INTERACTIVE ALGORITHMS TO SOLVE BIOBJECTIVE AND TRIOBJECTIVE DECISION MAKING 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
Tuğba Denktaş
May 2021

INTERACTIVE ALGORITHMS TO SOLVE BIOBJECTIVE AND TRIOBJECTIVE DECISION MAKING PROBLEMS

By Tuğba Denktaş

May 2021

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)

Firdevs Ulus(Co-Advisor)

Oya Karaşan

Serhat Gül

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

INTERACTIVE ALGORITHMS TO SOLVE BIOBJECTIVE AND TRIOBJECTIVE DECISION MAKING PROBLEMS

Tuğba Denktaş
M.S. in Industrial Engineering
Advisor: Özlem Karsu
Co-Advisor: Firdevs Ulus
May 2021

We propose interactive algorithms to find the most preferred solution of biobjective and triobjective integer programming problems. The algorithms can be used in any setting where the decision-maker has a general monotone utility function. They divide the image space of the problems into boxes and search them by solving Pascoletti-Serafini scalarizations, asking questions to the decision-maker so as to eliminate boxes whenever possible. We also propose a cone based approach that can be incorporated into both algorithms if the decision-maker is assumed to have a non-decreasing quasiconcave utility function. We demonstrate the performances of the algorithms and their cone based extensions with computational experiments. The results of the experiments show that interactive algorithms are very useful in terms of solution time compared to a posteriori algorithms that find the whole Pareto set. The results of the experiments also show that the cone based approach leads to less interaction with the decision-maker.

Keywords: Multiobjective integer programming, Interactive algorithms, Convex cones.

ÖZET

İKİ AMAÇLI VE ÜÇ AMAÇLI KARAR VERME PROBLEMLERİ İÇİN ALGORİTMALAR

Tuğba Denktaş
Endüstri Mühendisliği, Yüksek Lisans
Tez Danışmanı: Özlem Karsu
İkinci Tez Danışmanı: Firdevs Ulus
Mayıs 2021

İki ve üç amaçlı tam sayılı programlama problemleri için en çok tercih edilen çözümü bulan iki interaktif algoritma geliştirilmiştir. Bu algoritmalar, karar vericinin genel monoton fayda fonksiyonuna sahip olduğu tüm durumlarda kullanılabilir. Algoritmalar, görüntü uzayını kutulara böler ve bu kutuları Pascoletti-Serafini skalarizasyon modelini çözerek arar. Bu sırada, karar vericiye sorular sorarak mümkün olduğu durumlarda bazı kutuları aramadan eler. Ek olarak, karar vericinin azalmayan yarı-konkav yarar fonksiyonuna sahip olduğu durumlar için, geliştirilen algoritmalara koni tabanlı bir yaklaşım sunulmuş ve algoritmaların ve koni yaklaşımlarının performansları bilgisayımsal denemeler ile test edilmiştir. Deney sonuçları, interaktif algoritmaların, tüm Pareto çözümleri bulan algoritmalara göre çözüm süresi açısından daha iyi olduğunu göstermektedir. Ayrıca, koni yaklaşımının karar verici ile olan etkileşim sayısını azaltmakta yardımcı olduğu gözlemlenmiştir.

Anahtar sözcükler: Çok amaçlı tam sayılı programlama, İnteraktif algoritmalar, Konveks koniler.

Acknowledgement

I would like to express my gratefulness to Asst. Prof. Özlem Karsu and Asst. Prof. Firdevs Ulus, for all the help and guidance they have given me. They have been so helpful and understanding to me for the past two years and I have learned so much from them, I am very thankful.

I would like to thank Prof. Oya Karaşan and Asst. Prof. Serhat Gül for reading and evaluating my thesis.

I'm thankful to my parents, my father Levent and my mother Berrin for being by my side and supporting every decision I make. I could not be where I am now without them by my side. I am forever grateful to them. I'm also thankful to my brother Semih, for letting me know that he will be there whenever I need and I will never be alone.

Furthermore, I'd like to thank to the people who made my life wonderful these past two years. Damla Akoluk, her vivacious soul showed me there's always more to life, Aleyna Kof, she always listened to me and reminded me to wear sunscreen all the time, Dilara Sönmez, she's been one of the most understanding people in my life and was always there for me, Ege Bilaloğlu, he kept my secrets, shared my musics and was always sincere with me. I had the most fun with them even when having fun was impossible. Also, I'm thankful to my best friends who couldn't be next to me physically but always have been by my side, Mert Erdemir and Bartu Gündoğmuş, the shoulders I cried on, the biggest laughs I shared with. Those people are the ones I can never imagine my life without, so I'm very grateful for all the love and joy we have shared and will share.

I also want to thank to my sister Ayca Şahin, who has been my rock through my university life. She has been the best partner I could ever ask for, and the other half

of the infamous "Parlayan Yıldızlar Takımı". She always did her best to make me happy and I will always appreciate it. Berfin Küçük, she always gives me the best advices yet I never listen, but in the end I find out she was right all along. I thank to her for being so patient and loving with me. She first taught me how to share a dorm room and then how to share happiness and dreams, therefore I am also thankful to all the people led her into my life.

Finally, I would like to thank all the people in our department and the ones who have touched my life in some way.

This study was supported by TUBITAK (The Scientific and Technological Research Council of Turkey) under Grant no: 218M606.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In real life, there are many problems that can be formulated as an integer programming problem such as knapsack, assignment and transportation. Most of those problems have multiple objectives and it is usually not possible to optimize all objectives at the same time, which leads to multiple solutions with changing performances with respect to the criteria (*nondominated solutions*). Therefore, one aims to find the most preferred solution of such multi-criteria optimization problems by interacting with the decision-maker.

In this thesis, we focus on biobjective integer programming (BOIPs) and triobjective integer programming problems (TOIPs). We propose two interactive algorithms for BOIPs and TOIPs and a cone based approach that interact with the decision-maker and return the most preferred nondominated solution without finding the whole set of nondominated solutions of the problem. There are algorithms proposed in the literature for this purpose. Some of them find the whole nondominated solution set first and then interact with the decision-maker [1, 2, 3, 4]. In the algorithms

we propose, the interactions with the decision-maker are done throughout the solution process of the problem. Algorithms with similar approach to ours have so far used methods such as Tchebychev scalarization [5, 6, 7, 8], Simplex method [9] and branch-and-bound technique [10, 11]. Convex cone approach is also broadly used in order to decrease the number of interactions with the decision-maker [12, 13, 14, 15].

The proposed algorithms are based on dividing the search space in regions and searching them by solving Pascoletti-Serafini scalarizations with a fixed direction and changing reference points. In each iteration, a region is either directly searched or asked to the decision-maker to be searched (we will specify the type of the questions we ask later on). If the region is searched, a new nondominated solution is found if exists. Then, the decision-maker is asked to make a preference between the newest solution and the most preferred one (i.e. incumbent solution) among the previously obtained solutions. According to the answers of the decision-maker, some solutions and regions may be directly eliminated before taken into consideration. Algorithms continue until there is no region left to search for nondominated solutions. We also introduce a cone based approach for the algorithms where cone elimination is introduced to decrease the number of questions asked to the decision-maker. We test the performance of our algorithms in bicriteria and tricriteria knapsack and assignment problems.

The rest of the thesis is as follows. In Chapter 2, we give the preliminaries and the problem definition. In Chapter 3, we make a literature review. In Chapters 4 and 5, we give the proposed interactive algorithms for BOIPs and TOIPs, respectively. In Chapter 6, we introduce the cone based approach to both algorithms. Chapter 7 contains the computational results. We conclude our discussion in Chapter 8.

Chapter 2

Problem Definition and Preliminaries

We consider the following multiobjective integer programming problem (MOIP)

$$\begin{aligned} \max. \quad & z(x) \\ \text{s.t.} \quad & x \in X \end{aligned} \tag{P}$$

where $z : \mathbb{R}^n \to \mathbb{R}^p$ is a vector valued function and $X \subseteq \mathbb{Z}^n$ is the feasible region of the problem. $\mathcal{N} := \{z(x) \mid x \in X\}$ denotes the feasible set in the objective space. For brevity, we also denote an arbitrary element from $\mathcal{N}$ by z. When $p = 2$, the problem is called a biobjective integer programming problem (BOIPs) and when $p = 3$, it is called triobjective integer programming problem (TOIPs).

Let $z^1, z^2 \in \mathbb{Z}^p$. The following notation is used throughout the thesis:

- $z^1 > z^2 \iff z^1_j > z^2_j \quad \forall j = 1, \ldots, p$

- $z^1 \geq z^2 \iff z_j^1 \geq z_j^2 \quad \forall j = 1, \ldots, p$
- $z^1 \gneq z^2 \iff z^1 \geq z^2$ and $z^1 \neq z^2$

Definition 2.1. *Let $z(x) \in \mathcal{N}$ be a feasible objective function vector of (P). $z(x)$ is said to be dominated if there is $z(\hat{x}) \in \mathcal{N}$ such that $z(\hat{x}) \gneq z(x)$. If $z(\hat{x}) > z(x)$, then $z(\hat{x})$ strictly dominates $z(x)$. If there exists no $z(\hat{x}) \in \mathcal{N}$ that (strictly) dominates $z(x)$, then $z(x)$ is (weakly) nondominated.*

Set of all nondominated and weakly nondominated points are denoted by $\mathcal{N}_N$ and $\mathcal{N}_{WN}$, respectively.

Definitions above are illustrated in Figure 2.1, for which we have $\mathcal{N}_{WN} = \{a, b, d, e, f, g\}$, $\mathcal{N}_N = \{a, d, e, f, g\}$, b is dominated by a and, c is strictly dominated by d and e.

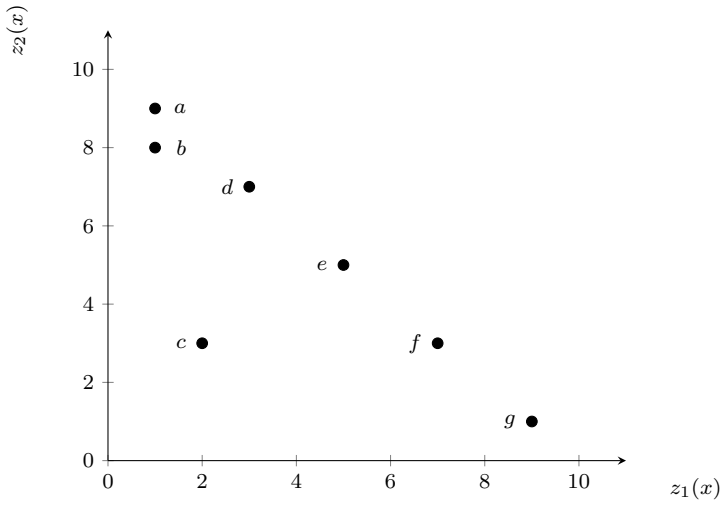


Figure 2.1: Example setting

Definition 2.2. *A function $f : \mathbb{R}^p \to \mathbb{R}$ is called a monotone function if it is monotonically non-decreasing or non-increasing:*

- *If for all $x \leq y$, $f(x) \leq f(y)$ holds, then f is a monotonically non-decreasing function.*

- *If for all $x \leq y$, $f(x) \geq f(y)$ hold then f is a monotonically non-increasing function.*

Definition 2.3. *A function $f : \mathbb{R}^p \to \mathbb{R}$ is called a quasi-concave function if $f(\lambda x + (1-\lambda)y) \geq \min\{f(x), f(y)\}$ for each $x, y \in \mathbb{R}^p$ and $\lambda \in [0,1]$.*

A function f is increasing quasi-concave if it is both increasing and quasi-concave.

Definition 2.4. *We define the preference relations in the following way:*

- *If a decision-maker prefers y at least as much as x, it is said that the decision-maker weakly prefers y to x and it is denoted by $x \preceq y$.*
- *If $x \preceq y$ and $y \npreceq x$, then y is strictly preferred to x and it is denoted by $x \prec y$.*
- *If $x \preceq y$ and $y \preceq x$, then the decision-maker is indifferent between x and y and it is denoted by $x \sim y$.*

A general monotone utility function $f : \mathbb{R}^p \to \mathbb{R}$ represents a preference relation $\preceq$ if it is true for all $x, y \in \mathbb{R}^p$ that $f(y) \geq f(x)$ if and only if $x \preceq y$.

If f represents the preference relation $\preceq$, then the following relations can be obtained using Definition 2.4:

$$f(y) > f(x) \Longleftrightarrow x \prec y$$

$$f(y) = f(x) \Longleftrightarrow x \sim y$$

The following concepts are important for the algorithms that will be explained in the next chapters.

Ideal point of a multiobjective programming problem (P) is obtained by solving the following model for all objectives $i = \{1, \ldots, p\}$:

$$\begin{aligned} \max. \quad & z_i(x) \\ \text{s.t.} \quad & x \in X. \end{aligned}$$

Let x^i be an optimal solution of the model $\max_{x \in X} z_i(x)$. Then, the vector $z^I := [z_1(x^1), \ldots, z_p(x^p)]$ is the ideal point for the problem (P).

The vector z^N, where $z_i^N = \min_{z \in \mathcal{Y}_N} z_i$ for all $i = 1, \ldots, p$ is the nadir point of the problem (P).

Chapter 3

Literature Review

Many algorithms are proposed to solve multiobjective integer programming problems (MOIPs) in the literature. The algorithms where all nondominated solutions of an MOIP found are called *a posteriori algorithms*. These algorithms mostly benefit from scalarizations in order to solve MOIPs and commonly used scalarization models are the weighted sum [16, 17, 18], the ϵ-constraint [19, 20, 21, 22, 23], and the weighted Tchebychev scalarization [24, 25].

The other type of algorithms are the ones where there is an interaction with a decision-maker, and they are called *interactive algorithms*. These algorithms gather preference information from the decision-maker and utilize it to solve the problem. The way they elicit information from the decision-maker depends on the problem type. It can be a grouping problem, where the aim is grouping a subset of the alternatives into classes; a ranking problem, where the aim is to rank the given points from the most preferred to the least; a choice problem, where the decision-maker is given a set of alternatives and has to choose one; or an optimization problem, in which the nondominated solutions are implicitly defined by the constraints and

the aim is to find a subset of those solutions until the most preferred solution by the decision-maker is obtained. Detailed reviews on the interactive algorithms to solve these problems are present in [26, 27, 28]. Here, we will focus on more recent algorithms in the literature. In Table 3.1 we provide a categorization of the studies regarding the interactive algorithms with respect to the type of the problem that they solve: grouping (G), choice/ranking (C/R), optimization (O); the underlying utility function assumed for the decision-maker: linear (L), quasiconcave (QC), symmetric quasiconcave (SQC), general monotone (GM), additive (A); and whether convex cones are used (Y) or not (N) in elimination. As it will be detailed later, with the help of convex cones generated using the answers of the decision-maker to the previously asked preference questions, it may be possible to eliminate some alternatives without asking further questions to the decision-maker.

Table 3.1: Articles grouped by their type, utility assumptions and based on existence of cone approach

Articles	Type	Utility	Cone	Articles	Type	Utility	Cone
[29]	G	L	N	[7, 8]	O	-	N
[30]	G	L,QC,GM	N	[5, 11, 6]	O	GM	N
[1, 2, 23]	C/R	-	N	[9, 10]	O	L	N
[3]	C/R	A	N	PA[1]	O	GM	N
[4]	C/R	L	N	[12, 31, 32]	O	QC	Y
[13, 33, 34, 35, 14, 15, 36]	C/R	QC	Y	[37]	O	SQC	Y
[38, 39]	C/R	SQC	Y	CBA[2]	O	QC	Y

In the algorithms designed to solve grouping or choice/ranking problems, the decision-maker is already presented with the set of alternatives. For instance, in [29], a multiobjective decision making problem of partitioning the given alternatives into preference groups is considered. The decision-maker is first asked to place some of the alternatives into classes. Based on the information gained from the decision-maker's grouping, and using dominance relation and the linearity assumption on the utility function, the algorithm aims to place the other alternatives into the groups.

[1]The algorithms that we propose to solve BOIP and TOIP

[2]The cone based approach that we propose

In [30], Köksalan and Ulu apply the same procedure to the problem of admitting students to a master's program by placing the alternatives into "acceptable" and "unacceptable" sets under the linearity assumption for the utility function. According to the decision-maker's previous preferences and by checking the dominance relations, their algorithm aims to identify the students that should be accepted or rejected without asking the decision-maker. They propose algorithms for quasiconcave and general monotone utility functions as well.

In choice/ranking problems, the main objective is to find a rank ordering of the whole set or a subset of alternatives [40]. In [1, 2, 3], the authors are interested in finding a complete ranking of alternatives under the linearity assumption on the decision-maker's utility function. Moreover, the approach in [4] aims to achieve the ranking of alternatives without any assumption on the utility function; the algorithm is tested on a discrete alternative multicriteria choice problem. In the first iteration, random weights are generated and the alternative that maximizes the corresponding weighted sum of the objectives is selected as the reference alternative. It is presented to the decision-maker along with the other alternatives to indicate his preference. Based on his answers, the reference alternative and the weights are updated. The algorithm stops when the current reference point is preferred to all other points. Final weights are used to rank the alternatives.

For the algorithms designed to solve grouping or choice/ranking problems, one of the aims is to decrease the number of interactions with the decision-maker (see e.g., [23]). There are several approaches proposed for this purpose. Among those approaches to limit the number of interactions, the most common one utilizes the convex cones [34] under the assumption of an underlying nondecreasing quasiconcave [33, 14, 15] or a symmetric quasiconcave utility function when applied to equitable decision making problems [38, 39]. In most of the algorithms, the convex cones are generated by the preference relations between the nondominated points but there are exceptions where dummy points are generated by taking the convex combinations

of the nondominated points [13], or considering dominated or dominating points [35, 36].

The previous part briefly summarized the grouping and choice/ranking problems. Since we focus on optimization problems in this thesis we will explain the algorithms developed for this type, in detail. When the problem type is optimization, the decision-maker is interacted throughout the solution procedure of an MOIP in order to determine the most preferred solution. In general, these interactions lead to finding a subset of the nondominated solution set until the most preferred point is obtained. Such an interactive approach is introduced with weighted Tchebychev scalarization model under the assumption of increasing utility function [5, 6]. The procedure aims to find the subset of solutions by finding the nondominated points that are "closest" to the reference point, which is the ideal point of the problem. They solve the Tchebychev scalarization model in every iteration with random weights and ask the decision-maker to make preferences between obtained solutions. The weights are updated based on the answers. In a similar approach [7], the decision-maker is also asked to change the position of the reference point or indicate the objective she wants to improve. The procedure continues until the predetermined iteration limit is reached. [8] applies a similar idea using branch-and-bound method by solving a Tchebychev scalarization model, combined with cutting plane method in every branch.

There are other interactive algorithms utilizing a branch-and-bound framework as well. [10] utilizes Zionts and Wallenius method [9] under the assumption of linear utility function, where the decision-maker is asked in every iteration to make comparisons between the new solution and the incumbent solution, which is the most preferred solution by the decision-maker so far. The decision-maker can also interact to change the search direction. Through these interactions, the algorithm captures the decision-maker's preference structure while aiming to find the most preferred solution. In the initial step, the linear relaxation of the problem is solved. If the

solution is integer, then it is also optimal to the integer programming problem. If the solution is not integer, the process goes on until a solution that is both integer and preferred by the decision-maker is found. A node in a branch is called a candidate problem or a subproblem, and it can be eliminated in three cases: (1) if there is no feasible solution to the linear relaxation of it, (2) if the current most preferred solution by the decision-maker is preferred to the optimal solution of the multicriteria linear relaxation of it, (3) if the optimal solution to the problem is integer and preferred by the decision-maker. If it is the last case, then the most preferred solution is updated.

Another interactive branch-and-bound algorithm [11] can be applied if the feasible set of the problem is convex or discrete and bounded. The algorithm splits the main problem into branches by introducing constraints on the objectives. In each node, an upper bound which is the ideal point of that subproblem is determined. Then, decision-maker ranks the ideal points of subproblems and selects the one he prefers the most. The ideal point is compared to the incumbent solution. If the ideal point of the subproblem is preferred, then it is solved. The algorithm stops if the ideal point of the subproblem is not preferred to the incumbent point or there is no node left to explore.

There is also an interactive method utilizing simplex method under the assumption of a linear utility function [9]. It starts with random choice of weights for the objectives. For each of the nonbasic variables obtained through the procedure, the decision-maker is asked if he wants to add a nonbasic variable into the simplex table. Based on the answers, the weights of the objectives are updated. The resulting weights are then used to maximize the initial problem in order to find a new efficient solution. The process continues until there are no nonbasic variables left. This method guarantees convergence but there is a risk that there may not always be an increase in utility from one efficient solution to another.

Unlike the grouping and choice/ranking problems, in optimization problems we do not have a set of nondominated points presented to the decision-maker beforehand. Instead, we obtain them throughout the iterations by solving scalarization models or applying some of the procedures mentioned previously. This may result in asking questions to the decision-maker after every new solution or set of solutions found. Therefore, the convex cone method by Korhonen et al. [34], which is proposed originally for choice/ranking and grouping problems, can also be used in optimization problems to decrease these interactions with the decision-maker. Since we utilize convex cones for optimization problems, the respective literature will be reviewed in detail.

Convex cone method detects the inferior points under the assumption that the decision-maker has a nondecreasing quasiconcave utility function. In grouping and choice/ranking problems, convex cones are generated using the decision-maker's preferences among a subset of the points. Then, the points that are not included in this subset are checked if they are in the cone dominated region. If a point is in the cone dominated region, it is directly eliminated from consideration. Else, the decision-maker provides a preference between the point and the incumbent solution.

For optimization problem type, elimination can be done during the solution process of the underlying MOIP. For instance, in [12], in every iteration of the algorithm the decision-maker provides preference information between the incumbent and the most recently found solution. A convex cone is generated using these two points and the preference relation among them. Then, the region dominated by the convex cone is defined with inequalities. Adding these inequalities as constraints to the main scalarization model guarantees that the next solution obtained will not be in a cone dominated region. In every iteration, as the new preference information is obtained from the decision-maker, another set of constraints is added to the model until there is no new feasible solution left for the problem with additional constraints. The same approach is applied to equitable multiobjective integer programming problems

in [37, 32]. An equitably nondominated challenger (another equitably nondominated point) is generated by a scalarization model and the preferences are made among the challenger and the incumbent solution. For such algorithms utilizing convex cones, if an approximation to the most preferred nondominated solution is sufficient, then α-cones are used where α is the approximation parameter [31]. It is also possible to find the most preferred solution with this approach if it is not in one of the α-cone dominated regions. The α parameter is determined and can be changed during the solution process by the decision-maker. The performance of the convex cones is experimented through nondecreasing quasiconcave utility functions such as linear, quadratic and Tchebychev in the literature [15].

We propose interactive algorithms to solve optimization problems to find the most preferred solution by the decision-maker assuming the utility function is general monotone. These algorithms divide search space into boxes and solve Pascoletti-Serafini scalarizations to explore them. They interact with the decision-maker before exploring a box, or after a new solution is found. In order to decrease the number of interactions, we propose a cone based approach to those algorithms when the utility function is nondecreasing quasiconcave.

Chapter 4

An Interactive Algorithm for BOIPs

In this chapter, we explain the proposed interactive algorithm for BOIP. Our aim is to find the most preferred solution of a BOIP without having to find all of the nondominated solutions. We aim to achieve this by interacting with the decision-maker. The decision-maker is assumed to have a general monotone utility function.

The algorithm is based on dividing the search space into boxes and exploring these boxes in a specified order. The boxes yet-to-be explored are kept in a list, which is initialized by a box obtained by solving lexicographic optimization models, which contains the set of all nondominated points. The proposed algorithm solves a Pascoletti-Serafini scalarization model in each of the boxes to find a weakly nondominated solution. In order to guarantee finding a nondominated solution, a secondary model is solved. If a new solution is found, the decision-maker is asked to give preference information between this new solution and the previous best solution known so far (*incumbent solution*). Based on the information, incumbent solution is

updated. Also, this preference information is used to eliminate some of the boxes to be searched without asking the decision-maker, if possible. In every iteration, two new boxes are added to the search list if a new nondominated solution is found. The searched box is removed from the box list.

The pseudocode of the algorithm is given in Algorithm 1. In order to explain the algorithm explicitly, we first explain the initial steps, initialization of the boxes, and then discuss the main loop in detail.

In the initialization step, we initialize $\mathcal{Y}_N$ (the set of nondominated solutions found), and $\mathcal{B}$ (the boxes to be explored) which is defined by the upper and lower bounds of the complete search space. To determine the complete search space, we solve the following models in sequence:

$$\max\{z_1(x) \mid x \in X\}$$

Let $x^{'}$ be the optimal solution of this problem. Then the following model is solved:

$$\max\{z_2(x) \mid x \in X, z_1(x) = z_1(x^{'})\}$$

Let x^1 be the optimal solution. We obtain the nondominated point $n^1 = z(x^1)$ which is at the lower right-hand corner of the search space of (P).

Next, we solve the following models in sequence:

$$\max\{z_2(x) \mid x \in X\}$$

Let x^* be the optimal solution of this problem. Then the following model is solved:

$$\max\{z_1(x) \mid x \in X, z_2(x) = z_2(x^*)\}$$

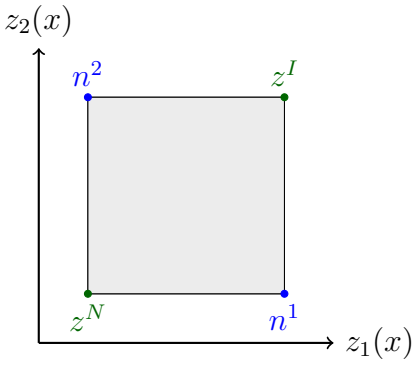
Let x^2 be the optimal solution. We obtain the nondominated point $n^2 = z(x^2)$ which is at the upper left-hand corner of the search space of (P). We initialize $\mathcal{Y}_N$ as $\{n^1, n^2\}$. Note that the first component of n^1, n^1_1 sets an upper bound on the first

objective and the second component of n^2, n_2^2 sets an upper bound on the second objective whereas n_2^1 and n_1^2 set a lower bound for first and second objectives of all nondominated points, respectively. It follows that the ideal point of the search space is $z^I = (n_1^1, n_2^2)$ and the nadir point is $z^N = (n_1^2, n_2^1)$. Throughout, $\tilde{z} = z(\tilde{x})$ denotes the incumbent solution.

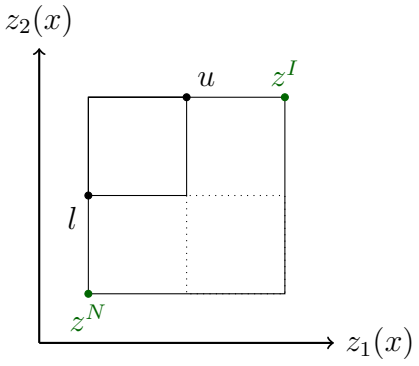
At an arbitrary iteration of the main loop of the algorithm, a box $b \in \mathcal{B}$ is defined by its lower and upper bounds l and u, respectively and denoted as follows:

$$b = b(l, u) := \{y \in \mathbb{R}^2 \mid l \leq y \leq u\}$$

Figure 4.1 shows the initial box and an arbitrary box defined by their upper and lower bounds.



(a) Initial box.



(b) A box with an upper corner u and lower corner l at an arbitrary iteration.

Figure 4.1: Boxes in two dimensional search space

Initial box $b(z^N, z^I)$ is added to set $\mathcal{B}$ to start the searching process. Decision-maker is asked to make a preference between n^1 and n^2 to determine the initial incumbent (lines 5-9 of Algorithm 1). After this step, the algorithm continues with the main loop. In each iteration of the main loop, if $\mathcal{B}$ is not empty, a box is selected and removed from $\mathcal{B}$ (lines 10-12 of Algorithm 1). We select the boxes in the order that they are added to $\mathcal{B}$.

The following remark shows that we can eliminate some boxes without searching due to integrality of the problem.

Remark 4.1. *For $b \in \mathcal{B}$, if $u_i - l_i \leq 1$ for any $i = 1, 2$, then there are no nondominated points in b.*

If a box b is not eliminated by Remark 4.1, we check if the current incumbent solution is on one of the upper edges of b, i.e. we check if $\tilde{z}_1 = u_1$ or $\tilde{z}_2 = u_2$ (lines 13-14 of Algorithm 1). If this is the case, $\tilde{z}$ is dominated by u.

Remark 4.2. *If $\tilde{z}$ is dominated by u, it follows that $u \succ \tilde{z}$.*

If $\tilde{z}$ is on one of the edges of b, the box is directly searched without asking the decision-maker due to Remark 4.2. Otherwise, we ask the decision-maker to make a preference between $\tilde{z}$ and u. If the decision-maker prefers u, the box is searched (lines 15-18 of Algorithm 1). Otherwise, it is not searched as this implies $u \prec \tilde{z}$ and we have $z \preceq u$ for any point $z \in b$. That is, the decision-maker would not be interested in solutions in box b.

If b is decided to be searched, the following optimization model, namely, Pascoletti-Serafini [41] scalarization with additional box constraints is solved (lines 1-6 of Procedure 1):

$$\begin{aligned} \min. \quad & \alpha \\ \text{s.t.} \quad & z(x) \geq l \\ & z(x) \geq u - \alpha d \\ & x \in X \\ & \alpha \in \mathbb{R} \end{aligned} \qquad (PS_1(l, u, d))$$

Here u and l are the upper and lower bounds of the current box b, respectively, $d \in \mathbb{R}^p_+$ is the direction vector of choice and $x \in X$ denotes the constraints of the

biobjective programming problem. The first constraint defines the lower bound of the nondominated solution to be returned in box b. The second constraint selects the point inside box b that is closest to the upper bound of box b. Note that the model is always feasible as it will return one of the corners (nondominated points) of the box if the interior is empty. Otherwise, it will return a new solution in the interior. If no new solution is returned, we move on with the next box. Else we move on as follows:

Assume (α^*, x^*) is the optimal solution of $PS_1(l, u, d)$. Note that $z(x^*)$ is a weakly nondominated point. If $z(x^*)$ is in the interior of the box b, we solve the following model in order to ensure finding a nondominated point:

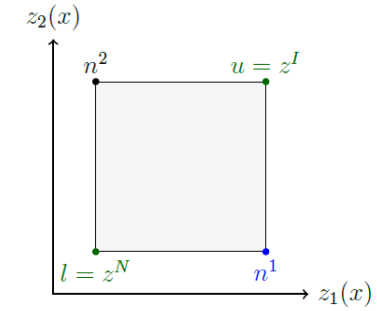
$$\begin{aligned} \max \quad & \sum_i z_i(x) \\ \text{s.t.} \quad & z(x) \geq z(x^*) \qquad\qquad (PS_2(x^*)) \\ & x \in X \end{aligned}$$

We denote the sequence of $PS_1(l, u, d)$ and $PS_2(x^*)$ models by $PS(l, u, d)$ and it means the following:

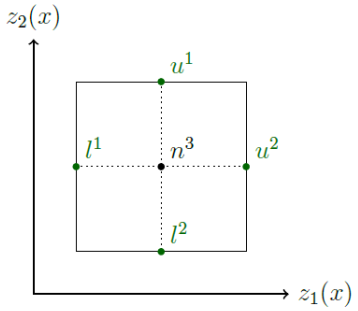
Solve $PS_1(l, u, d)$: If (α^*, x^*) is an optimal solution, then solve $PS_2(x^*)$.

Let $\hat{x}$ be the optimal solution of $PS_2(x^*)$. Then, $z(\hat{x})$ is the nondominated solution obtained from searching box b. We add $z(\hat{x})$ to $\mathcal{Y}_N$. Next, we ask the decision-maker to make a preference between $\tilde{z}$ and $z(\hat{x})$ (line 7 of Procedure 1). If $z(\hat{x})$ is preferred, we update the incumbent solution. We then find the upper and lower bounds of each box formed by finding the new solution. The upper and lower bounds of the upper box are $(z_1(\hat{x}), u_2)$ and $(l_1, z_2(\hat{x}))$, respectively. Similarly, the bounds of the lower box are $(u_1, z_2(\hat{x}))$ and $(z_1(\hat{x}), l_2)$ (lines 11-12 of Procedure 1). This procedure basically splits a box into two boxes, removing the regions with points dominated by $z(\hat{x})$ and dominating $z(\hat{x})$.

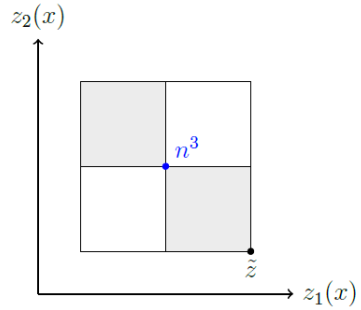
After the formation of new boxes, $\mathcal{B}$ is updated accordingly (line 12 of Procedure 1). The algorithm iterates until there is no box left in $\mathcal{B}$. See Figure 4.2 for an iteration of the main loop of the algorithm.



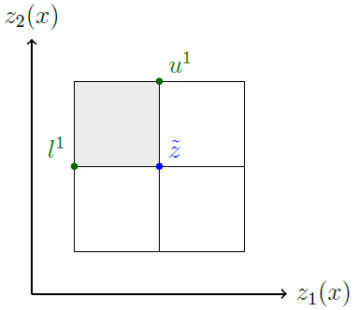
(a) Initial box to be searched. $\tilde{z} = n_1$, shown as blue.



(b) By solving $PS(l, u, d)$, n^3 is found. n^3 divides the initial box in two, $b(l^1, u^1)$ and $b(l^2, u^2)$. New boxes are added to $\mathcal{B}$.



(c) We ask the decision-maker to make preference between n^3 and $\tilde{z}$. For this example, we assume that n^3 is preferred. So, we update the incumbent accordingly.



(d) $b(l^1, u^1)$ is the next box to be searched. We solve $PS(l^1, u^1, d)$. Algorithm continues until there is no box left.

Figure 4.2: An iteration (a)-(d) of the proposed BOIP algorithm

Algorithm 1 Proposed Interactive Algorithm for BOIP

Let $x^i \in \arg\max_{x \in X} z_i(x)$, $z_i^I = z_i(x^i)$, $z_i^N = (z_1(x^2), z_2(x^1))$
Let $u = z^I$ and $l = z^N$ be the upper and lower bounds of of box b, respectively.
$\mathcal{B} = b$, $\tilde{z} = \emptyset$, solve $= 1$.
$\mathcal{Y}_N = \{z(x^1), z(x^2)\}$
(Ask the decision-maker about her preference between z^1 and z^2)
if $z^1 \succeq z^2$ **then**
    $\tilde{z} = z^1$
else
    $\tilde{z} = z^2$
end if
while $\mathcal{B} \neq \emptyset$ **do**
    Let $b \in \mathcal{B}$
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \{b\}$
    if $\mid u_1 - l_1 \mid > 1$ and $\mid u_2 - l_2 \mid > 1$ **then**
        if $\tilde{z}_1 = u_1$ or $\tilde{z}_2 = u_2$ **then**
            $\mathcal{B} \leftarrow Explore(\mathcal{B}, b, d, \tilde{z})$;
        else
            (Ask the decision-maker her preference between u and $\tilde{z}$)
            if $u \succeq \tilde{z}$ **then**
                $\mathcal{B} \leftarrow Explore(\mathcal{B}, b, d, \tilde{z})$;
            end if
        end if
    end if
end while

Procedure 1 *Explore*$(\mathcal{B}, b, d, \tilde{z})$

Solve $PS_1(l, u, d)$.
if the problem is feasible **then**
  Let (α^*, x^*) be the optimal solution.
  if $l < z(x^*) < u$ **then**
    Solve $PS_2(z(x^*))$. Let $x^{'}$ be the optimal solution.
    $\mathcal{Y}_N \leftarrow \mathcal{Y}_N \cup \{z(x^{'})\}$
    (Ask the decision-maker about her preference between $z(x^{'})$ and $\tilde{z}$)
    if $z(x^{'}) \succeq \tilde{z}$ **then**
      $\tilde{z} = z(x^{'})$
    end if
    Let $\hat{b} = (l_1, z_2(x^{'}), z_1(x^{'}), u_2)$ and $\bar{b} = (z_1(x^{'}), l_2, u_1, z_2(x^{'}))$ be the new boxes.
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\hat{b}, \bar{b}\}$
  end if
end if
return $\mathcal{B}$

Chapter 5

An Interactive Algorithm for TOIPs

In this chapter, we explain the proposed interactive algorithm for TOIP. Our aim is to find the most preferred solution of a TOIP without having to find all of the nondominated solutions. We aim to achieve this by interacting with the decision-maker of the problem, who is assumed to have general monotone utility function.

Similar to the algorithm proposed for BOIPs, this algorithm is also based on splitting the search space. Here, the search space becomes a three-dimensional box. To initialize the search space, we solve single objective problems to determine the ideal point and find a lower bound for the nadir point of TOIP. The initial box is defined by these two points. We keep yet-to-be explored boxes and previously explored boxes in lists. The algorithm solves $PS(l, u, d)$ for a box in the search list. If a new solution is found, the decision-maker is asked to give preference information between this solution and the incumbent solution. Incumbent solution is updated accordingly. Also, this preference information is used to eliminate some of boxes to

be searched before even asking the decision-maker, if possible. In every iteration, new boxes are added to the search list if a new nondominated solution is found. Searched box is removed from the box list. The algorithm stops when there is no box left to search in the list.

The pseudocode of the algorithm is given in Algorithm 2. In order to explain the algorithm explicitly, we first explain the initial steps, initialization of the search space, and then the main loop in detail.

In the initialization step, we define $\mathcal{Y}_N$ as the set of nondominated solutions, $\mathcal{B}$ as the boxes to be explored and $PrevBox$ as the boxes searched previously. Initial box is defined by the corner points of the complete search space. To determine the complete search space of the problem, we solve the following augmented problem for each objective $i = 1, 2, 3$.

$$\max\{z_i(x) + \epsilon \sum_{j \neq i} z_j(x) \mid x \in X\}$$

Here, $\epsilon > 0$ is a sufficiently small positive real number.

An optimal solution found for this problem is denoted by n^i, for $i = 1, 2, 3$. We add these points to $\mathcal{Y}_N$. We note that the ideal point is $z^I = (n_1^1, n_2^2, n_3^3)$ for the complete search space. Then, we solve the following model for each $i = 1, 2, 3$ to find a lower bound $\hat{z}^N$ for the nadir point z^N.

$$\begin{aligned} \text{min.} \quad & z_i(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

Let the solution of the above model be x^i. We set $\hat{z}_i^N = z_i(x^i)$ for all $i = 1, 2, 3$.

An example of initial search space is shown in Figure 5.1.

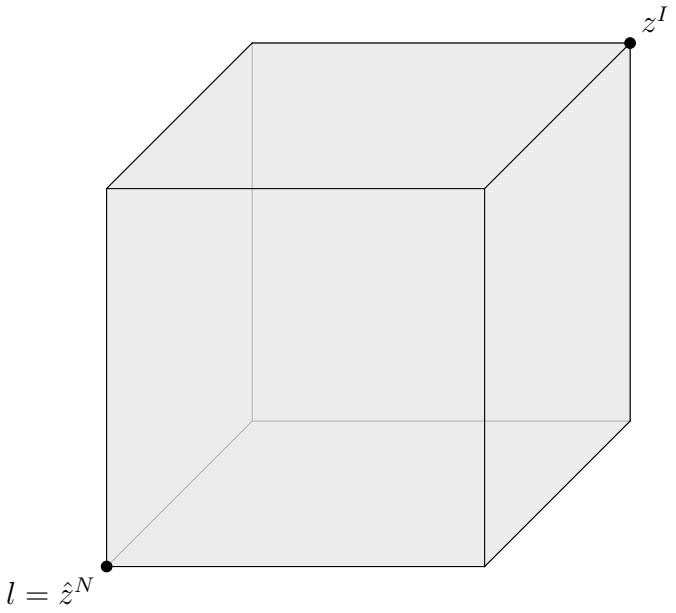


Figure 5.1: Initial search space of a TOIP

At an arbitrary iteration of the main loop of the algorithm, a box $b \in \mathcal{B}$ is defined by two points, lower corner of the box l and the ideal point z^I as follows:

$$b = b(l, z^I) := \{y \in \mathbb{R}^3 \mid l \leq y \leq z^I\}$$

We note that $l = \hat{z}^N$ for the initial box.

We search for nondominated points within box b by solving $PS(l, z^I, d)$. As opposed to the case in BOIP, in TOIP we may have intersecting boxes. The part of a box b that does not intersect with other boxes is called the *individual part* of b [42]. Individual parts also have an upper bound and it is denoted by b^I. We determine the individual upper bound of a box while forming it. We also note that unlike the case in $p = 2$, the upper corner of all boxes are the same (z^I) in this setting ($p = 3$). Hence elimination of the boxes through preference information is not possible. However, one can shrink the boxes using individual upper bound and this is how we use

preference information in TOIP.

To start the searching process, the initial box is added to set $\mathcal{B}$. Decision-maker is asked to make preferences between n^1, n^2 and n^3 to determine the initial incumbent. After this step, the algorithm continues with the main loop (lines5-36 of Algorithm 2). In each iteration of the main loop, if $\mathcal{B}$ is not empty, a box b is selected from $\mathcal{B}$. We add b to list D, which keeps the boxes to be deleted from $\mathcal{B}$ after every iteration.

We keep information on the previously explored boxes in a list called $PrevBox$. Each element in this list is denoted by (n^b, b) where n^b is the nondominated solution found by searching b. There may be cases when there is no solution in b and in these cases we denote n^b as $n^b = \emptyset$. If $b \in \mathcal{B}$ and $b \subseteq \hat{b}$ for a $\hat{b} \in Prevbox$, we can eliminate b without searching by the following proposition:

Proposition 5.1. *[43, Proposition 2.2] Let $b, \hat{b} \subseteq \mathbb{R}^p$ with lower bounds l and $\hat{l}$ respectively and $d \neq 0$. Then, the following holds:*

1. *Let $l \geq \hat{l}$. If $PS_1(\hat{l}, z^I, d)$ is infeasible, then $PS_1(l, z^I, d)$ is also infeasible.*
2. *Let $l \geq \hat{l}$. If x is an optimal solution of $PS(l, z^I, d)$ and $z(x) \geq l$, then x is also an optimal solution of $PS(\hat{l}, z^I, d)$.*

Proof. Both follow by the fact that feasible region of $PS(\hat{l}, z^I, d)$ is a subset of feasible region of $PS(l, z^I, d)$. □

We first check if information from previous boxes can be used to explore b without solving $PS(l, z^I, d)$ (line 7 of Algorithm 2). CheckSolve procedure checks if there is a box $\hat{b} \in PrevBox$ such that $b \subseteq \hat{b}$. If there is such box and it has no nondominated points inside, then b is not explored since it is also empty by Proposition 5.1, and we say b is eliminated by infeasibility. If there is such a box $\hat{b} \in PrevBox$ and the

nondominated point $n^{\hat{b}}$ was found in $\hat{b}$, then there is no need to explore box b by Proposition 5.1, and we say b is eliminated by optimality.

If b is not eliminated by infeasibility or optimality, the algorithm goes on as follows. We first check if $b^I = z^I$. If they are equal, we directly search the box. Otherwise, we ask the decision-maker to compare $\tilde{z}$ and b^I. If decision-maker prefers $\tilde{z}$, since any solution found in this region will be as good as b^I at most, and b^I is not preferred to $\tilde{z}$, we eliminate the region $\{y : \hat{z}^N \leq y \leq b^I\}$ (lines 9-14 of Algorithm 2). We perform this elimination by decomposing b into two smaller boxes and add those boxes to $\mathcal{B}$, which will be explained in detail later. Then, we select the next box from $\mathcal{B}$ and follow the same steps up to here.

If a box is to be explored, we solve $PS(l, z^I, d)$ for b. If the scalarization model is feasible, we denote the image of the solution found as n^b. Otherwise, b is empty and n^b is set as $n^b = \emptyset$. We update the $PrevBox$ accordingly (lines 16-24 of Algorithm 2).

If $n^b \neq \emptyset$, we add n^b to $\mathcal{Y}_N$ and update $\mathcal{B}$. NewBoxes procedure considers the current set of boxes in $\mathcal{B}$ that contains n^b including b, adds those boxes to list D of boxes to be deleted and splits them with respect to n^b (see lines 26-28 of Algorithm 2). Then, checks if the new boxes are redundant by Proposition 5.1 (see line 16 of Procedure 5 and isredundant procedure). It adds the boxes that are not redundant into the box list $\mathcal{B}$ (line 23 of Procedure 5). Figure 5.2 shows the new boxes after a new solution n^b is found. Note that the only region that is not covered by these boxes is the set of points that dominate n^b. This region can not contain any feasible points since n^b is nondominated.

Next, we ask the decision-maker about his preference between n^b and $\tilde{z}$, update $\tilde{z}$ accordingly and delete the boxes in D from $\mathcal{B}$ (lines 29-33 of Algorithm 2).

While defining a new box $\hat{b}$, we keep a variable $\hat{b}^c$ which keeps track of which

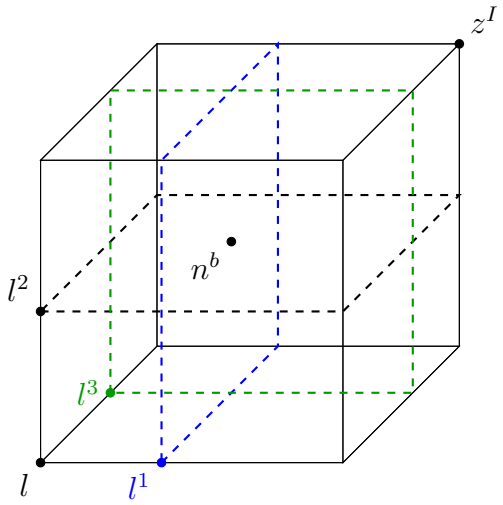


Figure 5.2: New boxes after a new solution n^b found

Nondominated point n^b divides the initial search space into three new boxes, b^1, b^2 and b^3. The points l^1, l^2, l^3 denote the lower bounds of b^1, b^2 and b^3, respectively. Note that the upper corner of each box is z^I.

component of n^b is used while the box was formed (lines 9-15 of NexBoxes procedure). There are 3 rules for $\hat{b}^c$ depending on the lower bound of the new box:

- If $\hat{l} = (n_1^b, l_2, l_3)$, then $\hat{b}^c = 1$
- If $\hat{l} = (l_2, n_2^b, l_3)$, then $\hat{b}^c = 2$
- If $\hat{l} = (l_1, l_2, n_3^b)$, then $\hat{b}^c = 3$

where b is the box that was searched and n^b is the new solution found in this box. For a better understanding, see Figure 5.3.

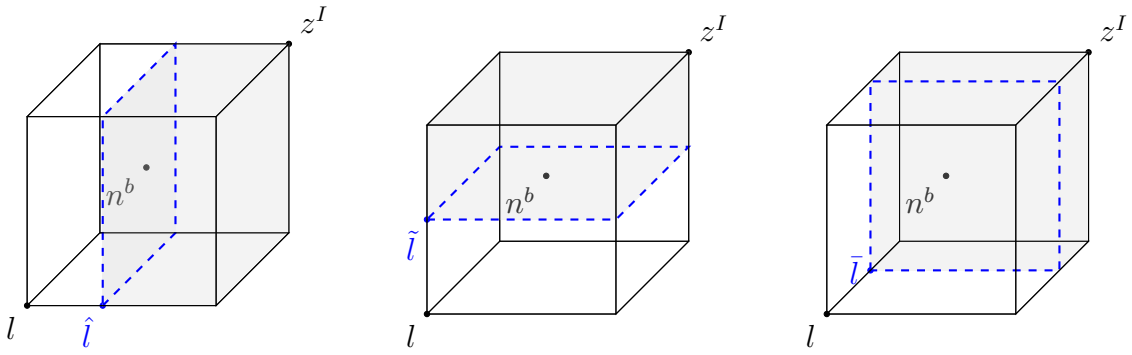


Figure 5.3: Finding the lower bounds of newly formed boxes

Figures show the three new boxes formed after a new solution n^b is found. Lower bound of each box is shown on the figures by $\hat{l}$, $\tilde{l}$ and $\bar{l}$. Note that each of these lower bounds share one component with n^b and two components with l. We keep information on the shared component through b^c.

Intersections of these boxes are noticable in Figure 5.3. As we have mentioned before, the parts that do not intersect with other boxes are individual subsets and they have their own individual upper bounds. Figure 5.4 shows individiual parts of the boxes in 5.3.

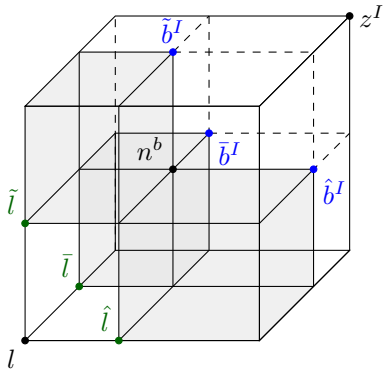


Figure 5.4: Individual parts of the boxes formed after n^b is found

We perform two operations on a box: splitting and decomposing. Splitting is performed after a newly found nondominated solution is within the box. We simply create child boxes (see line 27 of Algorithm 2 and lines 5-8 of the NewBoxes procedure). Decomposing is performed based on preference information. The reason we keep the individual upper bounds is to decide if a box b will be decomposed or not. Decomposing simply means eliminating the region of b that will not have a solution preferred to $\tilde{z}$. In other words, we diminish the search region of b by decomposing. When we eliminate the determined region from b, we obtain two intersecting boxes with different lower bounds (see line 12 of Algorithm 2 and decompose procedure). Figures 5.5 (for $b^c = 1$) and 5.6 (for $b^c = 2$ and $b^c = 3$) show the search region of boxes before and after decomposition.

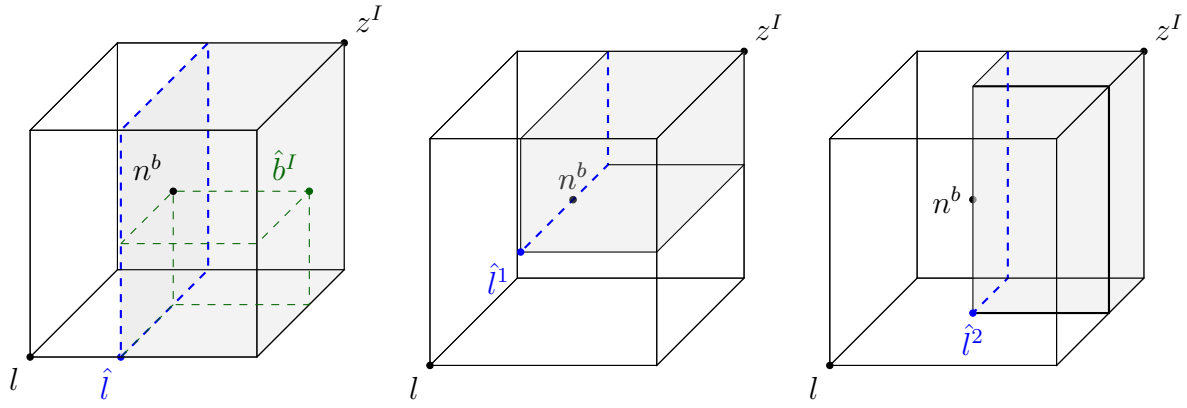


Figure 5.5: Decomposition of a box with $b^c = 1$

Since $\tilde{z} \succ \hat{b}^I$, the region $\{y \mid \hat{l} < y < \hat{b}^I\}$ can not contain the most preferred solution, hence eliminated, which results in boxes with lower corners $\hat{l}^1$ and $\hat{l}^2$. This is for a box with $b^c = 1$.

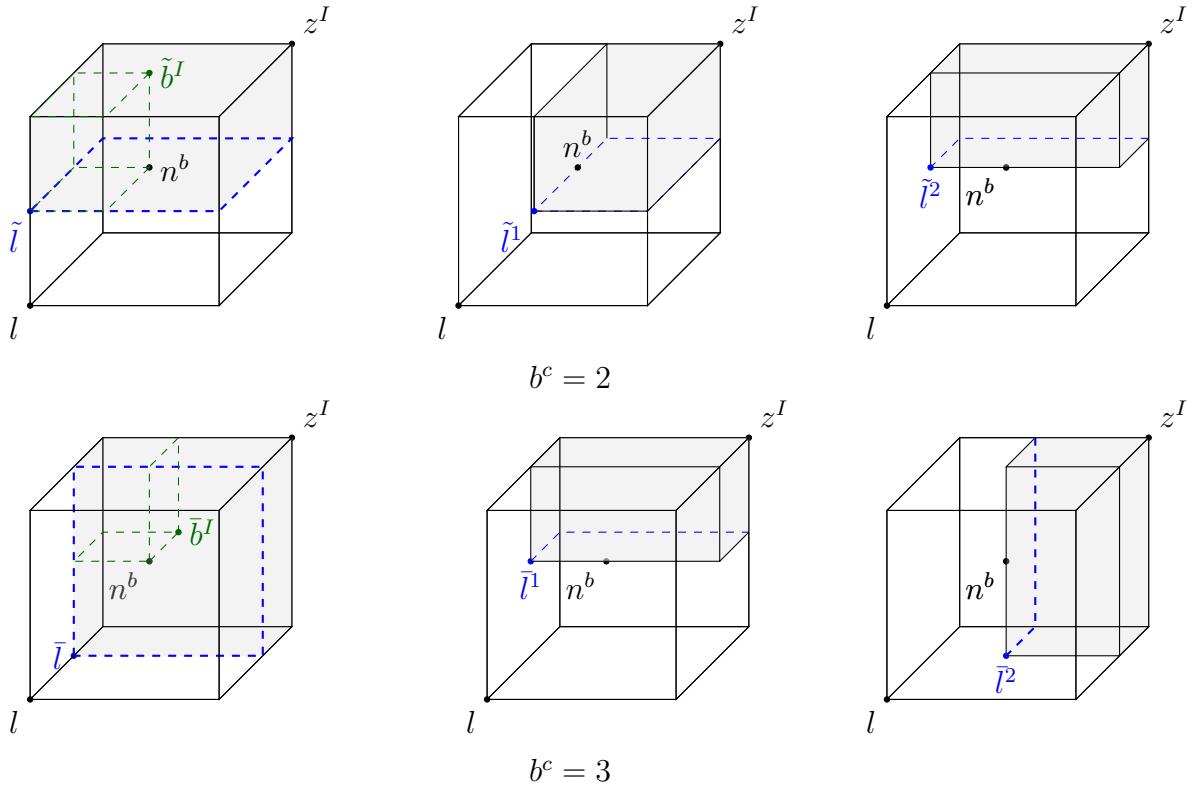


Figure 5.6: Decomposition of boxes with $b^c = 2$ and $b^c = 3$

If $b^I \neq z^I$, we decompose b into two boxes b^1 and b^2. We follow some rules to define new lower bounds and we utilize b^c to achieve this:

- If $b^c = 1$ then $l^1 = (l_1, l_2, u_3)$ and $l^2 = (l_1, u_2, l_3)$
- If $b^c = 2$ then $l^1 = (l_1, l_2, u_3)$ and $l^2 = (u_1, l_2, l_3)$
- If $b^c = 3$ then $l^1 = (l_1, u_2, l_3)$ and $l^2 = (u^1, l_2, l_3)$

If a box b is obtained through decomposition, we set its b^I as z^I (line 3 of decompose procedure), which also means that a box can not be decomposed more than once (lines 9-15 of Algorithm 2).

Algorithm 2 Proposed Interactive Algorithm for TOIP

Let $x^i \in \arg\max_{x \in X} z_i(x)$, $z_i^I = z_i(x^i)$, $\hat{z}_i^N = \min_{x \in X} z_i(x)$ for $i = 1, 2, 3$;
Let $b = b(\hat{z}^N, z^I)$, $b^I = z^I$, $b^c = 1$.
$\mathcal{B} = b$, $PrevBox = \emptyset$, $incumbent = \emptyset$, $\mathcal{Y}_N = \{z(x^1), z(x^2), z(x^3)\}$.
Decision-maker chooses the most preferred solution among $z(x^1), z(x^2), z(x^3)$. $\tilde{z}$ is initialized.
while $\mathcal{B} \neq \emptyset$ **do**
  Let $b \in \mathcal{B}$ and $D := \{b\}$.
  $[n^b, solve] \leftarrow CheckSolve(b, PrevBox)$
  if solve $= 1$ **then**
    if $z^I \neq b^I$ **then**
      (Ask the decision-maker to compare $\tilde{z}$ and b^I)
      if $\tilde{z} \succeq b^I$ **then**
        $\mathcal{B} \leftarrow$ decompose$(b, \mathcal{B}, b^I, b^c)$
        solve $= 0$
      end if
    end if
    if solve $= 1$ **then**
      Solve (PS(l, z^I, d));
      if The problem is feasible **then**
        Let x^* be an optimal solution;
        $n^b = z(x^*)$
        $PrevBox \leftarrow PrevBox \cup \{(b, n^b)\}$;
      else
        $PrevBox \leftarrow PrevBox \cup \{(b, \emptyset)\}$;
      end if
      if $n^b \neq \emptyset$ **then**
        $\mathcal{Y}_N \leftarrow \mathcal{Y}_N \cup \{n^b\}$
        $[\mathcal{B}, D] \leftarrow NewBoxes(\mathcal{B}, n^b, b^c)$;
        (Ask the decision-maker to make a comparison between n^b and $\tilde{z}$)
        if $n^b \succeq \tilde{z}$ **then**
          $incumbent = n^b$
        end if
      end if
    end if
  end if
  $\mathcal{B} \leftarrow \mathcal{B} \setminus D$;
end while

Procedure 3 CheckSolve(b, $PrevBox$)

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

Procedure 4 decompose($b, \mathcal{B}, b^I, b^c$)

Let $\hat{b}$ and $\bar{b}$ denote the parts of b after individual part is cut off.
$\hat{b} = b$, $\bar{b} = b$
$\hat{b}^I = \bar{b}^I = z^I$
if $b^c = 1$ **then**
  $\bar{l} = [l_1, l_2, u_3]$
  $\hat{l} = [l_1, u_2, l_3]$
else if $b^c = 2$ **then**
  $\bar{l} = [l_1, l_2, u_3]$
  $\hat{l} = [u_1, l_2, l_3]$
else if $b^c = 3$ **then**
  $\bar{l} = [l_1, u_2, l_3]$
  $\hat{l} = [u_1, l_2, l_3]$
end if
$\mathcal{B} \leftarrow \mathcal{B} \cup (\hat{b}, \bar{b})$
return $\mathcal{B}$

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

$D \leftarrow \emptyset$, Btemp := $\mathcal{B}$;
for $b \in \mathcal{B}$ **do**
  if $n^b \geq l$ **then**
    $D \leftarrow D \cup \{b\}$;
    for $j \in \{1, ..., p\}$ **do**
      if $z_j^I - n_j^b \geq 1$ **then**
        $\bar{l}_i \leftarrow l_i$ for $i \neq j$ and $\bar{l}_j \leftarrow n_j^b + \epsilon$
        Let $\bar{b}$ be a box with lower bound $\bar{l}$.
        if $\bar{l} = [z_1^I, n_2^b, n_3^b]$ **then**
          $\bar{b}^c = 1$
        else if $\bar{l} = [n_1^b, z_2^I, n_3^b]$ **then**
          $\bar{b}^c = 2$
        else if $\bar{l} = [n_1^b, n_2^b, z_3^I]$ **then**
          $\bar{b}^c = 3$
        end if
        if not isredundant($\bar{b}, \mathcal{B}, b$) **then**
          Btemp $\leftarrow$ Btemp $\cup$ $\{\bar{b}\}$;
        end if
      end if
    end for
  end if
end for
$\mathcal{B} \leftarrow$ Btemp;
return $[\mathcal{B}, D]$;

Procedure 6 isredundant($\bar{b}$,$\mathcal{B}$,b)

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

Chapter 6

Cone Based Approach

In this chapter, we propose a cone based approach to interactive algorithms that we have proposed. Recall that the algorithms that are presented in Chapters 4 and 5 are valid for all general monotone utility function types. However, cone based algorithm can only be used when the utility function is nondecreasing and quasi-concave.

Throughout the cone based approach, we keep the preference information obtained by the decision-maker in a list called *pref*. Each element in this list is denoted as (z^m, z^k) where z^m and z^k are the points that were compared by the decision-maker. Order of the points in the list are important and the first point is preferred to the second point, i.e. $z^k \preceq z^m$. We use the preference information obtained from the decision-maker to form two-point convex cones denoted by $C(z^m, z^k)$ and defined as:

$$C(z^m, z^k) = \{z \in \mathbb{R}^p \mid z = z^k + \mu(z^k - z^m), \mu \geq 0\}$$

Then, we utilize the following theorem:

Theorem 6.1. *[34, Theorem 1] Let $f : \mathbb{R}^p \to \mathbb{R}$ be a quasiconcave and nondecreasing function. Consider distinct points z^m, $z^k \in \mathbb{R}^p$ such that $f(z^k) < f(z^m)$. For any point $z^* \in \mathbb{R}^p$, consider the following linear programming problem:*

$$
\begin{aligned}
\max. \quad & \epsilon \\
s.t. \quad & \mu(z^k - z^m) - \epsilon e \geq z^* - z^k \qquad (\mathrm{CM}(z^m, z^k, z^*)) \\
& \mu \geq 0
\end{aligned}
$$

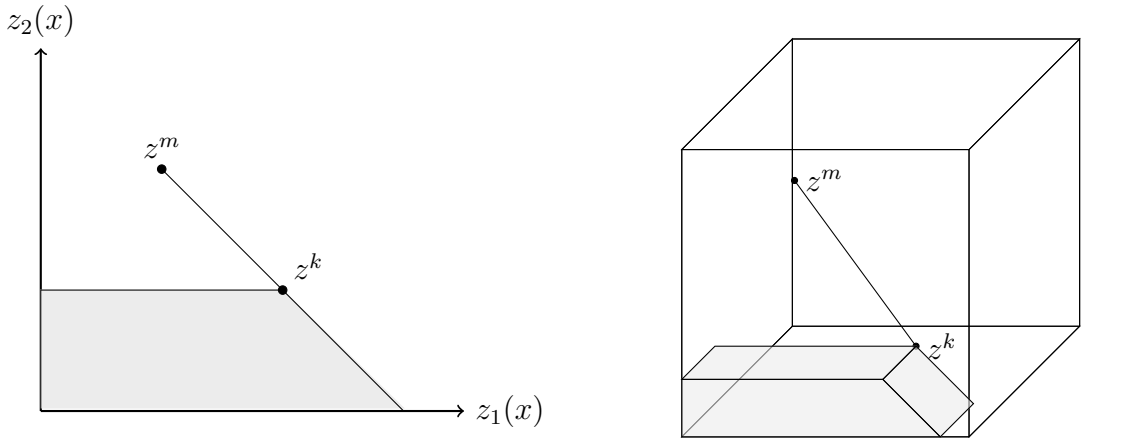
where e is a vector with all components equal to one. If the optimal value ϵ of $CM(z^m, z^k, z^*)$ is non-negative, then $f(z^k) \geq f(z^*)$.

A region defined by $C(z^m, z^k)$ is called cone dominated region and denoted as

$$CDR(z^m, z^k) = \{z \mid z \leq z^* \text{ for some } z^* \in C(z^m, z^k)\}.$$

In the model above, the first constraint checks if z^* is in one of the cone dominated regions by finding the maximum ϵ that satisfies the constraint. The second constraint gives a lower bound for μ.

It is known by Theorem 6.1 that any point that lies in the cone dominated region will not be preferred by the decision-maker. Therefore, we aim to eliminate some points and boxes without asking the decision-maker for preference information. Our purpose with this algorithm is to decrease the total number of questions asked to the decision-maker. Cone eliminated regions in bicriteria and tricriteria optimization settings can be seen in Figure 6.1.



(a) Cone dominated region in bicriteria optimization setting, $z^k \preceq z^m$

(b) Cone dominated region in tricriteria optimization setting, $z^k \preceq z^m$

Figure 6.1: Examples of cone eliminated regions

We note that the regions shown above are only one examples of each problem setting. Different shapes may occur for the regions depending on the positions of the points forming convex cones.

We provide the pseudocode of the cone based approach in Algorithms 7 and 8 for BOIP and TOIP, respectively. In bicriteria and tricriteria algorithms discussed in Chapters 4 and 5, we ask decision-maker to make preference between the incumbent and the new solution or between the incumbent and the individual ideal of the box. With the cone based approach, we first check if the point-to-be-asked lies within one of the cone dominated regions that were formed by the previous preferences of the decision-maker. To do so, we take the point to be checked and a pair from *pref* list. Using Theorem 6.1, we check if the point is in the cone eliminated region formed by the chosen pair. If not, we select the next pair from the *pref* list and check for the point again. We continue to select pairs from *pref* list until a cone eliminates the point or there is no cone left to check. Whenever we find such a pair, we stop checking procedure and eliminate the point (see Procedure 9). If the point that we checked is an upper corner of a box, we directly eliminate the box without searching and move on with the new box in $\mathcal{B}$ (lines 23-30 of Algorithm 7 for BOIP, lines 11-20

of Algorithm 8 for TOIP).

If it is a nondominated solution, we eliminate the point before asking decision-maker to make a comparison between the point and incumbent (lines 16-22 of ExploreCone procedure for BOIP and 33-34 of Algorithm 7 for TOIP). If the point can not be eliminated by any of the cones formed by the pairs in *pref* list, we ask the decision-maker's preference. Note that *pref* list is updated whenever the decision-maker makes a preference and when a point is eliminated by a cone.

We note that adding cone elimination model to the algorithm increases the number of models solved to find the most preferred solution. Increased number of models causes the solution time to increase. Therefore, we implement a variant for the cone based approach where we benefit from the inequalities to determine the cone dominated region [12].

6.1 Cone Dominated Region with Inequalities

Instead of solving the cone elimination model $CM(z^m, z^k, z^*)$, we can write the inequalities that define the cone dominated region and check if a point is in that region. Let z^m and z^k be two nondominated points and $z^m \succeq z^k$. To check if point z^* is eliminated by the cone generated by z^m and z^k, we use the following theorem which is valid for all p-criteria problems.

Theorem 6.2. *[12, Theorem 2] Let $f : \mathbb{R}^p \to \mathbb{R}$ be a quasiconcave and nondecreasing function. Consider two distinct nondominated points z^m and z^k such that $f(z^k) < f(z^m)$. Then, a point z^* is in the cone dominated region defined by z^m and z^k if and only if the following conditions hold:*

1. $z_i^* \leq z_i^k \qquad \forall i \in S_{\leq}^{m,k}$,

2. $z_i^*(z_j^k - z_j^m) + z_j^*(z_i^m - z_i^k) \leq z_j^k z_i^m - z_i^k z_j^m \qquad \forall i \in S_{\leq}^{m,k}, \quad \forall j \in S_{>}^{m,k},$

where $S_{\leq}^{m,k} = \{i : z_i^k - z_i^m \leq 0\}$ *and* $S_{>}^{m,k} = \{j : z_j^k - z_j^m > 0\}$.

Theorem 6.2 shows that instead of solving $CM(z^m, z^k, z^*)$, one can check the inequalities given in Theorem 6.2.1 and 6.1.2 in order to decide if z^* can be eliminated.

Algorithm 7 Cone Based Approach to BOIP

Let $x^i \in \arg\max_{x \in X} z_i(x)$, $z_i^I = z_i(x^i)$, $z_i^N = (z_1(x^2), z_2(x^1))$
Let $u = z^I$ and $l = z^N$ be the upper and lower bounds of box b, respectively.
$\mathcal{B} = b$, $\tilde{z} = \emptyset$,
$pref = \emptyset$, eliminate $= 0$.
$\mathcal{Y}_N = \{z^1, z^2\}$
(Ask the decision-maker to give preference information between z^1 and z^2)
if $z^1 \succeq z^2$ **then**
  $\tilde{z} = z^1$
  $pref \leftarrow pref \cup \{(z^1, z^2)\}$
else
  $\tilde{z} = z^2$
  $pref \leftarrow pref \cup \{(z^2, z^1)\}$
end if
while $\mathcal{B} \neq \emptyset$ **do**
  Let $b \in \mathcal{B}$
  $\mathcal{B} \leftarrow \mathcal{B} \setminus \{b\}$
  if $\mid u_1 - l_1 \mid > 1$ and $\mid u_2 - l_2 \mid > 1$ **then**
    if $\tilde{z}_1 = l_1$ or $\tilde{z}_2 = u_2$ **then**
      $eliminate = 0$
    else
      $[pref, eliminate] \leftarrow ConeSearch(pref, u)$
    end if
    if eliminate $= 0$ **then**
      (Ask the decision-maker to compare the box ideal with the incumbent)
      if $u \succeq \tilde{z}$ **then**
        $pref \leftarrow pref \cup \{(u, \tilde{z})\}$
        $\mathcal{B} \leftarrow ExploreCone(\mathcal{B}, b, d, \tilde{z})$
      else
        $pref \leftarrow pref \cup \{(\tilde{z}, u)\}$
      end if
    end if
  end if
end while

Procedure 7 ExploreCone($\mathcal{B}, b, d, \tilde{z}$)

eliminated = 0
Solve $PS_1(l, u, d)$.
if the problem is feasible **then**
  Let x^* be the optimal solution.
  Solve $PS_2(z(x^*))$. Let $x^{'}$ be the optimal solution.
  if $l < z(x^{'}) < u$ **then**
    $\mathcal{Y}_N \leftarrow \mathcal{Y}_N \cup \{z(x^{'})\}$
    $[pref, eliminate] \leftarrow ConeSearch(pref, u)$
    if eliminated = 0 **then**
      if $z^{'} \succeq \tilde{z}$ **then**
        $pref \leftarrow pref \cup \{(z^{'}, \tilde{z})\}$
        $\tilde{z} = z^{'}$
      else
        $pref \leftarrow pref \cup \{(\tilde{z}, z^{'})\}$
      end if
      Let $\hat{b} = (l_1, z_2(x^{'}), z_1(x^{'}), u_2)$ and $\bar{b} = (z_1(x^{'}), l_2, u_1, z_2(x^{'}))$ be the new boxes.
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\hat{b}, \bar{b}\}$
    end if
  end if
end if
return $\mathcal{B}$

Algorithm 8 Cone Based Approach to TOIP

Let $x^i \in argmax_{x \in X} z_i(x)$, $z_i^I = z_i(x^i)$, $\hat{z}_i^N = min_{x \in X} z_i(x)$ for $i = 1, 2, 3$;
Let $b = b(\hat{z}^N, z^I)$, $b^I = z^I$, $b^c = 1$.
$\mathcal{B} = b$, $PrevBox = \emptyset$, $incumbent = \emptyset$, $\mathcal{Y}_N = \{z(x^1), z(x^2), z(x^3)\}$, $pref = \emptyset$.
Decision-maker chooses the most preferred solution among $z(x^1), z(x^2), z(x^3)$. $\tilde{z}$ is initialized. *pref* is updated accordingly.
while $\mathcal{B} \neq \emptyset$ **do**
    Let $b \in \mathcal{B}$ and $D := \{b\}$.
    $[n^b, solve] \leftarrow CheckSolve(b, PrevBox)$
    if solve = 1 **then**
        if $z^I \neq b^I$ **then**
            [*pref*, solve] $\leftarrow$ ConeSearch(*pref*,b^I)
            if solve = 1 **then**
                (Ask the decision-maker to search the box)
                if $\tilde{z} \succeq b^I$ **then**
                    $pref \leftarrow pref \cup (\tilde{z}, b^I)$
                    $\mathcal{B} \leftarrow$ decompose$(b, \mathcal{B}, b^I, b^c)$
                    solve = 0
                else
                    $pref \leftarrow pref \cup (b^I, \tilde{z})$
                end if
            end if
        end if
        if solve = 1 **then**
            Solve (PS(l, z^I, d))
            if The problem is feasible **then**
                Let x^* be an optimal solution, $n^b = z(x^*)$
                $PrevBox \leftarrow PrevBox \cup \{(b, n^b)\}$;
            else
                $PrevBox \leftarrow PrevBox \cup \{(b, \emptyset)\}$;
            end if
            if $n^b \neq \emptyset$ **then**
                $\mathcal{Y}_N \leftarrow \mathcal{Y}_N \cup \{n^b\}$
                $[\mathcal{B}, D] \leftarrow NewBoxes(\mathcal{B}, n^b, b^c)$;
                check = 0
                [*pref*, check] $\leftarrow$ ConeSearch(*pref*, n^b)
                if check = 1 **then**
                    (Ask the decision-maker to give preference information)
                    if $n^b \succeq \tilde{z}$ **then**
                        $\tilde{z} = n^b$
                        $pref \leftarrow pref \cup (n^b, \tilde{z})$
                    else
                        $pref \leftarrow pref \cup (\tilde{z}, n^b)$
                    end if
                end if
            end if
        end if
    end if
    $\mathcal{B} \leftarrow \mathcal{B} \setminus D$;
end while

Procedure 9 ConeSearch(*pref*, z^*)

1: **for** each (z^m, z^k) pair in *pref* **do**
2: Solve CM(z^m, z^k, z^*) (or check the inequalities)
3: **if** $\epsilon^* \geq 0$ (or in one of the cone dominated regions) **then**
4: *pref* $\leftarrow$ *pref* $\cup$ (z^k, z^*)
5: **return** [*pref*, 1];
6: **end if**
7: **end for**
8: **return** [*pref*, 0];

Chapter 7

Computational Experiments

We tested the performances of the proposed algorithms on two sets of problem instances. For the interactive algorithm for BOIP, there are instances of bicriteria assignment problem (B-AP) with 200 $\times$ 200 (A,B) binary variables and instances of bicriteria knapsack (B-KP) problem with 375 (A) and 500 (B) binary variables. There are 5 instances in each set (A) and (B) for both problems. For the TOIP, we consider the sets of tricriteria assignment problem (T-AP) with 10 $\times$ 10 (A), 20 $\times$ 20 (B) and 30 $\times$ 30 (C) binary decision variables and tricriteria knapsack problem (T-KP) with 20 (A), 40 (B), 60 (C), 80 (D) and 100 (E) binary variables with 10 instances in each set.

Knapsack problem instances consist of 0-1 binary decision variables (x). In these problems, we select among the given items each having characteristics and values. The objective is to maximize the total value of the selected items while considering the characteristic constraints. Let v_i^k be the value of the item for k^{th} objective and w_i^m be the m^{th} characteristic value of item i, where there are n items, p objectives and t characteristics for the items. W^m denotes the upper limit for the total value of

the m^{th} characteristic of the items included in knapsack. The problem is modelled as follows:

$$\begin{aligned}
\text{max.} \quad & \sum_{i=1}^{n} v_i^k x_i && k = 1, \ldots, p \\
\text{s.t.} \quad & \sum_{i=1}^{n} w_i^m x_i \leq W^m && m = 1, \ldots, t \\
& x_i \in \{0, 1\} && i = 1, \ldots, n
\end{aligned}$$

$x_i = 1$ if item i is selected and 0 otherwise.

Assignment problems consist of a number of tasks and agents to perform these tasks. The problem is to assign tasks to agents in a way that maximizes the profit. Let n be the number of tasks and agents, v_{ij}^k is the profit in the k^{th} objective obtained from assigning agent i to task j, and p is the number of objectives. The mathematical model is as follows:

$$\begin{aligned}
\text{max} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij}^k x_{ij} && k = 1, \ldots, p \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1 && j = 1, \ldots, n \\
& \sum_{j=1}^{n} x_{ij} = 1 && i = 1, \ldots, n \\
& x_{ij} \in \{0, 1\} && i, j = 1, \ldots, n
\end{aligned}$$

$x_{ij} = 1$ if agent i is assigned to task j, 0 otherwise.

We have coded the algorithms in MATLAB and solved the mathematical models with CPLEX 12.8. All of the experiments on the algorithms are done on a computer with Intel Xeon CPU E5-1650 3.6 GHz processor and 64 GB RAM. All computation times are provided in (CPU) seconds.

Throughout the chapter, the utility functions we use for simulating the decision-maker's responses are as follows:

$$(L) \qquad u(z) = \sum_{i=1}^{p} z_i \qquad \text{(linear)}$$

$$(Q) \qquad u(z) = -\sum_{i=1}^{p} (z_i - z^I)^2 \qquad \text{(quadratic)}$$

$$(T) \qquad u(z) = -\max_i (z_i - z^I) \qquad \text{(Tchebychev)}$$

These functions are used in the literature for testing purposes [13, 33].

7.1 Computational Results of Interactive Algorithms for BOIP

We first present the results of the computational experiments of the interactive algorithm proposed for BOIP and the cone approach on assignment and knapsack problems. For each of the utility functions above, we tested the Interactive Algorithm (IA) and the Cone Based Approach (CBA). The results for BOIP are shown in Table 7.1.

$\mathcal{N}$ shows the average number of total number of nondominated solutions for each set of instances. NS and PS denote the number of nondominated solutions found and

number of Pascoletti-Serafini scalarization models solved until the most preferred solution by the decision-maker is obtained. Change denotes the number of changes of the incumbent solution throughout the algorithm. These three metrics are the same for IA and the CBA for each problem instance. The rest of the variables reflect the average differences of the performances of the algorithms. Time is the solution time of the algorithms in seconds, BoxQ is the number of interactions made with the decision-maker to ask if a box should be explored or not, IncQ is the number of questions asked to the decision-maker to make a preference between the incumbent and newly found solution. ElDirect denotes the number of boxes that are eliminated by a direct interaction with the decision-maker, i.e. through pairwise comparison questions asked. Note that in the cone based approach, indirect elimination is also possible through cones. The amount of such eliminations can be deduced from the difference between IA and CBA in terms of ElDirect. Finally, NIneq is only valid for Cone Based Approach and it shows how many cones are checked in average.

Table 7.1: Computational results of interactive algorithm for BOIP on assignment and knapsack instances

Problem	$\mathcal{N}$	Utility	NS	PS	Change	Alg.	Time	BoxQ	IncQ	ElDirect	NIneq
B-AP	A: 708.4	L	87.8	94.8	4.6	IA	342.67	116.8	86.8	38.8	-
						CBA	346.24	83.4	16.0	5.4	13341.0
		Q	62.4	66.4	2.6	IA	245.22	76.2	61.4	26.8	-
						CBA	248.74	69.8	32.4	20.4	8397.2
		T	18.4	17.0	1.0	IA	77.68	14.0	17.4	14.0	-
						CBA	85.63	13.8	17.4	13.8	478.4
	B: 1416.2	L	119.6	137.8	4.4	IA	551.36	176.6	118.6	58.2	-
						CBA	557.45	128.4	21.4	10.0	29863.4
		Q	78.6	91.0	4.6	IA	371.09	109.8	77.6	38	-
						CBA	374.99	98.4	33.8	26.6	15305
		T	20.6	19.6	1.2	IA	95.81	15.8	19.6	15.8	-
						CBA	98.31	15.8	19.4	15.8	609.8
B-KP	A: 975.4	L	97.0	105.6	3.4	IA	58.06	168.0	96.0	78.0	-
						CBA	58.82	101.6	13.6	11.6	22029.4
		Q	56.4	59.6	2.2	IA	34.95	89.0	55.4	45.0	-
						CBA	36.17	74.8	21.6	30.8	9473.0
		T	17.4	17.0	1.2	IA	9.20	14.0	16.4	14.0	-
						CBA	9.91	13.6	16.4	13.6	454.4
	B:1539.4	L	129.4	142.6	5.4	IA	106.09	229.2	128.4	101.8	-
						CBA	108.17	133.4	18.0	6.0	39707.2
		Q	70.4	76.2	3.0	IA	59.29	115.6	69.4	55.8	-
						CBA	60.19	95.6	24.8	35.8	14830.6
		T	19.8	19.6	1.4	IA	14.07	16.0	18.8	16.0	-
						CBA	14.41	15.8	18.8	15.8	588.8

Note that the main difference between the two algorithms is the use of cones so as to make indirect eliminations. Hence the number of solutions found (NS) and the number of models solved (PS) are the same in both algorithms. We see that the algorithms return the most preferred solution by finding a small fraction of the whole set of nondominated points. For set A, the number of solutions found is in the 1.78% - 9.94% range of total solutions, and for set B, it is in the 1.28% - 8.41% range, considering all utility functions 1, 2 and 3. It shows that, both IA and CBA work efficiently in terms of number of solutions found.

We can see in the tables above that the number of questions asked to the decision-maker before exploring a box increases as the problem size gets larger. CBA eliminates some of those questions by determining the boxes with an upper bound in cone dominated regions. Therefore, we see a significant decrease in BoxQ for CBA. In IA, it is seen that IncQ = NS - 1. This is due to asking a preference question whenever

a new solution is found. When we compare IA and CBA, we see that the number of interactions with the decision-maker is much less for the CBA than it is for the IA, especially for the linear and quadratic utility functions. In the worst case, we know that we can find all nondominated solutions ($\mathcal{N}$) first and then ask decision maker to make comparisons by $\mathcal{N}-1$ pairwise comparison questions to find the most preferred solution. Therefore, both IA and CBA work well in terms of asking less questions.

The number of eliminations done directly by the decision-maker (ElDirect) decreases as well in line with the decrease in the number of questions asked to the decision-maker. The decrement is less for the experiments with Tchebychev utility function because the most preferred solution is usually found in the early steps of the algorithm. The reason for this is PS scalarization returns the point with minimum (unweighted) Tchebychev distance to the upper bound, which is the ideal point at the beginning. This leads to the elimination of many boxes before exploring them, hence the algorithm stops in a short time (which can be seen in *Time* column), before collecting enough amount of preference information to start elimination by convex cone method. It can also be seen on the *Change* column of the Table 7.1 that for Tchebychev utility function, the average incumbent changes is very close to 1, which means that the most preferred solution is found in the early steps and it does not change again until the algorithm stops. For the linear and quadratic functions, the average number of incumbent changes is higher. When we compare the solution times of the algorithms, even though CBA checks the cone dominated regions (it can be seen that the number of checks are high, see NIneq column, the solution times are almost the same for both IA and CBA.

Moreover, we compare the solution time of the algorithms with the solution time of finding the set of all nondominated solutions. If we take out the interactive part of our proposed IA, the remaining part becomes an a posteriori algorithm to solve BOIP. The average time (in seconds) to find all nondominated points are given in

Table 7.2 for each instance set A and B for B-AP and B-KP. SolTime shows the time to find all nondominated solution set, IATime and CBATime show the solution times of IA and CBA, respectively. Note that, we consider solution times of all utility functions to find the minimum, average and maximum of solution times. To see the time range, we provide minimum, average and maximum of average solution times for both IA and CBA. *Range* shows the range of the solution times of the algorithms compared to finding the whole solution set. Table 7.2 shows that there is a significant decrease in the solution time of the algorithms compared to finding all nondominated solutions.

Table 7.2: Solution time comparison for finding the whole nondominated set and the proposed algorithms

Problem	Set	SolTime	IATime				CBATime			
			Min	Avg	Max	Range (%)	Min	Avg	Max	Range (%)
B-AP	A	3065.34	77.68	221.85	342.67	2.5 - 11.17	85.63	226.87	346.24	2.79 - 11.29
	B	6220.02	95.81	339.42	551.36	1.54 - 8.86	98.31	343.59	557.45	1.58 - 8.96
K-AP	A	642.59	9.2	34.07	58.06	1.43 - 9.03	9.91	34.96	58.82	1.54 - 9.15
	B	1518.89	14.07	59.81	106.09	0.92 - 6.98	14.41	60.92	108.17	0.94 - 7.12

In Chapter 6, we explain that CBA can be used only when the utility function of the decision-maker is quasiconcave. We tested CBA with a non-quasiconcave utility function in order to see how much utility value of the preferred solution would deviate from the actual one. We consider the following non-quasiconcave underlying utility function:

$$u(z) = \sum_{i=1}^{p} z_i^2$$

and implement CBA. Let the solution returned by the algorithm be z' and true best solution be z^*. We then calculate the deviation as follows:

$$\frac{u(z^*) - u(z')}{u(z^*)} \times 100$$

For the assignment problems, the average deviation from the actual utility value is found to be 0.01% and 0% for sets A and B, respectively. For the knapsack problems,

the average deviation is 2.58% for set A and 1.87% for set B. We conclude that there is a possibility of CBA to work correct for non-quasiconcave utility functions but it is not always the case. One can use CBA for non-quasiconcave utility functions as well, if finding the most preferred solution is not the main concern.

7.2 Computational Results of Interactive Algorithms for TOIP

We present similar results for the proposed interactive algorithm for TOIP in the Tables 7.3 and 7.4. Notations and utility functions are the same with the previous section.

Table 7.3: Computational results of interactive algorithm for TOIP on assignment instances

$\mathcal{N}$	Utility	NS	PS	Change	Algorithm	Time	BoxQ	IncQ	ElDirect	NIneq
A: 171.7	L	114.2	295	3.2	IA	9.773	260.4	113.2	41.1	-
					CBA	9.707	240.6	39.8	21.3	64836.4
	Q	65.7	174.1	2.7	IA	5.268	137	64.7	45.9	-
					CBA	5.188	114.6	30.8	23.5	19152.2
	T	15.3	41.9	1.6	IA	0.873	20.7	14.3	20.7	-
					CBA	0.864	14.2	6.8	14.2	506.5
B: 1860.5	L	1024.5	2447.6	3.5	IA	212.692	2300.2	1023.5	206.3	-
					CBA	208.777	2157.3	155.4	63.4	4370058
	Q	426.2	1064.7	2.2	IA	88.948	904.9	425.2	223.1	-
					CBA	87.683	735	135.6	53.2	739617.4
	T	44.5	120	0	IA	7.723	67.5	43.5	67.5	-
					CBA	7.708	29.3	6.7	29.3	2957.2
C: 6181.3	L	3352.6	7716.6	7.5	IA	1470.71	7289.2	3351.6	607.7	-
					CBA	1462.427	6800.5	249.6	119	40067346
	Q	1353.3	3308	3.9	IA	600.879	2810.5	1352.3	702.2	-
					CBA	597.191	2199.7	246	91.4	6124525
	T	73.3	194.9	1.1	IA	27.783	111.2	72.3	111.2	-
					CBA	27.628	44.6	6.6	44.6	6630

Table 7.4: Computational results of interactive algorithm for TOIP on knapsack instances

$\mathcal{N}$	Utility	NS	PS	Change	Alg.	Time	BoxQ	IncQ	ElDirect	NIneq
A: 38	L	23.7	69.5	1.6	IA	1.107	53.3	22.7	13.3	
					CBA	1.172	49.9	13.7	9.9	3185.8
	Q	17.1	49.3	1.7	IA	0.625	32.2	16.1	12.9	
					CBA	0.618	29	12	9.7	1574.6
	T	7.5	20.5	1.1	IA	0.19	8.2	6.5	8.2	
					CBA	0.196	7	5.5	7	118.3
B: 311.2	L	204.7	607.5	2.5	IA	23.686	560.6	203.7	57.2	
					CBA	23.607	534.2	43.7	30.8	268029.6
	Q	95.4	95.4	2.3	IA	9.754	230.9	94.4	65.3	
					CBA	9.657	200.4	37.8	34.8	51959.2
	T	13.2	37.4	1.3	IA	0.822	17.8	12.2	17.8	
					CBA	0.832	13.1	6.9	13.1	394.9
C: 917.1	L	587.4	1745.1	3.8	IA	141.628	1651.7	586.4	130	
					CBA	142.151	1578.3	89.1	56.6	2272722
	Q	248.1	735.9	2.5	IA	55.01	649.7	247.1	132.7	
					CBA	54.819	568.8	81.9	51.8	419056.5
	T	21.1	61	1.7	IA	2.398	31.8	20.1	31.8	
					CBA	2.446	18.7	7.2	18.7	966.7
D: 2295.8	L	1505.4	4449.2	5.2	IA	782.803	4280.5	1504.4	232.4	
					CBA	781.586	4134	146.5	85.9	15007654
	Q	605.7	1787.9	4.1	IA	320.45	1630.5	604.7	245.4	
					CBA	320.629	1468.5	144.9	83.4	2488973
	T	29.8	86.9	1.4	IA	5.188	46.8	28.8	46.8	
					CBA	5.159	26.9	8.2	26.9	1832.6
E: 5849	L	3881.7	11479	7.7	IA	4658.217	11200.5	3880.7	401.3	
					CBA	4656.291	10930.6	287.8	131.4	111212524
	Q	1547.9	4570.3	4.9	IA	1968.745	4245.3	1546.9	526.1	
					CBA	1961.065	3852.9	270.8	133.7	17354015
	T	50	147.3	1.3	IA	22.764	79.2	49	79.2	
					CBA	22.683	37.8	9.6	37.8	4034.4

Similar conclusions to the biobjective approach can be made in triobjective case. We see that NS is quite low compared to $\mathcal{N}$, especially when the utility function is Tchebychev. The solution times are close for both algorithms even though CBA checks many cone inequalities. CBA decreases the number of interactions with the decision-maker when compared to IA for TOIP.

Finally, in Table 7.5, the average solution time (seconds) to find all nondominated solutions and average solution times of IA and CBA are higher compared. Solution times for IA and CBA are compared to the BOIP case due to increased number of boxes to be explored. However, we can still see an improvement on the times.

Table 7.5: Average solution time comparison for finding the whole nondominated set and the proposed algorithms

Problem	Set	SolTime	IATime				CBATime			
			Min	Avg	Max	Range (%)	Min	Avg	Max	Range (%)
T-AP	A	22.06	0.87	5.3	9.73	3.95 - 44.30	0.84	5.25	9.70	3.91 - 44.00
	B	423.867	7.72	103.12	212.69	1.82 - 50.17	7.70	101.38	208.77	1.81 - 49.25
	C	2770.118	27.78	699.79	1470.71	1.00 - 53.09	27.62	695.74	1462.42	0.99 - 52.79
T-KP	A	1.83	0.19	0.64	1.10	10.33 - 60.20	0.19	0.66	1.17	10.66 - 63.74
	B	33.51	0.82	11.42	23.68	2.45 - 70.68	0.82	11.37	23.60	2.48 - 70.44
	C	195.72	2.39	66.35	141.62	1.22 - 72.35	2.44	66.47	142.15	1.25 - 72.62
	D	1008.07	5.18	369.48	782.80	0.51 - 5.15	5.15	369.12	781.58	0.51 - 77.53
	E	5995.393	22.76	2216.58	4658.21	0.38 - 77.69	22.68	2213.35	4656.29	0.37 - 77.66

A potential drawback of the algorithms that we propose for TOIP is that they may require too much interaction with the decision-maker even when CBA is utilized. As the size of the problem set gets larger, like B and C of T-AP and D and E of K-AP, the number of preference questions increases as well. In some cases, we can see that the total interactions with the decision-maker exceeds the worst case, which is $\mathcal{N}-1$. The reason is, as we previously explained in Chapter 5, whenever we have a box to explore, we first check its individual upper bound and if it is not equal to the ideal point of the problem, we decompose the box into three new boxes. We learn the decision-maker's preference on the upper bounds of the boxes before we explore them. Therefore, the number of boxes triples, and the algorithms may end up asking too many questions to the decision-maker. The number of preference questions also increases as the problem size gets larger. So, we explored two ideas to decrease the number of questions asked in the following sections 7.2.1 and 7.2.2.

7.2.1 Asking Based on the Volumes of the Box

In order to decrease the number of questions asked to explore a box or not (BoxQ), we propose a volume based approach (Volume). We ask the decision-maker to compare the incumbent and the individual upper bound of the box if the volume of the box to be explored is between 0.05% and 0.1% of the volume of the initial box, otherwise we

directly explore the box without asking the decision-maker. The underlying reasoning for this method is that when a box is too small, it is possibly empty, therefore asking a question for an empty box is useless. When the box is too large, upper bound of the box is usually attractive hence would not be eliminated anyway. We experiment this approach on a set of tricriteria assignment and knapsack problems. The results are shown in Table 7.6

Table 7.6: Comparison of the CBA with the proposed volume approach

Problem	$\mathcal{N}$	Utility Function	Approach	NS	BoxQ	IncQ
T-AP	B: 1860.5	L	CBA	1024.5	2157.3	155.4
			Volume	899.8	6.8	67.2
		Q	CBA	426.2	735	135.6
			Volume	831.3	2.7	73.1
		T	CBA	44.5	29.3	6.7
			Volume	793.6	2.0	65.5
T-KP	C: 6181.3	L	CBA	587.4	1578.3	89.1
			Volume	764.2	19.6	120.7
		Q	CBA	248.1	568.8	81.9
			Volume	591.2	15.4	112.8
		T	CBA	21.1	18.7	7.2
			Volume	202.9	2.0	25.1

We can see that the number of questions asked to the decision-maker to explore a box (BoxQ) decreases significantly compared to the CBA. Average number of solutions found (NS) mostly increases with a few exceptions. The reason is, with the new approach we do not ask most of the questions that we ask in CBA and do not obtain the corresponding preference information. Hence, we may search some boxes that would not be searched in CBA and obtain some solutions that will not be preferred by the decision-maker. When we ask the decision-maker to compare these solutions with the incumbent, we obtain different preference information from the one that we would obtain with CBA. In some instances, we eliminate more solutions with this preference information and we end up with a smaller NS value. Relatively, the number of pairwise comparison questions (IncQ) asked to the decision-maker decreases too. However, this is not the case for all instances. Especially when the utility function is Tchebychev, NS and IncQ increases significantly. Hence, it is not possible to have a general conclusion in terms of NS and IncQ. However, we conclude that this variant works efficiently in terms of decreasing BoxQ.

7.2.2 Weight Estimation Model

In order to decrease the number of preference questions (IncQ) asked to the decision-maker, we utilize an estimation for the utility function of the decision-maker to decide which pairwise comparison questions are worth asking. We utilize the following weight assumption model [44]:

$$
\begin{aligned}
\text{max.} \quad & \epsilon \\
\text{s.t.} \quad & \sum_{i=1}^{p} w_i z_i^m \geq \sum_{i=1}^{p} w_i z_i^k + \epsilon \qquad \forall (z^m, z^k) \in \text{pref} \\
& \sum_{i=1}^{p} w_i = 1 \\
& w_i \geq \epsilon \qquad \forall i = 1, \ldots, p
\end{aligned}
$$

where p is the number of criteria, *pref* is the list of preference information obtained from the decision-maker, and w_i is the weight of each criteria in the utility function. The objective function of the model aims to find the ϵ that maximizes the utility differences, i.e., find a point that is far from the closest weight constraint. The first constraint provides the preference of the decision-maker in terms of linear utility function. The second constraint adds weights of the objectives up to 1. The last constraint provides a lower bound on the weights.

We propose two variants for this method. For the first variant (Variant 1), in the first iterations of the algorithm, we ask the decision-maker preference questions and obtain information. After sufficient amount of information obtained, we solve the model above and find the estimated weights (w_i) for all $i = 1, \ldots, p$. For the rest of the algorithm, we estimate the utilities of the newly found solutions based on the latest weights. If the estimated utility is greater than the utility of the incumbent solution, we ask the decision-maker to make a pairwise comparison between the current solution and the incumbent. According to the decision-maker's answers, we update the weights by solving the model above. If the estimated utility of a solution is less than the utility of the incumbent, we take this solution into a solution pool and go on, i.e. we do not ask a pairwise comparison question to the decision-maker. When the algorithm stops, we calculate the estimated utilities of the solutions in the solution pool with the final weights. We take the solution with the maximum estimated utility and ask the decision-maker to make one last comparison between

that solution and incumbent.

For the second variant (Variant 2), we limit the number of questions that will be asked to the decision-maker. After the limit is reached, we estimate the weights of the utility function using the information obtained from the decision-maker. For the rest of the algorithm, whenever we need a pairwise comparison, we estimate the utilities and decide based on these estimations. We return the solution with the highest estimated utility as the most preferred solution for the decision-maker.

We test the variants on a set of instances of each of knapsack and assignment problems with the following utility functions:

$$(L): \qquad u(z) = 0.5z_1 + 0.3z_2 + 0.2z_3 \qquad \text{(linear)}$$

$$(C): \qquad u(z) = z_1^2 + z_2^2 + z_3^2 \qquad \text{(convex)}$$

We prefer to experiment on a linear utility function to see how the variants work since we have a linearity assumption, and on a convex utility function to see how much variants deviate from the true utility when the assumption does not hold.

We do not limit the number of questions in the first variant but limit it by 10 for the second variant and provide the results for both variants along with the consideration of the box volumes before asking a question in Table 7.7. Deviation shows how much of the utility of the most preferred solution found by a variant deviates from the utility of the true best solution.

Table 7.7: Comparison of the CBA with the proposed variants

Problem	$\mathcal{N}$	Utility Function	Variant	NS	BoxQ	IncQ	Deviation (%)
T-KP	837	L	CBA	464.6	1235.6	79.6	-
			Variant 1	690.0	18.9	21.9	0.39
			Variant 2	790.7	18.9	10.0	0.22
		C	CBA	566.0	1515.7	78.1	-
			Variant 1	710.6	18.9	19.5	2.48
			Variant 2	837.7	19.6	10.0	1.37
T-AP	2404	L	CBA	798.8	1650.2	140.8	-
			Variant 1	1294.1	11.6	19.6	2.08
			Variant 2	1513.7	18.0	10.0	7.57
		C	CBA	1120.6	2328.4	154.1	-
			Variant 1	1269.7	15.6	17.2	2.11
			Variant 2	1431.1	18.0	10.0	0.80

We can see from the table that the proposed variants decrease the number of questions asked to the decision-maker significantly but increase the number of solutions found. When we compare Variant 1 and Variant 2, we see that Variant 2 is better in terms of number of preference questions (IncQ) since we limit the questions. We can also see that Variant 1 usually causes small deviations from the real utility (deviation is between 0.39%-2.48%). On the other hand, it is not possible to make such generalization for Variant 2 since it has both 0.22% and 7.57% average deviations for different instances. We also see that the deviations are usually higher for the convex utility function than they are for the linear utility function. This is expected as the estimated utility function we use is also linear. Still, the results indicate that if the main concern is to decrease the interactions with the decision-maker, these variants can be utilized as heuristic methods.

Chapter 8

Conclusion and Further Research

We propose two interactive algorithms based on Pascoletti-Serafini scalarization to find the most preferred nondominated solution of a biobjective or triobjective programming problem by a decision-maker who has a general monotone utility function. We also generate a cone based approach by utilizing the convex cones and the region they cover for the cases where the utility function of the decision-maker is quasiconcave.

We compare the performances of the algorithms and the cone based approach via computational experiments we perform on knapsack and assignment problems. We observe that the cone based approach is better in terms of number of questions asked to the decision-maker and that the solution time of each algorithm is close to its cone based approach. We propose exact and heuristic variants in order to decrease the total number of questions. Our experiments show that these variants work efficiently.

We also compare the solution times required to find the most preferred solution in our algorithms to the time required to find the whole nondominated solution set. Our experiments demonstrate that our algorithms decrease the solution time by finding

only a subset of the nondominated solutions.

As a future work, more experiments can be conducted on different type of problems. We also suggest further research on understanding the effect of the utility function structure in decreasing the number of interactions with the decision-maker. We solved our PS models with fixed direction. Observing the effect of using other parameter setting mechanisms such as changing direction is also an interesting question that can be answered in future research.

Bibliography

[1] P. Korhonen and M. Soismaa, "An interactive multiple criteria approach to ranking alternatives," *Operational Research Society*, vol. 32, pp. 577–585, 1981.

[2] B. Malakooti, "Ranking and screening multiple criteria alternatives with partial information and use of ordinal and cardinal strength of preferences," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 30, no. 3, pp. 355–368, 2000.

[3] J.Siskos, "A way to deal with fuzzy preferences in multi-criteria decision problems," *European Journal of Operational Research*, vol. 10, no. 3, pp. 314–324, 1982.

[4] S. Zionts, "A multiple criteria method for choosing among discrete alternatives," *European Journal of Operational Research*, vol. 7, no. 2, pp. 143–147, 1981.

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

[6] R. Steuer, J. Silverman, and A. Whisman, "A combined tchebycheff/aspiration criterion vector interactive multi-objective programming procedure," *Management Science*, vol. 39, no. 10, pp. 1255–1260, 1993.

[7] M. Alves and J. Climaco, "Using cutting planes in an interactive reference point approach for multiobjective integer linear programming problems," *European Journal of Operational Research*, vol. 117, pp. 565–577, 1999.

[8] M. Alves and J. Climaco, "An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound," *European Journal of Operational Research*, vol. 124, pp. 478–494, 2000.

[9] S. Zionts and J. Wallenius, "An interactive programming method for solving the multiple criteria problem," *Management Science*, vol. 22, pp. 652–663, 1976.

[10] R. Ramesh, S. Zionts, and M. H. Karwan, "A class of practical interactive branch and bound algorithms for multicriteria integer programming," *European Journal of Operational Research*, vol. 26, pp. 161–172, 1986.

[11] O. Marcotte and R. Soland, "An interactive branch-and-bound algorithm for multiple criteria optimization," *Management Science*, vol. 32, no. 1, pp. 61,75, 1986.

[12] B. Lokman, M. Köksalan, P. J. Korhonen, and J. Wallenius, "An interactive algorithm to find the most preferred solution of multi-objective integer programs," *Annals of Operations Research*, vol. 245, no. 1, pp. 67–95, 2016.

[13] M. M. Köksalan, M. H. Karwan, and S. Zionts, "An improved method for solving multiple criteria problems involving discrete alternatives," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-14, no. 1, pp. 24–34, 1984.

[14] M. M. Köksalan and O. V. Taner, "An approach for finding the most preferred alternative in the presence of multiple criteria," *European Journal of Operational Research*, vol. 60, no. 1, pp. 52–60, 1992.

[15] O. V. Taner and M. M. Köksalan, "Experiments and an improved method for solving the discrete alternative multiple-criteria problem," *Journal of the Operational Research Society*, vol. 42, no. 5, pp. 383–391, 1991.

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

[17] Y. Aneja and K. Nair, "Bicriteria transportation problem," *Management Science*, vol. 27, pp. 73–78, 1979.

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

[19] V. Chankong and Y. Haimes, "Multiobjective decision making: Theory and methodology," 1983.

[20] H. W.Hamacher, C. R. Pedersen, and S. Ruzika, "Finding representative systems for discrete bicriterion optimization problems," *Operations Research Letters*, vol. 35, no. 3, pp. 336–344, 2007.

[21] N. Boland, H. Charkhgard, and M. Savelsbergh, "A criterion space search algorithm for biobjective integer programming: The balanced box method," *INFORMS Journal on Computing*, vol. 27, no. 4, pp. 735–754, 2015.

[22] 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, pp. 25–35, 2009.

[23] A. Sinha, P. Korhonen, J. Wallenius, and K. Deb, "An interactive evolutionary multi-objective optimization algorithm with a limited number of decision maker calls," *European Journal of Operational Research*, vol. 233, pp. 674–688, 2014.

[24] V. Bowman, *Multiple Criteria Decision Making*, ch. On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives, pp. 76–85. Berlin: Springer Verlag, 1976.

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

[26] M. J. Alves and J. Climaco, "A review of interactive methods for multiobjective integer and mixed-integer programming," *European Journal of Operational Research*, vol. 180, pp. 99–115, 2007.

[27] S. Greco, J. Figueira, and M. Ehrgott, "Multiple criteria decision analysis: state of the art surveys," *International series in operations research and management science*, vol. 233, 2016.

[28] W. S.Shin and A.Ravindran, "Interactive multiple objective optimization: Survey i—continuous case," *Computers and Operations Research*, pp. 97–114, 1991.

[29] M. Köksalan and C. Ulu, "An interactive approach for placing alternatives in preference classes," *European Journal of Operational Research*, vol. 144, no. 2, pp. 429–439, 2003.

[30] M. Köksalan and C. Ulu, "An interactive procedure for selecting acceptable alternatives in the presence of multiple criteria," *Naval Research Logistics*, vol. 48, no. 7, pp. 592–606, 2001.

[31] B. Lokman, M. Köksalan, P. J. Korhonen, and J. Wallenius, "An interactive approximation algorithm for multi-objective integer programs," *Computers and Operations Research*, vol. 96, pp. 80–90, 2018.

[32] Ö. Karsu and H. Erkan, "Balance in resource allocation problems: a changing reference approach," *OR Spectrum*, vol. 42, p. 297–326, 2020.

[33] Ö. Özpeynirci, S. Özpeynirci, and A. Kaya, "An interactive approach for multiple criteria selection problem," *Computers and Operations Research*, vol. 78, pp. 154–162, 2017.

[34] P. Korhonen, J. Wallenius, and S. Zionts, "Solving the discrete multiple criteria problem using convex cones," *Management Science*, vol. 30, no. 11, pp. 1268–1387, 1984.

[35] P. Korhonen, M. Soleimani-damaneh, and J. Wallenius, "Dual cone approach to convex-cone dominance in multiple criteria decision making," *European Journal of Operational Research*, vol. 249, no. 3, pp. 1139–1143, 2016.

[36] M. M. Köksalan and P. N. S. Sagala, "An interactive approach for choosing the best of a set of alternatives," *The Journal of the Operational Research Society*, vol. 43, no. 3, pp. 259–263, 1992.

[37] B. Bashir and Ö. Karsu, "Solution approaches for equitable multiobjective integer programming problems," *Annals of Operations Research*, 2020.

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

[39] Ö. Karsu, A. Morton, and N. Argyris, "Capturing preferences for inequality aversion in decision support," *European Journal of Operational Research*, vol. 264, pp. 686–706, 2018.

[40] Ö. Karsu, "Using holistic multicriteria assessments: The convex cones approach," *Wiley Encyclopedia of Operations Research and Management Science*, pp. 1–14, 2013.

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

[42] K. Dächert and K. Klamroth, "A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems," *Journal of Global Optimization*, vol. 61, no. 4, pp. 643–676, 2015.

[43] Ö. Karsu and F. Ulus, "Split algorithms for multiobjective integer programming problems," *Working paper*.

[44] G. Karakaya, M. Köksalan, and S. Ahipaşaoğlu, "Interactive algorithms for a broad underlying family of preference functions," *European Journal of Operational Research*, vol. 265, p. 248–262, 2018.