Optimisation over the Non-dominated Set of a Multi-objective Optimisation Problem

Zhengliang Liu

Management Science

Lancaster University

This dissertation is submitted for the degree of

Doctor of Philosophy

Management School August 2016

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Zhengliang Liu

August 2016

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Professor Matthias Ehrgott for his continuous support of my Ph.D study. His immense knowledge and insights in the field of multi-objective optimisation provided clear guidance throughout the research. His advice helped me tackle numerous problems encountered during the study. His experience and expertise in academic writing exemplified high quality of research presentation. In this journey of study, constantly I was deeply touched by his passion and dedication to research. He persisted in weekly meetings with me in spite of heavy duties as the Head of Department of Management Science. In my difficult times, Matthias helped me seek solutions patiently and stood by me as a mentor and also as a friend. I can never express my appreciation too much to him and felt lucky to have the privilege to work with him.

I also would like to thank Dr. Andrea Raith, who co-supervised me during my study in the University of Auckland. It was such a wonderful time to work with her. Her expertise in optimisation and in computer programming is of great help to me. Andrea showed great patience to help me solve problems in my computer programming and writing. She provided support to me not only in my research but also in my decision of Ph.D study, which I believed changed my life.

Besides my supervisors, I would like to thank Professor Horst W. Hamacher and research fellows in the University of Kaiserslautern for their advice to my research and kind hospitality during my visit in Germany.

My sincere thanks also goes to my annual review committee, Professor Adam Letchford, Professor John Boylan, Professor Mike Wright and Professor Konstantinos Zografos. I appreciate their opinions and feedback to my study.

I am grateful to Ms Kim Williams, Ms Gay Bentinck and Ms Lindsay Newby, who provided professional support in administration, especially in my transfer from the University of Auckland to Lancaster University.

It was impossible for me to have the opportunity of Ph.D. study without the financial support from the University of Auckland and the US Airforce Office for Scientific Research.

Last but not the least, I would like to thank my families for loving, supporting and understanding me.

Abstract

In this thesis we are concerned with optimisation over the non-dominated set of a multi-objective optimisation problem. A multi-objective optimisation problem (MOP) involves multiple conflicting objective functions. The non-dominated set of this problem is of interest because it is composed of the "best" trade-off for a decision maker to choose according to his preference. We assume that this selection process can be modelled by maximising a function over the non-dominated set.

We present two new algorithms for the optimisation of a linear function over the non-dominated set of a multi-objective linear programme (MOLP). A primal method is developed based on a revised version of Benson's outer approximation algorithm. A dual method derived from the dual variant of the outer approximation algorithm is proposed. Taking advantage of some special properties of the problem, the new methods are designed to achieve better computational efficiency. We compare the two new algorithms with several algorithms from the literature on a set of randomly generated instances. The results show that the new algorithms are considerably faster than the competitors.

We adapt the two new methods for the determination of the nadir point of (MOLP). The nadir point is characterized by the componentwise worst values of the non-dominated points of (MOP). This point is a prerequisite for many multi-criteria decision making (MCDM) procedures. Computational experiments against another exact method for this purpose from the literature reveal that the new methods are faster than the competitor.

The last section of the thesis is devoted to optimising a linear function over the non-dominated set of a convex multi-objective problem. A convex multi-objective problem (CMOP) often involves nonlinear objective functions or constraints. We extend the primal

and the dual methods to solve this problem. We compare the two algorithms with several existing algorithms from the literature on a set of randomly generated instances. The results reveal that the new methods are much faster than the others.

Contents

Chapter 1

Introduction

Mathematical optimisation is concerned with applying mathematical models to aid decision making. In practice a decision is made to achieve a goal given by a decision maker. For example a driver may aim to travel on the shortest route from a starting location to a destination. This can be modelled by minimising a function of the travel distance between the two locations. This function is known as the objective function in the model. To achieve this goal, a shortest route has to be selected among many alternatives, which are represented by feasible solutions (the set containing all feasible solutions is called the feasible set). However, sometimes a decision maker has to take into consideration multiple objectives simultaneously. For example the driver may also intend to avoid traffic congestion. This target can be incorporated into the model as the second objective function quantifying traffic congestion. However, these two objectives are often conflicting with each other. The shortest route is often rather congested. Therefore, the decision maker has to seek a trade-off between these two objectives. This scenario can be modelled as a multi-objective optimisation problem (MOP). This problem has many applications in practice, such as minimising cost versus minimising adverse environmental impacts in infrastructure projects (Ehrgott et al., 2010), minimising risk versus maximising return in financial portfolio management (Markowitz (1952), Ehrgott et al. (2009b)) or maximising tumour control versus minimising normal tissue complications in radiotherapy treatment

design (Ehrgott et al., 2009a). Because a feasible solution simultaneously optimising all of the objectives does not usually exist, the goal of (MOP) is to identify a set of so-called efficient solutions. Efficient solutions have the property that it is not possible to improve any of the objectives without deteriorating at least one other objective. The set of all efficient solutions is known as the efficient set. A vector of the objective function values derived from an efficient solution is called a non-dominated point. All of the non-dominated points compose the non-dominated set. The determination of this set is essential to (MOP) because this set contains the "best" options, from which the decision maker can choose one option (a non-dominated point) to his preference and thereafter choose an efficient solution corresponding to the chosen option to implement. In Chapter 2 we review several algorithms for solving multi-objective linear programmes (MOLPs) and convex multi-objective optimisation problems (CMOPs). These algorithms are based on Benson's outer approximation algorithm (Benson, 1998) and its dual variant (Ehrgott et al., 2011a). In this study we present the up-to-date versions of the algorithms proposed by Löhne et al. (2014).

In practical applications of (MOP), the decision maker has to select one solution from the efficient set for implementation. We assume that this selection process can be modelled by means of a function which is to be maximised over the efficient set of the underlying (MOP). For example, an investor perhaps aims at minimising the transaction cost of establishing a portfolio with high return and low risk. Problems of this kind are known as optimisation over the efficient set of a multi-objective optimisation problem (OE). This problem arises in many applications. For instance, Benson (1984) describes a production planning problem. A manufacturing firm has ten factories for producing four different types of products. The firm's goal is to maximise its profit function. However, the firm also aims to maintain high employment levels at each of its ten factories. Therefore, instead of maximising the profit function over the set of all feasible production plans, the firm wants to maximise it over the set of all efficient solutions of (MOP), in which the objective functions consist of ten employment level functions of the factories. Thus, (OE)

will be to find a maximum-profit production plan among all plans that are efficient in terms of the employment levels at the ten factories.

Research interest in (OE) is motivated by many factors. First, in terms of computational effort, it may be easier to solve (OE) directly than to solve (MOP) and then obtain a most preferred efficient solution. Secondly, decision makers may be overwhelmed by the large size of the whole efficient set (even in the case of (MOLP), the efficient set usually has infinite cardinality) and may not be able to choose a preferred solution from it. In Chapter 3 a survey on the methods for solving (OE) is conducted. These methods can be categorised into two classes, namely, methods based in decision space and methods based in objective space. The former class explores the feasible set of (MOP) in decision space where the decision variables exist. The latter class investigates the the feasible set in objective space (the image of the feasible set in decision space) where the objective vectors exist. Several articles explore the structures and properties of the efficient set and the non-dominated set. Dauer (1987) notes that the feasible set in objective space often has a much simpler structure, i.e., fewer extreme points and proper faces, than the feasible set in decision space. This is due to the fact that in practice the dimension of the decision space is much greater than that of the objective space. Dauer (1993) illustrates the concept of "collapsing", which means that faces of the efficient set shrink into nonfacial subsets of the non-dominated set. Benson (1995) shows that the dimension of efficient faces in the feasible set always exceeds or equals the dimension of their images in the non-dominated set. Hence, it is more computationally efficient to employ techniques and methods to solve (OE) in objective space. In the hope that computational effort might be saved, solving (OE) in objective space has been attracting attention, and several algorithms have been developed since 2000. Most existing methods in the literature employ the branch and bound technique (A. H. Land (1960)). This technique is a systematic search scheme which can be represented by a "tree" structure. The "root" (the original problem) of the "tree" splits into two "branches" (two subproblems), from each of which upper bounds and lower bounds of the objective function are determined. A gap between the upper

bound and the lower bound of a "branch" indicates that further partitioning has be to conducted. If a "branch" does not provide better solutions than the incumbent solution (the solution providing the best objective function value so far), this "branch" is pruned without further branching. An optimal solution is obtained if the upper bound and the lower bound coincide. We review several algorithms (Nguyen Thi et al. (2008), Fülöp and Muu (2000) Kim and Thang (2013), Thoai (2000) and Benson (2011)) of this kind in Chapter 3.

In this thesis the problem of interest is the optimisation of a linear function over the non-dominated set of an MOP. In Chapter 4 we present two new algorithms for maximising a linear function over the non-dominated set of an MOLP, which is named problem (P). A primal method is developed based on a revised version of Benson's outer approximation algorithm to compute the non-dominated set of an MOLP. Benson's algorithm enumerates all of the non-dominated vertices of the objective polyhedron (the image of the feasible set in decision space). We first show that an optimal solution to (P) exists at a vertex of the objective polyhedron. Therefore, a naïve algorithm can be proposed with two phases. Firstly enumerate all non-dominated vertices through Benson's algorithm, and then evaluate the objective function at the vertices. The primal method we propose integrates the two phases. Specifically, the vertex evaluation step is embedded into the vertex enumeration step. When a vertex is generated in the iterations of Benson's algorithm, the objective function is evaluated at the vertex. If the vertex is a non-dominated point of (MOLP), we use a hyperplane serving as a "cut" to remove part of the search region where no better point exists.

Furthermore, an important conclusion is drawn that an optimal solution is obtained at a so-called incomplete vertex of the objective polyhedron. A vertex is incomplete if it has at least one dominated "neighbour" (an adjacent vertex connected by one edge). This property reveals that it is sufficient to search for the incomplete vertices to solve (P). However, it is rather difficult to confine the search to the set of incomplete vertices due to the complexity of the non-dominated set. Fortunately, the dual method we propose takes

advantage of this property in dual objective space. This method is derived from the dual variant of Benson's outer approximation algorithm. We have proved that an incomplete vertex of the primal polyhedron corresponds to a so-called incomplete facet of the dual polyhedron. The dual algorithm is designed to determine the incomplete facets in order to solve the problem. We compare the two new algorithms with several algorithms from the literature on a set of randomly generated instances. The results show that the new algorithms are considerably faster than the competitors.

In Chapter 5 the primal method and the dual method are adapted to determine the nadir point of an MOLP. The nadir point is characterized by the componentwise worst values of the non-dominated points of (MOP). The task of the determination of the nadir point is of significant importance. Firstly, the knowledge of the nadir point facilitates the normalisation of the objectives with inconsistent magnitude (Miettinen (1999)). In the field of multi-criteria decision making (MCDM), a decision maker aims at a desirable point out of the non-dominated set. The preferences of the decision maker are reflected via a set of weights associated with the criteria. However, if the objectives are of different magnitude, it is necessary to normalise them into the same interval, say between zero and one, in order to ensure that a preferred point is chosen in accordance to the weights. The normalisation procedure requires the range of the non-dominated set, which is delimited by the ideal point and the nadir point. The decision maker would make a decision within this range. Secondly, the nadir point is a pre-requisite for some interactive methods such as the STEM method (Benayoun et al. (1971)) and the GUESS method (Buchanan (1997)). In compromise programming, it serves as a reference point (Ehrgott and Tenfelde-Podehl (2003)). Additionally, another motivation is proposed by Deb et al. (2010) that the nadir point is crucial in terms of visualising the non-dominated set. For the reasons mentioned above, heuristics and optimisation methods for determining this point are of interested to many researchers and practitioners.

Unfortunately, it is a challenge to determine the nadir point, which is composed of the maxima of the individual objectives over the non-dominated set. The structure of this set

can be very complex causing the determination of the nadir point to be rather difficult. To the best of our knowledge, the only exact method for computing the nadir point for MOLPs from the literature is proposed by Alves and Costa (2009).

The nadir values can be determined by optimising a linear function over the non-dominated set of (MOLP), which is a special case of (P). Hence the primal method and the dual method can be utilised to solve this problem. In order to achieve better performance, we adapt these two methods to find the exact nadir point. We compare these two methods against the exact method proposed byAlves and Costa (2009). The results show that the new methods are faster.

In Chapter 6, we extend the primal and the dual methods to maximise a linear function over the non-dominated set of (CMOP). A CMOP is a multi-objective optimisation problem with nonlinear convex objective functions and feasible set. This problem can be solved by the outer approximation algorithm proposed by Löhne et al. (2014). This method uses polyhedra to approximate the non-dominated set of the feasible set in objective space from outside. We first show that the properties of (P) still hold for the nonlinear case. Then a primal method and a dual method are introduced, which are based on the outer approximation algorithms by Löhne et al. (2014). We compare the two new algorithms with several algorithms from the literature on a set of randomly generated instances. The results show that the new algorithms are more efficient than the competitors.

Chapter 2

Multi-objective Optimisation

This chapter is concerned with the multi-objective optimisation problem (MOP). This problem involves simultaneously optimising multiple conflicting objective functions. Solutions to (MOP) are known as efficient solutions. A solution is efficient if it is not possible to improve one objective function without deteriorating at least one other. One class of (MOP) is the multi-objective linear programme (MOLP), where all of the objective functions are linear and the feasible set is a convex polyhedron. In this chapter we review two objective space based methods to solve (MOLP), namely the revised version of Benson's outer approximation algorithm and its dual variant. The second part of the chapter focuses on another class of (MOP) known as the convex multi-objective optimisation problem (CMOP). This type of problem involves optimising multiple non-linear convex objective functions over a non-linear convex feasible set. Benson's method and its dual variant have been extended to solve this problem. The algorithms discussed in this chapter establish the foundation of algorithms in the subsequent chapters.

A multi-objective optimisation problem is formulated as

$$\min \{(f_1(x), \ldots, f_p(x))^T : x \in \mathcal{X}\}, \tag{MOP}$$

where $f : \mathbb{R}^n \to \mathbb{R}^p$ is a vector-valued function composed of p real-valued continuous functions $f_k : \mathbb{R}^n \to \mathbb{R}$, $k = 1, \ldots, p$. The set $\mathcal{X}$ is a feasible set in decision space $\mathbb{R}^n$. The

objective function f maps a solution $x \in \mathcal{X}$ to a point $y = f(x)$ in objective space $\mathbb{R}^p$. Let $\mathcal{Y} := \{f(x) : x \in \mathcal{X}\}$ denote the image of $\mathcal{X}$ in objective space. We assume that $\mathcal{X}$ is a nonempty and compact set. Therefore, $\mathcal{Y}$ is nonempty and compact, too.

We use the notation introduced by Ehrgott (2005) to compare vectors y^1, $y^2 \in \mathbb{R}^p$: $y^1 = y^2$ if $y^1_k = y^2_k$ for $k = 1, \ldots, p$; $y^1 < y^2$ if $y^1_k < y^2_k$ for $k = 1, \ldots, p$; $y^1 \leqq y^2$ if $y^1_k \leq y^2_k$ for $k = 1, \ldots, p$; $y^1 \leq y^2$ if $y^1 \leqq y^2$ and $y^1 \neq y^2$. We define $\mathbb{R}^p_{\geqq} := \{y \in \mathbb{R}^p : y_k \geqq 0, k = 1, \ldots, p\}$, and $\mathbb{R}^p_{\leqq} := \{y \in \mathbb{R}^p : y_k \leqq 0, k = 1, \ldots, p\}$; $\mathbb{R}^p_{\geq} := \{y \in \mathbb{R}^p : y_k \geq 0, k = 1, \ldots, p\}$, and $\mathbb{R}^p_{\leq} := \{y \in \mathbb{R}^p : y_k \leq 0, k = 1, \ldots, p\}$.

Definition 2.1. *A feasible solution $\hat{x} \in \mathcal{X}$ is an efficient solution of (MOP) if there is no $x \in \mathcal{X}$ such that $f(x) \leq f(\hat{x})$. The set of all efficient solutions is called the efficient set in decision space and denoted by $\mathcal{X}_E$. Correspondingly, $\hat{y} = f(\hat{x})$ is called a non-dominated point and $\mathcal{Y}_N := \{f(x) : x \in \mathcal{X}_E\}$ is the non-dominated set in objective space.*

Definition 2.2. *A feasible solution $\hat{x} \in \mathcal{X}$ is a weakly efficient solution of (MOP) if there is no $x \in \mathcal{X}$ such that $f(x) < f(\hat{x})$. The set of all weakly efficient solutions is called the weakly efficient set in decision space and denoted by $\mathcal{X}_{WE}$. Correspondingly, $\hat{y} = f(\hat{x})$ is called a weakly non-dominated point and $\mathcal{Y}_{WN} := \{f(x) : x \in \mathcal{X}_{WE}\}$ is the weakly non-dominated set in objective space.*

A non-dominated point of (MOP) is characterized as a vector of the objective values, none of which can be improved without deteriorating at least one other value. Therefore, the set of non-dominated points is of interest to decision makers. To solve (MOP) is generally understood as obtaining $\mathcal{Y}_N$ or a representation of this set. Once $\mathcal{Y}_N$ is determined, the efficient set $\mathcal{X}_E$ consists of those solutions in the set $\{x \in \mathbb{R}^n : f(x) \in \mathcal{Y}_N\}$. Normally in practice decision makers choose a non-dominated point $y \in \mathcal{Y}_N$, and one efficient solution $x \in \{x : x \in \mathcal{X}_E, f(x) = y\}$ is adopted to implement.

For (MOP) two points, namely the ideal point and the anti-ideal point, are of interest. The ideal point is a vector in objective space composed of the minimal values of the minimisation of each objective function individually. On the contrary, the anti-ideal

point contains the maximal values of maximising the objective functions individually. For most of the algorithms introduced in this study, these two points are often determined to initiate the computation. For example these two points can be used to construct the initial polyhedron containing $\mathcal{Y}$ to start an outer approximation scheme. The formal definitions of these two points are in Definition 2.3 below.

Definition 2.3.

- *The ideal point* $y^I \in \mathbb{R}^p$ *of (MOP) is defined to be the vector of componentwise minima of* $\mathcal{Y}$*, i.e.,* $y^I_k := \min\{f_k(x) : f(x) \in \mathcal{Y}\}$*,* $k = 1, \ldots, p$*.*
- *The anti-ideal point* $y^{AI} \in \mathbb{R}^p$ *of (MOP) is defined to be the vector of componentwise maxima of* $\mathcal{Y}$*, i.e.,* $y^{AI}_k := \max\{f_k(x) : f(x) \in \mathcal{Y}\}$*,* $k = 1, \ldots, p$*.*

Figure 2.1 shows the ideal and the anti-ideal point of a polyhedron $\mathcal{Y}$, for the case $p = 2$.

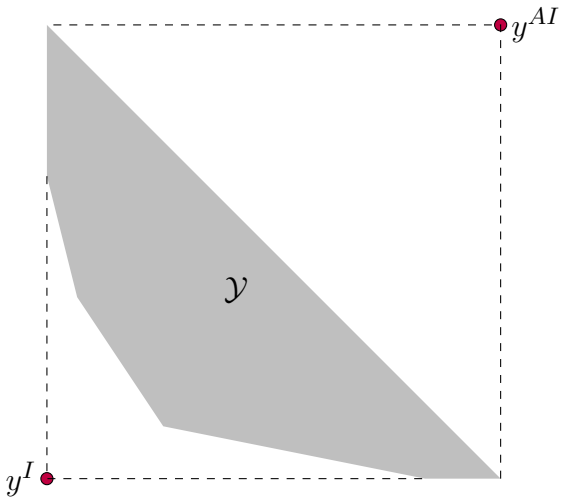


Figure 2.1 Ideal point and anti-ideal point.

2.1 Multi-objective linear programming

A multi-objective linear programme is a special case of (MOP), where all objectives and constraints are linear. It can be formulated as

$$\min \{Cx : Ax \geqq b\}, \tag{MOLP}$$

where C is a $p \times n$ matrix, p is the number of objective functions and n is the number of decision variables. The feasible set $\mathcal{X}$ is a polyhedral convex set defined by $Ax \geqq b$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. We assume that $\mathcal{X}$ is bounded. A polyhedral convex set such as $\mathcal{X}$ has a finite number of faces. A subset $\mathcal{F}$ of $\mathcal{X}$ is a face if and only if there are $\omega \in \mathbb{R}^n \setminus \{0\}$ and $\gamma \in \mathbb{R}$ such that $\mathcal{X} \subseteq \{x \in \mathbb{R}^n : \omega^T x \geqq \gamma\}$ and $\mathcal{F} = \{x \in \mathbb{R}^n : \omega^T x = \gamma\} \cap \mathcal{X}$. We call a hyperplane $H = \{x \in \mathbb{R}^n : \omega^T x = \gamma\}$ supporting to $\mathcal{X}$ if $\omega^T x \geqq \gamma$ for all $x \in \mathcal{X}$ and there is some $x^0 \in \mathcal{X}$ such that $\omega^T x^0 = \gamma$. The proper $(r-1)$-dimensional faces of an r-dimensional polyhedral set $\mathcal{X}$ are called facets of $\mathcal{X}$. Proper faces of dimension zero are called extreme points or vertices of $\mathcal{X}$. For more details of the polyhedral theory the reader is referred to Nemhauser and Wolsey (1988).

In Example 2.1 we provide a numerical example of (MOLP).

Example 2.1.

$$\begin{aligned} &\min \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ \text{s.t.} \quad &\begin{pmatrix} 4 & 1 \\ 3 & 2 \\ 1 & 5 \\ -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geqq \begin{pmatrix} 4 \\ 6 \\ 5 \\ -6 \end{pmatrix} \\ &x_1, x_2 \geqq 0. \end{aligned}$$

Figure 2.2 shows the feasible set $\mathcal{X}$ of Example 2.1 in decision space. The red line segments compose the efficient set $\mathcal{X}_E$.

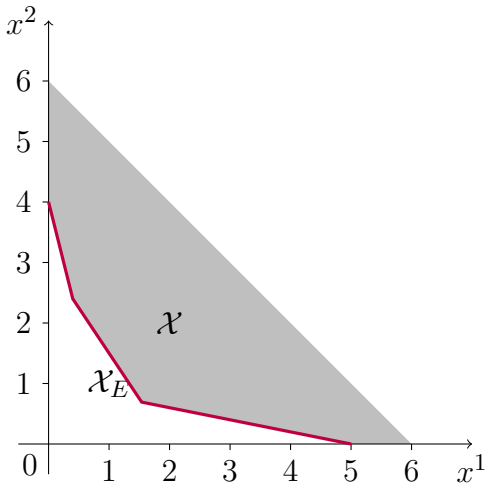


Figure 2.2 The feasible set and the efficient set of Example 2.1 in decision space.

Figure 2.3 shows the feasible set $\mathcal{Y}$ of Example 2.1 in objective space. The bold line segments compose the non-dominated set $\mathcal{Y}_N$. In this example, the feasible set $\mathcal{X}$ in decision space is identical to $\mathcal{Y}$. Furthermore $\mathcal{X}_E$ is the same as $\mathcal{Y}_N$. This is because C is the identity matrix in this example.

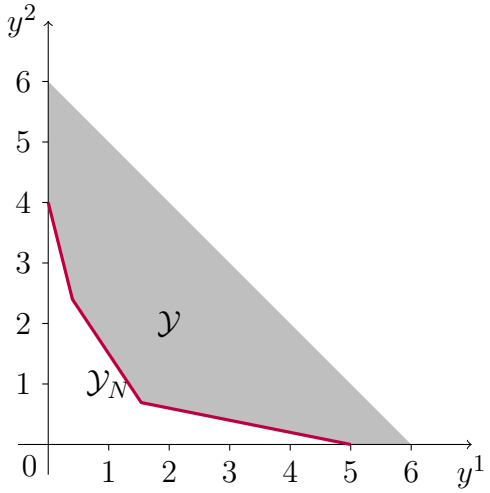


Figure 2.3 The feasible set and non-dominated set of Example 2.1 in objective space.

Let $\mathcal{P} := \mathcal{Y} + \mathbb{R}^p = \{y + \hat{y} : y \in \mathcal{Y}, \hat{y} \in \mathbb{R}^p\}$ denote the extended set of $\mathcal{Y}$. The non-dominated set of $\mathcal{P}$ is the same as that of $\mathcal{Y}$, i.e., $\mathcal{P}_N = \mathcal{Y}_N$ (Ehrgott (2005) Proposition 2.3). Therefore in terms of solving (MOP) these two sets are equivalent. In this study we use $\mathcal{P}$ instead of $\mathcal{Y}$ because $\mathcal{P}$ has some beneficial properties to the algorithms that we will present in the following sections. For example $\mathcal{P}$ has dimension p, i.e., $dim(\mathcal{P}) = p$, and the non-dominated set of $\mathcal{P}$ belongs to the boundary of $\mathcal{P}$, i.e., $\mathcal{P}_N \subset \text{bd}(\mathcal{P})$ (Ehrgott (2005) Proposition 2.4). We will explain the importance of these properties in the next section.

A number of methods for solving (MOLP) have been reviewed by Ehrgott (2005). The existing methods fall into two main categories, namely methods based in decision space and methods based in objective space. The former search $\mathcal{X}$ for efficient solutions in decision space, whereas the latter explore the structure of $\mathcal{Y}$ in objective space. In the following sections we review several objective space based methods which play important roles in this research.

2.1.1 A revised version of Benson's outer approximation algorithm for (MOLP)

An objective space based method for solving (MOLP) was initially proposed by Benson (1998). Then Ehrgott et al. (2011a) revised this method with a few modifications. Hamel et al. (2014) further developed this method so that only one linear programme (LP) needs to be solved in each iteration. In this section the state-of-the-art version by Hamel et al. (2014) is reviewed. We first provide notation that will facilitate the description of this algorithm. For $y \in \mathbb{R}^p$ and $v \in \mathbb{R}^p$, let

$$\lambda(v) := \left(v_1, ..., v_{p-1}, 1 - \sum_{i=1}^{p-1} v_i \right)^T, \tag{2.1}$$

$$\lambda^*(y) := (y_1 - y_p, ..., y_{p-1} - y_p, -1)^T. \tag{2.2}$$

Ehrgott et al. (2011a) define a coupling function $\varphi : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ by

$$\varphi(y, v) := \sum_{i=1}^{p-1} y_i v_i + y_p \left(1 - \sum_{i=1}^{p-1} v_i\right) - v_p. \tag{2.3}$$

Consider the following weighted sum problem of (MOLP):

$$\min \{\lambda(v)^T Cx : x \in \mathbb{R}^n, Ax \geqq b\}. \tag{$P_1(v)$}$$

Proposition 2.1. *If $\lambda \in \mathbb{R}^p_{\geq}$, then an optimal solution x to $(P_1(v))$ is a weakly efficient solution to (MOLP).*

Proposition 2.1 is well known, see Ehrgott (2005).

The dual problem of $P_1(v)$ is

$$\max \{b^T u : u \in \mathbb{R}^m, u \geqq 0, A^T u = C^T \lambda(v)\}. \tag{$D_1(v)$}$$

The following LPs play an important role in the algorithms:

$$\min \{z : (x, z) \in \mathbb{R}^n \times \mathbb{R}, Ax \geqq b, Cx - ez \leqq y\}, \tag{$P_2(y)$}$$

where e is a column vector with all elements being one. Given a point $y \in \mathbb{R}^p$ in the objective space, if the optimal value $z \leqq 0$, then $y \in \mathcal{P}$, otherwise $y \notin \mathcal{P}$. An optimal solution $\hat{x} \in \mathcal{X}$ to $(P_2(y))$ provides a point $\hat{y} = C\hat{x}$ which is on the boundary of $\mathcal{P}$. Moreover $\hat{y}$ is a weakly non-dominated point, i.e., $\hat{y} \in \mathcal{P}_{WN}$ (Benson, 1998).

The following LP $(D_2(y))$ is the dual problem of $(P_2(y))$.

$$\max \{b^T u - y^T \lambda : (u, \lambda) \in \mathbb{R}^m \times \mathbb{R}^p, (u, \lambda) \geqq 0, A^T u = C^T \lambda, e^T \lambda = 1\}. \tag{$D_2(y)$}$$

Proposition 2.2. *Let* (u^*, λ^*) *be an optimal solution to* $(D_2(y))$*. Then*

$$\sum_{k=1}^{p} \lambda_k^* y_k \geqq b^T u^*$$

provides a supporting hyperplane to $\mathcal{P}$ *at* $\hat{y}$*.*

Proof. See Proposition 4.2 in Hamel et al. (2014). □

Therefore, by solving $(P_2(y))$, we not only check if y belongs to $\mathcal{P}$ but also obtain the dual variable values (u^*, λ^*) (an optimal solution to $(D_2(y))$), through which a supporting hyperplane to $\mathcal{P}$ at $\hat{y}$ can be constructed.

The algorithm discussed in this section is the revised version of Benson's algorithm proposed by Hamel et al. (2014). It first constructs a p-dimensional polyhedron $\mathcal{S}^0 := y^I + \mathbb{R}^p_{\leqq}$ such that $\mathcal{P} \subseteq \mathcal{S}^0$. In every iteration it chooses a vertex s^k from $V_{\mathcal{S}^{k-1}}$, the vertex set of $\mathcal{S}^{k-1}$, which is not in $\mathcal{P}$ and constructs a supporting hyperplane to $\mathcal{P}$ by solving $(P_2(s^k))$ and obtaining its dual optimal solution to $(D_2(s^k))$. $\mathcal{S}^k$ is defined by intersecting $\mathcal{S}^{k-1}$ with the halfspace of the supporting hyperplane containing $\mathcal{P}$. The algorithm terminates as soon as no such $s^k \in \mathcal{S}^{k-1} \setminus \mathcal{P}$ can be found and $\mathcal{S}^{k-1} = \mathcal{P}$. Some notation used in Algorithm 2.1 is listed below.

Y_N set of non-dominated points found in the algorithm.

$\mathcal{S}^k$ outer approximation set of $\mathcal{P}$ in iteration k.

y^I ideal point of $\mathcal{P}$.

$V_{\mathcal{S}^k}$ vertex set of $\mathcal{S}^k$.

e^i_ϵ $e^i_{\epsilon j} = \epsilon$ if $i \neq j$; otherwise, $e^i_{\epsilon j} = 1 - (p-1)\epsilon$, $\epsilon > 0$ is a small positive number.

Note that in Line 1 in stead of using $(P_1(e^i))$, $(P_1(e^i_\epsilon))$ is employed to assure that non-dominated points are obtained.

Algorithm 2.1 Revised Benson's algorithm

Input: (MOLP)
Output: Y_N; $\mathcal{S}^k$

1: Compute an optimal solution x^i and an optimal value y_i^I of $(P_1(e_\epsilon^i))$, for $i = 1, \ldots, p$. $Y_N := \{Cx^i\}$, for $i = 1, \ldots, p$.
2: Set $\mathcal{S}^0 := \{y^I\} + \mathbb{R}^p_{\leqq}$; $V_{\mathcal{S}^0} := \{y^I\}$ and $k := 1$.
3: **while** $V_{\mathcal{S}^{k-1}} \not\subset \mathcal{P}$ **do**
4: Choose a vertex s^k of $\mathcal{S}^{k-1}$.
5: Compute an optimal solution (x^k, z^k) to $P_2(s^k)$ and an optimal solution (u^k, λ^k) to $D_2(s^k)$.
6: **if** $z^k > 0$ **then**
7: Set $\mathcal{S}^k := \mathcal{S}^{k-1} \cap \{y \in \mathbb{R}^p : \varphi(y, (\lambda_1^k, \ldots, \lambda_{p-1}^k, b^T u^k)) \geqq 0\}$; Update $V_{\mathcal{S}^k}$; $Y_N := Y_N \cup Cx^k$.
8: **else**
9: $Y_N := Y_N \cup s^k$.
10: **end if**
11: Set $k := k + 1$
12: **end while**

Example 2.2. In Figure 2.4, we illustrate the revised version of Benson's algorithm with Example 2.1.

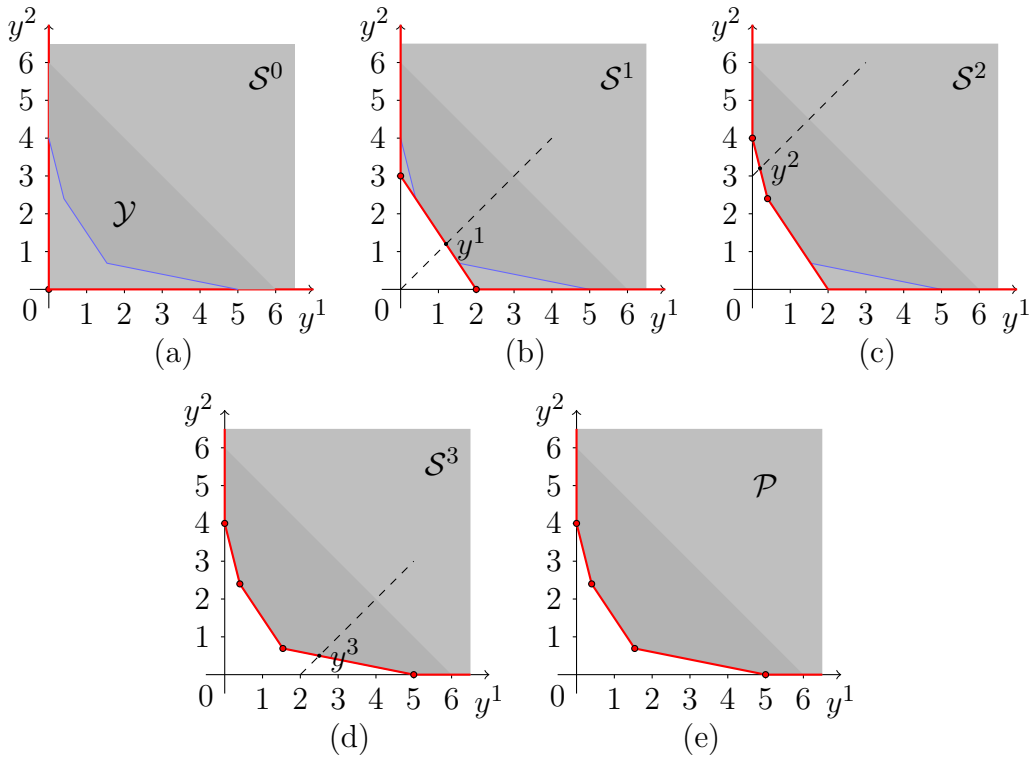


Figure 2.4 The revised version of Benson's algorithm.

Table 2.1 Iterations of Algorithm 2.1 in Example 2.2.

Iteration k	Vertex s^k	Vertices of $\mathcal{S}^k$ in $\mathcal{P}$	Vertices of $\mathcal{S}^k$ not in $\mathcal{P}$
1, Figure 2.4 (b)	$(0,0)^T$	$\emptyset$	$(2,0)^T$,$(0,3)^T$
2, Figure 2.4 (c)	$(0,3)^T$	$(0,4)^T, (0.4,2.4)^T$	$(2,0)^T$
3, Figure 2.4 (d)	$(2,0)^T$	$(0,4)^T, (0.4,2.4)^T$, $(20/13,9/13)^T, (5,0)^T$	$\emptyset$

The algorithm starts with the initial polyhedron $\mathcal{S}^0$, see Figure 2.4 (a). The first vertex s^1 is $(0,0)^T$. By solving $(P_2(s^1))$, y^1, a weakly non-dominated point of $\mathcal{P}$ is obtained. A supporting hyperplane to $\mathcal{P}$ at y^1 is generated and two vertices of $\mathcal{S}^1$, $(0,3)^T$ and $(2,0)^T$, are obtained. In the second iteration, the vertex $(0,3)^T$ is chosen and a new hyperplane is generated. Then we obtain two new extreme points, $(0,4)^T$ and $(0.4,2.4)^T$, which belong to $\mathcal{P}$. The process is iterated until all the extreme points of $\mathcal{S}^{k-1}$ are in $\mathcal{P}$. At termination, both the non-dominated vertices and the hyperplanes defining $\mathcal{P}$ are known, as shown in Figure 2.4 (e). More discussion and details of this algorithm can be found in Hamel et al. (2014).

2.1.2 Geometric duality

Heyde and Löhne (2008) introduced a concept of geometric duality for (MOLP). This theory relates (MOLP) with a dual multi-objective linear programme (DMOLP) in dual objective space $\mathbb{R}^p$. We use the following notation to compare two vectors v^1, $v^2 \in \mathbb{R}^p$ in dual objective space. We write $v^1 >_{\mathcal{K}} v^2$ if $v^1_k = v^2_k$ for $k = 1,\ldots,p-1$ and $v^1_p > v^2_p$; $v^1 \geqq_{\mathcal{K}} v^2$ if $v^1_k = v^2_k$ for $k = 1,\ldots,p-1$ and $v^1_p \geqq v^2_p$. Moreover $v^1 \geq_{\mathcal{K}} v^2$ is the same as $v^1 >_{\mathcal{K}} v^2$.

The dual of (MOLP) is

$$\max_{\mathcal{K}} \{(\lambda_1,...,\lambda_{p-1}, b^T u)^T : (u,\lambda) \geqq 0, A^T u = C^T\lambda, e^T\lambda = 1\}, \qquad \text{(DMOLP)}$$

where $(u,\lambda) \in \mathbb{R}^m \times \mathbb{R}^p$. $\mathcal{K} := \{v \in \mathbb{R}^p : v_1 = v_2 = \cdots = v_{p-1} = 0, v_p \geqq 0\}$ is the ordering cone in the dual objective space, and maximisation is with respect to the order defined by $\mathcal{K}$. Let $\mathcal{V}$ denote the feasible set in the dual objective space, then the extended feasible set in the dual objective space is $\mathcal{D} := \mathcal{V} - \mathcal{K} = \{v - \hat{v} : v \in \mathcal{V}, \hat{v} \in \mathcal{K}\}$. For (DMOLP) it is of interest to determine the so-called $\mathcal{K}$-maximal set, which is

$$\mathcal{D}_{\mathcal{K}} = \max_{\mathcal{K}} \{(\lambda_1, ..., \lambda_{p-1}, b^T u)^T : (u,\lambda) \geqq 0, A^T u = C^T \lambda, e^T \lambda = 1\}.$$

Figure 2.5 shows the extended feasible set $\mathcal{D}$ in the dual objective space of Example 2.1. The bold line segments compose $\mathcal{D}_{\mathcal{K}}$.

Heyde and Löhne (2008) define the two set-valued maps H and H^* to relate $\mathcal{P}$ and $\mathcal{D}$.

$$H : \mathbb{R}^p \rightrightarrows \mathbb{R}^p, H(v) := \{y \in \mathbb{R}^p : \lambda(v)^T y = v_p\} \text{ and} \tag{2.4}$$

$$H^* : \mathbb{R}^p \rightrightarrows \mathbb{R}^p, H^*(y) := \{v \in \mathbb{R}^p : \lambda^*(y)^T v = -y_p\}. \tag{2.5}$$

Note that $\lambda(v)$ and $\lambda^*(y)$ are defined in (2.1) and (2.2).

Given a point $v \in \mathbb{R}^p$ in dual objective space, H defines a hyperplane $H(v)$ in primal objective space. On the other hand a hyperplane $H(v)$ in primal objective space corresponds to a point v in dual objective space. Similarly, given a point $y \in \mathbb{R}^p$ in primal objective

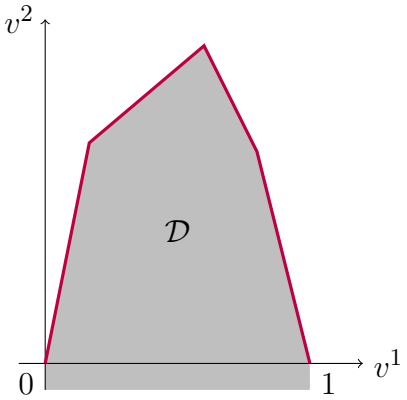


Figure 2.5 Extended dual objective polyhedron and the $\mathcal{K}$-maximal set of Example 2.1.

space, $H^*(y)$ is a hyperplane in dual objective space. On the contrary, a hyperplane $H^*(y)$ in primal objective space can be mapped to a point y in primal objective space through H^*. Theorems 2.1 and 2.2 state a relationship between proper $\mathcal{K}$-maximal faces of $\mathcal{D}$ and proper weakly non-dominated faces of $\mathcal{P}$.

Theorem 2.1. *(Heyde and Löhne (2008))*

1. *A point v is a $\mathcal{K}$-maximal vertex of $\mathcal{D}$ if and only if $H(v) \cap \mathcal{P}$ is a weakly non-dominated facet of $\mathcal{P}$.*

2. *A point y is a is a weakly non-dominated vertex of $\mathcal{P}$ if and only if $H^*(y) \cap \mathcal{D}$ is a $\mathcal{K}$-maximal facet of $\mathcal{D}$.*

Heyde and Löhne (2008) define a duality map $\Psi : 2^{\mathbb{R}^p} \to 2^{\mathbb{R}^p}$. Let $\mathcal{F}^* \subset \mathbb{R}^p$, then

$$\Psi(\mathcal{F}^*) := \bigcap_{v \in \mathcal{F}^*} H(v) \cap \mathcal{P}.$$

Theorem 2.2. *(Heyde and Löhne (2008))* *Ψ is an inclusion reversing one-to-one map between the set of all proper $\mathcal{K}$-maximal faces of $\mathcal{D}$ and the set of all proper weakly non-dominated faces of $\mathcal{P}$ and the inverse map is given by*

$$\Psi^{-1}(\mathcal{F}) = \bigcap_{y \in \mathcal{F}} H^*(y) \cap \mathcal{D}.$$

Moreover, for every proper $\mathcal{K}$-maximal face $\mathcal{F}^*$ of $\mathcal{D}$ it holds that $dim\mathcal{F}^* + dim\Psi(\mathcal{F}^*) = p - 1$.

2.1.3 A dual variant of Benson's outer approximation algorithm for (MOLP)

Ehrgott et al. (2011a) propose a dual variant of Benson's algorithm to solve (DMOLP). This method is further developed by Hamel et al. (2014). The revised version of Benson's

algorithm applies an outer approximation to $\mathcal{P}$ in primal objective space, whereas its dual variant does the same to $\mathcal{D}$ in dual objective space. Hamel et al. (2014) detail the dual algorithm. It iteratively generates supporting hyperplanes of $\mathcal{D}$, which correspond to non-dominated faces of $\mathcal{P}$. Eventually a complete set of hyperplanes that define $\mathcal{D}$ as well as the set of all vertices of $\mathcal{D}$ is obtained.

Algorithm 2.2 Dual variant of Benson's algorithm

Input: (MOLP)
Output: $V_{\mathcal{S}^{k-1}}$ ($\mathcal{K}$-maximal vertex set of $\mathcal{D}$); $\mathcal{S}^k$ (inequality representation of $\mathcal{D}$)

1: Choose some $\hat{d} \in \text{int}\mathcal{D}$.
2: Compute an optional solution x^0 of $P_1(\hat{d})$.
3: Set $S^0 := \{v \in \mathbb{R}^p : \lambda(v) \geqq 0, \varphi(Cx^0, v) \geqq 0\}$ *and* $k := 1$.
4: **while** $V_{\mathcal{S}^{k-1}} \not\subset \mathcal{D}$ **do**
5: Choose a vertex s^k of $\mathcal{S}^{k-1}$ such that $s^k \notin \mathcal{D}$.
6: Compute an optimal solution x^k of $(P_1(s^k))$.
7: Set $\mathcal{S}^k := \mathcal{S}^{k-1} \cap \{v \in \mathbb{R} : \varphi(Cx^k, v) \geqq 0\}$.
8: Set $k := k + 1$.
9: **end while**

Example 2.3. We illustrate the dual variant of Benson's algorithm with Example 2.1 in Figure 2.6. Algorithm 2.2 solves (DMOLP) in dual objective space. It firstly chooses an interior point in $\mathcal{D}$ in the way stated in Algorithm 2 (i1) in Ehrgott et al. (2011a). Then a polyhedron containing $\mathcal{D}$ is constructed as shown in Figure 2.6 (a). Iteratively infeasible vertices of $\mathcal{S}^{k-1}$ are used to generate supporting hyperplanes of $\mathcal{D}$. The algorithm terminates as soon as all vertices of $\mathcal{S}^{k-1}$ are feasible, i.e., they belong to $\mathcal{D}$. Table 2.2 shows details of the solution process for Example 2.1.

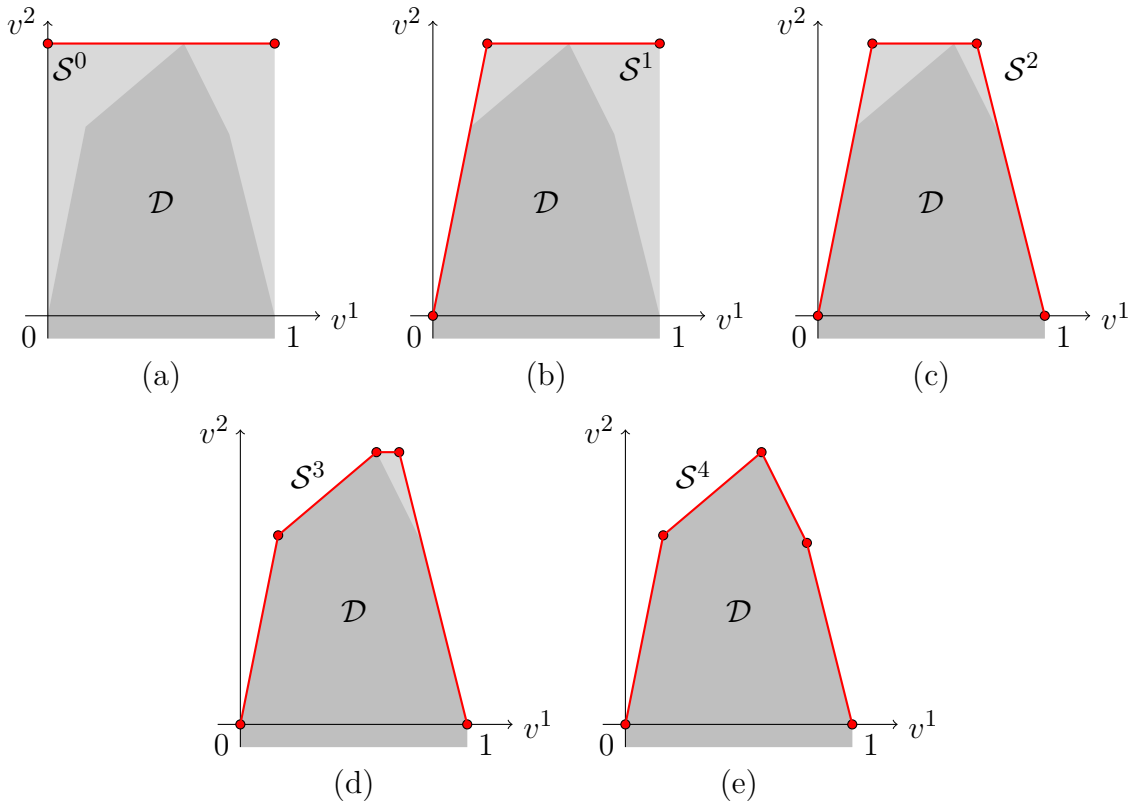


Figure 2.6 Dual variant of Benson's algorithm.

Table 2.2 Iterations of Algorithm 2.2 in Example 2.3.

Iteration k	Vertex s^k	Feasible vertices	Infeasible vertices
1 Figure 2.6 (b)	$(0,6/5)^T$	$(0,0)^T$	$(6/25,6/5)^T$,$(1,6/5)^T$
2 Figure 2.6 (c)	$(1,6/5)^T$	$(0,0)^T$, $(1,0)^T$	$(6/25,6/5)^T$,$(14/20,6/5)^T$
3 Figure 2.6 (d)	$(6/25,6/5)^T$	$(0,0)^T$, $(1,0)^T$,	$(14/20,6/5)^T$
		$(1/6,5/6)^T$, $(3/5,6/5)^T$	$\emptyset$
4 Figure 2.6 (e)	$(14/20,6/5)^T$	$(0,0)^T$, $(1,0)^T$,	
		$(1/6,5/6)^T$, $(3/5,6/5)^T$,	
		$(4/5,4/5)^T$	

Table 2.2 lists the iterations of Algorithm 2.2 in Example 2.3. In the first iteration an infeasible vertex $(0,6/5)^T$ is used to generate the first hyperplane, which leads to a feasible point $(0,0)^T$. The second iteration selects an infeasible vertex $(1,6/5)^T$ and finds another

feasible point $(1,0)^T$. So far the infeasible points are $(6/25,6/5)^T$ and $(14/20,6/5)^T$. In the third and fourth iteration these two points are removed by two new hyperplanes. Eventually all vertices of $\mathcal{S}^4$ are feasible and $\mathcal{S}^4$ is the same as $\mathcal{D}$. Thus all $\mathcal{K}$-maximal vertices and facets of $\mathcal{D}$ are found.

2.2 Convex multi-objective optimisation

A convex multi-objective programme (CMOP) is a multi-objective optimisation problem with convex objective functions and convex feasible set. Consider a multi-objective optimisation problem

$$\min \{f(x) : x \in \mathbb{R}^n, g(x) \leqq 0\}, \tag{CMOP}$$

where $f(x) = (f_1(x), \ldots, f_p(x))^T$ and $g(x) = (g_1(x), \ldots, g_m(x))^T$. If $f(x)$ and $g(x)$ are convex, i.e., the objectives and the constraints are all convex functions, the problem is a convex multi-objective optimisation problem. The problem discussed in this section involves non-linear objectives or constraints.

In practice it is adequate to obtain an approximation of the non-dominated set $\mathcal{P}_N$ of (CMOP) with a tolerance $\epsilon \in \mathbb{R}^p$ and $\epsilon > 0$ given by the decision maker. In this study this concept is modelled by (weakly) ϵ-non-dominance, which serves to measure the quality of the approximation.

Definition 2.4. *Let $\epsilon \in \mathbb{R}^p_{\geq}$. A point y is called (weakly) ϵ-non-dominated if $y + \epsilon \in \mathcal{Y}$ and there does not exist any $\hat{y} \in \mathcal{Y}$ such that $\hat{y} \leq (<) y$.*

Example 2.4. Below is an example of (CMOP). The image of the feasible set in the objective space is illustrated in Figure 2.7. The red curve is the non-dominated set of the

CMOP.

$$\min \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$
$$\text{s.t. } (x_1 - 2)^2 + (x_2 - 2)^2 \leqq 4.$$

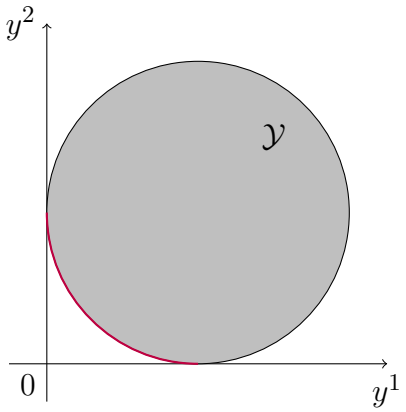


Figure 2.7 The image of the feasible set of Example 2.4 in objective space.

2.2.1 An outer approximation algorithm for (CMOP)

In order to solve (CMOP) an algorithm is proposed by Ehrgott et al. (2011b). This algorithm is an extension of Benson's outer approximation algorithm. It provides a set of ϵ-non-dominated points by means of approximating the non-dominated set of (CMOP). This algorithm first constructs a polyhedron containing $\mathcal{P}$. In each iteration, a vertex that is not an ϵ-non-dominated point of $\mathcal{P}$ is chosen to generate a supporting hyperplane to $\mathcal{P}$. Then the approximation polyhedron is updated by intersecting it with the half space containing $\mathcal{P}$ defined by the supporting hyperplane. The algorithm terminates when all of the vertices are ϵ-non-dominated. In this section we review the revised version of this algorithm proposed by Löhne et al. (2014), in which only one optimisation problem needs to be solved in each iteration. We first introduce two pairs of single objective optimisation

problems to facilitate the description of the algorithms later. Problem $\mathbb{P}_1(v)$ is a weighted sum problem. Solving $\mathbb{P}_1(v)$ results in a weakly non-dominated point. Problem $\mathbb{D}_1(v)$ is the Lagrangian dual of $\mathbb{P}_1(v)$. Problems $\mathbb{P}_2(y)$ and $\mathbb{D}_2(y)$ are employed to generate supporting hyperplanes. These four optimisation problems are the nonlinear extension of the LPs $P_1(v)$, $D_1(v)$, $P_2(y)$ and $D_2(y)$ respectively, which serve similar purposes as in Algorithm 2.1 and Algorithm 2.2.

$$\min \{\lambda(v)^T f(x) : x \in \mathbb{R}^n, g(x) \leqq 0\}. \tag{$\mathbb{P}_1(v)$}$$

$$\max \left\{ \min_{x \in \mathcal{X}} \left[\lambda(v)^T f(x) + u^T g(x) \right] : u \geqq 0 \right\}. \tag{$\mathbb{D}_1(v)$}$$

$$\min \{z \in \mathbb{R} : g(x) \leqq 0, f(x) - ze - y \leqq 0\}. \tag{$\mathbb{P}_2(y)$}$$

$$\max \left\{ \min_{x} \{u^T g(x) + \lambda^T f(x)\} - \lambda^T y : u \geqq 0, e^T \lambda = 1, \lambda \geqq 0 \right\}. \tag{$\mathbb{D}_2(y)$}$$

In this context, these four optimisation problems involve nonlinear convex terms making them hard to solve. However, given that $f(x)$ and $g(x)$ are differentiable , the nonlinear term can be linearized in the way stated in Section 5 of Ehrgott et al. (2011b), to which we refer the readers for more details. Below is the revised outer approximation algorithm by Löhne et al. (2014).

Example 2.5. Figure 2.8 illustrates the first few iterations of Algorithm 2.3 when solving Example 2.4. In this example the ideal point is $(0, 0)^T$. The first cut is generated as shown in Figure 2.8 (b). This cut generates two new vertices. If both of them are ϵ-non-dominated points, the algorithm terminates. Otherwise a new cut is computed through the vertices shown in Figure 2.8 (c) and (d). Eventually the algorithm ends with a set of ϵ-non-dominated points and a set of hyperplanes which form an outer approximation of $\mathcal{P}$.

Algorithm 2.3 Outer approximation algorithm for (CMOP)

Input: (CMOP), ϵ (tolerance given by DM).
Output: $Y_{\epsilon N}$ (set of ϵ-non-dominated points).

Compute an optimal solution x^i and an optimal value y_i^I of $(\mathbb{P}_1(e_\epsilon^i))$, for $i = 1, \ldots, p$. $Y_{\epsilon N} := \{f(x^i)\}$, for $i = 1, \ldots, p$.
Set $\mathcal{S}^0 := \{y^I\} + \mathbb{R}^p_{\leqq}$; $V_{\mathcal{S}^0} := \{y^I\}$ and $k := 1$.
while $V_{\mathcal{S}^{k-1}} \not\subset \mathcal{P}_{\epsilon N}$ **do**
  Choose a vertex s^k of $\mathcal{S}^{k-1}$.
  Compute an optimal solution (x^k, z^k) to $\mathbb{P}_2(s^k)$ and an optimal solution (u^k, λ^k) to $\mathbb{D}_2(s^k)$.
  if $z^k > \epsilon$ **then**
    Set $\mathcal{S}^k := \mathcal{S}^{k-1} \cap \{y \in \mathbb{R}^p : \varphi(y, (\lambda_1^k, \ldots, \lambda_{p-1}^k, b^T u^k)) \geqq 0\}$; Update $V_{\mathcal{S}^k}$; $Y_{\epsilon N} := Y_{\epsilon N} \cup f(x^k)$.
  else
    $Y_{\epsilon N} := Y_{\epsilon N} \cup s^k$.
  end if
  Set $k := k + 1$
end while

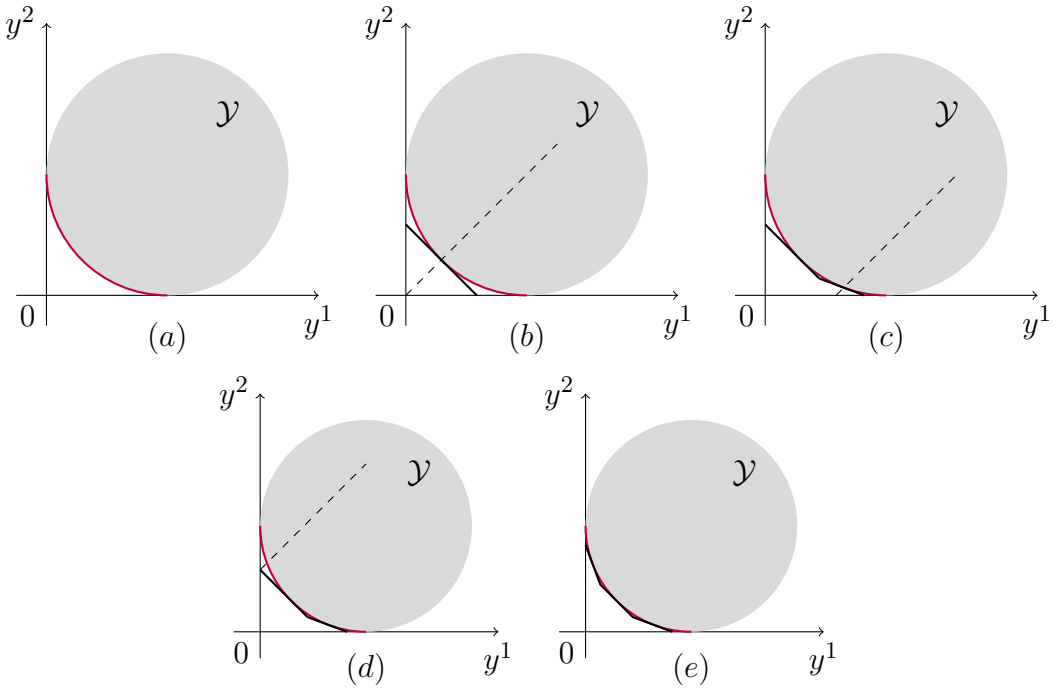


Figure 2.8 Outer approximation algorithm for (CMOP)

2.2.2 Dual method for (CMOP)

Löhne et al. (2014) propose a dual variant of Algorithm 2.3 to solve (CMOP) in the dual objective space. The geometric dual of (CMOP) is defined as

$$\max_{\mathcal{K}} \{D(v) : v \in \mathbb{R}^p, \lambda(v) \geq 0\}, \tag{DCMOP}$$

where $D(v) = \left\{v_1, \ldots, v_{p-1}, \min_{x\in\mathcal{X}} \left[\lambda(v)^T f(x)\right]\right\}$. The ordering cone $\mathcal{K} := \{v \in \mathbb{R}^p : v_1 = v_2 = \cdots = v_{p-1} = 0, v_p \geqq 0\}$ is the same as in (DMOLP), and maximisation is with respect to the order defined by $\mathcal{K}$. Let $\mathcal{V}$ denote the feasible set in the dual objective space, then the extended feasible set in the dual objective space is $\mathcal{D} := \mathcal{V} - \mathcal{K}$. The $\mathcal{K}$-maximal set of (DCMOP) is

$$\mathcal{D}_{\mathcal{K}} = \max_{\mathcal{K}}\{(\lambda_1, ..., \lambda_{p-1}, \min_{x\in\mathcal{X}} \left[\lambda(v)^T f(x)\right])^T : (u, \lambda) \geqq 0, e^T\lambda = 1\}.$$

Figure 2.9 shows the extended feasible set of Example 2.4 in dual objective space and the red curve is the $\mathcal{K}$-maximal set. For (CMOP) we employ the idea of ϵ-non-dominance. Similarly for (DCMOP) we use the idea of $\epsilon\mathcal{K}$-maximum.

Definition 2.5. *Let $\epsilon \in \mathbb{R}$ and $\epsilon > 0$. A point v is called an $\epsilon\mathcal{K}$-maximal point if $v - \epsilon e^p \in \mathcal{D}$ and there does not exist any $\hat{v} \in \mathcal{D}$ such that $\hat{v}_j = v_j$ for $j = 1, \ldots, p-1$ and $\hat{v}_p > v_p$.*

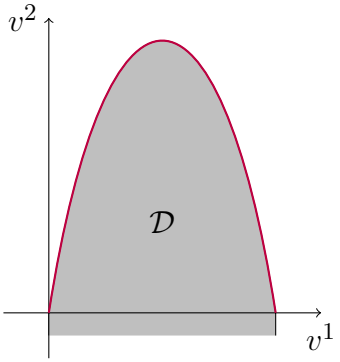


Figure 2.9 The extended feasible set of Example 2.4 in dual objective space.

Algorithm 2.4 for solving (CMOP) performs an outer approximation to $\mathcal{D}$ in dual objective space. This algorithm firstly chooses an interior point in $\mathcal{D}$. This is implemented in the way stated in Algorithm 2 (i1) in Ehrgott et al. (2011a). Then a polyhedron $\mathcal{S}^0$ containing $\mathcal{D}$ is constructed. In each iteration a vertex $s^k \notin \mathcal{D}_{\epsilon\mathcal{K}}$ of $\mathcal{S}^k$ is chosen. By solving $(\mathbb{P}_1(s^k))$, a supporting hyperplane is determined. Eventually, a set of $\epsilon\mathcal{K}$-maximal points of $\mathcal{D}$ is obtained.

Algorithm 2.4 Dual variant of Algorithm 2.3

Input: (CMOP)
Output: $V_{\epsilon\mathcal{K}}$ a set of $\epsilon\mathcal{K}$-maximal points of $\mathcal{D}$.
1: Choose some $\hat{d} \in \text{int}\mathcal{D}$.
2: Compute an optimal solution x^0 of $\mathbb{P}_1(\hat{d})$;
3: Set $S^0 := \{v \in \mathbb{R}^p : \lambda(v) \geqq 0, \{v \in \mathbb{R} : \varphi^*(f(x^0), v) \geqq 0\}$ *and* $k := 1$.
4: **while** $V_{\mathcal{S}^{k-1}} \subsetneq \mathcal{D}_{\epsilon\mathcal{K}}$ **do**
5: Choose a vertex s^k of $\mathcal{S}^{k-1}$.
6: Compute an optimal solution x^k of $(\mathbb{P}_1(s^k))$ and an optimal value y^k to $(\mathbb{P}_1(s^k))$.
7: **if** $s_q^k - y^k > \epsilon$ **then**
8: Set $\mathcal{S}^k := \mathcal{S}^{k-1} \cap \{v \in \mathbb{R} : \varphi^*(f(x^k), v) \geqq 0\}$. Update $V_{\mathcal{S}^k}$.
9: **else**
10: $V_{\epsilon\mathcal{K}} := V_{\epsilon\mathcal{K}} \cup s^k$.
11: **end if**
12: Set $k := k + 1$.
13: **end while**

Example 2.6. Figure 2.10 shows the first few iterations of Algorithm 2.4 by solving Example 2.4. In Figure 2.10 (a) the initial polyhedron containing $\mathcal{D}$ is constructed. In each iteration a cut is derived from a vertex that is not an $\epsilon\mathcal{K}$-maximal point. If there does not exist such a vertex the algorithm terminates with a set of $\epsilon\mathcal{K}$-maximal points and $\mathcal{K}$-maximal points and a set of hyperplanes which form an outer approximation of $\mathcal{D}_{\mathcal{K}}$.

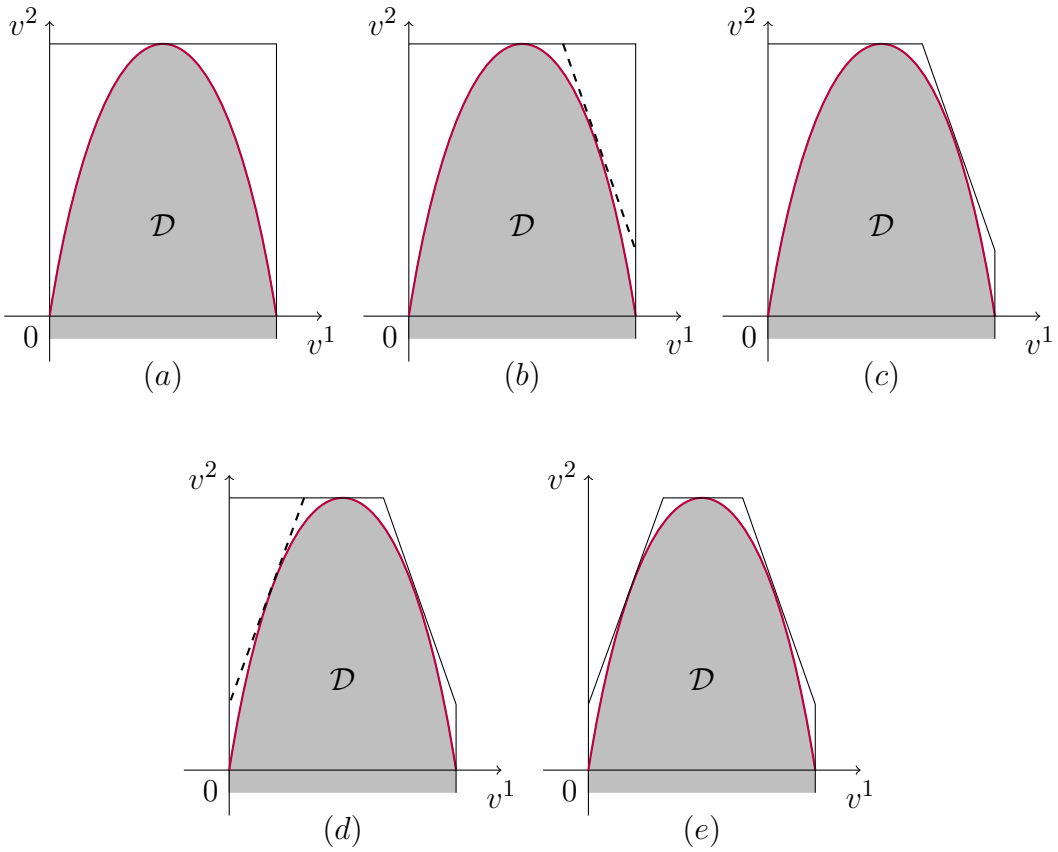


Figure 2.10 First few iterations of Algorithm 2.4 for Example 2.4.

Chapter 3

Literature Review

In Chapter 2 we introduced the multi-objective optimization problem (MOP) and several methods to approximate the non-dominated set $\mathcal{P}_N$. For any point $y \in \mathcal{P}_N$, there is a set of efficient solutions that map to y through f, i.e., $\{x \in \mathcal{X} : f(x) = y\}$. In practical applications of multi-objective optimisation, it is of course necessary that the decision maker has to select one solution from the efficient set for implementation. We assume that this selection process can be modelled by means of a function which is to be maximised over the efficient set of the underlying (MOP). Problems of this kind are known as optimisation over the efficient set of a multi-objective optimisation problem,

$$\max \{\Phi(x) : x \in \mathcal{X}_E\}, \tag{OE}$$

where $\Phi(x) : \mathbb{R}^n \to \mathbb{R}$ is a function of x; $\mathcal{X}_E$ is the efficient set of (MOP).

In this research the problem of interest is optimisation over the non-dominated set of (MOP), which is closely associated with (OE). We assume that the objective function Φ of (OE) is a composite function of a function $M : \mathbb{R}^p \to \mathbb{R}$ and the objective function f of (MOP), i.e., $\Phi = M \circ f$. Therefore, $\Phi(x) = M(f(x))$. Substituting $y = f(x)$ into (OE), we derive the problem:

$$\max \{M(y) : y \in \mathcal{P}_N\}. \tag{ON}$$

Under the assumption (ON) is essentially the same as (OE). However, the (ON) formulation is more intuitive than the (OE) formulation because in practice decision makers choose a preferred solution based on the objective function values rather than the decision variables. In the next section we briefly review some of the decision space based methods to solve this problem.

3.1 Decision space based algorithms

Philip (1972), Ecker and Song (1994), and Fülöp (1994), propose algorithms to solve a special case of (OE), where $\Phi(x)$ is a linear function and the underlying multi-objective optimisation problem is a multi-objective linear programme (MOLP). Bolintineanu (1993) optimises a quasi-convex function over the efficient set of (MOLP). All of these algorithms are based on two techniques. One technique is moving from one efficient vertex of $\mathcal{X}$ to an efficient neighbouring vertex with a better objective function value via an efficient edge. This is achievable due to the fact that the efficient set is connected (Steuer, 1986). The other technique is cutting off part of the feasible set where $\Phi(x)$ takes worse values than the incumbent value, i.e., the best function value found so far.

Benson (1992) proposes a nonadjacent vertex search algorithm to solve (OE) with an underlying (MOLP), which dispenses with vertex enumeration. The nonadjacent vertex search algorithm solves a sequence of linear programmes whose optimal solutions are efficient solutions for (MOLP). The sequence of the optimal solutions converges to an optimal solution of (OE). This algorithm can be regarded as an inner approximation algorithm.

Sayin (2000) introduces a face search algorithm which decomposes the efficient set into a finite number of faces. These faces are represented through index sets of non-zero components of decision variables. On each of the faces an optimisation problem needs to be solved in order to determine if a face is to be fathomed or to be explored further.

However the implementation of the method involves the list-management of the index sets. The rapid growth of the list gives rise to impracticality in computation.

White (1996), Dauer and Fosnaugh (1995) and An et al. (1996) introduce a gap function to reformulate (OE) as an optimization problem with a reverse convex constraint. This extra constraint can be relaxed as an objective function through introducing a Lagrangian multiplier. This Lagrangian relaxed problem has been further studied in order to solve (OE). This method is also used in Tuyen and Muu (2001), where (OE) is reformulated as a biconvex programming problem.

Thach et al. (1996) develop a duality approach to solve (OE). This approach is based on the nonconvex duality theory of Thach (1991). In this article a dual problem of (OE) is proposed and this dual problem is reformulated as a quasi-convex optimisation problem which can be solved iteratively. In Yamada et al. (2000) and Yamada et al. (2001) this dual approach is employed to generate cuts to approximate the feasible set from inside, i.e., an inner approximation procedure in decision space.

In Thai Quynh and Hoang Quang (2000), a bisection method is used for locating optima. It starts with an interval containing an optimal value of $\Phi(x)$, and then the interval is reduced until an approximate solution of desired quality is obtained.

A method designed by Le Thi et al. (2002) incorporates a global search scheme and a local search scheme. This method combines a branch and bound scheme and d.c. programming (optimisation problems with an objective function that is the difference of two convex functions). The bounding procedure is performed by using the weak duality theorem of Lagrange duality, whereas the d.c. programming formulation provides lower bounds.

3.2 Objective space based algorithms

Several articles explore the structures and properties of $\mathcal{X}_E$ and $\mathcal{Y}_N$ (Dauer (1987), Dauer (1993), Benson (1995)). Dauer (1987) notes that $\mathcal{Y}$ often has a much simpler structure,

i.e., fewer extreme points and faces, than $\mathcal{X}$. In the hope that computational effort might be saved, several algorithms have been developed since 2000. In this section, we review some of the objective space based methods, which explore the feasible polyhedron $\mathcal{Y}$ in objective space.

3.2.1 Polyblock approximation method

The polyblock approximation method, proposed by Nguyen Thi et al. (2008), is concerned with minimising $M(y)$ over $\mathcal{Y}_N$, rather than maximising as stated in (ON), where $M(y)$ is a continuous and increasing function. Note that a function M is increasing if for y', $y \in \mathbb{R}^p$ and $y' \geq y$ we have $M(y') \geqq M(y)$. We call this specific (ON) problem (SON),

$$\min \{M(y) : y \in \mathcal{Y}_N\}. \tag{SON}$$

For a, $b \in \mathbb{R}^p$ the set $[a,b] := \{y \in \mathbb{R}^p : a \leqq y \leqq b\}$ is called a "box". Obviously, the ideal point y^I and the anti-ideal point y^{AI} form a "box" $[y^I, y^{AI}]$ containing $\mathcal{Y}$.

Example 3.1. Figure 3.1 illustrates this algorithm by minimising $4y_1 + 5y_2$ over the non-dominated set of Example 2.1.

The algorithm starts with the initial box $[y^I, y^{AI}]$ shown in Figure 3.1 (a). In Figure 3.1 (b), a line connecting y^I and y^{AI} is drawn, which intersects with $\mathcal{Y}_N$ at $q = (1.2, 1.2)^T$, which is a non-dominated point. The function M is evaluated at q, $M(q) = 10.8$. Then $[y^I, q]$ is removed from $[y^I, y^{AI}]$ generating two new vertices, $v^1 = (0, 1.2)^T$ and $v^2 = (1.2, 0)^T$ with $M(v^1) = 6$; $M(v^2) = 4.8$. Since $M(v^1) > M(v^2)$, in Figure 3.1 (c) v^2 is chosen to be connected with y^{AI} resulting in an updated q and $M(q) \approx 10.17$. $M(v^3) \approx 8.08$, $M(v^4) \approx 6.90$. After a finite number of iterations, an approximate solution is obtained.

For details of the algorithm the reader is referred to Nguyen Thi et al. (2008). However, Theorem 3.1 below shows that it is unnecessary to use this algorithm to solve (SON), which in fact is equivalent to a linear programming problem shown in Theorem 3.1.

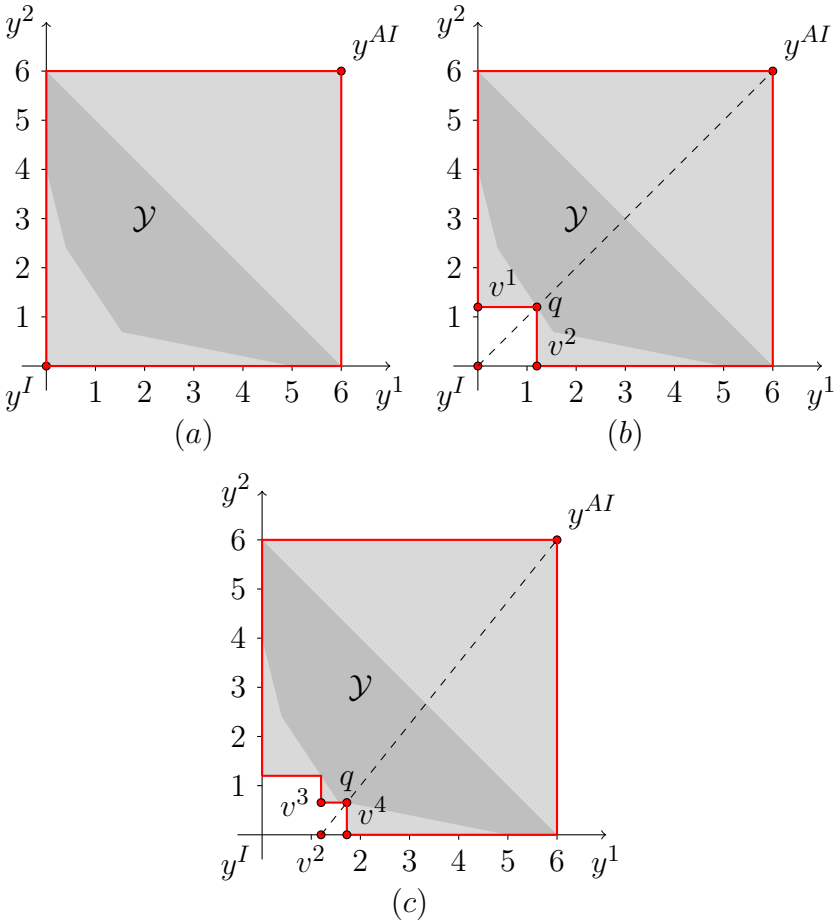


Figure 3.1 Polyblock approximation algorithm.

Theorem 3.1. *If $M(y)$ is continuous and increasing, then there exists an optimal solution to (RON) that is also optimal to (SON), where (RON) is*

$$\min \{M(y) : y \in \mathcal{Y}\}. \tag{RON}$$

Proof. Assume that y^* is an optimal solution to (RON) and $y^* \notin \mathcal{Y}_N$. Due to the fact that $\mathcal{Y}_N$ is externally stable, there exists a point $y' \in \mathcal{Y}_N$ such that $y' \leq y^*$ (see, e.g., Ehrgott (2005), Theorem 2.21). Since $M(y)$ is increasing, $M(y') \leqq M(y^*)$. If $M(y') < M(y^*)$ a point $y' \in \mathcal{Y}$ is obtained with $M(y')$ that is better than that of the presumed optimal point y^*, a contradiction. Otherwise $M(y') = M(y^*)$ and a non-dominated point $y' \in \mathcal{Y}$ which minimises (SON) is obtained. □

3.2.2 Bi-objective branch and bound algorithm

This section is dedicated to the description of a bi-objective branch and bound algorithm to solve a special case of (ON), where $M(y)$ is a lower-semicontinuous function on the non-dominated set of a bi-objective linear programming problem. The bi-objective linear programming problem possesses some special properties, which help exploit the structure of the problem. This method was first proposed by Benson and Lee (1996), and further improved by Fülöp and Muu (2000). The revised version by Fülöp and Muu (2000) is reviewed in this section.

Example 3.2. We demonstrate the bi-objective branch and bound algorithm by maximising $y_1 + y_2$ over the non-dominated set of Example 2.1 in Figure 3.2.

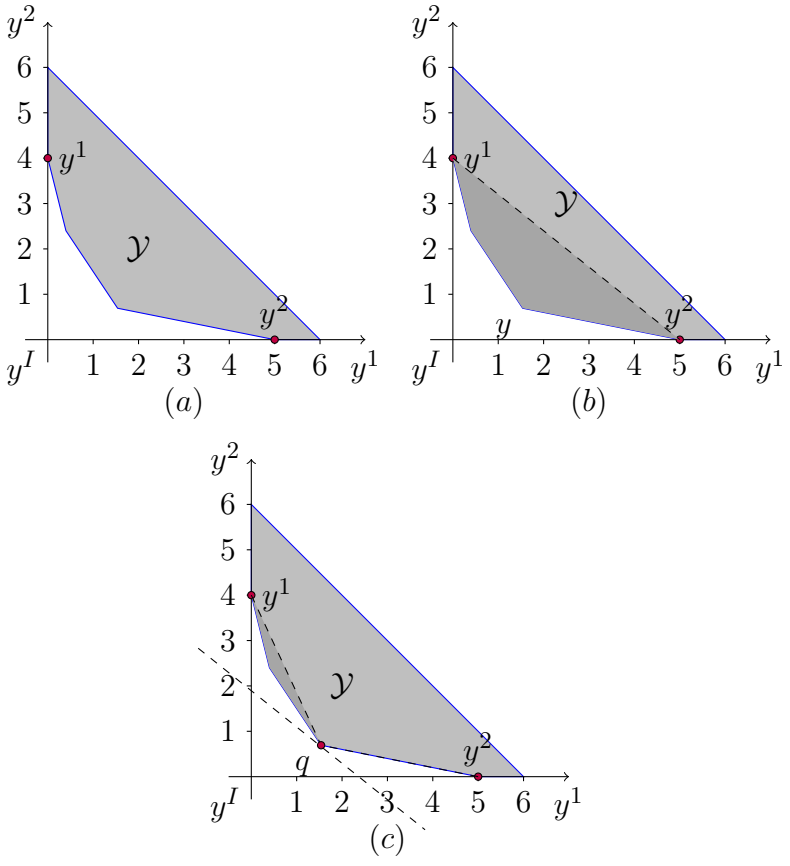


Figure 3.2 Bi-objective branch and bound algorithm.

Let $m_1 = \min\{y_1 : y_2 = y_2^I, y \in \mathcal{Y}\}$, and $m_2 = \min\{y_2 : y_1 = y_1^I,\ y \in \mathcal{Y}\}$. Let $y^1 = (y_1^I, m_2)^T$, and $y^2 = (m_1, y_2^I)^T$. These two points are non-dominated and therefore feasible to (ON), hence they can be used to find a lower bound on $y^1 + y^2$. In Figure 3.2 (a), $y^1 = (0,4)^T$, $y^2 = (5,0)^T$. A lower bound of 5 is obtained at y^2. Then optimisation problem (3.1) is solved to obtain an upper bound,

$$\max \left\{ M(y) : (y_2^2 - y_2^1)y_1 + (y_1^1 - y_1^2)y_2 \leqq y_1^1 y_2^2 - y_2^1 y_1^2, y \in \mathcal{Y} \right\}. \tag{3.1}$$

In the objective space, (3.1) means to find an optimal point over the shaded region which is below the dashed line segment shown in Figure 3.2 (b). Having solved problem (3.1), we have found an upper bound of 5 at y^2. At this point, the upper bound and the lower bound coincide. Therefore, we have solved the problem. In the case that $\Phi(x)$ is more complex (e.g., $\Phi(x)$ is nonlinear), branching steps may take place as shown in Figure 3.2 (c). Let the line segment connecting y^1 and y^2 shift parallel until it becomes a supporting hyperplane to $\mathcal{Y}$ at some point q. By connecting y^1 and y^2 with q, then the branching process splits the problem into two subproblems. For each of the subproblems the same process is repeated until the upper bound and the lower bound coincide. Computational experiments can be found in Fülöp and Muu (2000).

Kim and Thang (2013) extend this method to maximise an increasing function $M(y)$ over the non-dominated set of a convex bi-objective optimisation problem (CBOP). This algorithm starts with a simplex, from which a lower bound and an upper bound on the optimal objective function value can be attained. Then this simplex is divided into two simplices, each of which is to be explored in the subsequent iterations. The simplices with upper bounds that are worse than the incumbent objective function values are pruned. We use Figure 3.3 to illustrate this algorithm.

Example 3.3. Figure 3.3 shows an example of maximising an increasing function $M(y)$ over the non-dominated set of (CBOP). The first step (Figure 3.3 (a)) is to determine y^1 and y^2 in the same way as the linear version by Fülöp and Muu (2000). Notice that

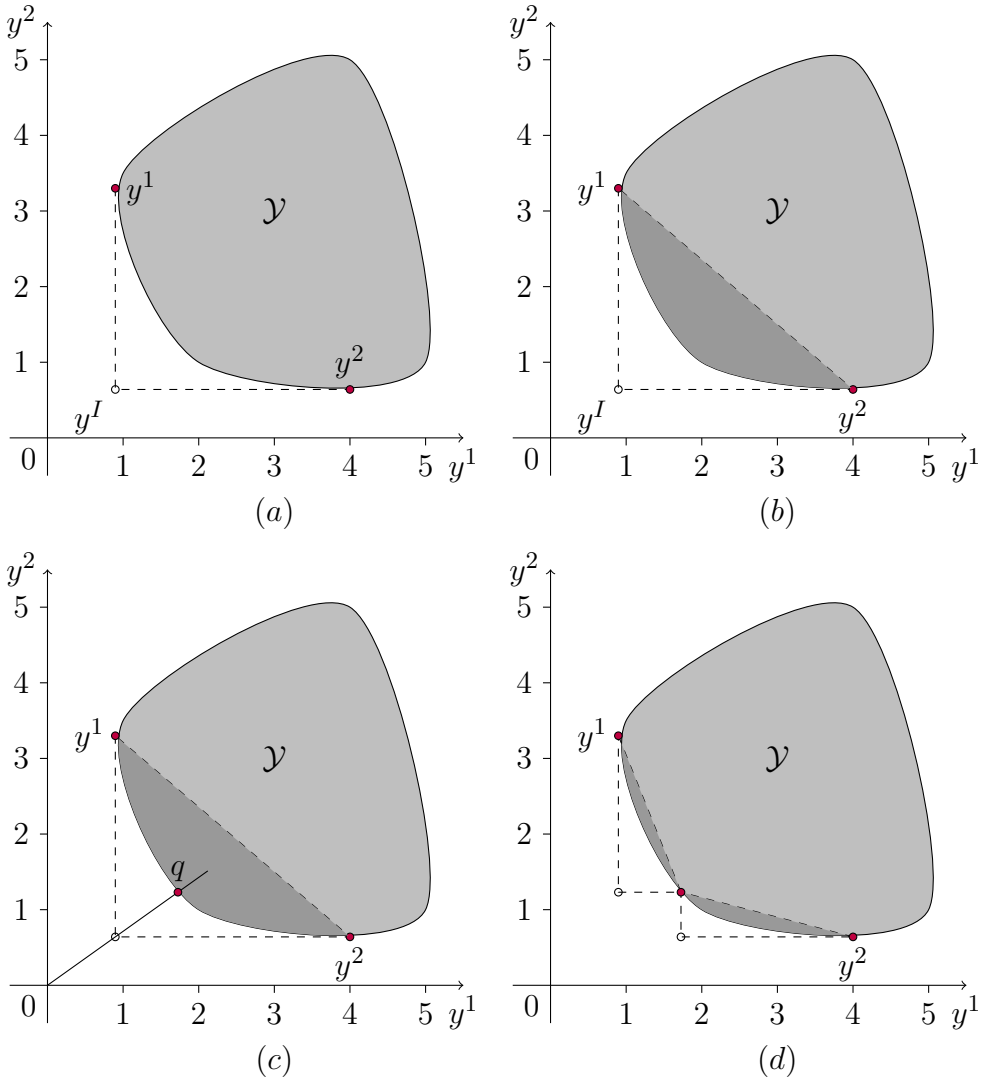


Figure 3.3 Convex bi-objective branch and bound method.

y^1 and y^2 are non-dominated. Therefore, a lower bound can be determined. These two points and the ideal point y^I define a simplex shown in Figure 3.3 (b). The objective function $M(y)$ is maximised over the intersection of $\mathcal{Y}$ and the simplex (shaded region on (b)). This step provides an upper bound. In Figure 3.3 (c) and (d), the branching step takes place. Point q splits the simplex into two. This point is at the intersection of a ray emanating from the origin and $\mathcal{Y}_N$. This process is iterated until the gap between the upper bound and the lower bound is within a tolerance determined initially by the decision maker.

3.2.3 Conical branch and bound algorithm

Another branch and bound algorithm was proposed by Thoai (2000), which optimises a continuous function over the non-dominated set of (MOP). A conical partition technique is employed as the branching process.

Example 3.4. In Figure 3.4, the conical branch and bound algorithm is demonstrated by maximising $y_1 + y_2$ over the non-dominated set of Example 2.1.

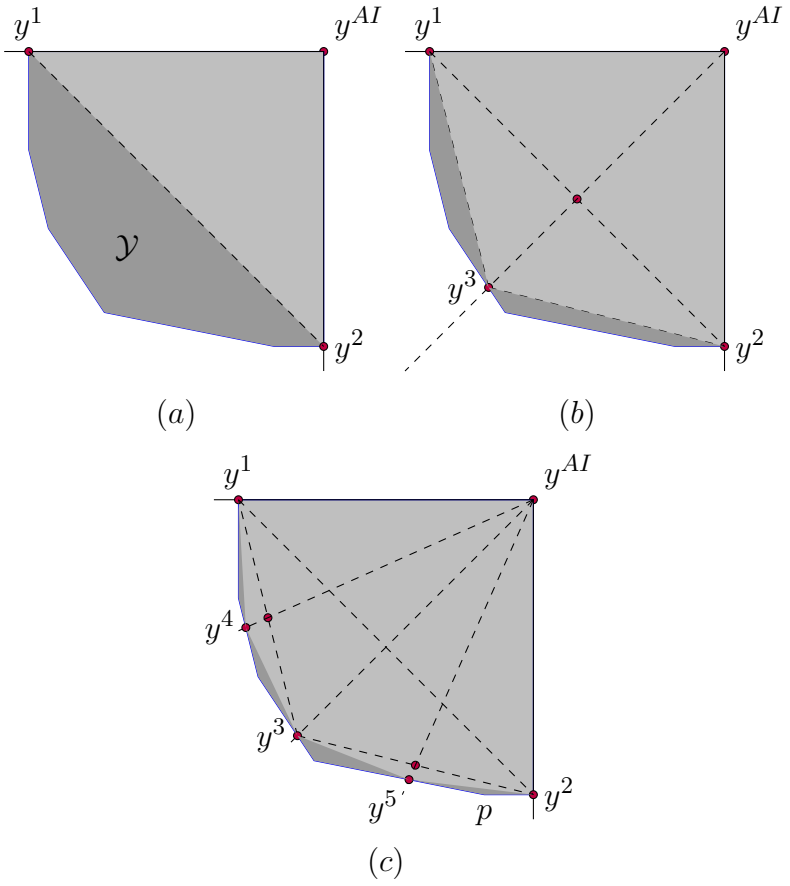


Figure 3.4 Conical branch and bound algorithm.

In Figure 3.4 (a), cone $y^{AI} - \mathbb{R}^p_{\leqq}$ with vertex y^{AI} is constructed. This cone contains $\mathcal{Y}$. Along each extreme direction of the cone, the intersection point of the direction and the weakly non-dominated set $\mathcal{P}_{WN}$ of the extended feasible set $\mathcal{P}$ is obtained. Here $y^1 = (0,6)^T$, $y^2 = (6,0)^T$. Neither y^1 nor y^2 is non-dominated, otherwise, a lower bound is

found. Then solving problem (3.2), a relaxation of (ON), finds an upper bound, here 6, at some point on the line segment connecting y^1 and y^2.

$$\begin{aligned} \max \quad & M(y) \\ \text{s.t.} \quad & y - U\lambda = y^{AI} \\ & \sum_{i=1}^{p} \lambda_i \geqq 1 \\ & \lambda \geqq 0 \\ & y \in \mathcal{Y}, \end{aligned} \tag{3.2}$$

where U is a matrix containing column vectors $y^1 - y^{AI}$ and $y^2 - y^{AI}$ in the example. Figure 3.4 (b) illustrates the branching step. A ray emanating from y^{AI} and passing through the mid-point of y^1 and y^2 hits the boundary of $\mathcal{Y}$ at y^3 $(1.2, 1.2)^T$. Since y^3 is a non-dominated point, a lower bound of 2.4, is achieved. The initial cone is partitioned into two cones defined by y^1, y^3, y^{AI} and y^2, y^3, y^{AI}, respectively. At this stage, neither the lower bound nor the upper bound is changed. In Figure 3.4 (c), the cones are further refined. By evaluating each cone, the gap between the upper bound and the lower bound is narrowed. An optimal point is obtained when the upper bound coincides with the lower bound. Table 3.1 shows the iterations of this method in Example 3.4. A cone is called active if there is a gap between the upper bound and lower bound. An active cone will be further explored. A cone is incumbent if the upper bound meets the lower bound with the best objective value so far. A cone is fathomed if the best feasible solution found in this cone is suboptimal. The convergence of the algorithm is discussed in Thoai (2000).

From Table 3.1, the reader may notice that in this example the cones defined by y^1 or y^2 stay active until they shrink to a ray. This is due to the fact that these cones contain points which are weakly non-dominated but not non-dominated (i.e., points belonging to $\mathcal{Y}_{WN} \backslash \mathcal{Y}_N$), e.g., the line segment between p and y^2 in Figure 3.4 (c). For solving (OE), $\mathcal{Y}_{WN} \backslash \mathcal{Y}_N$ is not worthy of exploration. Therefore, it is necessary to introduce a scheme to

Table 3.1 Iterations of the conical branch and bound algorithm in Example 3.4.

Iteration	Points defining cone	Upper bound	Lower bound	Status
1	y^1,y^2	6.00	$-\infty$	active
2	y^1,y^3	6.00	2.40	active
	y^3,y^2	6.00	2.40	active
3	y^1,y^4	6.00	3.35	active
	y^4,y^3	3.35	3.35	fathomed
	y^3,y^5	3.78	3.78	incumbent
	y^5,y^2	6.00	3.78	active
4	...	...	...	...

eliminate this part. The algorithm given in Thoai (2000) does not explicitly provide a way to do so. Proposition 3.1 provides a sufficient condition for detecting cones of this kind.

Proposition 3.1. *Given a cone considered in the conical branch and bound algorithm such that all intersection points y^i of extreme rays with $\mathcal{Y}_{WN}$ belong to $\mathcal{Y}_{WN}\backslash\mathcal{Y}_N$ assume that there there exists $j \in \{1,\dots,p\}$ such that y^i_j are equal for all i. Then for an optimal solution y^* to problem (3.2), then $y^* \in \mathcal{Y}_{WN}\backslash\mathcal{Y}_N$.*

Proof. If there exists j such that y^i_j are equal for all i, then the y^is lie on a facet of $\mathcal{Y}$. Furthermore, if all $y^i \in \mathcal{Y}_{WN}\backslash\mathcal{Y}_N$, then every point on the facet is in the set $\mathcal{Y}_{WN}\backslash\mathcal{Y}_N$. Therefore, an optimal solution to problem (3.2) is from the set $\mathcal{Y}_{WN}\backslash\mathcal{Y}_N$. □

Proposition 3.1 provides a way to detect the case when a cone is not worthy of further exploration. The modified version of the conical branch and bound algorithm removes the cone once such a case is found.

This method also deals with optimisation over the non-dominated set of (CMOP). Figure 3.5 illustrates a few steps of the algorithm for maximising a continuous function $M(y)$ over the non-dominated set of (CMOP). In Figure 3.5 (a) a cone is constructed to contain $\mathcal{Y}$. An upper bound of $M(y)$ can be found through maximising $M(y)$ over the shaded region of $\mathcal{Y}$. Figure 3.5 (b) illustrates the first branching step splitting the initial cone into two. A ray emanating from y^{AI} hits the non-dominated set of $\mathcal{Y}$ at y^3, a non-dominated point. A lower bound $M(y^3)$ is attained. The objective function $M(y)$ is maximised over each of the shaded regions of the cones, which provides new upper bounds.

In Figure 3.5 (c) each of the cones splits into two cones with new upper bounds and lower bounds. This process is repeated until the upper bound and the lower bound coincide or the gap between them is small enough.

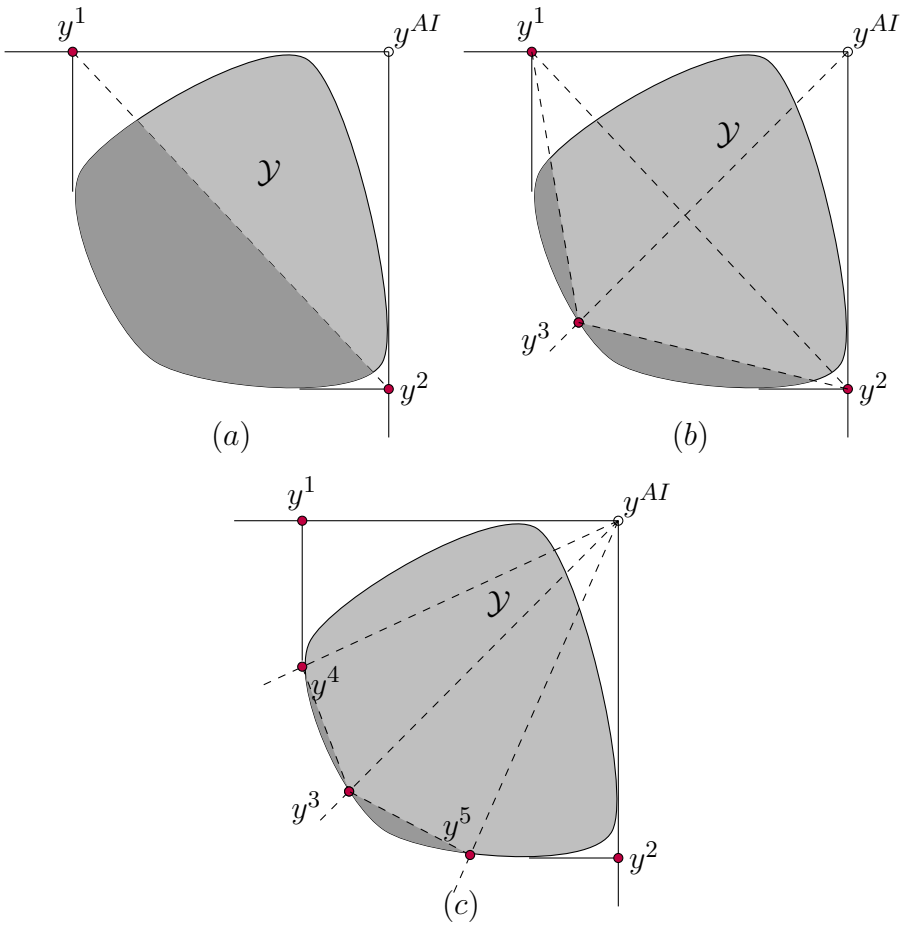


Figure 3.5 Conical branch and bound algorithm.

3.2.4 An outcome space algorithm

The branch and bound technique also plays an essential role in the algorithm designed by Benson (2011). This method was designed for optimising a finite, convex function over the weakly efficient set of a multi-objective nonlinear programming problem that has nonlinear objective functions and a convex, nonpolyhedral feasible set. In this paper the problem is reformulated as a convex programming problem with a single reverse

convex constraint. Then the branch and bound technique is employed to globally solve this problem. The initial simplex containing $\mathcal{Y}$ is first constructed. In each iteration a simplex is subdivided into two simplices. However, compared to the case where all of the constraints and objectives are linear, it is more complicated to find a feasible point and therefore, to establish lower bounds. A subroutine is developed that requires solving several optimisation problems to detect feasible points. A relaxed problem is used to find upper bounds by using a convex combination of the vertices of the simplex. The reader is referred to Benson (2011) for more details.

Chapter 4

Linear Optimisation over the Non-dominated Set of a Multi-objective Linear Programme

This chapter is devoted to a primal method and a dual method to maximise a linear function over the non-dominated set of a multi-objective linear programme. This problem is formulated as

$$\max \{\mu^T y : y \in \mathcal{P}_N\}, \tag{P}$$

where μ is a column vector with p elements, and $\mathcal{P}_N$ is the non-dominated set of the extended feasible set $\mathcal{P}$. In this chapter we investigate the properties of (P), which we will exploit in our primal algorithm. The dual method for (P) is developed based on some important properties of (P) in dual objective space.

Theorem 4.1. *An optimal solution y^* of (P) is obtained at a vertex of $\mathcal{P}$, i.e., $y^* \in V_{\mathcal{P}}$, the set of vertices of $\mathcal{P}$.*

Proof. Let y be an optimal solution of (P). Therefore, y can be expressed by a convex combination of the vertices of $\mathcal{P}$.

$$y = \sum_{i=1}^{r} \alpha_i \hat{y}^i,$$

where $\{\hat{y}^1, \ldots, \hat{y}^r\} = V_{\mathcal{P}}$ such that $0 \leqq \alpha_i \leqq 1$ for $i = 1, \ldots, r$ and $\sum_{i=1}^{r} \alpha_i = 1$.

Therefore we get

$$\begin{aligned} \mu^T y = \mu^T \sum_{i=1}^{r} \alpha_i \hat{y}^i &= \sum_{i=1}^{r} \alpha_i \mu^T \hat{y}^i \\ &\leqq \sum_{i=1}^{r} \alpha_i \mu^T \bar{y} \\ &= \mu^T \bar{y}, \end{aligned}$$

where $\bar{y} \in \operatorname{argmax}\{\mu^T \hat{y}^i : \hat{y}^i \in V_{\mathcal{P}}\}$.

It follows that $\mu^T y = \mu^T \bar{y}$ and therefore at least one of the extreme points of $\mathcal{P}$ is an optimal solution of (P). □

Theorem 4.1 implies that a naïve algorithm for solving (P) is to simply enumerate the vertices of $\mathcal{P}$ using Algorithm 2.1 and determine which one has the largest value of $\mu^T y$. Notice that the set of vertices of $\mathcal{P}$ is a subset of the non-dominated set, i.e., $V_{\mathcal{P}} \subset \mathcal{P}_N$. This is summarised in Algorithm 4.1.

Algorithm 4.1 Brute force algorithm

Input: (P)
Output: y^* (an optimal solution to (P))
Phase 1: Obtain $V_{\mathcal{P}}$ through Algorithm 2.1.
Phase 2: $y^* \in \operatorname{argmax}\{\mu^T y : y \in V_{\mathcal{P}}\}$.

However, taking advantage of some properties of (P) may dispense with the enumeration of all vertices of $\mathcal{P}$. We start this discussion by investigating the vertices of $\mathcal{Y}$ more closely. Let $[a-b]$ denote an edge of polyhedron $\mathcal{Y}$ with vertices a and b. We call points a and b neighbouring vertices and denote with $N(a)$ the set of all neighbouring vertices of vertex a.

Definition 4.1. *Let $[a-b]$ be an edge of $\mathcal{Y}$. $[a-b]$ is called a non-dominated edge if for some point y in the relative interior of $[a-b]$, $y \in \mathcal{Y}_N$.*

Yu (1985) (Chapter 8) shows that if a point in the relative interior of a face of a polyhedron is non-dominated, then the entire face is non-dominated.

Definition 4.2. *If $[a-b]$ is a non-dominated edge for all $b \in N(a)$, then a is called a complete vertex, otherwise it is called an incomplete vertex. Let $V_{\mathcal{Y}}^{c}$ denote the set of complete vertices of $\mathcal{Y}$, and define $V_{\mathcal{Y}}^{ic} := V_{\mathcal{Y}} \setminus V_{\mathcal{Y}}^{c}$ as the set of incomplete vertices.*

Notice that a complete vertex must be a non-dominated vertex. Figure 4.1 shows the set $\mathcal{Y}$ of Example 2.1. The complete vertices are b and c. The incomplete vertices are a, d, e and f. The distinction between $\mathcal{Y}$ and $\mathcal{P}$ is important here: All vertices of $\mathcal{P}$ are complete, whereas $\mathcal{Y}$ also has dominated vertices, and hence incomplete vertices must exist.

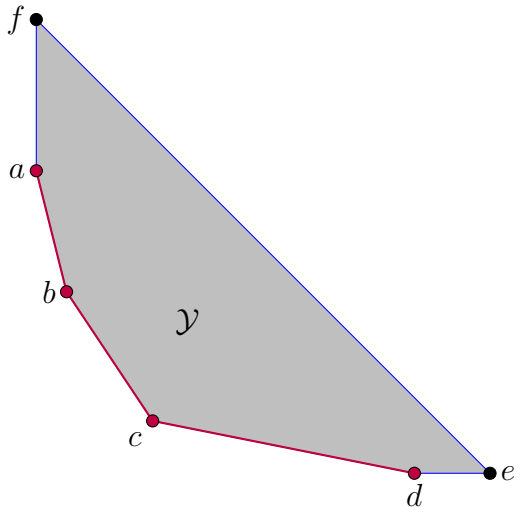


Figure 4.1 Complete and incomplete vertices.

Proposition 4.1. *Let a be a vertex of $\mathcal{Y}$. If there does not exist a vertex $b \in N(a)$ such that $\mu^T(b-a) > 0$, then a is an optimal solution of the linear programme*

$$\max \{\mu^T y : y \in \mathcal{P}\}. \tag{RP}$$

Proof. Assume a is not an optimal solution, then there exists at least one vertex $b \in N(a)$ such that $\mu^T b > \mu^T a$, which means $\mu^T(b-a) > 0$, a contradiction. □

Theorem 4.2. *(P) has the same optimal solution as (RP), if and only if* $\mu \in \mathbb{R}^p_{\leq}$.

Proof.

1. First, let $\mu \in \mathbb{R}^p_{\leq}$ and let y^* be an optimal solution of (RP). We show that y^* is an optimal solution of (P). Let $\mu' := -\mu$, then $\mu' \in \mathbb{R}^p_{\geq}$. We rewrite (RP) as

$$\min\{\mu'^T Cx : x \in \mathcal{X}\},$$

which is a weighted sum scalarisation of the underlying (MOLP). If $\mu' = 0$, any x is optimal, so y^* is an optimal solution of (P). If $\mu' \geq 0$, it is well known that there exists an efficient solution $x^* \in \mathcal{X}$ which is an optimal solution of (RP). Therefore, $y^* = Cx^* \in \mathcal{P}_N$. Hence, y^* is an optimal solution of (P).

2. Let $\mu \in \mathbb{R}^p_{\leq}$, and let $y^* \in \mathcal{P}_N$ be an optimal solution of (P). Assume there exists $\hat{y} \in \mathcal{P} \backslash \mathcal{P}_N$ such that $\mu^T \hat{y} > \mu^T y^*$. Since $\hat{y} \notin \mathcal{P}_N$, $\hat{y} = y^* + d$, where $d \geq 0$. Therefore, $\mu^T \hat{y} = \mu^T(y^* + d) = \mu^T y^* + \mu^T d$. Hence, $\mu^T d > 0$. Since $d_k \geq 0$, $k = 1, \ldots, p$, there exists k such that $\mu_k > 0$. This contradicts $\mu \in \mathbb{R}^p_{\leq}$, and therefore, y^* is an optimal solution of (RP).

3. Now, let $\mu \notin \mathbb{R}^p_{\leq}$ and assume that y^* is an optimal solution of both (P) and (RP). Choose $d \in \mathbb{R}^p \setminus \mathbb{R}^p_{\leq}$ such that $d_j = 1$ if $\mu_j > 0$ and $d_j = 0$ otherwise. Let $y' = y^* + d$. Since $\mathcal{P} = \mathcal{Y} + \mathbb{R}^p_{\geq}$, $y' \in \mathcal{P}$. However, $\mu^T y' = \mu^T y^* + \mu^T d > \mu^T y^*$, which contradicts the optimality of y^* for (RP).

□

Proposition 4.2. *Let* $\mu \in \mathbb{R}^p \setminus \mathbb{R}^p_{\leqq}$. *If* $y \in V^c_{\mathcal{Y}}$, *then there exists* $y' \in N(y)$ *such that* $\mu^T(y' - y) > 0$.

Proof. Let μ and y be as in the proposition. If there were no $y' \in N(y)$ such that $\mu^T(y' - y) > 0$, then y would be an optimal solution to (RP). As y is also a non-dominated point, y would be an optimal solution to (P), according to Theorem 4.2 this implies $\mu \in \mathbb{R}^p_{\leqq}$, a contradiction. □

The main result of this section is Theorem 4.3.

Theorem 4.3. *For all $\mu \in \mathbb{R}^p \setminus \mathbb{R}^p_{\leqq}$, an optimal solution y^* of (P) is attained at an incomplete non-dominated vertex, i.e., $y^* \in V^{ic}_{\mathcal{Y}} \cap V_{\mathcal{P}}$.*

Proof. Under the assumptions of the theorem, assume that $V^{ic}_{\mathcal{Y}}$ does not contain an optimal solution of (P), then because of Theorem 4.1 there exists an optimal solution $y^* \in V^c_{\mathcal{Y}}$. According to Proposition 4.2, there exists $y \in N(y^*)$ such that $\mu^T y > \mu^T y^*$. Since $y \in \mathcal{Y}_N$, y is feasible for (P) and a contradiction is obtained. □

According to Theorem 4.2, whenever $\mu \leqq 0$, (P) can be solved by solving the LP (RP). In case $\mu \in \mathbb{R}^p \setminus \mathbb{R}^p_{\leqq}$, an optimal solution of (P) must be obtained at an incomplete vertex of $\mathcal{Y}$. Algorithm 4.1 can therefore be restricted to incomplete vertices. Unfortunately, due to the fact that the structure of $\mathcal{P}_N$ can be very complex (see for example Fruhwirth and Mekelburg (1994) for an investigation of the structure of the non-dominated set of tri-objective linear programmes), a necessary and sufficient condition for checking a vertex to be incomplete is not known. In the next section, we provide an algorithm that uses cutting planes to eliminate the enumeration of all vertices of $\mathcal{P}$.

4.1 The primal method for (P)

Algorithm 4.1 has two distinct phases, vertex generation and vertex evaluation. Combining and integrating both phases and exploiting the properties of (P), we expect that we can save computational effort, since not all vertices of $\mathcal{P}$ need to be enumerated. More specifically, once a new hyperplane is generated and added to the approximation polyhedron $\mathcal{S}^{k-1}$ as a cut, a set of new extreme points of $\mathcal{S}^k$ is found. Evaluating $\mu^T y$ at these extreme points, we select the one with the best function value as s^k in the next iteration. If the selected point is infeasible, a cut called improvement cut $\{y \in \mathbb{R}^p : \varphi(y, (\lambda^k_1, \ldots, \lambda^k_{p-1}, b^T u^k)) \geqq 0\}$ is added. This cut is used in the revised version of Benson's algorithm. Otherwise another type of cut, called threshold cut, is added. A threshold cut is $\{y \in \mathbb{R}^p : \mu^T y \geqq \mu^T \hat{y}\}$, where

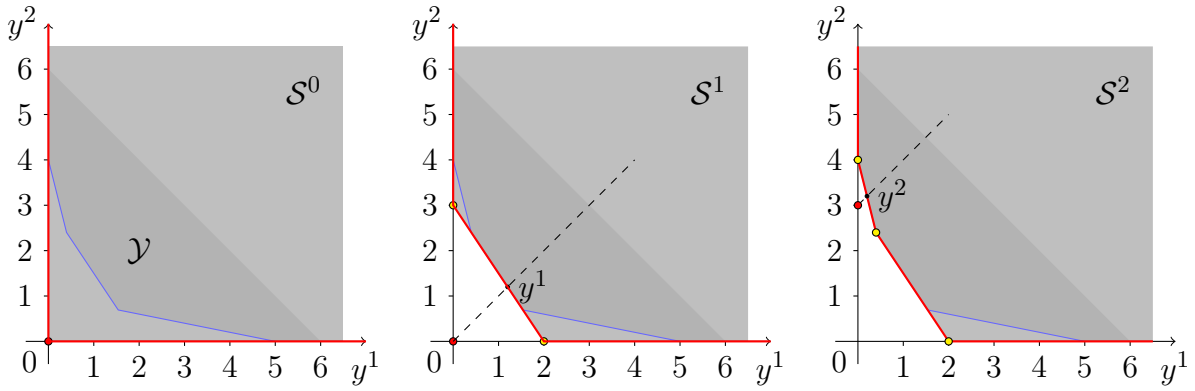
$\hat{y}$ is the incumbent solution, i.e., the best feasible non-dominated point found so far. A threshold cut removes the region where points are worse than $\hat{y}$. This primal method is detailed in Algorithm 4.2.

Algorithm 4.2 Primal algorithm

Input: (P)
Output: y^* (an optimal solution to (P))

Compute the optimal value y_i^I of $(P_1(e^i))$, for $i = 1, \ldots, p$.
Set $\mathcal{S}^0 := \{y^I\} + \mathbb{R}^p_{\geqq}$, $k := 1$ and $V_{\mathcal{S}^0} := \{y^I\}$.
Threshold := False.
while $V_{\mathcal{S}^{k-1}} \not\subset \mathcal{P}$ **do**
  $s^k \in \operatorname{argmax}\{\mu^T y : y \in V_{\mathcal{S}^{k-1}}\}$,
  if $s^k \in \mathcal{P}$ and Threshold = False **then**
    $\mathcal{S}^k := \mathcal{S}^{k-1} \cap \{y \in \mathbb{R}^p : \mu^T y \geqq \mu^T s^k\}$. Update $V_{\mathcal{S}^k}$. Threshold := True.
  else
    Compute an optimal solution (x^k, z^k) to $P_2(s^k)$ and its dual optimal variable values (u^k, λ^k).
    Set $\mathcal{S}^k := \mathcal{S}^{k-1} \cap \{y \in \mathbb{R}^p : \varphi(y, (\lambda_1^k, \ldots, \lambda_{p-1}^k, b^T u^k)) \geqq 0\}$. Update $V_{\mathcal{S}^k}$. Threshold := False.
  end if
  Set $k := k + 1$.
end while
$y^* := s^k$.

Example 4.1. The primal method is illustrated in Figure 4.2 by maximising $y_1 + y_2$ over the non-dominated set of Example 2.1.



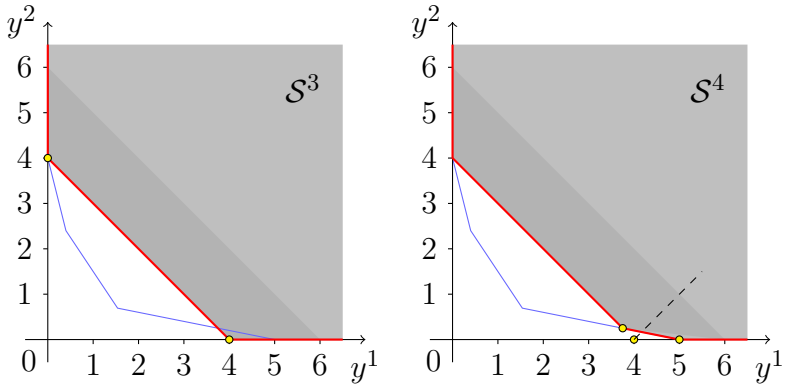


Figure 4.2 The primal method.

Table 4.1 Iterations of the primal algorithm in Example 4.1.

Iteration k	Vertex s^k	Cut type	Candidates	Non-dominated points	$\mu^T y$
1	(0,0)	Improvement	(2,0),(0,3)	$\emptyset$	3
2	(0,3)	Improvement	(2,0)	(0,4), (0.4,2.4)	4
3	(0,4)	Threshold	(4,0)	(0,4)	4
4	(4,0)	Improvement	$\emptyset$	(0,4), (5,0), (0.25,3.75)	5

Figure 4.2 and Table 4.1 show in each iteration the extreme point chosen, the type of cut added and the objective function value. In the first iteration, an improvement cut is added generating two new vertices, $(2,0)^T$ and $(0,3)^T$. Then $(0,3)^T$ is chosen in the next iteration because it provides the best objective function value so far. The second iteration improves the function value to 4. Since $(0,4)^T$ is feasible, a threshold cut, $y_1 + y_2 \geqq 4$, is then added. Although it does not improve the objective function value, this cut generates a new infeasible vertex $(4,0)^T$. An optimal point, $(5,0)^T$, is found after another improvement cut has been generated.

4.2 The dual method for (P)

In this section, we explore some properties of (P) in dual objective space, which enable us to solve (P) more efficiently than by the primal algorithm of Section 4.1. We call this the dual algorithm.

Let y^{ex} be an extreme point of $\mathcal{P}$. Hence $y^{ex} \in \mathcal{Y}_N$. Via set-valued map H^*, y^{ex} corresponds to a hyperplane $H^*(y^{ex})$ in the dual objective space,

$$\begin{aligned} H^*(y^{ex}) &= \{v \in \mathbb{R}^p : \lambda^*(y^{ex})^T v = -y_p^{ex}\}, \\ &= \{v \in \mathbb{R}^p : (y_1^{ex} - y_p^{ex})v_1 +, ..., +(y_{p-1}^{ex} - y_p^{ex})v_{p-1} - v_p = -y_p^{ex}\}. \end{aligned}$$

Moreover, μ and y^{ex} define a hyperplane $H_{y^{ex}}$ in primal objective space

$$H_{y^{ex}} = \left\{y \in \mathbb{R}^p : \mu^T y = \mu^T y^{ex}\right\}. \tag{4.1}$$

We notice that without loss of generality we can assume that $\mu \geq 0$. Otherwise we set $\hat{\mu}_k = -\mu_k$ and $\hat{y}_k = -y_k$ whenever $\mu_k < 0$ and $\hat{\mu}_k = \mu_k$ and $\hat{y}_k = y_k$ whenever $\mu_k \geqq 0$ to rewrite (4.1) as

$$H_{y^{ex}} = \left\{\hat{y} \in \mathbb{R}^p : \hat{\mu}^T \hat{y} = \hat{\mu}^T \hat{y}^{ex}\right\}. \tag{4.2}$$

Let us now define $\mu' := \sum_{i=1}^{p} \mu_i$ and divide both sides of the equation in (4.2) by μ' to obtain

$$H_{y^{ex}} = \left\{y \in \mathbb{R}^p : \frac{\mu^T y}{\mu'} = \frac{\mu^T y^{ex}}{\mu'}\right\}. \tag{4.3}$$

Since (4.3) is a hyperplane in the primal objective space, it corresponds to a point v^μ in dual objective space. According to geometric duality theory, in particular (2.1), this point is nothing but

$$v^\mu = \left(\frac{\mu_1}{\mu'}, ..., \frac{\mu_{p-1}}{\mu'}, \frac{\mu^T y^{ex}}{\mu'}\right)^T.$$

Notice that only the last element of v^μ varies with y^{ex}, i.e., the first $p-1$ elements of v^μ are merely determined by μ. Geometrically, it means that v^μ with respect to various extreme points y^{ex} lies on a vertical line $L^\mu := \{v \in \mathbb{R}^p : v_1 = v_1^\mu, ..., v_{p-1} = v_{p-1}^\mu\}$. Furthermore, the last element of v^μ is equal to the objective function value of (P) at y^{ex} divided by μ'. Hence, geometrically, (P) is equivalent to finding a point v^μ with the largest last element along L^μ.

Theorem 4.4. *The point v^μ lies on $H^*(y^{ex})$.*

Proof. Substitute the point v^μ into the equation of $H^*(y^{ex})$. The left hand side is

$$\begin{aligned} LHS &= (y_1^{ex} - y_p^{ex})\frac{\mu_1}{\mu'} + , ..., + (y_{p-1}^{ex} - y_p^{ex})\frac{\mu_{p-1}}{\mu'} - \frac{\mu^T y^{ex}}{\mu'} \\ &= \left(\sum_{i=1}^{p-1} \mu_i y_i^{ex} - y_p^{ex} \sum_{i=1}^{p-1} \mu_i - \sum_{i=1}^{p-1} \mu_i y_i^{ex} - \mu_p y_p^{ex} \right) \frac{1}{\mu'} \\ &= -\frac{\sum_{i=1}^{p} \mu_i}{\mu'} y_p^{ex} = -y_p^{ex}. \end{aligned}$$

□

This discussion shows that, because we are just interested in finding a $H^*(y^{ex})$ that intersects L^μ at the highest point, i.e., the point with the largest last element value, it is unnecessary to obtain the complete $\mathcal{K}$-maximal set of $\mathcal{D}$. We now characterise which elements of this set we need to consider.

In Section 4.1, we reached the conclusion that an optimal solution to (P) can be found at an incomplete vertex of $\mathcal{Y}$. An analogous idea applies to the facets of the dual polyhedron. In the rest of this section, we derive this idea through the association between the primal and the dual polyhedra, and implement this idea to propose the dual algorithm.

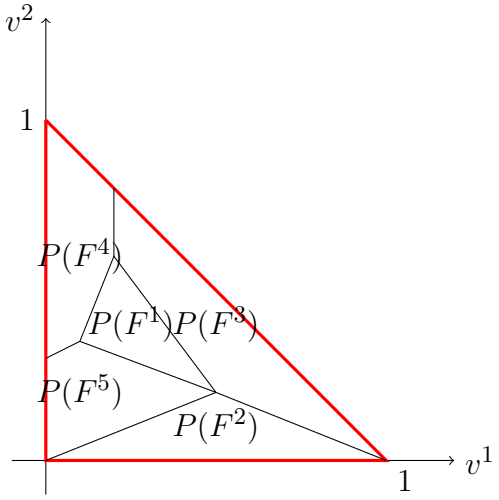


Figure 4.3 Projection of a three dimensional dual polyhedron onto the v^1-v^2 coordinate plane.

Figure 4.3 shows the projection of a three dimensional dual polyhedron onto the v^1 and v^2 coordinate plane. The cells $P(F^1)$ to $P(F^5)$ are the projection of facets F^1 to F^5 of $\mathcal{D}$, respectively.

Definition 4.3. *Two $\mathcal{K}$-maximal facets F^i and F^j of polyhedron $\mathcal{D}$ are called neighbouring facets if $dim(F^i \cap F^j) = p-2$.*

Figure 4.3 shows that the neighbouring facets of facet F^2 are F^3 and F^5. Neither F^4 or F^1 is a neighbouring facet of F^2.

Proposition 4.3. *If y^i, $y^j \in V_{\mathcal{P}}$, and y^i and y^j are non-dominated neighbouring vertices of $\mathcal{P}$, then facets $F^i = H^*(y^i) \cap \mathcal{D}$ and $F^j = H^*(y^j) \cap \mathcal{D}$ are neighbouring facets of $\mathcal{D}$.*

Proof. Since y^i and y^j are neighbouring vertices of $\mathcal{Y}$ there is an edge $[y^i - y^j]$ connecting them. Since the edge $[y^i - y^j]$ has dimension one and since according to Theorem 2.2 $dim([y^i - y^j]) + dim(\Psi^{-1}([y^i - y^j])) = p-1$. Therefore, $dim(\Psi^{-1}([y^i - y^j])) = p-2$. Moreover, $\Psi^{-1}([y^i - y^j]) = (H^*(y^i) \cap \mathcal{D}) \cap (H^*(y^j) \cap \mathcal{D}) = F^i \cap F^j$. Hence $dim(F^i \cap F^j) = p-2$ and F^i and F^j are neighbouring facets. □

Definition 4.4. *Let F be a $\mathcal{K}$-maximal facet of $\mathcal{D}$. If all neighbouring facets of F are $\mathcal{K}$-maximal facets, then F is called a complete facet, otherwise it is called an incomplete facet. The set of all complete facets of $\mathcal{D}$ is denoted by F^c, the set of all incomplete facets of $\mathcal{D}$ is denoted by F^{ic}.*

From the projection of the facets $P(F^1)$ to $P(F^5)$ in Figure 4.3 the incomplete facets are F^2, F^3, F^4 and F^5. The only complete facet is F^1.

Theorem 4.5. *There exists a one-to-one correspondence between incomplete facets of $\mathcal{D}$ and incomplete vertices of $\mathcal{Y}$.*

Proof. Proposition 4.3 states that the facets of $\mathcal{D}$ have the same neighbouring relations with the vertices of $\mathcal{Y}$, which implies the completeness of a facet of $\mathcal{D}$ remains the same as that of its corresponding vertex of $\mathcal{Y}$. Therefore, Theorem 4.5 is true. □

Theorem 4.6. *If y^* is an optimal extreme point solution of (P), then $(H^*(y^*) \cap \mathcal{D})$ is a $\mathcal{K}$-maximal incomplete facet of $\mathcal{D}$.*

Proof. Theorem 4.6 follows directly from Theorem 4.3 and Theorem 4.5. □

Theorem 4.6 says that a facet of $\mathcal{D}$ corresponding to an optimal solution to (P) is an incomplete facet. In other words, a complete facet cannot produce an optimal solution because in the primal objective space the vertex corresponding to this facet has only non-dominated neighbours, i.e., this vertex is a complete vertex, which cannot be an optimal solution of (P). On the other hand, an incomplete facet of $\mathcal{D}$ is the counterpart of an incomplete vertex of $\mathcal{Y}$. Hence an optimal solution to (P) can be obtained among the incomplete facets. In order to obtain the set of incomplete facets of $\mathcal{D}$, let us define

$$W_D := \bigcup_{i=1}^{p-1} \{v \in \mathbb{R}^p : v_i = 0, 0 \leqq v_j \leqq 1, j = 1 \dots (p-1), j \neq i\}$$

$$\cup \left\{ v \in \mathbb{R}^p : \sum_{i=1}^{p-1} v_i = 1, 0 \leqq v_i \leqq 1, i = 1 \dots (p-1) \right\}.$$

In Figure 4.3, the highlighted triangle is the projection of W_D onto the v^1-v^2 coordinate plane. The incomplete facets intersect with W_D because their neighbouring facets are not all $\mathcal{K}$-maximal facets. On the other hand, a complete facet "surrounded" by $\mathcal{K}$-maximal facets does not intersect with W_D. Hence, it is sufficient to consider the facets that intersect with W_D. The dual algorithm (Algorithm 4.3) is designed to solve (P) in the dual objective space through finding facets that intersect with W_D.

Algorithm 4.3 Dual algorithm

Input: (P)

Output: y^* (an optimal solution to (P))

1: Choose some $\hat{d} \in \text{int}\mathcal{D}$.

2: Compute an optimal solution x^0 of $(P_1(\hat{d}))$, set $y^* := Cx^0$, $M^* := \mu^T y^*$.

3: Set $\mathcal{S}^0 := \{v \in \mathbb{R}^p : \lambda(v) \geqq 0, \varphi(Cx^0, v) \geqq 0\}$ and $k := 1$.

4: **while** $W_D \cap V_{\mathcal{S}^{k-1}} \not\subset \mathcal{D}$ **do**

5: Choose $v^k \in W_D \cap V_{\mathcal{S}^{k-1}}$ such that $v^k \notin \mathcal{D}$.

6: Compute an optimal solution x^k of $(P_1(v^k))$, set $y^k := Cx^k$.

7: **if** $M^* < \mu^T y^k$ **then**

8: Set $y^* := y^k$ and $M^* := \mu^T y^k$.

9: **end if**

10: Set $\mathcal{S}^k := \mathcal{S}^{k-1} \cap \{v \in \mathbb{R} : \varphi(Cx^k, v) \geqq 0\}$.. Update $V_{\mathcal{S}}^k$.

11: Set $k := k + 1$.

12: **end while**

Example 4.2. Figure 4.4 below illustrates the dual algorithm by maximising $y_1 + y_2$ over the non-dominated set of Example 2.1. By employing the dual method, only two of the four $\mathcal{K}$-maximal hyperplanes need to be generated.

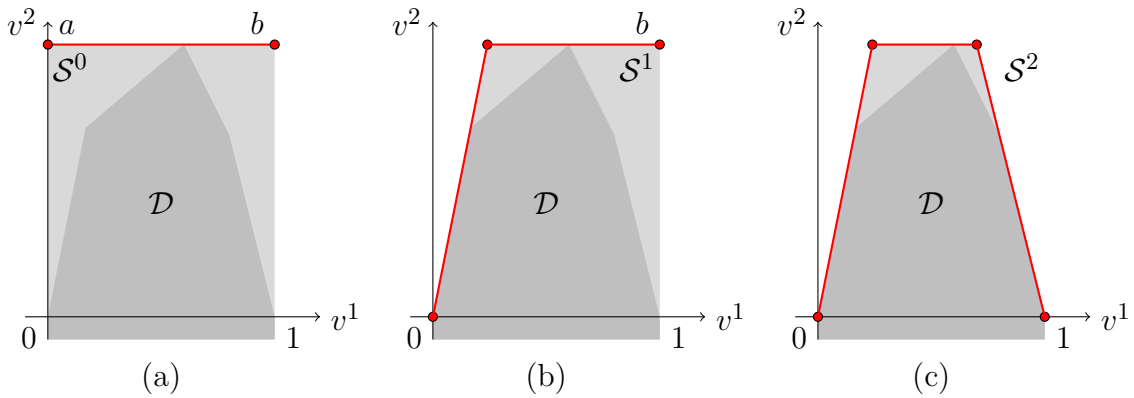


Figure 4.4 The dual algorithm.

In this example, $W_D = \{v \in \mathbb{R}^2 : v_1 = 0\} \cup \{v \in \mathbb{R}^2 : v_1 = 1\}$. In Figure 4.4 (a), $a, b \in W_D$. In Figure 4.4 (b), vertex a is used to generate a hyperplane, which corresponds to extreme point $(0,4)^T$ in the primal space. Meanwhile, a new vertex c is found. Since $c \notin W_D$, in Figure 4.4 (c)vertex b is employed to generate another hyperplane corresponding to extreme point $(5,0)^T$ in the primal space. At this stage, there is no infeasible vertex belonging to W_D and the algorithm terminates with optimal point $(5,0)^T$.

4.3 Computational experiments

In this section, we use randomly generated instances to compare some of the algorithms discussed in Chapter 3 and the primal and the dual algorithms. The method proposed by Charnes et al. (1974) is used to generate instances whose coefficients are uniformly distributed between -10 and 10. All of the algorithms were implemented in Matlab R2013b using CPLEX 12.5 as linear programming solver. The experiments were run on an Intel(R) Core(TM) i7 CPU computer with 16GB RAM and 1TB hard drive. Table 4.2 below shows the average CPU times (in seconds) of solving three instances of the same size for which the underlying (MOLP) has p objectives, m constraints and n variables. We tested six algorithms, namely

- the brute force algorithm (Algorithm 4.1), A1
- the bi-objective branch and bound algorithm of Section 3.2.2, A2
- the conical branch and bound algorithm of Section 3.2.3, A3
- Benson's branch and bound algorithm (Section 3.2.4), A4
- the primal algorithm (Algorithm 4.2), A5
- the dual algorithm (Algorithm 4.3), A6

Table 4.2 CPU times of six different algorithms.

p	$m=n$	A1	A2	A3	A4	A5	A6
	5	0.0427	0.0066	0.0128	0.0343	0.0065	0.0147
	10	0.0065	0.0016	0.0035	0.0063	0.0068	0.0067
2	50	0.0072	0.0038	0.0282	0.0732	0.0038	0.0084
	100	0.0239	0.0069	0.0541	0.1418	0.0197	0.0163
	500	0.3226	0.1634	2.5935	12.5185	0.2464	0.2789
	5	0.0678	-	0.0074	0.0209	0.0161	0.0269
	10	0.0294	-	0.0052	0.0077	0.0114	0.0178
3	50	0.0847	-	0.1005	0.2327	0.0402	0.0386
	100	0.1385	-	0.302	0.9642	0.0985	0.0596
	500	3.4191	-	3.9931	17.0181	1.4985	0.4342
	5	0.1373	-	0.0084	0.0232	0.0637	0.0414
	10	0.1951	-	0.1638	0.3575	0.2061	0.0445
4	50	0.5496	-	0.2004	0.066	0.5685	0.1103
	100	2.0857	-	3.6578	4.2433	0.7587	0.4348
	500	35.8634	-	66.1054	141.241	21.0746	9.1878
	5	5.1236	-	0.0934	0.024	0.8902	0.0618
	10	2.6293	-	0.0277	0.0099	2.7356	0.1623
5	50	10.9649	-	7.1391	3.249	3.5458	0.8399
	100	25.9835	-	50.4573	10.6344	6.4632	2.9714
	500	204.7300	-	354.8034	578.1003	89.6327	53.5104

Clearly, as the size of the instances grows, the CPU time increases rapidly. The crucial parameter here is the number of objective functions. While with $p = 2$ objectives, even problems with 500 variables and constraints can be solved in less than a second, this takes between less than 1 minutes and about 10 minutes with $p = 5$ objectives for the different

algorithms. We observe that for $p = 2$, the bi-objective branch and bound algorithm turns out to be the fastest algorithm, but it cannot be generalised to problems with more than two objectives. As p increases, the merit of the primal and the dual algorithms is revealed. Specifically, when solving problems with 5 objectives and 500 variables and constraints, the primal and the dual algorithms take much less time (one sixth, respectively one tenth) than Benson's branch and bound algorithm. Table 4.2 also shows the dual method performs better than the primal method in solving instances of large scale. In Figure 4.5, we plot the log-transformed CPU times of solving the instances with 500 variables and constraints for the five different applicable algorithms. It shows that, as expected, the time required to achieve optimality exponentially increases with the number of objectives, even for our primal and dual algorithm, making the speed-up obtained by our algorithms even more important.

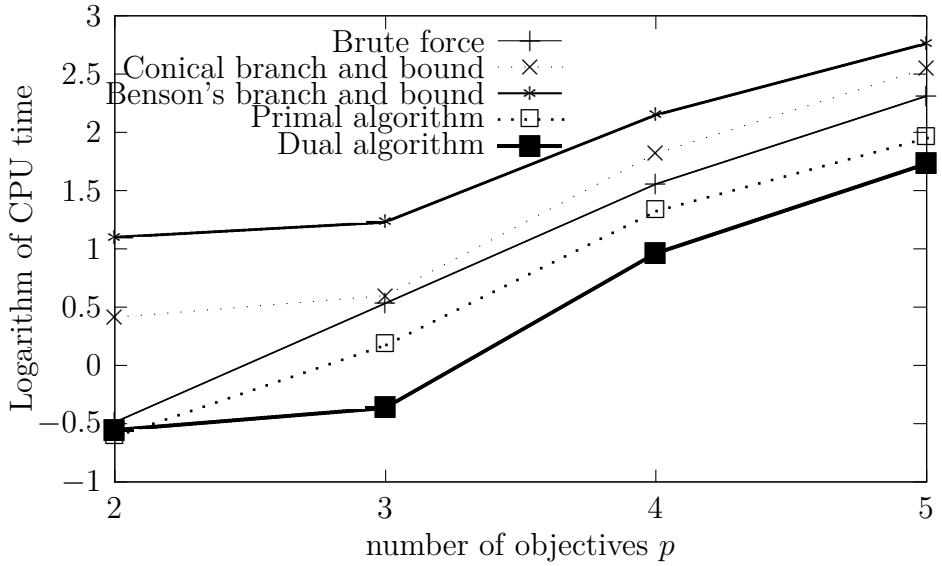


Figure 4.5 Log-transformed CPU times for instances with 500 variables and constraints.

4.4 Conclusion

Optimisation over the non-dominated set is a problem of concern when decision makers have to choose a preferred point among an infinite number of non-dominated points. We have addressed the case that this selection is based on the optimisation of a linear function. We have exploited primal and dual variants of Benson's algorithm, which compute all non-dominated extreme points and facets of multi-objective linear programmes, as the basis of algorithms to solve this problem. In addition, we have described structural properties of the problem of optimising a linear function over the non-dominated set of (MOLP) to dramatically reduce the need for complete enumeration of all non-dominated extreme points of (MOLP). We have compared our algorithms to several algorithms from the literature, and the complete enumeration approach, and obtained speed-ups of up to 10 times on instances with up to 5 objectives and 500 variables and constraints.

Chapter 5

The Determination of the Nadir Point

5.1 The nadir point of (MOP)

The nadir point is characterized by the componentwise worst values of the non-dominated points of a multi-objective optimisation problem (MOP). In this chapter, we propose two exact methods to find the nadir point of (MOLP). Computational experiments were performed to test the new methods against another exact method from the literature.

The task of the determination of the nadir point is of significant importance. Firstly the knowledge of the nadir point facilitates the normalisation of the objectives with inconsistent magnitude (Miettinen (1999)). Secondly the nadir point is a pre-requisite for some interactive methods such as the STEM method (Benayoun et al. (1971)) and the GUESS method (Buchanan (1997)). In compromise programming, it serves as a reference point (Ehrgott and Tenfelde-Podehl (2003)). Additionally, another motivation is proposed by Deb et al. (2010) that the nadir point is crucial in terms of visualising the non-dominated set.

Definition 5.1. *The* ***nadir point*** $y^N \in \mathbb{R}^p$ *of (MOP) is defined to be the vector of componentwise maxima of* $\mathcal{Y}_N$*.*

We consider the problem

$$y_k^N = \max\{y_k : y \in \mathcal{Y}_N\}, \qquad k = 1, ..., p, \tag{N}$$

where y_k^N denotes the k^{th} component of y^N, and $\mathcal{Y}_N$ is the non-dominated set of (MOLP).

Figure 5.1 below shows the objective polyhedron $\mathcal{Y}$ of Example 2.1 along with the ideal point y^I, the nadir point y^N and the anti-ideal point y^{AI}.

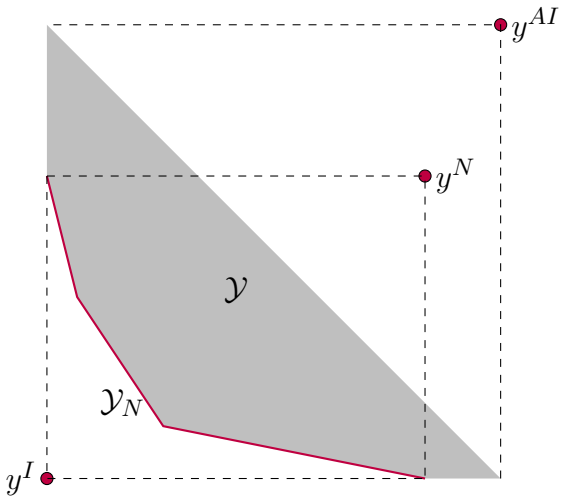


Figure 5.1 Ideal point, nadir point and anti-ideal point.

The nadir point contains the maximal objective values attained from the non-dominated set of (MOP), which is a nonconvex set and therefore burdensome to compute even for (MOLP) (Ehrgott and Tenfelde-Podehl (2003)). On the other hand, if the entire non-dominated set is found, it is not motivated to determine the nadir point since decisions can then be made based on the full knowledge of the non-dominated set. Hence, it is intuitive to make an estimate of the location of the nadir point.

Benayoun et al. (1971) established a payoff table method to approximate the nadir point. This method constructs a table consisting of the vectors optimising the objectives individually, and an estimation is made through the worst values of each of the

criteria. Assume $x^k \in \text{argmin}\{f_k(x) : x \in \mathcal{X}\}$, $k = 1 \dots p$. The estimated nadir point $\tilde{y}_i^N := \max\{f_i(x^k), k = 1 \dots p\}$, $i = 1 \dots p$. A payoff table of Example 2.1 is shown below.

Table 5.1 Payoff table of Example 2.1

	x	$f_1(x)$	$f_2(x)$
$x^1 \in \text{argmin}\{f_1(x) : x \in \mathcal{X}\}$	$(0,4)^T$	0	4
$x^2 \in \text{argmin}\{f_2(x) : x \in \mathcal{X}\}$	$(5,0)^T$	5	0
$\tilde{y}^N$		5	4

Example 5.1. In this example, the payoff table method finds the true nadir point $(5,4)^T$. Nevertheless, Isermann and Steuer (1988) illustrated the possible considerable discrepancy between the estimated point derived from the payoff table and the true nadir point in case x^k is not unique. For example the payyoff table below shows an overestimate of the true nadir point of Example 2.1.

Table 5.2 Payoff table of Example 2.1

	x	$f_1(x)$	$f_2(x)$
$x^1 \in \text{argmin}\{f_1(x) : x \in \mathcal{X}\}$	$(0,5)^T$	0	5
$x^2 \in \text{argmin}\{f_2(x) : x \in \mathcal{X}\}$	$(6,0)^T$	6	0
$\tilde{y}^N$		6	5

Ehrgott (2005) (Section 2.2) shows an example, in which the nadir values derived from the payoff table may over-estimate or under-estimate the true nadir values and the estimated nadir point can be arbitrarily far from the true nadir point.

More sophisticated heuristic methods have been developed to estimate the nadir point. One class of these procedures integrates evolutionary approaches. A review on the nadir point estimation by evolutionary approaches was made by Deb and Miettinen (2008). However, heuristics provide no guarantee for finding the true nadir point.

Ehrgott and Tenfelde-Podehl (2003) propose a general scheme to compute the nadir point based on some theoretical results on the relationship between the non-dominated sets of (MOP) and that of subproblems with fewer objectives. Theoretically, this method finds

the true nadir point of (MOP) with any number p of objectives. However, p subproblems of $p-1$ objectives have to be solved, which is computationally impracticable for p greater than 3.

To the best of our knowledge, the only exact method for computing the nadir point from the literature is proposed by Alves and Costa (2009). This method starts with the payoff table estimation. For each objective function, it searches for efficient solutions that deteriorate the objective values. This is achieved by investigating the so-called nadir region (the area in the weight space corresponding to the efficient solutions that provide worse objective values). The nadir value of a specific objective is determined if there exists no such nadir region.

For each objective function $f_k(x)$ $(k = 1, \ldots, p)$, the method checks if a value z is the nadir value of $f_k(x)$. If not, it searches the weight space defined by the set $\Lambda = \left\{ \lambda \in \mathbb{R}^p : \lambda_k > 0, k = 1, \ldots, p, \sum_{k=1}^{p} = 1 \right\}$ for efficient solutions that lead to larger nadir values.

Assuming z_k is the maximum of the k^{th} objective function already known, an auxiliary problem is solved:

$$\min \{c^k x : x \in \mathcal{X}, c^k x \geqq z_k + \delta\}. \qquad (Aux(k))$$

All of the alternative optimal bases to $Aux(k)$ are to be computed, of which only the efficient bases are considered. Then feasible pivot operations are performed on the efficient bases that leads to other efficient bases with greater objective values. Assume that s is the variable associated with the extra constraints of $Aux(k)$. If s is also an efficient nonbasic variable, allowing s to enter the basis leads to anther efficient basis with larger objective value. The reader is referred to Alves and Costa (2009) for more details.

5.2 The determination the nadir point by the primal method and the dual method

In this section we adapt the primal method (Algorithm 4.2) and the dual method (Algorithm 4.3) proposed in Chapter 4 to determine the exact nadir point of (MOLP).

Recall that (P) is

$$\max \{\mu^T y : y \in \mathcal{P}_N\}, \tag{P}$$

where μ is a column vector with p elements, and $\mathcal{P}_N$ is the non-dominated set of the extended feasible set of (MOLP). If $\mu = e^k$ (a unit vector with the k^{th} entry being one, others zero), (N) can be rewritten as

$$y_k^N = \max \{e^{kT} y : y \in \mathcal{P}_N\}, \quad k = 1, \ldots, p. \tag{N}$$

Obviously, we can solve (N) through solving p individual (P) with $\mu = e^k, k = 1, \ldots, p$. On the other hand, (P) can be solved by calculating the nadir values of (MOP) which is constructed by adding $\Phi(x)$ to the underlying (MOP) of (P) as a dummy objective function. To show this, we write (P) as

$$\max \{\mu^T Cx : x \in \mathcal{X}_E^{(\mathrm{MOP})}\},$$

where $\mathcal{X}_E^{(\mathrm{MOP})}$ denotes the efficient set of the following (MOP)

$$\min \{Cx : x \in \mathcal{X}\} \tag{MOP}$$

By adding $\mu^T Cx$ to (MOP) as an objective, we have (MOP$'$)

$$\min \left\{ \begin{pmatrix} Cx \\ \mu^T Cx \end{pmatrix} : x \in \mathcal{X} \right\}. \tag{MOP$'$}$$

Let $\mathcal{X}_E^{(\mathrm{MOP}')}$ denote the efficient set of (MOP′), and let y_{p+1}^N denote the $(p+1)^{th}$ element of the nadir point of (MOP′). Adding an extra dimension to the objective space of (MOP) may extend the efficient set, i.e., $\mathcal{X}_E^{(\mathrm{MOP})} \subseteq \mathcal{X}_E^{(\mathrm{MOP}')}$, therefore,

$$\begin{aligned} & \max\{\mu^T Cx : x \in \mathcal{X}_E^{(\mathrm{MOP})}\} \\ \leqq\ & \max\{\mu^T Cx : x \in \mathcal{X}_E^{(\mathrm{MOP}')}\} \\ =\ & y_{p+1}^N. \end{aligned}$$

We assume

$$\max\{\mu^T Cx : x \in \mathcal{X}_E^{(\mathrm{MOP})}\} < y_{p+1}^N.$$

Therefore, there exists $\bar{x} \in \mathcal{X}_E^{(\mathrm{MOP}')} \backslash \mathcal{X}_E^{(\mathrm{MOP})}$ such that

$$\mu^T C\bar{x} > \max\{\mu^T Cx : x \in \mathcal{X}_E^{(\mathrm{MOP})}\}. \tag{5.1}$$

Since $\bar{x} \notin \mathcal{X}_E^{(\mathrm{MOP})}$, there exists $x' \in \mathcal{X}_E^{(\mathrm{MOP})}$ such that $Cx' \leq C\bar{x}$. By (5.1) we have $\mu^T C\bar{x} > \mu^T Cx'$. Hence,

$$\begin{pmatrix} Cx' \\ \mu^T Cx' \end{pmatrix} \leq \begin{pmatrix} C\bar{x} \\ \mu^T C\bar{x} \end{pmatrix}.$$

This contradicts $\bar{x} \in \mathcal{X}_E^{(\mathrm{MOP}')}$. Therefore,

$$y_{p+1}^N = \max\{\mu^T Cx : x \in \mathcal{X}_E^{(\mathrm{MOP})}\}.$$

In this section a modified version of Algorithm 4.2 is designed to solve (N) more efficiently than solving (P) by the primal method (Algorithm 4.2) p times independently.

Remark 1. For (MOP), if $y \in \mathcal{Y}_N$, then $y_k \leqq y_k^N$, $k = 1, \ldots, p$.

Remark 1 says that any non-dominated point provides lower bounds to the nadir values of all of the p objective functions. Throughout the primal algorithm, non-dominated points are attained, each of which provides lower bounds for the p nadir values. The best lower

bounds for all p objective functions are kept. Through the lower bounds threshold cuts can be constructed to eliminate regions of $\mathcal{P}_N$ which are not worthy of further exploration. We employ the payoff table method in the beginning of the algorithm to obtain non-dominated points (therefore lower bounds of the nadir values), based on which threshold cuts are constructed. However, a numerical example illustrated by Ehrgott (2005)(section 2.2) shows that the nadir values can be over-estimated or under-estimated if the solutions to $\{f_k(x) : x \in \mathcal{X}\}$ are not unique. This is because the (LP) may have multiple optimal solutions, which are not necessarily all efficient. If weakly efficient (but not efficient) solutions are attained, weakly non-dominated (but not non-dominated) points derived from these solutions may cause over-estimation of the nadir value. In this case, the weakly non-dominated points cannot be used to derive lower bounds of the nadir values. To amend this we add two steps in the beginning of Algorithm 4.2. It is well known that $\hat{x}^k \in \text{argmin}\{c^k x : x \in \mathcal{X}\}$ is weakly efficient. By solving $\min\{\sum_{i \neq k} c^i x : x \in \mathcal{X}, c^k x = c^k \hat{x}\}$, an efficient solution can be guaranteed. These steps are listed in Line 3 and Line 4 of Algorithm 5.1 below.

Algorithm 5.1 Primal method for the nadir point

Input: (N).

Output: y^N.

1: $N := \emptyset$.

2: **for** $k = 1 \dots p$ **do**

3: Compute $\hat{x}^k \in \operatorname{argmin}\{c^k x : x \in \mathcal{X}\}$.

4: Compute $x^k \in \operatorname{argmin}\{\sum_{j \neq k} c^j x : x \in \mathcal{X}, c^k x = c^k \hat{x}\}$. $N = N \cup \{Cx^k\}$.

5: Set $\mathcal{S}^0 := \{y^I\} + \mathbb{R}^p_{\geqq}$.

6: $y_k := \max\{y_k : y \in N\}$.

7: $\mathcal{S}^1 := \mathcal{S}^0 \cap \{y \in \mathbb{R}^p : e^{kT} y \geqq y_k\}$. Threshold = True. $i = 2$.

8: **while** $V_{\mathcal{S}^{k-1}} \not\subset \mathcal{P}$ **do**

9: $s^i := \max\{y_k : y \in N \cup V_{\mathcal{S}^{i-1}}\}$.

10: **if** $s^i \in \mathcal{P}$ and if Threshold = False **then**

11: $\mathcal{S}^i := \mathcal{S}^{i-1} \cap \{y \in \mathbb{R}^p : e^{kT} y \geqq s^i_k\}$. Threshold = True.

12: **else**

13: Compute an optimal solution (x^i, z^i) to $P_2(s^i)$ and its dual variable values (u^i, λ^i). $N = N \cup \{Cx^i\}$.

14: Set $\mathcal{S}^i := \mathcal{S}^{i-1} \cap \{y \in \mathbb{R}^p : \varphi(y, (\lambda^i_1, \dots, \lambda^i_{p-1}, b^T u^i)) \geqq 0\}$.

15: **end if**

16: $N = N \cup (V_{\mathcal{S}^i} \cap \mathcal{P})$.

17: Set $i := i + 1$.

18: **end while**

19: $y^N_k = s^i_k$.

20: **end for**

In Line 1 of Algorithm 5.1, we create a set N to store non-dominated points discovered throughout the algorithm. In Line 13, an optimal solution to $P_2(s^i)$ provides a weakly non-dominated point Cx^i. Another source of non-dominated points is the vertex set $V_{\mathcal{S}^i}$

of $\mathcal{S}^i$. This set is updated when there is a cut added in Line 11 or Line 14 (φ is defined in (2.3)). If a new vertex is a non-dominated point, it is added to N in Line 16. Throughout the algorithm the set N collects the non-dominated points found. In this process our knowledge of $\mathcal{P}_N$ accumulates. Therefore, the determination of one nadir value benefits the subsequent ones because at the start of the exploration of a nadir value a threshold cut is generated based on the highest lower bound on that nadir value. As a result, a substantial part of $\mathcal{P}_N$ is removed by the cut. Therefore, it is obvious that Algorithm 5.1 is more efficient than solving p problems (P) individually. Eventually, this algorithm terminates when all of the nadir values are determined.

In Chapter 4 we introduced the dual method (Algorithm 4.3) to solve (P). We now modify Algorithm 4.3 to solve (N). Recall that this algorithm enumerates all of the incomplete facets (therefore all of incomplete vertices in primal objective space), which contains all the information needed to compute each component of y^N. Therefore, it is sufficient to perform the dual method once to determine y^N. Throughout the algorithm non-dominated points y^k are determined, some of which are incomplete vertices. We revise Algorithm 4.3 by storing all y^k in a set Y, and propose Algorithm 5.2 to determine the nadir point.

Algorithm 5.2 Dual method for nadir point

Input: (N).

Output: y^N.

1: Perform Algorithm 4.3 to obtain all of the incomplete vertices and store them in Y.

2: **for** $k = 1, \ldots, p$ **do**

3: $\quad y_k^N = \max\{e^{kT} y : y \in Y\}$.

4: **end for**

5.3 Computational experiments

In this section, we use the instances from Alves and Costa (2009) to compare the primal and the dual method against Alves and Costa's method discussed in Section 5.1. All of the algorithms were implemented in Matlab R2013b using CPLEX 12.5 as the linear programming solver. The experiments were run on an Intel(R) Core(TM) i7 CPU computer with 16GB RAM and 1TB hard drive. Table 5.3 below shows the average CPU times (in seconds) to solve five instances of the same size for which the underlying (MOLP) has p objectives, m constraints and n variables. Table 5.3 shows the CPU times of the three algorithms solving instances of different sizes.

Table 5.3 CPU times of three different algorithms.

p	n,m	Alves and Costa's method	Primal algorithm	Dual algorithm
	60,30	1.6333	0.4595	0.3825
3	80,40	4.6131	0.6755	0.5967
	100,50	8.4728	1.3887	0.9956
	60,30	53.8227	54.1272	43.5604
4	80,40	3072.7837	595.6945	52.7250
	100,50	5176.4915	1275.2579	498.6906
	60,30	787.7661	187.9240	104.5201
5	80,40	9456.3452	2781.8934	563.0983
	100,50	96034.3420	48720.9043	14893.3421

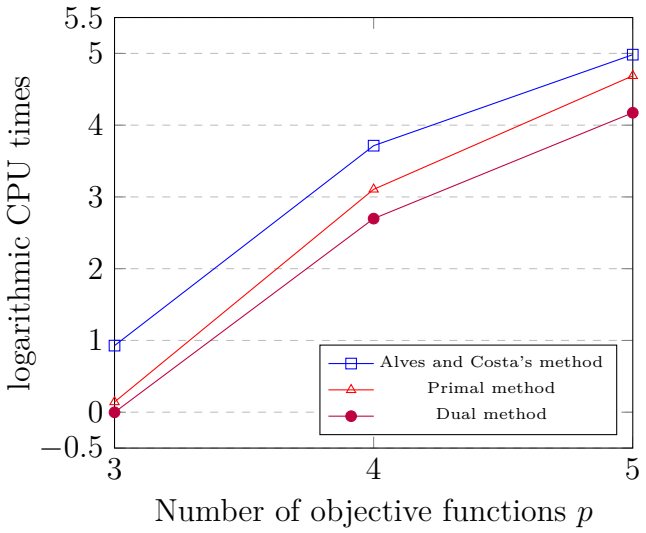


Figure 5.2 Log-transformed CPU times for instances with 100 variables and 50 constraints

Figure 5.2 shows the logarithmic CPU times for instances with 100 variables and 50 constraints. As the size of the instances grows, the CPU time increases rapidly. The crucial parameter here is the number of objective functions. With $p = 3$ objectives, even problems with 100 variables and 50 constraints can be solved in less than a second. As p increases, the merit of the primal and the dual algorithms is revealed. Specifically, when solving problems with 4 objectives and 100 variables and 50 constraints, the primal and the dual algorithms take much less time than Alves and Costa's method. Table 5.3 also shows that the dual method performs better than the primal method in solving instances of large scale.

5.4 Conclusions

The nadir point is characterized by the componentwise maximal values of the non-dominated points of (MOP). The determination of the nadir point is of significant importance because of its indispensable role as a prerequisite for many optimisation problems. However it is a challenging task to obtain the exact nadir values. It is obvious that this problem is a special case of optimising a linear function over the non-dominated

set of (MOP), which is achievable by the two methods, namely, the primal method and the dual method, proposed in Chapter 4. In this chapter, we apply these two methods to compute the exact nadir values. We have compared our methods against the only other exact method from the literature. It is evident that the new methods outperform this method.

Chapter 6

Optimisation over the Non-dominated Set of a Convex Multi-objective Optimisation Problem

In Chapter 2 the convex multi-objective optimisation problem (CMOP) is defined as

$$\min \{f(x) : x \in \mathbb{R}^n, g(x) \leqq 0\}, \tag{CMOP}$$

where $f(x) = (f_1(x), \ldots, f_p(x))^T$, and $g(x) = (g_1(x), \ldots, g_m(x))^T$. If $f(x)$ and $g(x)$ are convex, i.e., the objectives and the constraints are all convex functions, the problem is a convex multi-objective optimisation problem (CMOP). Example 6.1 is a numerical example of (CMOP). Figure 6.1 shows the feasible set in objective space and the non-dominated set of this example.

Example 6.1.

$$\min \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$
$$\text{s.t. } \frac{(x_1-3)^2}{9} + \frac{(x_2-2)^2}{4} \leqq 1.$$

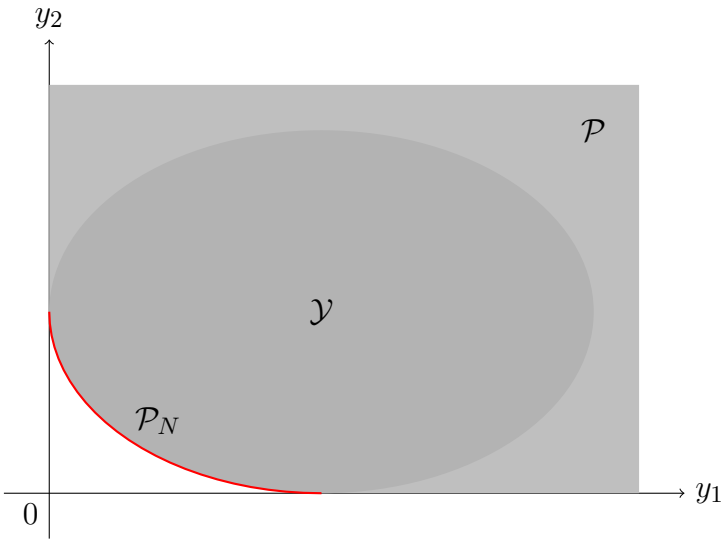


Figure 6.1 The non-dominated set of Example 6.1.

In this chapter we are concerned with an optimisation problem which maximises a linear function over the non-dominated set of (CMOP). To solve this problem we extend the primal algorithm (Algorithm 4.2) and the dual algorithm (Algorithm 4.3), which are designed to optimise a linear function over the non-dominated set of (MOLP) (see Chapter 4 for more details of the methods). In this chapter we firstly introduce the problem of interest, and then we propose two methods to solve this problem. The two new methods are tested against comparable methods from the literature on a set of randomly generated instances. The results are presented and discussed in the end.

6.1 Optimisation over the non-dominated set of (CMOP)

Consider the following problem

$$\max \{\mu^T y : y \in \mathcal{P}_{\epsilon N}\}, \tag{Q}$$

where $\mathcal{P}_{\epsilon N}$ is the ϵ-non-dominated set of the extended feasible set $\mathcal{P}$ ($\mathcal{P} := \mathcal{Y} + \mathbb{R}^p_{\leqq}$) for $\epsilon \in \mathbb{R}^p_{\geq}$. The concept of ϵ-non-dominance is defined in Definition 2.4. Column vector μ contains the coefficients of the linear function to be maximised. Example 6.2 below is a numerical example of (Q).

Example 6.2. A linear function $5y_1 + 7y_2$ is to be maximised over the non-dominated set of the (CMOP) in Example 6.1:

$$\max \{5y_1 + 7y_2 : y \in \mathcal{P}_{\epsilon N}\},$$

where $\mathcal{P}_{\epsilon N}$ is the ϵ-non-dominated set of Example 6.1.

6.2 Primal method to solve (Q)

In Chapter 4, we investigated some properties of (P) to facilitate solving the problem. In this section, we show that some properties of (P) still hold for (Q). Then we present a primal method to solve (Q).

Through Theorem 4.1 we know that an optimal solution to (P) is obtained at an extreme point of $\mathcal{P}$ of (MOLP). However, for problem (Q) the underlying (MOP) has a non-polyhedral feasible set $\mathcal{P}$ which may have no extreme point at all (see for example Example 6.1). Another way to view this case is that every point of $\mathcal{P}_N$ is an extreme point.

The outer approximation algorithm for (CMOP) (Algorithm 2.3) approximates this non-polyhedral $\mathcal{P}$ through a polyhedron $\mathcal{S}^i$, which is refined iteratively, i.e., $\mathcal{S}^0 \supseteq \mathcal{S}^1 \supseteq$

$\mathcal{S}^2 \ldots \mathcal{S}^i \supset \mathcal{P}$. This algorithm terminates if all of the vertices of $\mathcal{S}^i$ are ϵ-non-dominated. Without loss of generality assume that Algorithm 2.3 terminates after n iterations. Thus all of the vertices of $\mathcal{S}^n$ are ϵ-non-dominated, i.e., $V_{\mathcal{S}^n} \subset \mathcal{P}_{\epsilon N}$. Consider the following problem

$$\max \{\mu^T y : y \in V_{\mathcal{S}^n}\}. \tag{Q1}$$

Since the vertices of $\mathcal{S}^n$ are ϵ-non-dominated points, solving (Q1) provides a ϵ-non-dominated point that optimise the objective function of (Q). An optimal solution to (Q1) can be obtained at a vertex of $\mathcal{S}^n$. This implies that a naïve algorithm for solving (Q) is to obtain a set of ϵ-non-dominated points of $\mathcal{P}$ through Algorithm 2.3 and determine which one has the largest value of $\mu^T y$. This algorithm is summarised in Algorithm 6.1.

Algorithm 6.1 Brute force algorithm

Input: (Q), ϵ
Output: y^* (an optimal solution to (Q))
Phase 1: Obtain $Y_{\epsilon N}$ (a set of ϵ-non-dominated points) through Algorithm 2.3.
Phase 2: $y^* \in \text{argmax} \{\mu^T y : y \in Y_{\epsilon N}\}$.

In Chapter 4 we introduced the concept of completeness of a vertex of the feasible polyhedron $\mathcal{Y}$ of (MOLP). Theorem 4.3 concludes that an optimal solution to (P) is an incomplete vertex of $\mathcal{Y}$. Therefore (Q1) is equivalent to (Q2).

$$\max \{\mu^T y : y \in V^{ic}_{\mathcal{S}^n}\}, \tag{Q2}$$

where $V^{ic}_{\mathcal{S}^n}$ is the set of incomplete vertices of $\mathcal{S}^n$. This mean that it is sufficient to find the incomplete vertices of $\mathcal{S}^n$ to solve (Q). However, due to the complex structure of $\mathcal{Y}$, it is difficult to design an algorithm to confine the search to the set of incomplete vertices. Fortunately the dual method proposed in the following section takes advantage of this property to achieve better efficiency.

The primal method for (Q) is detailed in Algorithm 6.2. In each iteration, a hyperplane is generated and added as a cut resulting in a set of extreme points. Evaluating $\mu^T y$

at these points, we select the one with the best function value to construct the cut for the next iteration. If the selected point is an ϵ-non-dominated point, a threshold cut is added; otherwise an improvement cut ($\{y \in \mathbb{R}^p : \varphi(y, (\lambda_1^k, \ldots, \lambda_{p-1}^k, \lambda^{kT} y^k)) \geqq 0\}$) used in the revised version of Benson's algorithm is added, which leads to better objective function value in the next iteration. A threshold cut is $\{y \in \mathbb{R}^p : \mu^T y \geqq \mu^T \hat{y}\}$, where $\hat{y}$ is the incumbent solution. A threshold cut removes the region where points are worse than $\hat{y}$. This primal method is detailed in Algorithm 6.2.

Algorithm 6.2 Primal method for (Q)

Input: (Q), ϵ

Output: y^* (an optimal solution to (Q))

Compute an optimal solution $\bar{u}^i$ and an optimal value y_i^I of $(\mathbb{D}_1(e^i))$, for $i = 1, \ldots, p$.
Set $\mathcal{S}^0 := \{y^I\} + \mathbb{R}^p_{\geqq}$ and $k = 1$.
Threshold = False.
while $V_{\mathcal{S}^{k-1}} \cap \mathcal{P}_{\epsilon N} \neq \emptyset$ **do**
    $s^k \in \operatorname{argmax}\{\mu^T y : y \in V_{\mathcal{S}^{k-1}}\}$.
    if $s^k \in \mathcal{P}_{\epsilon N}$ and if Threshold = False **then**
        $\mathcal{S}^k := \mathcal{S}^{k-1} \cap \{y \in \mathbb{R}^p : \mu^T y \geqq \mu^T s^k\}$. Threshold = True.
    else
        Compute an optimal solution (u^k, λ^k) of $\mathbb{D}_2(y^k)$.
        Set $\mathcal{S}^k := \mathcal{S}^{k-1} \cap \{y \in \mathbb{R}^p : \varphi(y, (\lambda_1^k, \ldots, \lambda_{p-1}^k, \lambda^{kT} y^k)) \geqq 0\}$.
    end if
    Set $k := k + 1$.
end while

We illustrate Algorithm 6.2 through solving Example 6.2.

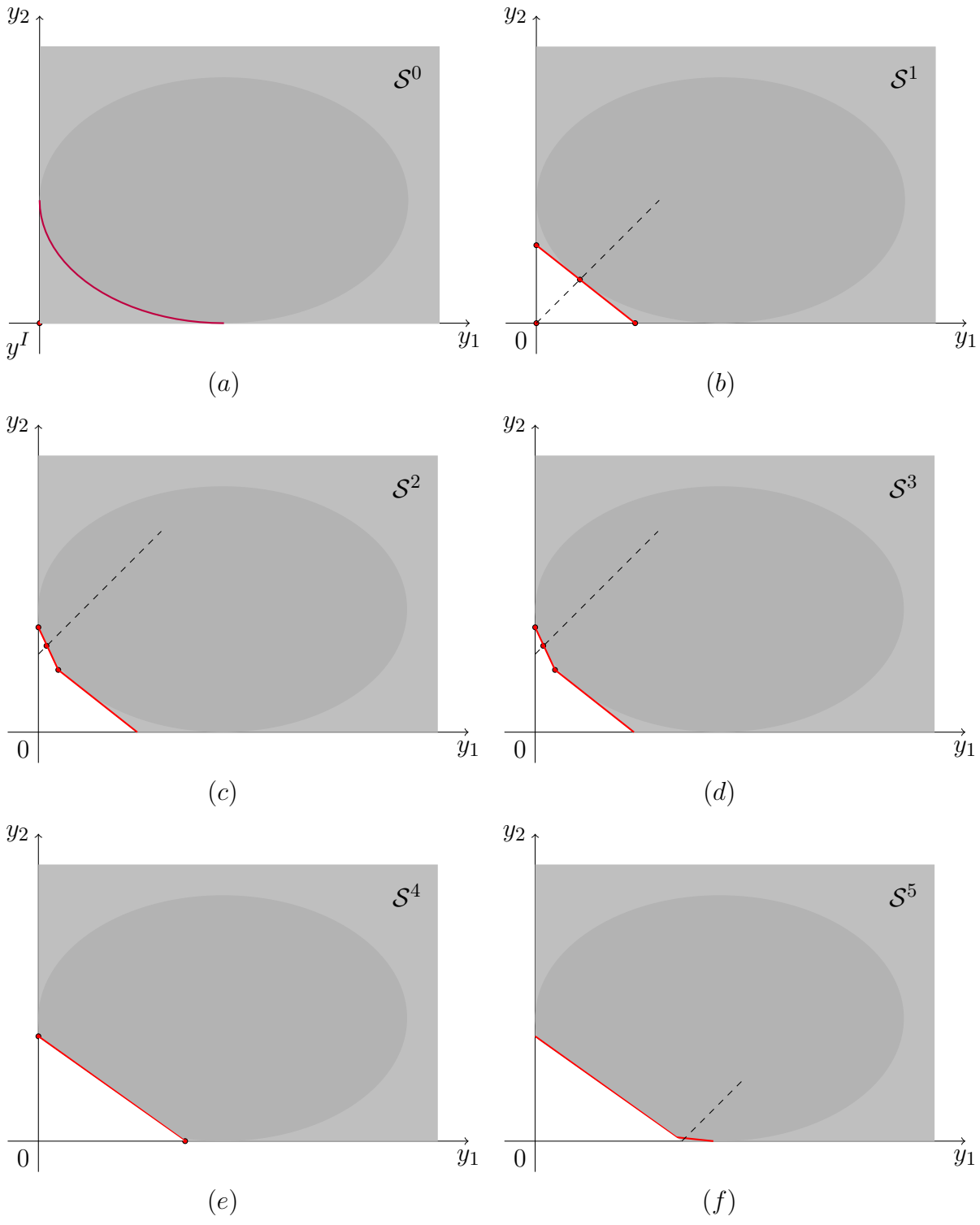


Figure 6.2 Iterations of Algorithm 6.2 in Example 6.2.

Table 6.1 Iterations of Algorithm 6.2 in Example 6.2.

Iteration k	Vertex s^k	Cut type	Candidates	ϵ-non-dominated points	$\mu^T y$
1	(0,0)	Improvement	(0,1.27),(1.61,0)	$\emptyset$	8.88
2	(0,1.27)	Improvement	(1.61,0)	(0.32,1.01), (0,1.71)	11.94
3	(0,1.71)	Threshold	(2.39,0)	(0,1.71)	11.94
4	(2.39,0)	Improvement	$\emptyset$	(2.9,0), (2.3,0.06)	14.5

Example 6.3. Figure 6.2 and Table 6.1 show the iterations of Algorithm 6.2 in solving Example 6.2. In the first iteration, an improvement cut is added generating two new vertices, $(0, 1.27)^T$ and $(1.61, 0)^T$. Then $(0, 1.27)^T$ is chosen in the next iteration because it provides the best objective function value so far. The second iteration improves the function value to 11.94. Since $(0, 1.71)^T$ is ϵ-non-dominated, a threshold cut, $5y_1 + 7y_2 \geqq 11.94$, is then added. Although it does not improve the objective function value, this cut generates a new infeasible vertex $(2.39, 0)^T$. An ϵ-non-dominated point, $(2.9, 0)^T$, is found after another improvement cut has been generated. This point is an approximately optimal solution to Example 6.2.

6.3 Dual method to solve (Q)

In Chapter 2 the geometric dual of (CMOP) is defined as

$$\max_{\mathcal{K}} \{D(v) : v \in \mathbb{R}^p, \lambda(v) \geq 0\}, \tag{DCMOP}$$

where $D(v) = \left\{v_1, \ldots, v_{p-1}, \min_{x \in \mathcal{X}} \left[\lambda(v)^T f(x)\right]\right\}$. Let $\mathcal{K} := \{v \in \mathbb{R}^p : v_1 = v_2 = \cdots = v_{p-1} = 0, v_p \geqq 0\}$ denote the ordering cone in the dual objective space. Maximisation in (DCMOP) is with respect to the order defined by $\mathcal{K}$. Let $\mathcal{V}$ denote the feasible set in the dual objective space, then the extended set of $\mathcal{V}$ in the dual objective space is $\mathcal{D} := \mathcal{V} - \mathcal{K} = \{v - \hat{v} : v \in \mathcal{V}, \hat{v} \in \mathcal{K}\}$. For (DCMOP) it is of interest to determine the

$\mathcal{K}$-maximal set, which is

$$\mathcal{D}_{\mathcal{K}} = \max_{\mathcal{K}}\{(\lambda_1, ..., \lambda_{p-1}, \min_{x\in\mathcal{X}} \left[\lambda(v)^T f(x)\right])^T : (u, \lambda) \geqq 0, e^T\lambda = 1\}.$$

Figure 6.3 shows the extended feasible set and $\mathcal{K}$-maximal set of Example 6.1 in dual objective space.

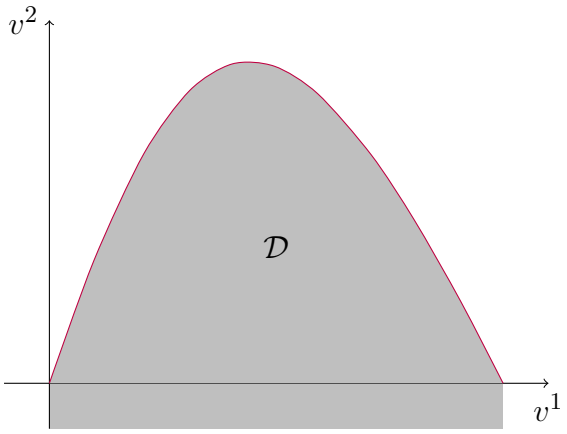


Figure 6.3 Extended feasible set and $\mathcal{K}$-maximal set of Example 6.1 in dual objective space.

In practice it is often sufficient to have an approximation of the $\mathcal{K}$-maximal set. This is modelled by the concept of $\epsilon\mathcal{K}$-maximum, which is defined in Definition 2.5. A point v is called an $\epsilon\mathcal{K}$-maximal point if $v - \epsilon \in \mathcal{D}$ and there does not exist any $\hat{v} \in \mathcal{D}$ such that $\hat{v}_j = v_j$ for $j = 1, \ldots, p-1$ and $\hat{v}_p > v_p$.

In this section, a dual algorithm for solving (Q) is presented. In Section 4.2, we investigate the properties of (P) in the dual objective space and propose a dual algorithm to solve (P). The dual algorithm (Algorithm 4.3) for (P) determines incomplete $\mathcal{K}$-maximal facets of $\mathcal{D}$. However the underlying (DCMOP) of (Q) has non-polyhedral $\mathcal{D}$, which means it may have no facets (see Figure 6.3 for example).

Fortunately the dual variant of Benson's outer approximation algorithm for (CMOP) (Algorithm 2.4) performs an outer approximation to $\mathcal{D}$. This is done by iteratively refining a polyhedron that contains $\mathcal{D}$, i.e., $\mathcal{S}^0 \supseteq \mathcal{S}^1 \supseteq \mathcal{S}^2 \ldots \mathcal{S}^i \supset \mathcal{D}$. This algorithm terminates if

all of the vertices of $\mathcal{S}^i$ are $\epsilon\mathcal{K}$-maximal. Without loss of generality assume Algorithm 2.4 terminates after n iterations. Thus all of the vertices of $\mathcal{S}^n$ are $\epsilon\mathcal{K}$-maximal, i.e., $V_{\mathcal{S}^n} \subset \mathcal{D}_{\epsilon\mathcal{K}}$. According to the geometric duality theory discussed in Chapter 2 the facets of $\mathcal{S}^n$ correspond to the vertices of the dual of $\mathcal{S}^n$, i.e., the corresponding polyhedron in primal space. Furthermore in the dual objective space the set of incomplete facets of $\mathcal{S}^n$ corresponds to the set of incomplete vertices of the dual polyhedron of $\mathcal{S}^n$ (Theorem 4.5). Theorem 4.6 states that the facet corresponding to an optimal vertex to (P) is an incomplete facet. Hence the set of incomplete facets of $\mathcal{S}^n$ is of interest. The dual algorithm for (Q) is designed to generate the set of incomplete facets.

Algorithm 6.3 Dual algorithm for (Q)

Input: (Q), ϵ

Output: y^* (an optimal solution to (Q))

1: Choose some $\hat{d} \in \text{int}\mathcal{D}$.
2: Compute an optimal solution x^0 of $(\mathbb{P}_1(\hat{d}))$, set $y^* := f(x^0)$, $M^* := \mu^T y^*$.
3: Set $S^0 := \{v \in \mathbb{R}^p : \lambda(v) \geqq 0, \varphi(y^*, v) \geqq 0\}$ *and* $k = 1$.
4: **while** $W_D \cap V_{\mathcal{S}^{k-1}} \not\subset \mathcal{D}_{\epsilon\mathcal{K}}$ **do**
5: Choose $v \in W_D \cap V_{\mathcal{S}^{k-1}}$ such that $v \notin \mathcal{D}_{\epsilon\mathcal{K}}$.
6: Compute $\alpha^k \in (0,1)$ such that $v^k := \alpha^k s^k + (1-\alpha^k)\hat{d} \in \mathbb{V}_{\mathcal{K}}$.
7: Compute an optimal solution x^k of $(\mathbb{P}_1(v^k))$, set $y^k := f(x^k)$, $M^k := \mu^T y^k$.
8: **if** $M^* < M^k$ **then**
9: $y^* := y^k$.
10: $M^* := M^k$.
11: **end if**
12: Set $\mathcal{S}^k := \mathcal{S}^{k-1} \cap \{v \in \mathbb{R} : \varphi(y^k, v) \geqq 0\}$. Update $V_{\mathcal{S}}^k$
13: Set $k := k+1$.
14: **end while**

Figure 6.4 below illustrates the dual algorithm in Example 6.2.

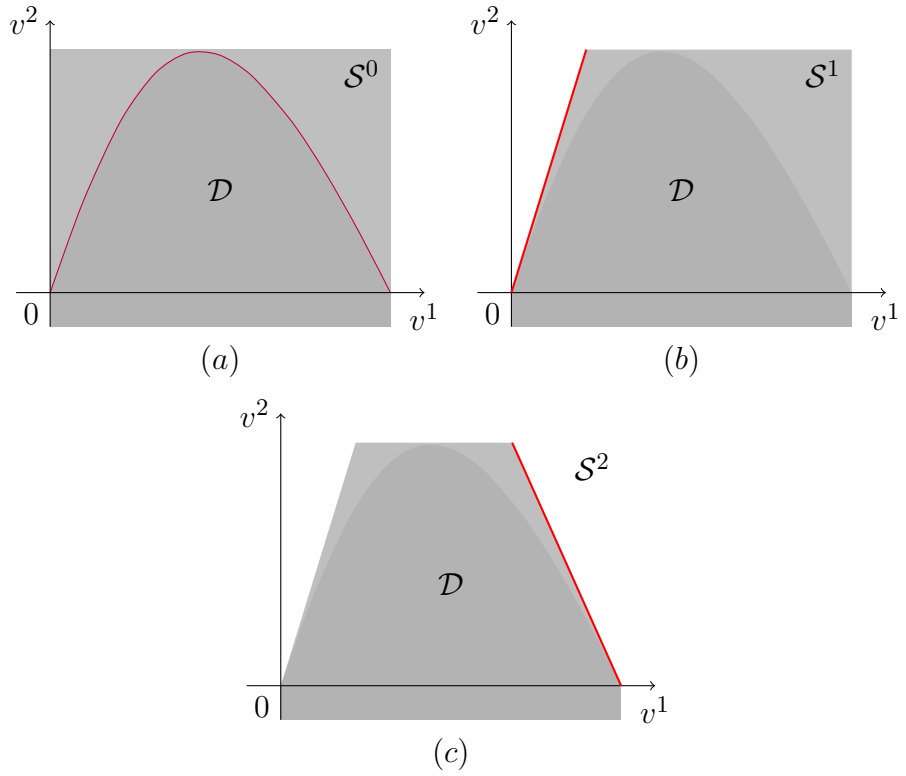


Figure 6.4 Dual algorithm for (Q).

6.4 Experimental results

In this section, we use randomly generated instances to compare some of the algorithms discussed in Chapter 3 and the primal and the dual algorithms for (Q). The method proposed by Charnes et al. (1974) is used to generate convex polyhedra as feasible sets of the underlying (CMOP). Quadratic functions are generated as the objective functions, which are in the form: $f(x) = x^T Hx + a^T x$, where H is a positive semi-definite matrix and a is a column vector. Matrix $H = S^T S$, where S is a square matrix. All coefficients are uniformly distributed between -10 and 10. All of the algorithms were implemented in Matlab R2013b using CPLEX 12.5 as a solver. The experiments were run on an Intel(R) Core(TM) i7 CPU computer with 16GB RAM and 1TB hard drive. Table 6.2 below shows the average CPU times (in seconds) of solving three instances of the same size for which

the underlying (CMOP) has p objectives, m constraints and n variables. We tested five algorithms, namely

- the brute force algorithm (Algorithm 2.3), A1
- the extended bi-objective branch and bound algorithm in Section 3.2.2, A2
- the conical branch and bound algorithm of Section 3.2.3, A3
- the primal algorithm (Algorithm 6.2), A4
- the dual algorithm (Algorithm 6.3), A5

Table 6.2 CPU times of six different algorithms.

p	$m = n$	A1	A2	A3	A4	A5
	5	2.3037	0.0397	0.2541	0.0821	0.1123
	10	4.8086	0.0549	0.4510	0.0869	0.1303
2	50	16.1997	0.4027	1.0291	0.6939	0.4150
	100	28.0444	1.8492	2.8614	2.2391	2.0439
	5	453.2516	-	246.2019	203.5841	160.5440
	10	3821.214	-	2921.345	2061.1652	1631.1805
3	50	17982.2314	-	14669.2419	10621.4219	7243.1480
	100	31987.2540	-	24613.2754	18213.4621	13717.6684

Obviously, the CPU time increases rapidly as the size of the instances grows. The largest size of instances we tested is 3 objectives and 100 variables and constraints due to the substantial amount of time required to solve the instances. We notice that the number of objective functions is a crucial factor. While with $p = 2$ objectives, all of the instances can be solved within 30 seconds. The most efficient algorithm is the extended bi-objective branch and bound algorithm (A2) proposed by Kim and Thang (2013), which solves the largest instances with 100 variables and constraints within 2 seconds. But it cannot be generalised to problems with more than two objectives. The difference in time between A3, A4 and A5 is not significant. We notice that adding one more dimension to the objective space (i.e., adding one more objective function) leads to substantial increase

in computational effort. For instances with 3 objectives, the required CPU time increases predominantly. Even for instances with 5 variables and constraints, it takes a few minutes to solve. Furthermore, for the largest instances with 100 variables and constraints, it takes a few hours. The merits of the primal and the dual methods are revealed. The dual method solves the largest instances in half of the time used by the conical branch and bound method (A3). We also notice that the dual method is faster than the primal method in most of the cases. Throughout the test, the slowest algorithm is the brute force algorithm (A1). This is due to the fact that this method enumerates a large number of vertices which are redundant. This also proves the advantage of the techniques employed in the new methods. Additionally, in the implementation of the algorithms, we notice that time required to solve instances are sensitive to the approximation accuracy (reflected by ϵ as stated in the algorithms). In this test the level of accuracy is 10^{-3}. It is expected that a lower level of accuracy results in faster solution time.

6.5 Conclusion

This section proposed two new methods to maximise a linear function over the non-dominated set of a convex multi-objective optimisation problem. These two methods are based on the primal and the dual methods introduced in Chapter 4. Computational experiments were conducted to compare the new methods against some other methods from the literature. The results reveal the merit of the new methods, especially in solving instances with three objectives.

Chapter 7

Conclusion

In this thesis we are concerned with optimisation over the non-dominated set of a multi-objective optimisation problem (MOP). An MOP is an optimisation problem that simultaneously optimise conflicting objectives. A solution that optimises every objective function concurrently dose not usually exist. A set of solutions that provide the "best" trade-off options of the objectives is of interest. This set is known as the efficient set. In Chapter 2 we review several algorithms to solve two classes of (MOP), namely the multi-objective linear programme (MOLP) and the convex multi-objective optimisation problem (CMOP). The algorithms discussed include a revised version of Benson's outer approximation algorithm. This algorithm is developed to find the non-dominated set of (MOLP), which is essential to the core problem of this study, optimisation over the non-dominated set of (MOP). We call this problem (ON). This problem is of great interest because it models a common situation where a decision maker has to choose one solution from the efficient set for implementation. By solving this problem a non-dominated point is obtained and the efficient solutions that lead to this point can be determined. Most methods for (ON) in the literature employ the branch and bound technique. These methods are discussed in Chapter 3.

In Chapter 4 we propose a primal method and a dual method to solve problem (P), which maximises a linear function over the non-dominated set of (MOLP). In this study

we discover some important properties of (P). Firstly an optimal solution occurs at a vertex of the feasible polyhedron. Hence Benson's outer approximation algorithm which enumerates the non-dominated vertices can be employed to solve (P). Furthermore, we notice that it is unnecessary to determine all of the non-dominated vertices, because an optimal solution to (P) is always obtained at a so-called incomplete vertex of the feasible polyhedron. A vertex is incomplete if not all of its neighbouring vertex are non-dominated. Taking advantage of this property we propose the primal algorithm to solve (P). This algorithm is based on the revised version of Benson's outer approximation method. It employs two types of cutting planes so that enumeration of vertices is avoided. We also discover that there is a one-to-one mapping between the the incomplete vertices of the primal polyhedron and the incomplete facets of the dual polyhedron. Obviously, it is sufficient to determine the incomplete facets to solve (P). The dual method is designed to attain these facets. Derived from the dual variant of Benson's outer approximation algorithm it dispenses with enumeration of all the facets. These two new algorithms are tested against several comparable algorithms in the literature on a set of randomly generated instances. The size of the instances varies from 2 objectives, 5 variables and constraints to 5 objectives 500 variables and constraints. The experimental results show that the new methods outperform the others in terms of computational efficiency. As the size of the instances grows, the time required to achieve optimality exponentially increases. While with two objectives, all instances can be solved quite quickly. As the number of objectives increases, the merit of the primal and the dual algorithms is revealed. Specifically, when solving problems with 5 objectives and 500 variables and constraints, the primal and the dual algorithms take much less time (one sixth, respectively one tenth) than Benson's branch and bound algorithm. We also notice that the dual method performs better than the primal method in solving instances of large scale.

The nadir point is a vector of component-wise maxima of the objective functions over the non-dominated set of (MOP). This point is of interest because it plays an important role in the field of multi-objective decision making and multi-objective optimisation. The

determination of the nadir point is a special case of (ON). Therefore the primal and the dual methods are capable of finding this point. We modify these two new methods to avoid solving (P) repeatedly. For the primal method, instead of solving individual (P) for p times we integrate the p iterations for the nadir values by means of keeping records of the non-dominated points found throughout the algorithm because each non-dominated point provides p lower bounds for the corresponding nadir values. In the beginning of each iteration, a cut derived from the highest lower bound of that nadir value is utilized to remove search regions which are not worthy of exploration. For the dual method, as a single performance of it determines all incomplete facets, all nadir values can be obtained at once. Computational experiments are conducted to compare the new methods against the only exact method for computing nadir point from the literature. The results show the new methods are faster then the competitor. The CPU times required to determine the exact nadir point increase rapidly as the number of objectives increases. All three algorithms can solve instances with 3 objectives and 60 variables and 30 constraints quite quickly. As the number of objectives increases to 5 and the number of variables to 100 and of constraints to 50 the advantage of the new methods is revealed. We also notice that the dual algorithm is faster than the primal algorithm.

In Chapter 6 we extend the primal method and the dual method to solve (Q), which maximises a linear function over the non-dominated set of a convex multi-objective optimisation problem. A convex multi-objective optimisation problem (CMOP) is featured with convex objectives and constraints, which may involve nonlinear terms. A CMOP usually has non-polyhedral feasible set. An outer approximation algorithm has been developed by Ehrgott et al. (2011b) to obtain an approximation of the non-dominated set of (CMOP). Based on this method we extend the primal algorithm to solve (Q). We first show that the properties of (P) still hold for the polyhedron that approximates the non-dominated set in the outer approximation algorithm. The primal algorithm aims at an incomplete ϵ-non-dominated vertex, which maximise the objective function of (Q). The dual method for (Q) is based on the dual variant of the outer approximation algorithm.

Similarly, an outer approximation scheme is performed to the dual feasible set through an approximating polyhedron that contains the dual feasible set. The dual method for (Q) determines an incomplete $\epsilon\mathcal{K}$-maximal facet, which provides an optimal solution to (Q).

The main contributions of the study are listed below.

1. A survey and an implementation of the existing objective space based algorithms for (OE).

2. Corrections and amendments on the existing algorithms for (OE).

3. Discovery of some important properties of (P).

4. A primal algorithm and a dual algorithm for (P).

5. Computational experiments comparing the new methods and the existing methods for (P).

6. A primal method and a dual method for the determination of the exact nadir point of (MOLP).

7. Computational experiments comparing the new methods and the existing algorithm for determining the nadir point.

8. A primal method and a dual method for (Q).

9. Computational experiments comparing the new methods and the existing methods for (Q).

Bibliography

A. H. Land, A. G. D. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520.

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

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

Benayoun, R., De Montgolfier, J., Tergny, J., and Laritchev, O. (1971). Linear programming with multiple objective functions: Step method (STEM). *Mathematical Programming*, 1(1):366–375.

Benson, H. (1995). A geometrical analysis of the efficient outcome set in multiple objective convex programs with linear criterion functions. *Journal of Global Optimization*, 6(3):231–251.

Benson, H. (1998). An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem. *Journal of Global Optimization*, 13:1–24.

Benson, H. (2011). An outcome space algorithm for optimization over the weakly efficient set of a multiple objective nonlinear programming problem. *Journal of Global Optimization*, 52(3):553–574.

Benson, H. and Lee, D. (1996). Outcome-based algorithm for optimizing over the efficient set of a bicriteria linear programming problem. *Journal of Optimization Theory and Applications*, 88(1):77–105.

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

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

Bolintineanu, S. (1993). Minimization of a quasi-concave function over an efficient set. *Mathematical Programming*, 61(1-3):89–110.

Buchanan, J. T. (1997). A naïve approach for solving MCDM problems: The GUESS method. *Journal of the Operational Research Society*, 48(2):202–206.

Charnes, A., Raike, W. M., Stutz, J. D., and Walters, A. S. (1974). On generation of test problems for linear programming codes. *Communications of the ACM*, 17(10):583–586.

Dauer, J. (1993). On degeneracy and collapsing in the construction of the set of objective values in a multiple objective linear program. *Annals of Operations Research*, 46-47(2):279–292.

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

Dauer, J. P. (1987). Analysis of the objective space in multiple objective linear programming. *Journal of Mathematical Analysis and Applications*, 126(2):579–593.

Deb, K. and Miettinen, K. (2008). A Review of Nadir Point Estimation Procedures Using Evolutionary Approaches: A Tale of Dimensionality Reduction. Technical report, Indian Institute of Technology, Kanpur, India.

Deb, K., Miettinen, K., and Chaudhuri, S. (2010). Toward an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches. *IEEE Transactions on Evolutionary Computation*, 14(6):821–841.

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

Ehrgott, M. (2005). *Multicriteria Optimization (2. ed.)*. Springer, Berlin.

Ehrgott, M., Güler, Ç., Hamacher, H. W., and Shao, L. (2009a). Mathematical optimization in intensity modulated radiation therapy. *Annals of Operations Research*, 175(1):309–365.

Ehrgott, M., Löhne, A., and Shao, L. (2011a). A dual variant of benson's "outer approximation algorithm" for multiple objective linear programming. *Journal of Global Optimization*, 52(4):757–778.

Ehrgott, M., Naujoks, B., Stewart, T. J., and Wallenius, J. (2010). *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, volume 634 of *Lecture Notes in Economics and Mathematical Systems*. Springer, Berlin.

Ehrgott, M., Shao, L., and Schöbel, A. (2011b). An approximation algorithm for convex multi-objective programming problems. *Journal of Global Optimization*, 50(3):397–416.

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

Ehrgott, M., Waters, C., Kasimbeyli, R., and Ustun, O. (2009b). Multiobjective programming and multiattribute utility functions in portfolio optimization. *Information Systems and Operational Research*, 47:117–128.

Fruhwirth, M. and Mekelburg, K. (1994). On the efficient point set of tricriteria linear programs. *European Journal of Operational Research*, 72:192–199.

Fülöp, J. (1994). A cutting plane algorithm for linear optimization over the efficient set. *Generalized Convexity*, 405:pp.374–385.

Fülöp, J. and Muu, L. D. (2000). Branch-and-bound variant of an outcome-based algorithm for optimizing over the efficient set of a bicriteria linear programming problem. *Journal of Optimization Theory and Applications*, 105(1):37–54.

Hamel, A., Löhne, A., and Rudloff, B. (2014). Benson type algorithms for linear vector optimization and applications. *Journal of Global Optimization*, 59(4):811–836.

Heyde, F. and Löhne, A. (2008). Geometric duality in multiple objective linear programming. *SIAM Journal on Optimization*, 19(2):836–845.

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

Kim, N. T. B. and Thang, T. N. (2013). Optimization over the efficient set of a bicriteria convex programming problem. *Pacific Journal of Optimization*, 9:103–115.

Le Thi, H. A., Pham, D. T., and Thoai, N. V. (2002). Combination between global and local methods for solving an optimization problem over the efficient set. *European Journal of Operational Research*, 142(2):258–270.

Löhne, A., Rudloff, B., and Ulus, F. (2014). Primal and dual approximation algorithms for convex vector optimization problems. *Journal of Global Optimization*, 60(4):713–736.

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

Miettinen, K. (1999). *Nonlinear multiobjective optimization*, volume 12 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers Dordrecht.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA.

Nguyen Thi, B. K., Le Thi, H. A., and Tran, M. T. (2008). Outcome-space polyblock approximation algorithm for optimizing over efficient sets. *Modelling, Computation and Optimization in Information Systems and Management Sciences*, 14:234–243.

Philip, J. (1972). Algorithms for the vector maximization problem. *Mathematical Programming*, 2(1):207–229.

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

Steuer, R. E. (1986). *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley, New York, 546 pp.

Thach, P. T. (1991). Quasiconjugates of functions, duality relationship between quasiconvex minimization under a reverse convex constraint and quasiconvex maximization under a convex constraint, and applications. *Journal of Mathematical Analysis and Applications*, 159(2):299–322.

Thach, P. T., Konno, H., and Yokota, D. (1996). Dual approach to minimization on the set of pareto-optimal solutions. *Journal of Optimization Theory and Applications*, 88(3):689–707.

Thai Quynh, P. and Hoang Quang, T. (2000). Bisection search algorithm for optimizing over the efficient set. *Vietnam Journal of Mathematics*, 28(3):217.

Thoai, N. V. (2000). Conical algorithm in global optimization for optimizing over efficient sets. *Journal of Global Optimization*, 18(4):321–336.

Tuyen, H. Q. and Muu, L. D. (2001). Biconvex programming approach to optimization over the weakly efficient set of a multiple objective affine fractional problem. *Operations Research Letters*, 28(2):81–92.

White, D. J. (1996). The maximization of a function over the efficient set via a penalty function approach. *European Journal of Operational Research*, 94(1):143–153.

Yamada, S., Tanino, T., and Inuiguchi, M. (2000). An inner approximation method for optimization over the weakly efficient set. *Journal of Global Optimization*, 16(3):197–217.

Yamada, S., Tanino, T., and Inuiguchi, M. (2001). An inner approximation method incorporating a branch and bound procedure for optimization over the weakly efficient set. *European Journal of Operational Research*, 133(2):267–286.

Yu, P.-L. (1985). *Multiple-Criteria Decision Making (Concepts, Techniques, and Extensions)*, volume 30 of *Mathematical Concepts and Methods in Science and Engineering*. Springer US.