using CPLEX V12.10.0 by a computer with Intel(r) Xeon(r) CPU E5-1650 v4 @ 3.60GHz Processor and 64 GB RAM. For each run, the CPU time is recorded and we report the following:

- **# of ESNs:** the number of extreme supported equitably non-dominated points found
- **# of ENs:** the total number of equitably non-dominated points found
- **# of Models per ESN:** the number of models needed to be solved for finding one ESN
- **Avg. CPU Time per ESN:** the average CPU time needed for finding one ESN

- **Last ESN CPU Time:** the total CPU time elapsed until the last ESN is found
- **First Part CPU Time:** the total CPU time elapsed until the end of the first part or reaching the time limit (if there is one)
- **# of Models in Split:** the total number of models solved in the split part
- **Avg. Split CPU Time:** the average CPU time for solving one split model
- **Completion CPU Time:** the total CPU time elapsed from beginning to end

6.1 Multiobjective Assignment Problem

For the multiobjective assignment problem (AP) we use the same instances used in [15]. In these problems, there are n agents, n tasks and $p = 3$. AP is formulated as below and our aim is to find all equitably non-dominated points of this problem.

Multiobjective Assignment Problem:

$$\min \sum_{r=1}^{n}\sum_{l=1}^{n} c_{jrl}x_{rl} \quad j = 1,\ldots,p \tag{6.1}$$

$$s.t. \sum_{l=1}^{n} x_{rl} = 1 \quad r = 1,\ldots,n \tag{6.2}$$

$$\sum_{r=1}^{n} x_{rl} = 1 \quad l = 1,\ldots,n \tag{6.3}$$

$$x_{rl} \in \{0,1\} \quad r,l = 1,\ldots,n \tag{6.4}$$

$$x_{rl} = \begin{cases} 1, & \text{if agent } r \text{ is assigned to task } l \\ 0, & \text{otherwise} \end{cases} \tag{6.5}$$

where $r,l = 1,\ldots,n$. c_{jrl} is the non-negative cost of assigning agent r to task l for the objective j, $j = 1,\ldots,p$ and $r,l = 1,\ldots,n$. c_{jrl}s are integers that are

generated randomly from the interval [1,20]. In these APs, $n \in \{5, 10, 15, \ldots, 50\}$ and for each n there are 10 different instances and n^2 variables in each instance. We have the number of non-dominated points for most of the AP instances and it can be seen in Table 6.1.

Table 6.1: Average # of non-dominated points of AP instances

p	# of Variables	Avg. # of non-dominated points
3	225	674.9
	400	1860.5
	625	3567.8
	900	6181.3
	1225	8972.5
	1600	14679.7
	2025	17702.3
	2500	24916.9

6.1.1 Performance of SBA Algorithm on AP

Table 6.2: Results of SBA algorithm on AP

p	# of Variables	Average								
		# of ESNs	# of ENs	# of Models per ESN	Avg. CPU Time per ESN	Last ESN CPU Time	First Part CPU Time	# of Models in Split	Avg. Split CPU Time	Completion CPU Time
3	25	2	2	1.749	0.072	0.204	1.436	1	0.01	1.505
	100	2.6	2.6*	1.899	0.096	0.334	4.025	1.4	0.019	4.113
	225	3.5	4.6*	2.433	0.125	0.609	11.403	7.4	0.044	11.917
	400	3.9	6.2	2.428	0.19	0.877	22.434	12	0.084	23.551
	625	2.8	3.7	1.83	0.194	0.589	1.55	5.4	0.108	2.31
	900	3.5	5	2.265	0.248	0.893	2.574	7.3	0.170	3.893
	1225	3.6	5.5*	1.951	0.318	1.2	3.031	8.9	0.209	5.112
	1600	3.3	5.2	2.48	0.361	1.364	6.783	7.5	0.235	8.728
	2025	3.6	5.3*	3.176	0.379	1.817	43.554	7.9	0.187	46.297
	2500	3.8	5.9	2.028	0.52	2.045	5.437	9.7	0.397	9.615

For these instances of AP, according to the Table 6.2 we can see that, although the number of ESNs is not affected by the number of variables, the total number of ENs slightly increase with the increasing number of variables. Moreover, the problem size does not have a direct effect on the average number of models solved per ESN. However, as the number of variables increases the average CPU time per ESN also increases since there are more variables to consider while solving

the model. SBA finds all the ESNs quickly but it cannot complete checking all stages to ensure that no ESN is left that fast. On average SBA spends only 10% of the time spent in ExA part for finding all ESNs. The remaining 90% percent of the time is spent for searching other stages which can be considered as the verification process so that SBA ensures that all ESNs are found. The time required to find all ESNs increases with the increasing number of variables. We can conclude that although SBA finds all ESNs in a relatively short time, it takes much more time to ensure that it searches all possible stages and be sure that there is no other ESN.

For the split part, the number of models solved in split increases with the number of variables since more variables leads to more ENs. The CPU time for solving a split model also increases with the number of variables for the same reason as mentioned above.

Completion times are generally increasing with the number of variables but there are some outliers in the instances. For instance, for the variable size of 400 the average completion CPU time is 23.551 but for the variable size of 2500 the average completion CPU time is 9.615.

The "*"s in the "# of ENs" column denote that for some instances, the algorithm could not find all ENs for the given variable size. For the variable size of 100, there is one missing solution in two different instances and for each variable sizes of 225, 1225, 2025, there is one missing solution for one instance. The underlying reason for this situation is probably the numerical representation in the memory because one of the objectives of missed solutions are always close to the same objective value of one of the solutions found.

To conclude, for this AP problem and instances, although there are some outliers, number of ENs, number of models solved and CPU times increase as the number of variables increases when SBA is used.

6.1.2 Performance of SOV Algorithm on AP

SOV is also implemented on the same AP instances to compare the results with SBA. The result can found in Table 6.3.

Table 6.3: Results of SOV algorithm on AP

p	# of Variables	Average			
		# of ENs	# of Models in Split	Avg. Split CPU Time	Completion CPU Time
3	25	2	5.8	0.006	0.126
	100	2.8	7.9	0.023	0.225
	225	4.7	13.2	0.035	0.526
	400	6.2	16.9	0.063	1.094
	625	3.7	10.2	0.080	0.863
	900	4.8*	13	0.130	1.71
	1225	5.6	15.1	0.189	2.986
	1600	5.2	13.7	0.222	3.195
	2025	5.4	14.3	0.264	4.3
	2500	5.8*	15.7	0.411	6.806

First of all, similar to SBA results, as the variable size increases,

- the number of ENs found
- the CPU time required for solving one split model,
- the CPU time elapsed until the end

increase. The number of split models solved is not directly affected by the number of variables for these instances. For each variable size, SOV takes only few seconds. However, for $n = 30$ and $n = 50$, SOV missed two and one solutions respectively. It is probably from working with small numbers and their rounding processes and comparisons. Most importantly, although the completion times are increasing with the number of variables, they are only a couple seconds on the average. In SOV, the verification of deciding whether the given box is empty or not happens more efficiently. The reason is that in the split part, the algorithm keeps

the information gathered from the boxes already searched and uses "CheckSolve" procedure to decide whether the box should be searched or not. Moreover, when generating new boxes, "isredundant" procedure helps eliminating the boxes that are dominated or infeasible.

6.2 Multiobjective Binary Knapsack Problem

For the multiobjective binary knapsack problem (KP) we use the same instances used in [15]. In these problems, there are n objects and each object has a positive integer weight w_r and non-negative integer profit v_{jr}, $r = 1, \ldots, n$, $j = 1, \ldots, p$. KP is formulated as below and our aim is to find all equitably non-dominated points of this problem.

Multiobjective Binary Knapsack Problem:

$$\max \sum_{r=1}^{n} v_{jr} x_r \quad j = 1, \ldots, p \tag{6.6}$$

$$s.t. \sum_{r=1}^{n} w_r x_r \leq W \tag{6.7}$$

$$x_r \in \{0, 1\} \quad r = 1, \ldots, n \tag{6.8}$$

$$x_r = \begin{cases} 1, & \text{if item } r \text{ is selected} \\ 0, & \text{otherwise} \end{cases} \tag{6.9}$$

where $r = 1, \ldots, n$. v_{jr} and w_r are integers that are generated randomly from the interval [1,1000] and the capacity W is set as $0.5 \times \sum_{r=1}^{n} w_r$. In these KP instances, $p \in \{3, 4, 5\}$ and there different variables sizes for each p. For $p = 3$, $n \in \{10, 20, \ldots, 100\}$.For $p = 4$, $n \in \{10, 20, 30, 40\}$. For $p = 5$, $n \in \{10, 20\}$. For each p and n pair, there are 10 instances. We have the number of non-dominated points for most of the KP instances and it can be seen in Table 6.4.

Table 6.4: Average # of non-dominated points of KP instances

p	# of Variables	Avg. # of non-dominated points
3	50	444.2
	60	917.1
	70	1643.4
	80	2295.8
	90	3107.8
	100	5849
4	10	11.6
	20	136.8
	30	397.6
	40	1808.6
5	10	16.2
	20	161.2

6.2.1 Performance of SBA Algorithm on KP

Similar to the AP instances, for the KP instance we see that for a fixed p, as the number of variables increases,

- the number of ENs found,
- the average average CPU time required to find one ESN,
- the CPU time elapsed until all ESNs are found,
- the CPU time elapsed in the first part,
- the number of models solved in split part,
- the average CPU time required for solving one split model,
- the CPU time elapsed until the end

also increase and it can be seen from Table 6.5. In addition to these, the number of ESNs and the number of models solved for finding one ESN also increase with the number of variables.

Table 6.5: Results of SBA algorithm on KP

p	# of Variables	Average # of ESNs	# of ENs	# of Models per ESN	Avg. CPU Time per ESN	Last ESN CPU Time	First Part CPU Time	# of Models in Split	Avg. Split CPU Time	Completion CPU Time
3	10	2.3	2.7	1.79	0.053	0.139	0.824	3.8	0.015	0.946
	20	3.6	4.7	2.279	0.058	0.239	3.818	8.5	0.024	4.076
	30	4.7	8.6	3.597	0.086	0.502	104.713	19.1	0.024	105.318
	40	5.8	14.3*	3.605	0.098	0.662	125.568	34.8	0.035	126.807
	50	5.9	13.4	5.934	0.155	1.185	479.946**	32.3	0.037	481.223
	60	9.1	27*	11.698	0.725	12.397	1227.64**	69.4	0.047	1231.297
	70	9.1	31.9*	16.887	5.509	113.574	1090.752**	83.9	0.056	1095.957
	80	7.9	29.2*	39.464	19.419	250.37	899.118**	77.1	0.067	904.856
	90	13.3	49*	66.306	22.728	355.644	2725.666**	129.9	0.078	2736.381
	100	12.7	74.6*	36.582	13.455	263.756	1953.744**	207.1	0.087	1976.89
4	10	1.8	2.1	1.734	0.11	0.214	0.612	4	0.017	0.866
	20	3	4.2	2.508	0.106	0.354	3.306	13.2	0.033	3.858
	30	4.9	10.4	4.722	0.11	0.596	626.368**	42.6	0.033	627.986
	40	8.2	28	53.366	30.236	399.89	2554.702**	132.1	0.045	2561.171
5	10	2.6	2.8	1.7	0.07	0.187	1.525	8.2	0.023	1.744
	20	4.5	7.4	22.456	21.252	275.578	920.011**	43.3	0.030	921.308

The "*" in "# of ENs" column show the p and variable size pairs that SBA find either one more solution or one less solution. SBA misses an ESN in only one instance. It is due to the time limit since SBA stops after reaching the time limit for the first part. For seven instances, SBA returned one more EN. However, when we checked the EN list, we figured out that the additional point found is almost identical to one of the ESNs, hence this is due to rounding process.

We also observed that in SBA, although it finds all ESNs relatively quicker, the search for ensuring there is no ESN left took much more time. Finding the non-extreme ENs are also quick in SBA. Similar to the AP instances, SBA spends only 13% of time spent in the first part for finding ESNs and remaining time is spent for verification of that all ESNs are found. Therefore, for this KP instances, we put a time limit for the first part of SBA as 3600 seconds since searching stages for verification takes much time. Although SBA finds all the ESNs before reaching time limit except one instance, in some instances verification process could not be completed within the time limit. For these instances, we continue with the all ESNs found within the time limit and continue to second part as usual. The "**" in the "First Part CPU Time" column shows the p and variable size pairs that has at least one instance exceeding the time limit. There are total such 29 instances and as the variable size increases, the number of such instances also increase. For

example, for $p = 3$ although there is only one instance that exceeds time limit for the variable size of 50, there are seven such instances for the variable size of 90. This observation makes sense since the number of ESNs increases with the variable size, more ESNs leads longer CPU times both for finding all ESNs and verification.

To conclude, for this KP problem and instances, all statistics collected increase with increasing number for variables within the same p when SBA is performed. Moreover, as the number of ESNs increases, the number of instances that exceeding the given limit also increases due to the fact that SBA searches more stages for the verification.

6.2.2 Performance of SOV Algorithm on KP

We also run the SOV on the same KP instances to compare the results with SBA. The result can found in Table 6.6.

First of all, similar to SBA results, for a fixed p, as the variable size increases,

- the number of ENs found
- the number of split models solved,
- the average CPU time required for solving one split model,
- the CPU time elapsed until the end

also increase. Similar to the AP result of SOV, for KP instances, SOV takes only few seconds on average since its verification process is more efficient.

Table 6.6: Results of SOV algorithm on KP

p	# of Variables	Average			
		# of ENs	# of Models in Split	Avg. Split CPU Time	Completion CPU Time
3	10	2.7	8.1	0.017	0.18
	20	4.7	14.1	0.018	0.284
	30	8.6	25.8	0.023	0.837
	40	14.2	42.5	0.030	1.312
	50	13.4	40.2	0.030	1.237
	60	26.9	80.5	0.042	3.575
	70	31.9	95.2	0.050	5.083
	80	29.1	86.9	0.055	5.446
	90	48.8	145	0.067	10.146
	100	74.5	221.8	0.085	22.327
4	10	2.1	8.8	0.007	0.078
	20	4.2	19.2	0.017	0.366
	30	10.4	50.5	0.023	1.354
	40	28	143.4	0.034	5.518
5	10	2.8	14.8	0.010	0.163
	20	7.4	51.8	0.021	1.1

6.3 Comparison of SBA and SOV

Both algorithms manage to find all ENs in general. SBA finds the ESNs first and keep their corresponding weight information that is used for finding these ESNs. Although this information is valuable as it is explained in section 6.5, verifying that there is no ESN is left takes much more time than finding all ESNs. Hence, SOV performs so much faster than the SBA for these KP instances. However, SOV cannot identify which points are ESN without performing post-processing.

As we mentioned, ensuring there is no more ESN left takes majority of the time spent in the first part. Moreover, the weight space information is useful since all ENs are presented to the DM and DM needs to pick one to implement. If DM adopts a preference among the cumulative ordered objective or ordered objectives, she can easily find the point that fits best for their preference. Hence, if there is a strict time limitation SOV should be used but if there is no such

limitation SBA also can be used since it also gives the information about ESNs and their corresponding weight space.

6.4 Performance of the Algorithm on Biobjective Binary Knapsack Problem

In Chapter 4 we developed an algorithm for E-BOIP problems. In order to test the performance of this algorithm, we use the same multiobjective knapsack problem instances for $p = 3$ and consider only the first two objectives. The result of the runs can be seen in Table 6.7.

Table 6.7: Results of the algorithm on KP for two objectives

		Average			
p	# of Variables	# of ENs	# of P Models	# of Q Models	CPU Time
2	10	1.6	3.6	1.6	0.045
	20	2.4	4.4	2.6	0.073
	30	3.5	5.5	3.6	0.136
	40	4.3	6.3	4.6	0.120
	50	5.3	7.3	5.5	0.156
	60	8.1	10.1	8.3	0.432
	70	7.4	9.4	7.5	0.297
	80	7.5	9.5	7.7	0.480
	90	15.5	17.5	15.6	0.848
	100	17.7	19.7	17.9	1.392

As the number of variables increase,

- the number of ENs found,
- the number of P models solved
- the number of Q models solved
- CPU time (in seconds)

also increase. The number of P models solved is greater than Q models solved since P models return weakly equitably non-dominated points and Q models use these points to return ENs. Similarly the number of ENs found is bounded by the number of Q models solved. Solution times are less than one second on average except the case of 100 variables. Hence, the algorithm performs efficiently for E-BOIP problem.

6.5 Weight Space Decomposition

We know that each ESN corresponds to a region in weight space such that it maximizes the sum of weighted cumulative ordered objectives for all the weights in that region. Recall that, our objectives are the cumulative ordered objectives in the models solved in the both algorithms. Let $\lambda_1, \lambda_2, \ldots, \lambda_p$ be the non-negative weights that sum up to 1. Then, λ_p can be written as $\lambda_p = 1 - \sum_{i=1}^{p-1} \lambda_i$. Hence, for the given p objectives, the weight space has the dimension of $p - 1$.

To explain the weight space better, we use $p = 3$ and consider a maximization problem. In this setting we have three weights as (λ_1, λ_2, λ_3), where $\lambda_3 = 1 - \lambda_1 - \lambda_2$, and the corresponding weight space can be seen in Figure 6.1. All possible values of weight space is shown as the shaded area. λ_1 values lie in the horizontal axis while λ_2 values lie in the vertical axis. The shaded area is bounded by the $\lambda_1 + \lambda_2 = 1$ line. On this line, value of $\lambda_3 = 0$ since $\lambda_1 + \lambda_2 = 1$. Similarly, $\lambda_1 = 0$ and $\lambda_2 = 0$ on the vertical and horizontal axes respectively.

We also know that λ_k corresponds to objective of $\sum_{i=1}^{k} \overrightarrow{z}_i$ for all $k = 1, \ldots, 3$. Therefore, each corner of the weight space corresponds a point that is the best in terms of different preferences among the objectives. For instance, when the weight is near the origin such that both λ_1 and λ_2 are close to 0, λ_3 becomes significantly larger and is close to 1. For that weight, the corresponding ESN is the one that maximizes the objective of $\sum_{i=1}^{3} \overrightarrow{z}_i$. In addition to this, this weight corresponds to the utilitarian solution since it focuses on maximizing the sum of all z_is for $i = 1, \ldots, p$, which is the sum of all objectives of MOIP. When

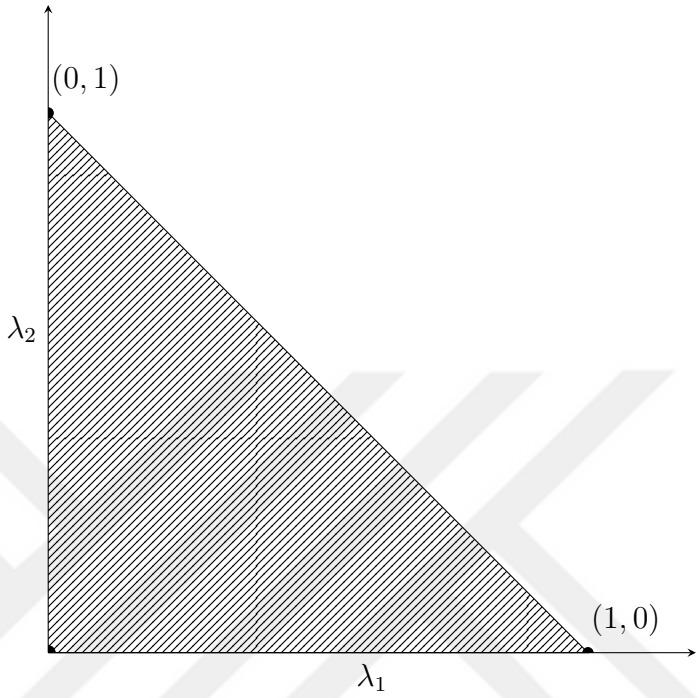


Figure 6.1: Weight space for $\theta_1(z)$, $\theta_2(z)$ and $\theta_3(z)$

λ_1 is close to 1 and the other weights are close to 0, the corresponding ESN for this weight is the one that focuses on maximizing the minimum objective $\overrightarrow{z}_1$. Moreover, this weight corresponds to the Rawlsian solution since the respective ESN has the highest $\overrightarrow{z}_1$ among the other ESNs.

We can also gather information about the preference among the ordered objectives by using the weight space information of ESNs. The ordered objectives are $\overrightarrow{z}_1$, $\overrightarrow{z}_2$ and $\overrightarrow{z}_3$. For the given weight (λ_1,λ_2,λ_3), we know the cumulative ordered weighted sum is $\sum_{i=1}^{3} \lambda_i \theta_i(z)$ which equals $\sum_{i=1}^{3} \lambda_i (\sum_{k=1}^{i} \overrightarrow{z}_k)$. If we arrange the terms, then we get the sum in terms of the ordered objectives which is $(\lambda_1 + \lambda_2 + \lambda_3)\overrightarrow{z}_1 + (\lambda_2 + \lambda_3)\overrightarrow{z}_2 + \lambda_3 \overrightarrow{z}_3$. Notice that the sum of weights of the weighted ordered sum is greater than 1 since $\sum_{i=1}^{3} \lambda_i = 1$. In order to normalize the weight ($\lambda_1 + \lambda_2 + \lambda_3$,$\lambda_2 + \lambda_3$,$\lambda_3$), we can divide it by their sum which is $\lambda_1 + 2\lambda_2 + 3\lambda_3$. Let ω be the normalized non-negative weight vector for ordered objectives such that $\omega_1 = \frac{\lambda_1+\lambda_2+\lambda_3}{\lambda_1+2\lambda_2+3\lambda_3}$, $\omega_2 = \frac{\lambda_2+\lambda_3}{\lambda_1+2\lambda_2+3\lambda_3}$ and $\omega_3 = \frac{\lambda_3}{\lambda_1+2\lambda_2+3\lambda_3}$.

The weight space for the ordered objectives can be seen in Figure 6.2 as the

shaded area. Notice that, the region is different from the Figure 6.1. To address this difference, let us find the bounds for each ω_i for $i = 1, \ldots, p$.

For ω_1:

- **Upper bound:** Since $\omega_1 = \frac{\lambda_1+\lambda_2+\lambda_3}{\lambda_1+2\lambda_2+3\lambda_3}$, in order to maximize ω_1, λ_1 needs to be 1 and for that case $\omega_1 = 1$ and $\omega = (1, 0, 0)$.
- **Lower bound:** Since $\omega_1 = \frac{\lambda_1+\lambda_2+\lambda_3}{\lambda_1+2\lambda_2+3\lambda_3}$, in order to minimize ω_1, λ_3 needs to be 1 and for that case $\omega_1 = \frac{1}{3}$ and $\omega = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

For ω_2:

- **Upper bound:** Since $\omega_2 = \frac{\lambda_2+\lambda_3}{\lambda_1+2\lambda_2+3\lambda_3}$, in order to maximize ω_2, λ_2 needs to be 1 and for that case $\omega_2 = \frac{1}{2}$ and $\omega = (\frac{1}{2}, \frac{1}{2}, 0)$.
- **Lower bound:** Since $\omega_2 = \frac{\lambda_2+\lambda_3}{\lambda_1+2\lambda_2+3\lambda_3}$, in order to minimize ω_2, λ_1 needs to be 1 and for that case $\omega_2 = 0$ and $\omega = (1, 0, 0)$.

For ω_3:

- **Upper bound:** Since $\omega_3 = \frac{\lambda_3}{\lambda_1+2\lambda_2+3\lambda_3}$, in order to maximize ω_3, λ_3 needs to be 1 and for that case $\omega_3 = \frac{1}{3}$ and $\omega = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.
- **Lower bound:** Since $\omega_3 = \frac{\lambda_3}{\lambda_1+2\lambda_2+3\lambda_3}$, in order to minimize ω_3, λ_3 needs to be 0 and for that case $\omega_3 = 0$ and $\omega = (w_1, 1 - w_1, 0)$.

In addition to these bounds, there is a hierarchy among them such that $\omega_1 \geq \omega_2 \geq \omega_3$ by their definition. Recall that $\omega_3 = 1 - \omega_1 - \omega_2$. Then $\omega_2 \geq \omega_3$ is equal to $\omega_2 \geq 1 - \omega_1 - \omega_2$ which is $\omega_2 \geq \frac{1-\omega_1}{2}$. Hence, these bounds result in a triangular region.

An example of weight space decomposition of a KP instance can be found below. It is from the data set where $p = 3$, variable size is 80 and the instance is 9.

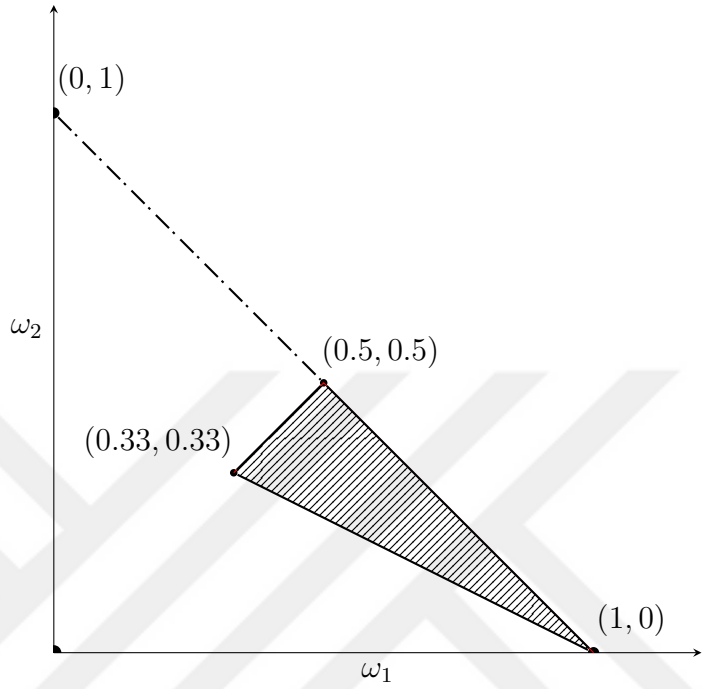


Figure 6.2: Weight space for $\overrightarrow{z_1}(x)$, $\overrightarrow{z_2}(x)$ and $\overrightarrow{z_3}(x)$

Example:

We have the following ESNs with their ordered objective values and cumulative ordered objective values.

- z^1**:** $(\overrightarrow{z}^1_1, \overrightarrow{z}^1_2, \overrightarrow{z}^1_3) = (30431, 30736, 30847) \quad \theta(z^1) = (30431, 61167, 92014)$
- z^2**:** $(\overrightarrow{z}^2_1, \overrightarrow{z}^2_2, \overrightarrow{z}^2_3) = (29400, 31583, 31814) \quad \theta(z^2) = (29400, 60983, 92797)$
- z^3**:** $(\overrightarrow{z}^3_1, \overrightarrow{z}^3_2, \overrightarrow{z}^3_3) = (30554, 30617, 30674) \quad \theta(z^3) = (30554, 61171, 91845)$
- z^4**:** $(\overrightarrow{z}^4_1, \overrightarrow{z}^4_2, \overrightarrow{z}^4_3) = (29912, 30985, 31679) \quad \theta(z^4) = (29912, 60897, 92576)$

The weight space for the cumulative ordered objectives can be seen in Figure 6.3. Firstly, we can see that for z^2, in its region λ_1 and λ_2 are low which leads a higher λ_3 value. Therefore, z^2 maximizes the sum of objectives which is $\theta_3(z)$ and it is the utilitarian solution. Similarly, the region of z^3 is located where λ_3 is low. Moreover, z^3 is the best point when the weight is (1,0,0) and (0,1,0) as z^3

maximizes both the minimum objective and the sum of the first two minimum objectives, which are $\theta_1(z)$ and $\theta_2(z)$, respectively. z^3 is also the Rawlsian solution since it maximizes the worst objective. z^1 and z^4 are not the points that maximize an objective directly. Instead they are the best in the area between z^2 and z^3.

To sum up, from the Figure 6.3 the DM can pick the point that suits her preference among the cumulative ordered objectives.

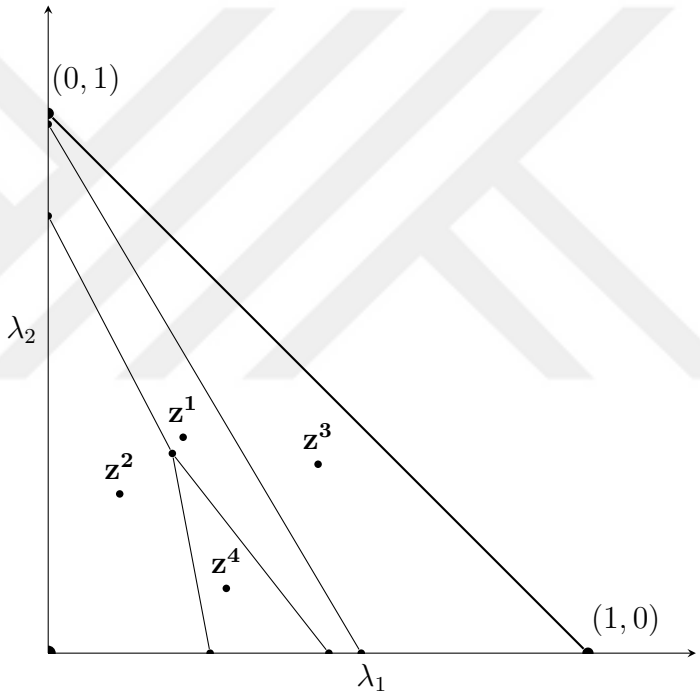


Figure 6.3: Weight space decomposition of the example for cumulative ordered objectives

The weight space for the ordered objectives can be seen in Figure 6.4 and its zoomed version can be found in Figure 6.5. In this weight graph, DM can determine which ESN leads to the best point according to her preference among the ordered objectives. In other words, if the DM wants to maximize the minimum objective, she should pick z^3 since it is the best when ω_1 is 1. Similarly, in the case of $\omega_1 = \omega_2 = 0.5$, z^3 is the best point as it maximizes the minimum objective and the second objective by giving them equal priority. Recall that in Figure 6.3, z^3 gives the best $\theta_1(z)$ and $\theta_2(z)$ values. Moreover, $\theta_1(z) = \overrightarrow{z}_1$ and $\theta_2(z) = \overrightarrow{z}_1 + \overrightarrow{z}_2$. Hence, we can see that there is an one to one correspondence

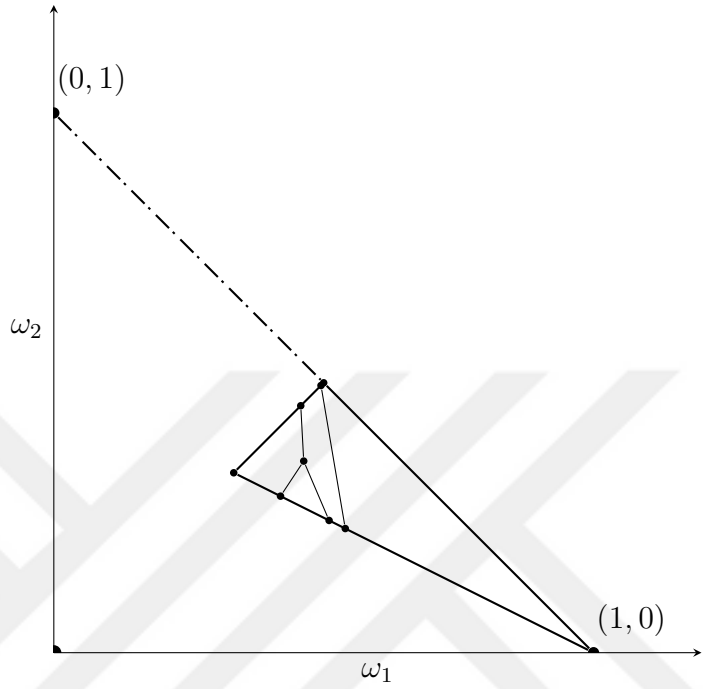


Figure 6.4: Weight space decomposition of example for ordered objectives

between the weight of cumulative ordered objectives and the weight of the ordered objectives. Similarly, recall that z^2 maximizes $\theta_3(z) = \sum_{i=1}^{3} \overrightarrow{z}_i$ and the weight corresponds to this value is $(\lambda_1, \lambda_2, \lambda_3) = (1, 1, 1)$. Its equivalent ω is $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Hence, we can see that when $\omega = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, z^2 is the respective non-dominated point. Similarly, the regions dominated by z^1 and z^4 lie in between the regions dominated by z^2 and z^3.

To sum up, from Figure 6.5 the DM can pick the point that suits her preference among the ordered objectives.

To conclude the weight space decomposition part, knowing which points are ESN is valuable since it helps us to construct the weight space for both cumulative ordered objectives and ordered objectives. For the cumulative ordered objectives, the corresponding weight space decomposition helps DM to determine which point is the best in terms of her preference among the cumulative ordered objectives. For instance, a DM with a utilitarian preference should pick the point corresponds to the weight of (0,0,1) which is the origin.

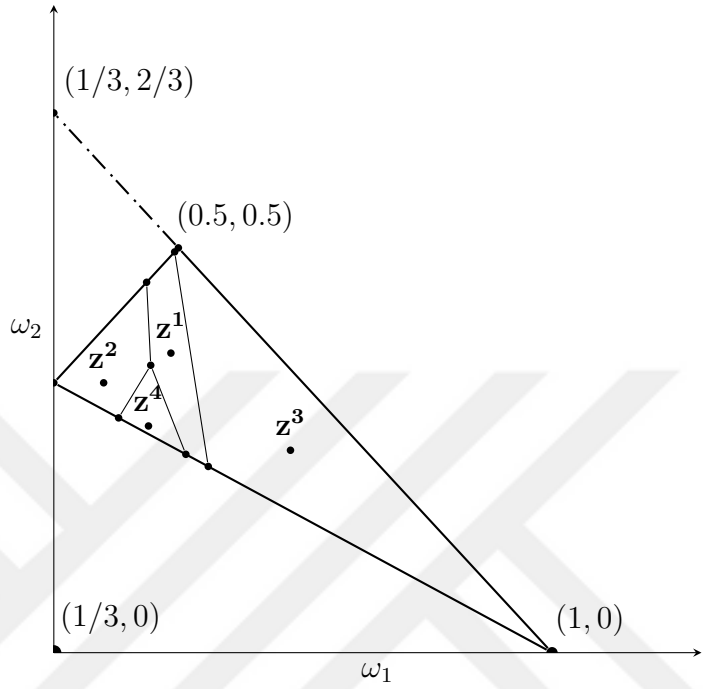


Figure 6.5: Zoomed weight space decomposition of the example for ordered objectives

Similarly, for the ordered objectives, DM is able to determine which point is the best in terms of their weight on ordered objectives. If the DM wants to maximize the worst objective and has a Rawlsian preference, she can look for the weight (1,0,0). However, if the DM also wants to maximize the second worst objective, she should pick the point corresponds to the weight (0.5,0.5,0).

In addition to these, the size of the areas for each ESN in both weight spaces also give useful information. Larger area corresponds to an ESN that is less sensitive to change in weights. Hence, the DM can use this area size information as a way of sensitivity analysis.

Therefore, thanks to the weight space decomposition, although the DM has no preference among the objectives at the beginning, if the DM decides to adopt a preference among the ordered or cumulative ordered objectives, she is free to look for and pick the ESN fits best for their preference.

Chapter 7

Conclusion and Future Research

We first introduce the algorithm for E-BOIP problems. For this algorithm we customize PS scalarization by defining new reference points and direction vectors as the algorithm processes. We first define a feasible region by constructing an upper reference point. We solve PS scalarization in the region bounded by the upper reference point and solve a followup model to get an equitably non-dominated point. Then we work on the two sides of the perfect equality line and compare the points with each other. After iteratively solving PS scalarization models and followup models, we compare them and update reference points and follow same steps until we get infeasible models for both sides. Finally, we return all equitably non-dominated points of the given problem. We also provide a computational experiment of this algorithm for the biobjective binary knapsack problem and it successfully solves each instance in less than few seconds.

We developed split based algorithm for E-MOIP problems with more than two objectives by combining and modifying existing algorithms for equitable optimization problems. First we use cumulative ordered weighted average and ExA algorithm in order to find all extreme supported equitably non-dominated points of the given problem. Then, we define boxes according to ESNs found and use a split algorithm to find the remaining equitably non-dominated points. After that, SBA returns all equitably non-dominated points of the given problem. We

also implemented split only variant for comparison.

We use instances of multiobjective assignment and knapsack problems for computational experiment. We compared the performances of SBA and SOV. For both problems, SOV is quicker. Although, SBA is also fast and finishes in a few seconds for AP, it takes longer time to finish for KP. In some KP instances SBA reaches the given time limit of 3600 seconds. Therefore, we observe that although SBA finds ESNs in a reasonable time, the verification process of ensuring there is no ESN left takes most of the time. Both algorithms find all equitably non-dominated points of the given problems.

Although SBA has a long verification process, the information gathered by finding ESNs first is useful for weight space decomposition. By using the ESN information, we can graph the weight space for cumulative ordered objectives and the regions that is dominated by each ESN. We also explain the weight space for ordered objectives. DM can look at these graphs and see which ESN performs best in terms of which preference. Therefore, if DM wants to adopt a preference among ordered or cumulative ordered objectives, she can easily know which ESN corresponds to her preference by looking at the respective weight values.

For future research, PS scalarization can be studied for the equitable settings that have more than two objectives. Other scalarization approaches can also be studied for E-MOP problems.

Bibliography

[1] Ö. Özpeynirci and M. Köksalan, "An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs," *Management Science*, vol. 56, no. 12, pp. 2302–2315, 2010.

[2] O. Kelechi and H. Tokos, "An milp model for the optimization of hybrid renewable energy system," in *26th European Symposium on Computer Aided Process Engineering* (Z. Kravanja and M. Bogataj, eds.), vol. 38 of *Computer Aided Chemical Engineering*, pp. 2193–2198, Elsevier, 2016.

[3] B. A. Bashir, *Generating Evenly Distributed Equitably Efficient Solutions in Multi-Objective Optimization Problems.* PhD thesis, Bilkent Universitesi (Turkey), 2018.

[4] T. Denktaş, *Interactive Algorithms to Solve Biobjective and Triobjective Decision Making Problems.* PhD thesis, Bilkent Universitesi (Turkey), 2021.

[5] A. Przybylski, X. Gandibleux, and M. Ehrgott, "A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme," *INFORMS Journal on Computing*, vol. 22, no. 3, pp. 371–386, 2010.

[6] M. M. Kostreva, W. Ogryczak, and A. Wierzbicki, "Equitable aggregations and multiple criteria analysis," *European Journal of Operational Research*, vol. 158, no. 2, pp. 362–377, 2004.

[7] G. Mavrotas and D. Diakoulaki, "Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming," *Applied mathematics and computation*, vol. 171, no. 1, pp. 53–71, 2005.

[8] M. De Santis and G. Eichfelder, "A decision space algorithm for multiobjective convex quadratic integer optimization," *Computers & Operations Research*, vol. 134, p. 105396, 2021.

[9] C. Bazgan, H. Hugot, and D. Vanderpooten, "Solving efficiently the 0–1 multi-objective knapsack problem," *Computers & Operations Research*, vol. 36, no. 1, pp. 260–279, 2009. Part Special Issue: Operations Research Approaches for Disaster Recovery Planning.

[10] D.-W.-I. S. Gausemeier, I. K.-P. Jäker, *et al.*, "Multi-objective optimization of a vehicle velocity profile by means of dynamic programming," *IFAC Proceedings Volumes*, vol. 43, no. 7, pp. 366–371, 2010.

[11] J. Mahmoudimehr and P. Sebghati, "A novel multi-objective dynamic programming optimization method: Performance management of a solar thermal power plant as a case study," *Energy*, vol. 168, pp. 796–814, 2019.

[12] M. Özlen and M. Azizoğlu, "Multi-objective integer programming: A general approach for generating all non-dominated solutions," *European Journal of Operational Research*, vol. 199, no. 1, pp. 25–35, 2009.

[13] M. Javadi, M. Lotfi, G. J. Osório, A. Ashraf, A. E. Nezhad, M. Gough, and J. P. Catalão, "A multi-objective model for home energy management system self-scheduling using the epsilon-constraint method," in *2020 IEEE 14th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG)*, vol. 1, pp. 175–180, IEEE, 2020.

[14] S. Tamby and D. Vanderpooten, "Enumeration of the nondominated set of multiobjective discrete optimization problems," *INFORMS Journal on Computing*, vol. 33, no. 1, pp. 72–85, 2021.

[15] G. Kirlik and S. Sayın, "A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems," *European Journal of Operational Research*, vol. 232, no. 3, pp. 479–488, 2014.

[16] J. Sylva and A. Crema, "A method for finding the set of non-dominated vectors for multiple objective integer linear programs," *European Journal of Operational Research*, vol. 158, no. 1, pp. 46–55, 2004.

[17] S. Rehman and S. A. Khan, "Multi-criteria wind turbine selection using weighted sum approach," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, 2017.

[18] T. K. Hua and N. Abdullah, "Weighted sum-dijkstra's algorithm in best path identification based on multiple criteria," *J. Comput. Sci. Comput. Math*, vol. 8, no. 3, pp. 2–8, 2018.

[19] S. K. Singh and M. Goh, "Multi-objective mixed integer programming and an application in a pharmaceutical supply chain," *International Journal of Production Research*, vol. 57, no. 4, pp. 1214–1237, 2019.

[20] M. Tavana, K. Govindan, A. K. Nasr, M. S. Heidary, and H. Mina, "A mathematical programming approach for equitable covid-19 vaccine distribution in developing countries," *Annals of Operations Research*, pp. 1–34, 2021.

[21] V. Chen and J. N. Hooker, "A just approach balancing rawlsian leximax fairness and utilitarianism," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 221–227, 2020.

[22] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: A review of applications, methods and models," *European journal of operational research*, vol. 153, no. 1, pp. 3–27, 2004.

[23] H. Luss, "A distributed algorithm for equitable bandwidth allocation for content distribution in a tree network," *Journal of the Operational Research Society*, vol. 63, no. 4, pp. 460–469, 2012.

[24] L. Galand and T. Lust, "Exact methods for computing all lorenz optimal solutions to biobjective problems," in *Algorithmic Decision Theory: 4th International Conference, ADT 2015, Lexington, KY, USA, September 27–30, 2015, Proceedings 4*, pp. 305–321, Springer, 2015.

[25] P. Xidonas, G. Mavrotas, C. Hassapis, and C. Zopounidis, "Robust multiobjective portfolio optimization: A minimax regret approach," *European Journal of Operational Research*, vol. 262, no. 1, pp. 299–305, 2017.

[26] Ö. Karsu and A. Morton, "Inequity averse optimization in operational research," *European journal of operational research*, vol. 245, no. 2, pp. 343–359, 2015.

[27] N. Kaynar and Ö. Karsu, "Equitable decision making approaches over allocations of multiple benefits to multiple entities," *Omega*, vol. 81, pp. 85–98, 2018.

[28] Ö. Karsu and H. Erkan, "Balance in resource allocation problems: a changing reference approach," *Or Spectrum*, vol. 42, no. 1, pp. 297–326, 2020.

[29] M. Yavuz, *Two approaches for fair resource allocation*. PhD thesis, Bilkent Universitesi (Turkey), 2018.

[30] J. Rawls, "Atheory of justice," *Cambridge (Mass.)*, 1971.

[31] M. M. Kostreva and W. Ogryczak, "Linear optimization with multiple equitable criteria," *RAIRO-Operations Research-Recherche Opérationnelle*, vol. 33, no. 3, pp. 275–297, 1999.

[32] D. Baatar and M. M. Wiecek, "Advancing equitability in multiobjective programming," *Computers & Mathematics with Applications*, vol. 52, no. 1-2, pp. 225–234, 2006.

[33] B. Bashir and Ö. Karsu, "Solution approaches for equitable multiobjective integer programming problems," *Annals of Operations Research*, pp. 1–29, 2022.

[34] V. K. Singh, *Equitable efficiency in multiple criteria optimization*. PhD thesis, Clemson University, 2007.

[35] A. M. Geoffrion, "Proper efficiency and the theory of vector maximization," *Journal of mathematical analysis and applications*, vol. 22, no. 3, pp. 618–630, 1968.

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

[37] E. U. Choo and D. R. Atkins, "Proper efficiency in nonconvex multicriteria programming," *Mathematics of Operations Research*, vol. 8, no. 3, pp. 467–470, 1983.

[38] Ö. Karsu and F. Ulus, "Split algorithms for multiobjective integer programming problems," *Computers & Operations Research*, vol. 140, p. 105673, 2022.

[39] N. Ö. Özpeynirci, *Approaches for multiobjective combinatorial optimization problems*. PhD thesis, Middle East Technical University, 2008.

Appendix A

Proof of Proposition 2

To the contrary, assume that $z(x^*)$ is not weakly non-dominated. Then $\exists\, x' \in \mathcal{X}$ such that $z_i(x') > z_i(x^*) \;\; \forall i = 1, \ldots, p$. Therefore,

$$
\begin{aligned}
z_i^R - z_i(x') &< z_i^R - z_i(x^*) \;\; \forall i = 1, \ldots, p \\
w_i(z_i^R - z_i(x')) &< w_i(z_i^R - z_i(x^*)) \;\; \forall i = 1, \ldots, p \\
\max_{i=1,\ldots,p} \{w_i(z_i^R - z_i(x'))\} &< \max_{i=1,\ldots,p} \{w_i(z_i^R - z_i(x^*))\} \\
\alpha' &< \alpha^*
\end{aligned}
$$

This **contradicts** the optimality of (α^*, x^*) for the Tchebychev scalarization problem.

Appendix B

Proof of Proposition 3

To the contrary, assume that $z(x^*)$ is not weakly non-dominated. Then $\exists\, x' \in \mathcal{X}$ such that $z_i(x') > z_i(x^*) \;\; \forall i = 1, \ldots, p$. Therefore,

$$
\begin{aligned}
z_i^R - z_i(x') &< z_i^R - z_i(x^*) \;\; \forall i = 1, \ldots, p \\
\frac{z_i^R - z_i(x')}{d_i} &< \frac{z_i^R - z_i(x^*)}{d_i} \;\; \forall i = 1, \ldots, p \\
\max_{i=1,\ldots,p} \{\frac{z_i^R - z_i(x')}{d_i}\} &< \max_{i=1,\ldots,p} \{\frac{z_i^R - z_i(x^*)}{d_i}\} \\
\rho' &< \rho^*
\end{aligned}
$$

This **contradicts** the optimality of (ρ^*, x^*) for the PS scalarization problem.

Appendix C

ExA Algorithm

Algorithm 6 ExA Algorithm

1: Let $Y_E^{'} = \emptyset$, $k = 0$, $\mathcal{V} = \emptyset$, $\mathcal{F} = \emptyset$, $\mathcal{L} = \{\{m^1, \ldots, m^p\}\}$.
2: Select a stage $R = \{r^1, r^2, \ldots, r^p\} \in \mathcal{L}$ and set $\mathcal{V} = \mathcal{V} \cup \{R\}$.
3: Solve λ for $\lambda r^1 = \lambda r^2 = \cdots = \lambda r^p$.
4: **if** $\lambda \in \mathbb{R}^p_>$ **then**
5: Solve MOMIP(λ) and let $r^* = (r_1^*, r_2^*, \ldots, r_p^*)$ be the optimal point.
6: **if** $r^* \in R$ **then**
7: Set $\mathcal{F} = \mathcal{F} \cup \{R\}$.
8: **else**
9: $\mathcal{L} = \mathcal{L} \cup \{\{r^1, \ldots, r^{p-1}, r^*\}, \{r^1, \ldots, r^*, r^p\}, \ldots, \{r^*, \ldots, r^{p-1}, r^p\}\}$ and $\mathcal{L} = \mathcal{L} - (\mathcal{L} \cap \mathcal{V})$.
10: **if** $r^* \notin Y_E^{'}$ **then**
11: $k = k + 1$, $y^k = r^*$ and $Y_E^{'} = Y_E^{'} \cup \{y^k\}$.
12: **end if**
13: **end if**
14: **else**
15: Find $y \in Y_{EM}^{'} \setminus R$ and $\lambda^{'} \in \mathbb{R}^p_>$ such that $\lambda^{'} y \leq \lambda^{'} r \ \forall r \in R$. Set $r^* = y$ and go to line 9.
16: **end if**
17: $\mathcal{L} = \mathcal{L} - R$.
18: **if** $\mathcal{L} = \emptyset$ **then**
19: **return** $Y_E^{'}$.
20: **else**
21: Go to line 2.
22: **end if**

MOMIP(λ):

$$\min \sum_{k=1}^{p} \lambda z(x) = \lambda_k z_k(x) \tag{C.1}$$

$$x \in \mathcal{X} \tag{C.2}$$