To appear in *Optimization*
Vol. 00, No. 00, Month 20XX, 1–28

Primal and Dual Algorithms for Optimisation over the Efficient Set

Zhengliang Liu[a*] and Matthias Ehrgott[b]

[a]*School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, United Kingdom*

[b]*Department of Management Science, Lancaster University Management School, Bailrigg, Lancaster LA1 4YX, United Kingdom*

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

Optimisation over the efficient set of a multi-objective optimisation problem is a mathematical model for the problem of selecting a most preferred solution that arises in multiple criteria decision making to account for trade-offs between objectives within the set of efficient solutions. In this paper we consider a particular case of this problem, namely that of optimising a linear function over the image of the efficient set in objective space of a convex multi-objective optimisation problem. We present both primal and dual algorithms for this task. The algorithms are based on recent algorithms for solving convex multi-objective optimisation problems in objective space with suitable modifications to exploit specific properties of the problem of optimisation over the efficient set. We first present the algorithms for the case that the underlying problem is a multi-objective linear programme. We then extend them to be able to solve problems with an underlying convex multi-objective optimisation problem. We compare the new algorithms with several state of the art algorithms from the literature on a set of randomly generated instances to demonstrate that they are considerably faster than the competitors.

Keywords: Multi-objective optimisation, optimisation over the efficient set, objective space algorithm, duality.

1. Introduction

Multi-objective optimisation deals with optimisation problems with multiple conflicting objectives. It has many applications in practice, e.g. minimising cost versus minimising adverse environmental impacts in infrastructure projects [13], minimising risk versus maximising return in financial portfolio management [23] or maximising tumour control versus minimising normal tissue complications in radiotherapy treatment design [11]. Because a feasible solution simultaneously optimising all of the objectives does not usually exist, the goal of multi-objective optimisation 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 of the others. In practical applications of multi-objective optimisation it is generally necessary for a decision maker to select one solution from the efficient set for implementation. This selection process can be modelled as the optimisation of a function over the efficient set of the underlying multi-objective optimisation problem. For example, an investor may aim at minimising the transaction cost of establishing a portfolio with high return and low risk. If such a function that describes the decision maker's preferences is explicitly available, one might expect that it is computationally easier to directly optimise the function over the efficient set rather than to solve the multi-objective optimisation problem first and then obtain a most preferred

*Corresponding author. Email: zliu082@gmail.com

efficient solution – in particular because the efficient set can in general be an infinite set (see Figures 1 and 2 below). Secondly, decision makers may be overwhelmed by the large size of the whole efficient set and may not be able to choose a preferred solution from it in a very effective manner. These considerations have motivated research on the subject of optimisation over the efficient set since the 1970s.

We contribute to this research with new algorithms and new results on how to identify optimal solutions of this problem. In Section 2 we provide the mathematical preliminaries on multi-objective optimisation and necessary notation. A revised version of Benson's outer approximation algorithm and its dual variant are summarised in Section 3. In Section 4, we survey algorithms for optimisation over the efficient set from the literature. As a new contribution, we identify a subset of the vertices of the feasible set in objective space at which an optimal solution must be attained. Based on the outer approximation algorithm we then propose a new primal algorithm in the all linear case by incorporating a bounding procedure in the primal algorithm of [18] in Section 5. Furthermore, this primal algorithm is extended to maximise a linear function over the non-dominated set of a convex multi-objective optimisation problem. Section 6 proposes a new dual algorithm for optimisation over the non-dominated set, which makes use of a new result providing a geometrical interpretation of optimisation over the non-dominated set. This dual algorithm is also further developed to maximise a linear function over the non-dominated set of a convex problem. The numerical experiments in Section 7 compare the performance of our new primal and dual algorithms with algorithms from the literature that we discuss in Section 4. The results reveal that our algorithms, in particular the dual algorithm in the linear case, are much faster (up to about 10 times) than comparable algorithms from the literature.

2. Preliminaries

A multi-objective optimisation problem (MOP) can be written as

$$\min\{f(x) : x \in \mathcal{X}\}, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^p$ is a function, which we assume to be continuous, and $\mathcal{X}$ is a feasible set in decision space $\mathbb{R}^n$. Let $\mathcal{Y} := \{f(x) : x \in \mathcal{X}\}$ denote the image of $\mathcal{X}$ in objective space $\mathbb{R}^p$. We assume that $\mathcal{X}$ is a nonempty and compact set. Therefore, $\mathcal{Y}$ is nonempty and compact, too. The feasible set $\mathcal{Y}$ in objective space is bounded by the vectors of componentwise minima and maxima denoted by y^I and y^{AI}, respectively, i.e., $y^I_k \leqq y_k \leqq y^{AI}_k$ for all $y \in \mathcal{Y}$. These two points are called the ideal and anti-ideal point, respectively.

We use the following notation to compare vectors $y^1, y^2 \in \mathbb{R}^p$: $y^1 < y^2$ if $y^1_k < y^2_k$ for $k = 1, \ldots, p$; $y^1 \leqq y^2$ if y^1_k is componentwise less than or equal to y^2_k and $y^1 \leq y^2$ if $y^1 \leqq y^2$ but $y^1 \neq y^2$. We also define $\mathbb{R}^p_{\geqq} := \{y \in \mathbb{R}^p : y \geqq 0\}$ and $\mathbb{R}^p_{\geq} := \{y \in \mathbb{R}^p : y \geq 0\} = \mathbb{R}^p_{\geqq} \setminus \{0\}$. Sets $\mathbb{R}^p_{\leqq}$, $\mathbb{R}^p_{\leq}$ and $\mathbb{R}^p_{<}$ are defined analogously.

Definition 1 *A feasible solution $\hat{x} \in \mathcal{X}$ is called a (weakly) efficient solution of* (1) *if there is no $x \in \mathcal{X}$ such that $f(x) \leq (<) f(\hat{x})$. The set of all (weakly) efficient solutions is called the (weakly) efficient set in decision space and is denoted by $\mathcal{X}_{(W)E}$. Correspondingly, $\hat{y} = f(\hat{x})$ is called a (weakly) non-dominated point and $\mathcal{Y}_{(W)N} := \{f(x) : x \in \mathcal{X}_{(W)E}\}$ is the (weakly) non-dominated set in objective space.*

Throughout this article we assume that the multi-objective optimisation problem is nontrivial, i.e. $y^I \notin Y$ and $Y \neq Y_N$, which means that the objectives do not have a common minimiser and that not every feasible point is non-dominated. If $y^I \in Y$ then $Y_N = \{y^I\}$.

In this case, as well as if $Y_N = Y$ the optimisation problem we study in this paper becomes either trivial or becomes a standard single objective optimisation problem.

In case all objectives of (1) are linear and the feasible set $\mathcal{X}$ is polyhedral (1) is called a multi-objective linear programme (MOLP) and can be written as

$$\min\{Cx : x \in \mathcal{X}\}, \tag{2}$$

where C is a $p \times n$ matrix. The feasible set $\mathcal{X}$ is defined by linear constraints $Ax \leqq b$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. 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 $\mathcal{X}$ at x^0 if $\omega^T x \geqq \gamma$ for all $x \in \mathcal{X}$ and $\omega^T x^0 = \gamma$. 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 $\mathcal{X}$.

In Example 1 we provide a numerical example of a multi-objective linear programme, which we shall use throughout the paper.

Example 1

$$\min \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

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

Figure 1 shows the feasible set $\mathcal{X}$ of Example 1 as well as its image $\mathcal{Y}$ in objective space (due to C being the identity matrix). The bold line segments compose the non-dominated set $\mathcal{Y}_N$. Furthermore, in this example $\mathcal{X}_E$ is the same as $\mathcal{Y}_N$.

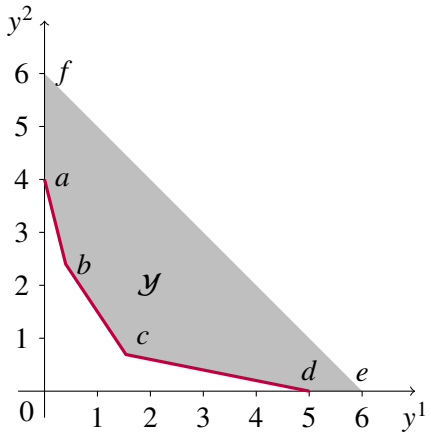

Figure 1. The image of the feasible set of the MOLP of Example 1 in objective space.

A convex multi-objective optimisation problem (CMOP) is a multi-objective optimisation problem with convex objective functions and a convex feasible set $\mathcal{X} = \{x \in \mathbb{R}^n : g(x) \leqq 0\}$, where $g : \mathbb{R}^n \to \mathbb{R}^m$. Example 2 provides a simple example of a CMOP that we shall use throughout the paper.

Example 2

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

The image of the feasible set in objective space is illustrated in Figure 2. The bold curve is the non-dominated set of the CMOP.

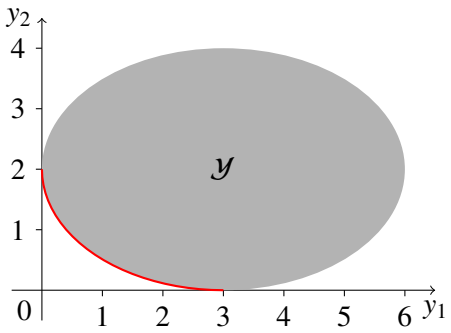

Figure 2. The image of the feasible set of the CMOP of Example 2 in objective space.

To solve multi-objective optimisation problems is generally understood as obtaining $\mathcal{Y}_N$, and for each point $y \in \mathcal{Y}_N$ some $x \in \mathcal{X}_E$ with $f(x) = y$. This paper is concerned with the optimisation of a function over the efficient set of a multi-objective optimisation problem,

$$\max \ \{\Phi(x) : x \in \mathcal{X}_E\}, \tag{3}$$

where $\Phi(x) : \mathbb{R}^n \to \mathbb{R}$ is a function of x.

In general, (3) is a difficult global optimisation problem [6, 17, 20, 24, 25]. This is due to the fact that the efficient set of even a multi-objective linear programme is nonconvex (see Example 1). As such, there exist local optima, which may differ from global optima. In Example 1, assuming $\Phi(x) = x_1 + x_2$, $(0, 4)^T$ is a local optimal solution. Along the efficient edges from $(0, 4)^T$ to $(0.4, 2.4)^T$ and $(9/13, 20/13)^T$ the objective value decreases, before increasing again to the global optimal solution at $(5, 0)^T$. Furthermore, since the feasible set of (3) is the efficient set of an MOP it can in general not be explicitly expressed in the form of a system of inequalities prior to solving the problem.

Fülöp [16] has shown the equivalence of optimisation over the efficient set to a bilevel optimisation problem and NP-hardness of the problem. In Example 3 we demonstrate that the bilevel optimisation approach even in the case of optimising a linear function over the efficient set of an MOLP does in general lead to a nonlinear optimisation problem.

Example 3 Consider an all linear version of (3), i.e.,

$$\max \ \{\mu^T C x : x \in \mathcal{X}_E\}, \tag{4}$$

where $\mathcal{X}_E$ is the efficient set of the MOLP

$$\min \{Cx : Ax \geqq b\}. \tag{5}$$

(4) and (5) can be regarded as a bilevel optimisation problem, where (4) is the upper level problem and (5) is the lower level problem. The constraint $x \in \mathcal{X}_E$ in (4) can be replaced by the well known optimality conditions for MOLP, see e.g. Ehrgott [10]. Feasible solution $x \in \mathcal{X}$ is efficient if and only if there exist $\lambda \in \mathbb{R}^p_>$ and $u \in \mathbb{R}^m_{\geqq}$ such that $A^T u = \lambda^T C$ and $\lambda^T C x = b^T u$.

Therefore (4) can be rewritten as

$$\max \left\{ \mu^T C x : Ax \geqq b, A^T u = C^T \lambda, \lambda^T C x = b^T u, u \geqq 0, \lambda > 0 \right\}. \tag{6}$$

While (6) has a linear objective function and some linear constraints, it also has quadratic constraints and requires that λ be strictly positive. Hence solving (6) will be considerably more difficult than solving a linear programme. Example 3 thus provides one motivation for research in algorithms to solve (3) in the linear case: is it possible to derive algorithms that only make use of linear programming techniques?

In multi-objective optimisation it is well known that, because the number of objective functions is usually much smaller than the number of decision variables, the structure of $\mathcal{Y}$ is most often simpler than the structure of $\mathcal{X}$. In particular, in multi-objective linear programming, the structure and properties of $\mathcal{X}_E$ and $\mathcal{Y}_N$ are well investigated. [9] notes that $\mathcal{Y}$ often has fewer extreme points and faces than $\mathcal{X}$. [8] illustrates the concept of "collapsing", which means that faces of $\mathcal{X}$ shrink into nonfacial subsets of $\mathcal{Y}$. [1] shows that the dimension of efficient faces in the feasible set always exceeds or equals the dimension of their images in $\mathcal{Y}$. Hence, it is arguably more computationally efficient to employ techniques and methods to solve (3) in objective space, and algorithms for optimisation over the efficient set have followed this trend since 2000. The algorithms we propose in this paper fall in this category, too.

In this context, we assume that the objective function Φ of (3) is a composite function of a function $M : \mathbb{R}^p \to \mathbb{R}$ and the objective function f of the underlying MOP, i.e., $\Phi = M \circ f$. Therefore, $\Phi(x) = M(f(x))$. Substituting $y = f(x)$ into (3), we derive the problem of optimising M over the non-dominated set $\mathcal{Y}_N$ of an MOP:

$$\max \{ M(y) : y \in \mathcal{Y}_N \}. \tag{7}$$

Problem (7) is essentially the same problem as (3) but appears to be more intuitive than (3), because in practice decision makers typically choose a preferred solution based on the objective function values rather than the value of decision variables.

In this article, the problems we are interested in are two special cases of (7) namely (8) and (10) defined below.

$$\max \left\{ \mu^T y : y \in \mathcal{P}_N \right\}, \tag{8}$$

where $\mu \in \mathbb{R}^p$ and $\mathcal{P}_N$ is the non-dominated set of the upper image $\mathcal{P} := \mathcal{Y} + \mathbb{R}^p_{\geqq}$ of an MOLP (2). Note that the set of vertices of $\mathcal{P}$, $V_{\mathcal{P}} \subset \mathcal{P}_N = \mathcal{Y}_N$, i.e., all vertices of $\mathcal{P}$ are non-dominated, and the non-dominated sets of $\mathcal{Y}$ and $\mathcal{P}$ coincide. Moreover, $\mathcal{P}_N$ is a subset of the boundary of $\mathcal{P}$, see for example [10], Proposition 2.4. It is easy to see that Theorem 1 holds for problem (8).

THEOREM 1 *There exists an optimal solution y^* of* (8) *at a vertex of $\mathcal{P}$, i.e., $y^* \in V_{\mathcal{P}}$, the set of vertices of $\mathcal{P}$.*

In the second special case, we consider a CMOP as underlying MOP. Then using once again $\mathcal{P}_N = \mathcal{Y}_N$, problem (9) optimises a linear function over the non-dominated set $\mathcal{P}_N$

of the upper image $\mathcal{Y} + \mathbb{R}^p_{\leqq}$ of of a CMOP.

$$\max \left\{\mu^T y : y \in \mathcal{P}_N\right\}. \tag{9}$$

Because the upper image $\mathcal{P}$ of a CMOP is a convex (but not necessarily polyhedral) set, we will in general not be able to compute it or its non-dominated subset exactly. Hence we consider approximations of $\mathcal{P}_N$ using the concept of ϵ-non-dominance as defined below.

Definition 2 *Let $\epsilon \in \mathbb{R}, \epsilon > 0$. A point y is called (weakly) ϵ-non-dominated if $y + \epsilon e \in \mathcal{Y}$, where e is a column vector with all elements being one, and there does not exist any $\hat{y} \in \mathcal{Y}$ such that $\hat{y} \leq (<) y$.*

Consequently we change Problem (9) by replacing $\mathcal{P}_N$ with $\mathcal{P}_{\epsilon N}$, an ϵ-non-dominated set of the upper image of a CMOP.

$$\max \left\{\mu^T y : y \in \mathcal{P}_{\epsilon N}\right\}. \tag{10}$$

3. Outer Approximation Algorithms in Multi-objective Optimisation

3.1. *Multi-objective linear programming*

Theorem 1 suggests that in order to solve (8) an algorithm to identify extreme points (vertices) of $\mathcal{P}$ is needed. Benson [2] proposed an outer approximation algorithm to compute the non-dominated set $\mathcal{Y}_N$ of an MOLP. The authors of [12] revised this method in such a way that it considers $\mathcal{P}$ rather than $\mathcal{Y}$ and in [18] it was further developed by reducing the number of LP subproblems that need to be solved (1 rather than 2 in each iteration). This revised version of Benson's algorithm 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^i from the vertex set $V_{\mathcal{S}^{i-1}}$ which is not in $\mathcal{P}$ and constructs a supporting hyperplane to $\mathcal{P}$ by solving an LP and obtaining its dual variable values. $\mathcal{S}^i$ is then defined by intersecting $\mathcal{S}^{i-1}$ with the halfspace of the supporting hyperplane containing $\mathcal{P}$ and updating the vertex set of $\mathcal{S}^i$. The algorithm terminates as soon as no such $s^i \in \mathcal{S}^{i-1} \setminus \mathcal{P}$ can be found and therefore $\mathcal{S}^i = \mathcal{P}$. At termination both a vertex and an inequality representation of $\mathcal{P}$ are known. [12] propose a dual variant of Benson's algorithm to solve (13), which has been further developed by [18].

We first provide notation that will facilitate the description of the subsequent algorithms. For $y \in \mathbb{R}^p$ and $v \in \mathbb{R}^p$, let

$$\lambda(v) := \left(v_1, ..., v_{p-1}, 1 - \sum_{i=1}^{p-1} v_i\right)^T, \tag{11}$$

$$\lambda^*(y) := \left(y_1 - y_p, ..., y_{p-1} - y_p, -1\right)^T. \tag{12}$$

Consider the weighted sum scalarisation $P_1(v)$ of (2)

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

Proposition 1 is well known, see e.g. [14].

Proposition 1 *Let $v \in \mathbb{R}^p_{\geq}$ such that $\sum_{k=1}^{p} v_k \leqq 1$. Then an optimal solution $\hat{x}$ to $(P_1(v))$ is a weakly efficient solution to the MOLP* (2).

The dual $D_1(v)$ of $P_1(v)$ is

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

Given a point $y \in \mathbb{R}^p$ in objective space, the following LP $(P_2(y))$ serves to check the feasibility of y, i.e., if the optimal value $\hat{z} > 0$, then y is infeasible, otherwise $y \in \mathcal{P}$. An optimal solution $(\hat{x}, \hat{z}) \in \mathcal{X} \times \mathbb{R}$ to $(P_2(y))$ provides a weakly non-dominated point $\hat{y} = C\hat{x}$ of $\mathcal{P}$.

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

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

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

Proposition 2 is the key result for the revised version of Benson's algorithm.

Proposition 2 *[18] Let (u^*, λ^*) be an optimal solution to $(D_2(y))$. Then*

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

is a supporting hyperplane to $\mathcal{P}$.

Therefore, by solving $(P_2(y))$, we not only check the feasibility of point y but also obtain the dual variable values (u^*, λ^*) as optimal solutions to $(D_2(y))$ by which we construct a supporting hyperplane to $\mathcal{P}$.

[19] introduced a concept of geometric duality for multi-objective linear programming. This theory relates an MOLP with a dual multi-objective linear programme in dual objective space $\mathbb{R}^p$. In dual objective space, we use the following notation to compare two vectors $v^1, v^2 \in \mathbb{R}^p$. We write $v^1 >_{\mathcal{K}} v^2$ if $v_k^1 = v_k^2$ for $k = 1, \ldots, p-1$ and $v_p^1 > v_p^2$; $v^1 \geqq_{\mathcal{K}} v^2$ if $v_k^1 = v_k^2$ for $k = 1, \ldots, p-1$ and $v_p^1 \geqq v_p^2$. Moreover $v^1 \geq_{\mathcal{K}} v^2$ is the same as $v^1 >_{\mathcal{K}} v^2$.

The dual of MOLP is

$$\max_{\mathcal{K}} \left\{ (\lambda_1, ..., \lambda_{p-1}, b^T u)^T : (u, \lambda) \geqq 0, A^T u = C^T \lambda, e^T \lambda = 1 \right\}, \tag{13}$$

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 its lower image is $\mathcal{D} := \mathcal{V} - \mathcal{K}$. The $\mathcal{K}$-maximal set of $\mathcal{D}$ is $\mathcal{D}_{\mathcal{K}} = \left\{ v \in \mathcal{V} : v = \max_{\mathcal{K}} \{ (\lambda_1, ..., \lambda_{p-1}, b^T u)^T : (u, \lambda) \geqq 0, A^T u = C^T \lambda, e^T \lambda = 1 \} \right\}$. Figure 3 shows the lower image $\mathcal{D}$ in the dual objective space of Example 1. The bold line segments compose $\mathcal{D}_{\mathcal{K}}$.

In [19] two set-valued maps H and H^* are defined to relate $\mathcal{P}$ and $\mathcal{D}$.

$$H : \mathbb{R}^p \rightrightarrows \mathbb{R}^p, H(v) := \left\{ y \in \mathbb{R}^p : \lambda(v)^T y = v_p \right\} \text{ and} \tag{14}$$

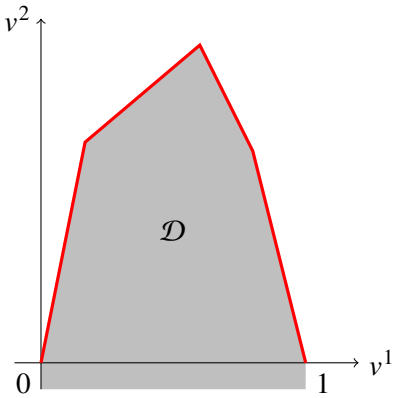

Figure 3. Lower image of (13) for the MOLP of Example 1.

$$H^* : \mathbb{R}^p \rightrightarrows \mathbb{R}^p, H^*(y) := \left\{ v \in \mathbb{R}^p : \lambda^*(y)^T v = -y_p \right\}. \tag{15}$$

Given a point $v \in \mathbb{R}^p$ in dual objective space $H(v)$ defines a hyperplane in primal objective space. Similarly, given a point $y \in \mathbb{R}^p$ in primal objective space $H^*(y)$ is a hyperplane in dual objective space.Theorems 2 and 3 state a relationship between proper $\mathcal{K}$-maximal faces of $\mathcal{D}$ and proper weakly non-dominated faces of $\mathcal{P}$.

THEOREM 2 *[19]*

(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 $\mathcal{D}$.

[19] 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 3 *[19] 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 $dim\mathcal{F}^* + dim\Psi(\mathcal{F}^*) = p - 1$.

Therefore, given a non-dominated extreme point $y^{ex} \in \mathcal{P}$, the corresponding $\mathcal{K}$-maximal facet of $\mathcal{D}$ is $H^*(y^{ex}) \cap \mathcal{D}$.

The revised version of Benson's algorithm applies an outer approximation to $\mathcal{P}$ in the primal objective space, whereas its dual variant does the same to $\mathcal{D}$ in the dual objective space. [18] detail the dual algorithm. It iteratively generates supporting hyperplanes of $\mathcal{D}$, which correspond to weakly non-dominated faces of $\mathcal{P}$. Eventually a complete set of hyperplanes that define $\mathcal{D}$ as well as the set of all extreme points of $\mathcal{D}$ are obtained.

3.2. *Multi-objective convex optimisation*

We review an extension of Benson's outer approximation algorithm proposed by [22]. It provides a set of ϵ-non-dominated points by means of approximating the non-dominated set of a convex MOP. Similar to the revised version of Benson's algorithm for computing $\mathcal{P}_N$ in the case of an MOLP this algorithm iteratively constructs a polyhedral outer approximation of the upper image $\mathcal{P}$ of a CMOP. It starts with a polyhedron $\mathcal{S}^0 = y^I + \mathbb{R}^p_{\leqq}$ containing $\mathcal{P}$. In each iteration i, a vertex of $\mathcal{S}^{i-1}$ that is not an ϵ-non-dominated point of $\mathcal{P}$ is randomly chosen to generate a supporting hyperplane to $\mathcal{P}$. Then the approximating 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 of $\mathcal{S}^i$ are ϵ-non-dominated with a vertex and inequality representation of polyhedron $\mathcal{S}^i$ containing $\mathcal{P}$.

We introduce two pairs of single objective optimisation problems to facilitate the description of 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 $\mathbb{P}_1(v)$. Because we work with Lagrangian duals, we will from now on assume that the functions defining the MOP are differentiable. Problems $\mathbb{P}_2(y)$ and $\mathbb{D}_2(y)$ are employed to generate supporting hyperplanes. These four optimisation problems are the nonlinear extensions of the LPs $P_1(v)$, $D_1(v)$, $P_2(y)$ and $D_2(y)$, respectively. They involve nonlinear convex terms making them harder to solve than their LP counterparts.

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

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

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

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

The outer approximation algorithm of [22] for convex MOPs approximates the non-polyhedral $\mathcal{P}$ (see Example 2) generating a set of ϵ-non-dominated points. Löhne et al. [22] propose a dual variant of the convex version of Benson's algorithm (see Section 5.3). The geometric dual of a convex MOP is defined as

$$\max_{\mathcal{K}} \{D(v) : v \in \mathbb{R}^p, \lambda(v) \geq 0\}, \tag{16}$$

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 (13), 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 its lower image in the dual objective space is $\mathcal{D} := \mathcal{V} - \mathcal{K}$. The $\mathcal{K}$-maximal set of (16) is

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

Figure 4 shows the lower image of Example 2 in dual objective space and the bold curve is the $\mathcal{K}$-maximal set. For computational purposes we consider an approximation of the

$\mathcal{K}$-maximal set. This is modelled by the concept of $\epsilon\mathcal{K}$-maximum, which is defined in Definition 3.

DEFINITION 3 *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 $\hat{v}_p > v_p$.*

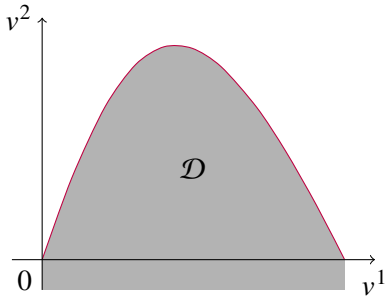

Figure 4. The lower image of the CMOP of Example 2 in dual objective space.

The dual version of the convex Benson algorithm for solving CMOPs performs an outer approximation to $\mathcal{D}$. This algorithm first chooses an interior point in $\mathcal{D}$. This is implemented in the way stated in Algorithm 2, Step i1 in [12]. Then a polyhedron $\mathcal{S}^0$ containing $\mathcal{D}$ is constructed. In each iteration a vertex s^i of $\mathcal{S}^{i-1}$ that does not belong to $\mathcal{D}_{\epsilon\mathcal{K}}$ is chosen. By solving $(\mathbb{P}_1(s^i))$, a supporting hyperplane to $\mathcal{D}$ is determined. Eventually, a set of $\epsilon\mathcal{K}$-maximal points of $\mathcal{D}$ is obtained, the convex hull of which extended by $-\mathcal{K}$ is an outer approximation polyhedron of $\mathcal{D}$.

4. A Review of Algorithms for Optimisation over the Efficient Set

In the literature there are two main categories of algorithms for optimisation over the efficient set: Algorithms based on decision space methods and algorithms based on objective space methods. The former approaches search for optimal solutions in decision space, whereas the latter ones explore the non-dominated set in objective space. In this section, we review some algorithms of the latter class, to which our new algorithms are compared in computational experiments. A more comprehensive review of the literature up to 2001 can be found in [25].

4.1. *Bi-objective branch and bound algorithm*

In this section we describe a bi-objective branch and bound algorithm to solve a special case of (7), where $M(y)$ is a lower semi-continuous 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 [4] and further improved by Fülöp and Muu [17].

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 feasible, hence they can be used to find a lower bound on $M(y)$. Then optimisation problem (17) is solved to obtain an upper bound,

$$\max \left\{ M(y) : (y_2^2 - y_2^1)y_1 + (y_1^1 - y_1^2)y_2 \geqq y_1^1 y_2^2 - y_2^1 y_1^2, y \in \mathcal{Y} \right\}. \tag{17}$$

In the objective space, (17) means to find an optimal point over the region bounded by the non-dominated set and the line connecting y^1 and y^2. Having solved problem (17), we have found an upper bound. In the case that $M(y)$ is nonlinear, branching steps may now take place. 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 both y^1 and y^2 with q, 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 [17].

In [21] this method is extended to maximise an increasing function $M(y)$ over the non-dominated set of a convex bi-objective optimisation problem. The first step is to determine y^1 and y^2 and a lower bound in the same way as the linear version by [17]. These two points and the ideal point y^I define a simplex. The objective function $M(y)$ is maximised over the intersection of $\mathcal{Y}$ and the simplex so that an upper bound on the optimal objective function value can be attained. A point q at the intersection of a ray emanating from the origin and $\mathcal{Y}$ is determined. Point q splits the simplex into two, 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. 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.

4.2. *Conical branch and bound algorithm*

Another branch and bound algorithm was proposed in [24] to optimise a continuous function over the non-dominated set of an MOP. A conical partition technique is employed as the branching process. Cone $y^{AI} - \mathbb{R}^p_{\geqq}$ with vertex y^{AI} is constructed. This cone contains $\mathcal{Y}$. Along each extreme direction $-e^k$ of the cone, the intersection point y^k of the direction and the weakly non-dominated set $\mathcal{P}_{WN}$ of the upper image $\mathcal{P} = \mathcal{Y} + \mathbb{R}^p_{\geqq}$ of the MOP is obtained. If any of these points is non-dominated a lower bound is found. Then solving problem (18), a relaxation of (7), finds an upper bound.

$$\max \left\{ M(y) : y - U\lambda = y^{AI}, \sum_{i=1}^{p} \lambda_i \geqq 1, \lambda \geqq 0, y \in \mathcal{Y} \right\}, \tag{18}$$

where U is a matrix containing column vectors $y^k - y^{AI}$ for $k = 1, \ldots, p$. For the branching step, a ray emanating from y^{AI} and passing through the centre point of the simplex spanned by the y^k vectors hits the boundary of $\mathcal{Y}$ at a non-dominated point and a lower bound is achieved. The initial cone is partitioned. By evaluating each new cone, the gap between the upper bound and the lower bound is narrowed. 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. An optimal point is obtained when the upper bound coincides with the lower bound.

This algorithm can also deal with optimisation over the non-dominated set of a CMOP. A cone is constructed that contains $\mathcal{Y}$. An upper bound for $M(y)$ can be found through maximising $M(y)$ over the intersection of the cone with $\mathcal{Y}$. A ray emanating from y^{AI} intersects the non-dominated set of $\mathcal{Y}$ at some point providing a lower bound and partitioning the cone. The objective function $M(y)$ is maximised over each of the regions

obtained by intersecting the smaller cones with the feasible set in objective space, which provides new upper bounds. This process is repeated until the upper bound and the lower bound coincide or the gap between them is small enough.

4.3. *Outcome space algorithm*

A branch and bound technique also plays an essential role in the algorithm of [3]. This algorithm is designed for globally optimising a finite, convex function over the weakly efficient set of a nonlinear multi-objective optimisation problem that has nonlinear objective functions and a convex, non-polyhedral feasible region. An initial simplex containing $\mathcal{Y}$ is constructed and a branching procedure is performed. 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. In order to find 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 [3] for more details.

5. The Primal Algorithms

5.1. *Properties of optimisation over the non-dominated set of an MOLP*

In this section, we investigate some properties of problem (8). It is well known that an optimal solution to (3) with linear function Φ and underlying MOLP is attained at an efficient vertex of $\mathcal{X}$, see [5]. The version of this result for problem (8), Theorem 1, implies that a naïve algorithm for solving (8) is to enumerate the vertices of $\mathcal{P}$ and determine which one has the largest value of $\mu^T y$. This is summarised in Algorithm 1.

Algorithm 1 Brute force algorithm.

Input: A, b, C, μ
Output: y^*, an optimal solution to (8)
Phase 1: Obtain $V_{\mathcal{P}}$.
Phase 2: $y^* = \operatorname{argmax}\left\{\mu^T y : y \in V_{\mathcal{P}}\right\}$.

In Phase 1 both the revised Benson algorithm and its dual variant are capable of finding $V_{\mathcal{P}}$. However, taking advantage of properties of (8) 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 *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 $[a - b]$, $y \in \mathcal{Y}_N$.*

According to [26] (Chapter 8) a point in the relative interior of a face of $\mathcal{Y}$ is non-dominated if and only if the entire face is non-dominated.

Definition 5 *Vertex $a \in V_{\mathcal{P}}$ is called a complete vertex if all faces $\mathcal{F}$ of $\mathcal{P}$ containing a are non-dominated, otherwise it is called an incomplete vertex. Let $V_{\mathcal{P}}^c$ denote the set of complete vertices of $\mathcal{P}$, and define $V_{\mathcal{P}}^{ic} := V_{\mathcal{P}} \setminus V_{\mathcal{P}}^c$ as the set of incomplete vertices.*

Definition 5 provides a partition of the vertices of $\mathcal{P}$ into complete and incomplete ones.In Figure 1 the complete vertices are b and c. The incomplete vertices are a and d,

whereas e and f are vertices of $\mathcal{Y}$ but not of $\mathcal{P}$.

Proposition 3 *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* (19)*,*

$$\max \left\{\mu^T y : y \in \mathcal{Y}\right\}. \tag{19}$$

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 (19) *has at least one optimal solution that is also an optimal solution to* (8) *if and only if $\mu \in \mathbb{R}^p_{\leqq}$.*

Proof. (1) Let $\mu \in \mathbb{R}^p_{\leqq}$. Define $\mu' := -\mu$, then $\mu' \in \mathbb{R}^p_{\geqq}$ and we rewrite (19) as $\min\{\mu'^T Cx : x \in \mathcal{X}\}$, which is a weighted sum scalarisation of the underlying MOLP. It is well known that there exists an efficient solution $x^* \in \mathcal{X}$ which is an optimal solution to this problem. Therefore, $y^* = Cx^* \in \mathcal{P}_N$. Hence, y^* is an optimal solution to (8).

(2) Now, let $\mu \notin \mathbb{R}^p_{\leqq}$ and assume that y^* is an optimal solution to (8). Choose $d \in \mathbb{R}_{\geq}$ such that $d_j = 1$ if $\mu_j > 0$ and $d_j = 0$ otherwise. Let $y' = y^* + \epsilon d$ and $\epsilon > 0$. Since $\mathcal{P} = \mathcal{Y} + \mathbb{R}^p_{\leqq}$, $y' \in \mathcal{Y}$ for sufficiently small ϵ. However, $\mu^T y' = \mu^T y^* + \epsilon \mu^T d > \mu^T y^*$ and so y^* is not an optimal solution to (19).

□

Using similar arguments as in the proof of Theorem 4 it is in fact possible to show that $\mu \in \mathbb{R}^p_{<}$ if and only if the set of optimal solutions of both (8) and (19) is identical.

Proposition 4 *Let $\mu \in \mathbb{R}^p \setminus \mathbb{R}^p_{\leqq}$. If $y \in V^c_{\mathcal{P}}$, 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 by Proposition 3, y would be an optimal solution to (19). As y is also a non-dominated point, y would be an optimal solution to (8). According to Theorem 4 this implies $\mu \in \mathbb{R}^p_{\leqq}$, a contradiction. □

The main result of this section is Theorem 5.

Theorem 5 *For all $\mu \in \mathbb{R}^p \setminus \mathbb{R}^p_{\leqq}$ there exists an optimal solution y^* of* (8) *in $V^{ic}_{\mathcal{P}} \cap V_{\mathcal{P}}$, i.e. at an incomplete non-dominated vertex.*

Proof. Under the assumptions of the theorem, assume that $V^{ic}_{\mathcal{P}}$ does not contain an optimal solution of (8), then because of Theorem 1 there exists an optimal solution $y^* \in V^c_{\mathcal{P}}$. According to Proposition 4, there exists $y \in N(y^*)$ such that $\mu^T y > \mu^T y^*$. Since $y \in \mathcal{Y}_N$, y is feasible for (8) and a contradiction is obtained. □

According to Theorem 4, whenever $\mu \leqq 0$, (8) can be solved by solving the LP (19). In case $\mu \in \mathbb{R}^p \setminus \mathbb{R}^p_{\leqq}$, an optimal solution to (8) must be obtained at an incomplete vertex of $\mathcal{P}$. Algorithm 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 [15] 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 avoid the enumeration of all vertices of $\mathcal{P}$.

Theorem 5 also explains why optimising over the non-dominated set of a bi-objective linear programme is easy. In the case $p = 2$ the extreme points of $\mathcal{P}$ can be ordered according to increasing values of one objective and therefore decreasing order of the other objective. It follows that $\mathcal{Y}$ has exactly two incomplete non-dominated extreme points (in Figure 1 these are a and d). These two extreme points are the non-dominated extreme points obtained by lexicographically optimising the objectives in the order (1,2) and (2,1) respectively. Hence for $p = 2$ objectives, (8) can be solved by linear programming, solving two lexicographic LPs or four single objective LPs.

5.2. *The primal algorithm for solving* (8)

Algorithm 1 has two distinct phases, vertex generation and vertex evaluation. Combining both phases and exploiting the results of Section 5.1 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 inequality representation of $\mathcal{P}$, a set of new extreme points of the outer approximating polyhedron $\mathcal{S}^i$ is found. Evaluating $\mu^T y$ at these extreme points, we select the one with the best function value as s^{i+1} in the next iteration. If the selected point is infeasible, we proceed with adding a cut $\left\{y \in \mathbb{R}^p : \sum_{k=1}^{p} \lambda_k y_k \geqq b^T u\right\}$ as in the revised version of Benson's algorithm. We call this an improvement cut. Otherwise we have a feasible solution to (8) and therefore a lower bound on its optimal value and we add what we call a threshold cut. 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 2.

Algorithm 2 Primal algorithm for solving (8).

Input: A, b, C, μ
Output: y^*, an optimal solution to (8)

Compute the optimal value y_k^I of $(P_1(e^k))$, for $k = 1, \ldots, p$.
Set $\mathcal{S}^0 := \left\{y^I\right\} + \mathbb{R}^p_{\geqq}$, $i := 1$ and $V_{\mathcal{S}^0} := \left\{y^I\right\}$.
Threshold := False.
while $V_{\mathcal{S}^{i-1}} \not\subset \mathcal{P}$ **do**
  $s^i \in \operatorname{argmax}\left\{\mu^T y : y \in V_{\mathcal{S}^{i-1}}\right\}$,
  if $s^i \in \mathcal{P}$ and Threshold = False **then**
    $\mathcal{S}^i := \mathcal{S}^{i-1} \cap \{y \in \mathbb{R}^p : \mu^T y \geqq \mu^T s^i\}$. Update $V_{\mathcal{S}^i}$. Threshold := True.
  else
    Compute an optimal solution (u^i, λ^i) of $\mathbb{D}_2(s^i)$.
    Set $\mathcal{S}^i := \mathcal{S}^{i-1} \cap \left\{y \in \mathbb{R}^p : \sum_{k=1}^{p} \lambda_k^i y_k^i \geqq b^T u^i\right\}$. Update $V_{\mathcal{S}^i}$. Threshold := False.
  end if
  Set $i := i + 1$.
end while
$y^* := s^i$.

Example 4 The primal algorithm is illustrated in Figure 5 by maximising $y_1 + y_2$ over the non-dominated set of Example 1.

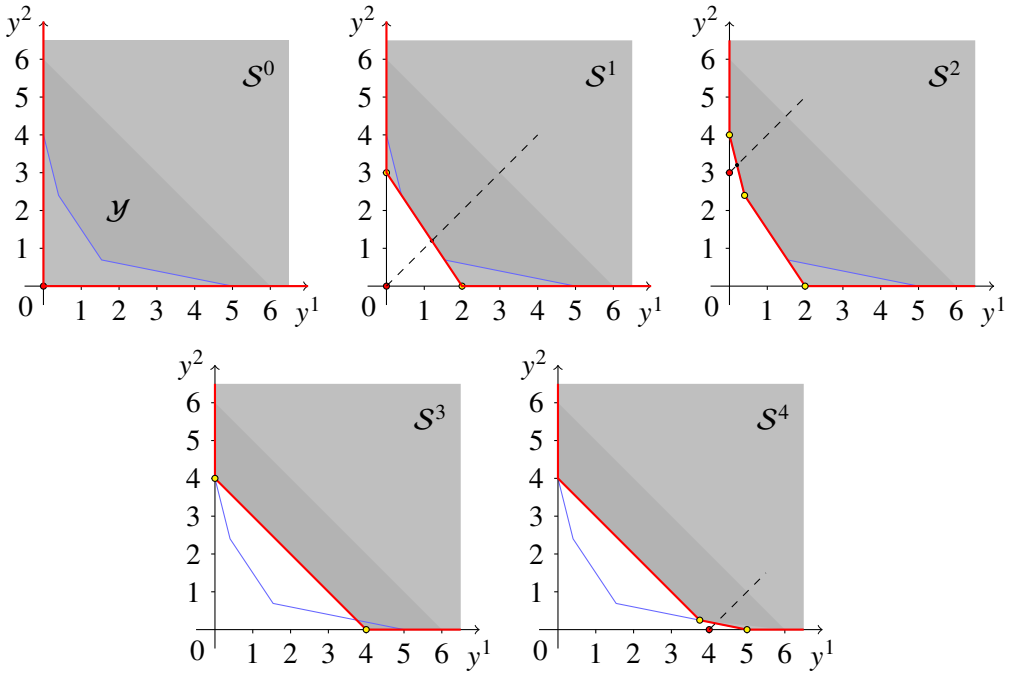

Figure 5. Iterations of the primal algorithm 2 in Example 4.

Table 1. Iterations of the primal algorithm 2 in Example 4.

i	Vertex s^i	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), (3.75,0.25)	5

Figure 5 and Table 1 show in each iteration the extreme point chosen, the type of cut added and the incumbent 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.

5.3. *The primal algorithm for solving* (10)

A naïve algorithm for (approximately) solving (10) is to obtain a set of ϵ-non-dominated points of $\mathcal{P}$ (vertices of $\mathcal{S}^i$) through the algorithm of [22] and to determine which one has the largest value of $\mu^T y$, in the same way as in Algorithm 1.

Now we extend the primal Algorithm 2 to solve (10). The algorithm starts with a polyhedron $\mathcal{S}^0$ containing $\mathcal{P}$. In each iteration a hyperplane is generated and added to the inequality representation of $\mathcal{S}^{i-1}$ resulting in a set of new 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 $\left\{y \in \mathbb{R}^p : \sum_{k=1}^{p} \lambda_k^i y_k^i \geqq b^T u^i\right\}$ as in the algorithm of [22] for convex MOPs 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. A threshold cut removes the region where points are worse than $\hat{y}$. At the end of the algorithm, an ϵ-non-dominated point y^* is obtained, which is an optimal solution to (10). Furthermore, By solving $(\mathbb{P}_2(y^*))$, we can find an element $\hat{y}$ of $\mathcal{P}_N$, which is an approximate solution to (9). We also know that $y^* \leqq \hat{y} \leqq y^* + \epsilon e$, where ϵ is the approximation error predetermined by the decision maker. As a result, we have solved (10) exactly and determined an approximate solution $\hat{y}$ to (9). This primal algorithm is detailed in Algorithm 3.

Algorithm 3 Primal algorithm for solving (10).

Input: f, g, μ, ϵ

Output: y^*, an optimal solution to (10) and $\hat{y}$, an approximately optimal solution to (9)

1: Compute the optimal value y_k^I of $(P_1(e^k))$, for $k = 1, \ldots, p$.
2: Set $\mathcal{S}^0 := \{y^I\} + \mathbb{R}^p_{\geqq}$ and $i := 1$.
3: Threshold = False.
4: **while** $V_{\mathcal{S}^{i-1}} \not\subset \mathcal{P}_{\epsilon N}$ **do**
5: $s^i \in \operatorname{argmax}\{\mu^T y : y \in V_{\mathcal{S}^{i-1}}\}$.
6: **if** $s^i \in \mathcal{P}_{\epsilon N}$ and if Threshold = False **then**
7: $\mathcal{S}^i := \mathcal{S}^{i-1} \cap \{y \in \mathbb{R}^p : \mu^T y \geqq \mu^T s^i\}$. Threshold := True.
8: **else**
9: Compute an optimal solution (u^i, λ^i) of $\mathbb{D}_2(s^i)$.
10: Set $\mathcal{S}^i := \mathcal{S}^{i-1} \cap \left\{y \in \mathbb{R}^p : \sum_{k=1}^{p} \lambda_k^i y_k^i \geqq b^T u^i\right\}$. Threshold = False. Update $V_{\mathcal{S}^i}$.
11: **end if**
12: Set $i := i + 1$.
13: **end while**
14: $y^* \in \operatorname{argmax}\{\mu^T y : y \in V_{\mathcal{S}^{i-1}}\}$.
15: Find an optimal solution $\hat{x}$ to $\mathbb{P}_2(y^*)$. An approximately optimal solution to (10) is $\hat{y} = f(\hat{x})$.

Example 5 We illustrate Algorithm 3 through maximising $5y_1 + 7y_2$ over the non-dominated set of the CMOP of Example 2.

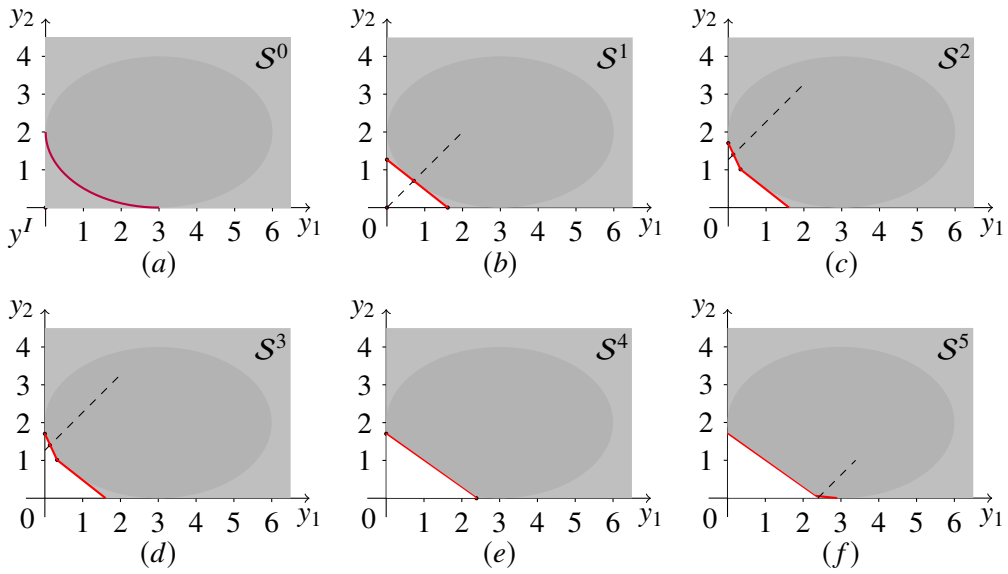

Figure 6. Iterations of Algorithm 3 in Example 5.

Table 2. Iterations of Algorithm 3 in Example 5.

i	Vertex s^i	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)	14.51

Figure 6 and Table 2 show the iterations of Algorithm 3. 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 $y^* = (2.9, 0)^T$, is found after another improvement cut has been generated. By solving $\mathbb{P}_2(y^*)$ we obtain an approximately optimal solution $(2.9001, 0.001)^T$ and objective function value 14.5075.

6. The Dual Algorithms

In this section, we introduce dual algorithms to solve (8) and (10) in dual objective space. The dual algorithms are derived from dual variants of Benson's algorithm [18, 22]. We commence the discussion with some properties of (8) in dual objective space. We then introduce the dual algorithms for solving (8) and (10).

6.1. *The dual algorithm for solving* (8)

In this section, we explore some properties of (8) in dual objective space, which enable us to solve (8) more efficiently than by the primal algorithm of Section 5. We call the resulting algorithm 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,

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

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

$$H_{y^{ex}} = \left\{y \in \mathbb{R}^p : \mu^T y = \mu^T y^{ex}\right\}. \tag{20}$$

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 (20) as

$$H_{\hat{y}^{ex}} = \left\{\hat{y} \in \mathbb{R}^p : \hat{\mu}^T \hat{y} = \hat{\mu}^T \hat{y}^{ex}\right\}. \tag{21}$$

Let us now define $\mu' := \sum_{i=1}^{p} \mu_i$ and divide both sides of the equation in (21) by μ' to obtain

$$H_{y^{ex}} = \left\{\hat{y} \in \mathbb{R}^p : \frac{\hat{\mu}^T \hat{y}}{\mu'} = \frac{\hat{\mu}^T \hat{y}^{ex}}{\mu'}\right\}. \tag{22}$$

Since (22) 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 (12), 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 (8) at y^{ex} divided by μ'. Hence, geometrically, (8) is equivalent to finding a point v^μ with the largest last element along L^μ.

THEOREM 6 *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 hyperplane $H^*(y^{ex})$ 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 5, we reached the conclusion that an optimal solution to (8) 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 develop this idea through the association between the upper image $\mathcal{P}$ of the primal MOLP and the lower image $\mathcal{D}$ of the dual MOLP, and exploit it to propose the dual algorithm.

In the discussion that follows, we make use of Theorem 2. Let $y^i, i = 1, \ldots, r$ be the extreme points of $\mathcal{P}$. Then we know that the facets of $\mathcal{D}$ are $F^i := \Psi^{-1}(y^i) \cap \mathcal{D}$ for $i = 1, \ldots, r$.

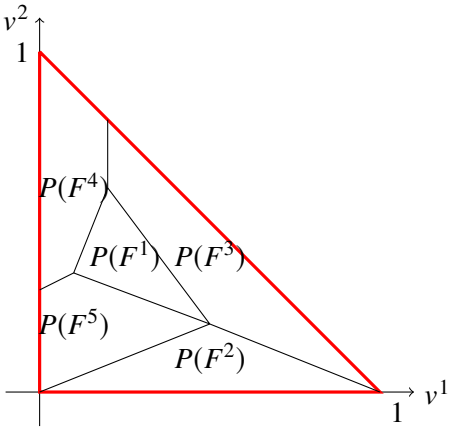

Figure 7. Projection of a three dimensional lower image $\mathcal{D}$ onto the v^1-v^2 coordinate plane.

Figure 7 shows the projection of a three dimensional lower image $\mathcal{D}$ onto the v^1 and v^2 coordinate plane. The cells $P(F^1)$ to $P(F^5)$ are the projections of facets F^1 to F^5 of $\mathcal{D}$, respectively. Notice that the projections of the facets have disjoint interiors, because by definition $\mathcal{D}$ consists of points $(v_1, \ldots, v_{p-1}, v_p)$ such that $v_p \leq v^*$ for the point $(v_1, v_{p-1}, v^*) \in \mathcal{D}_{\mathcal{K}}$, the $\mathcal{K}$-maximal subset of $\mathcal{D}$.

Definition 6 *$\mathcal{K}$-maximal facets F^i and F^j of $\mathcal{D}$ are called neighbouring facets if $dim(F^i \cap F^j) = p - 2$.*

Figure 7 shows that the neighbouring facets of facet F^2 are F^3 and F^5. Neither F^1 nor F^4 are neighbouring facets of F^2.

Proposition 5 *If y^i and y^j are 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. The edge $[y^i - y^j]$ has dimension one and according to Theorem 3 $dim([y^i - y^j]) + dim(\Psi^{-1}([y^i - y^j])) = p - 1$ we have that $dim(\Psi^{-1}([y^i - y^j])) = p - 2$. Moreover, $\Psi^{-1}(y^i) = H^*(y^i) \cap \mathcal{D} = F^i$ and $\Psi^{-1}(y^j) = H^*(y^j) \cap \mathcal{D} = F^j$ due to our notational convention. Hence $dim(F^i \cap F^j) = p - 2$ and F^i and F^j are neighbouring facets. □

Definition 7 *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}.*

In Figure 3 there are two incomplete facets, namely, the facet attached to the origin and the facet attached to the point $(1, 0)^T$. The other two facets are complete. In Figure 7 the projections of the incomplete facets are $P(F^2)$, $P(F^3)$, $P(F^4)$ and $P(F^5)$. The only complete facet is F^1, hence $P(F^1)$ is surrounded by the projections of the incomplete facets.

Theorem 7 *There exists a one-to-one correspondence between incomplete facets of $\mathcal{D}$ and incomplete vertices of $\mathcal{P}$.*

Proof. Proposition 5 states that the facets of $\mathcal{D}$ have the same neighbouring relations as the vertices of $\mathcal{P}$, which implies that the completeness of a facet of $\mathcal{D}$ remains the same as that of its corresponding vertex of $\mathcal{P}$. Therefore, Theorem 7 is true. □

THEOREM 8 *If y^* is an optimal solution of* (8) *at an incomplete vertex, then $(H^*(y^*) \cap \mathcal{D})$ is a $\mathcal{K}$-maximal incomplete facet of $\mathcal{D}$.*

Proof. Theorem 8 follows directly from Theorem 5 and Theorem 7. □

Theorem 8 says that a facet of $\mathcal{D}$ corresponding to an optimal extreme point solution to (8) is an incomplete facet. In other words, we do not neeed to investigate complete facets to find an optimal solution because in the primal 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 (8). 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 (8) can be obtained by investigating the incomplete facets of $\mathcal{D}$. In order to obtain the set of incomplete facets of $\mathcal{D}$, let us define

$$W_D := \bigcup_{i=1}^{p-1} \left\{ v \in \mathbb{R}^p : v_i = 0, 0 \leqq v_j \leqq 1, j = 1 \ldots (p-1), j \neq i \right\}$$
$$\cup \left\{ v \in \mathbb{R}^p : \sum_{i=1}^{p-1} v_i = 1, 0 \leqq v_i \leqq 1, i = 1 \ldots (p-1) \right\}.$$

In Fig. 7, 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 proposed below is designed to solve (8) in the dual objective space through finding facets that intersect with W_D.

Theorem 7 shows that there is a one-to-one correspondence between incomplete vertices of $\mathcal{Y}$ and incomplete facets of $\mathcal{D}$. But through intersections with W_D incomplete facets of $\mathcal{D}$ are easier to characterise than incomplete facets of $\mathcal{Y}$ and can therefore be handled algorithmically.

Algorithm 4 Dual algorithm for solving (8).

Input: A, b, C, μ
Output: y^*, an optimal solution to (8)
  Choose some $\hat{d} \in \text{int}\mathcal{D}$.
  Compute an optimal solution x^0 of $(P_1(\hat{d}))$, set $y^* := Cx^0$, $M^* := \mu^T y^*$.
  Set $\mathcal{S}^0 := \left\{ v \in \mathbb{R}^p : \lambda(v) \geqq 0, \sum_{k=1}^{p-1} (y_k^* - y_p^*) v_k - v_p \geqq -y_p^* \right\}$ and $i := 1$.
  while $W_D \cap V_{\mathcal{S}^{k-1}} \not\subset \mathcal{D}$ **do**
    Choose $v^i \in W_D \cap V_{\mathcal{S}^{i-1}}$ such that $v^i \notin \mathcal{D}$.
    Compute an optimal solution x^i of $(P_1(v^i))$, set $y^i := Cx^i$.
    if $M^* < \mu^T y^i$ **then**
      Set $y^* := y^i$ and $M^* := \mu^T y^i$.
    end if
    Set $\mathcal{S}^i := \mathcal{S}^{i-1} \cap \left\{ v \in \mathbb{R} : \sum_{k=1}^{p-1} (y_k^i - y_p^i) v_i - v_p \geqq -y_p^i \right\}$. Update $V_{\mathcal{S}^i}$.
    Set $i := i + 1$.
  end while

Example 6 Figure 8 below illustrates the dual algorithm by maximising $y_1 + y_2$ over the non-dominated set of Example 1. By employing the dual algorithm only two of the four

$\mathcal{K}$-maximal facets need to be generated.

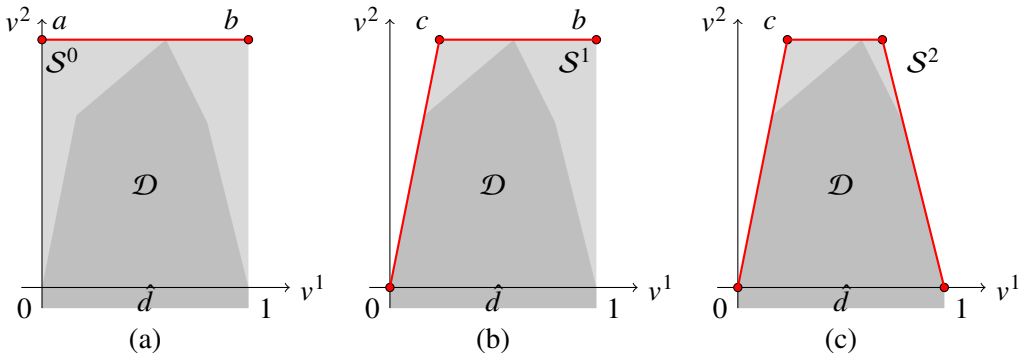

Figure 8. Iterations of Algorithm 4 in Example 6.

In this example, $W_D = \{v \in \mathbb{R}^2 : v_1 = 0\} \cup \{v \in \mathbb{R}^2 : v_1 = 1\}$. In Figure 8 (a), $a, b \in W_D$. In Figure 8 (b), vertex a is used to generate a supporting hyperplane of $\mathcal{D}$. This hyperplane corresponds to extreme point $(0, 4)^T$ in the primal space. A new vertex c is found. Since $c \notin W_D$, in Figure 8 (c) vertex b is employed to generate another supporting hyperplane of $\mathcal{D}$ 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 solution $(5, 0)^T$.

6.2. *The dual algorithm for solving* (10)

In Section 6.1, we investigated the properties of (8) in the dual objective space and proposed a dual algorithm to solve (8). The dual Algorithm 4 for (8) determines incomplete $\mathcal{K}$-maximal facets of $\mathcal{D}$. But since the underlying dual MOP (16) of (10) has a non-polyhedral lower image $\mathcal{D}$ it may have no facets (see Figure 4 for example). However, since the dual variant of Benson's outer approximation algorithm for solving CMOPs performs an outer approximation to $\mathcal{D}$ we can combine this approximation of $\mathcal{D}$ with a similar procedure as in Algorithm 4 to solve (10).

At termination all extreme points of the approximation polyhedron are $\epsilon\mathcal{K}$-maximal. The dual algorithm proposed below is designed to generate all incomplete facets of the polyhedron approximating $\mathcal{D}$ in dual objective space. Moreover, in every iteration, the dual algorithm computes a feasible solution to problem (9) by solving $(\mathbb{P}_1(s^i))$. Since it keeps track of the best solution, at termination, y^* is an optimal solution to (10), which is also a feasible approximate solution to (9) without ever computing an outer approximating polyhedron of $\mathcal{P}$.

Algorithm 5 Dual algorithm for solving (10).

Input: f, g, μ, ϵ

Output: y^*, an optimal solution to (10) and approximate solution to (9).

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

Compute an optimal solution x^0 of $(\mathbb{P}_1(\hat{d}))$, set $y^* := f(x^0)$, $M^* := \mu^T y^*$.

Set $\mathcal{S}^0 := \left\{ v \in \mathbb{R}^p : \lambda(v) \geqq 0, \sum_{k=1}^{p-1} (y_k^* - y_p^*)v_k - v_p \geqq -y_p^* \right\}$ and $i = 1$.

while $W_D \cap V_{\mathcal{S}^{i-1}} \not\subset \mathcal{D}_{\epsilon\mathcal{K}}$ **do**

  Choose $s^i \in W_D \cap V_{\mathcal{S}^{i-1}}$.

  Compute an optimal solution x^i of $(\mathbb{P}_1(s^i))$ and an optimal value z^i to $(\mathbb{P}_1(s^i))$; $y^i := f(x^i)$; $M^i := \mu^T y^i$.

  if $M^* < M^i$ **then**

    $y^* := y^i$ and $M^* := M^i$.

  end if

  if $s_p^i - z^i > \epsilon$ **then**

    Set $\mathcal{S}^i := \mathcal{S}^{i-1} \cap \left\{ v \in \mathbb{R} : \sum_{k=1}^{p-1} (y_k^i - y_p^i)v_i - v_p \geqq -y_p^i \right\}$. Update $V_{\mathcal{S}^i}$.

  else

    $V_{\epsilon\mathcal{K}} := V_{\epsilon\mathcal{K}} \cup s^i$.

  end if

  Set $i := i + 1$.

end while

Example 7 Figure 9 below illustrates the dual algorithm in Example 5.

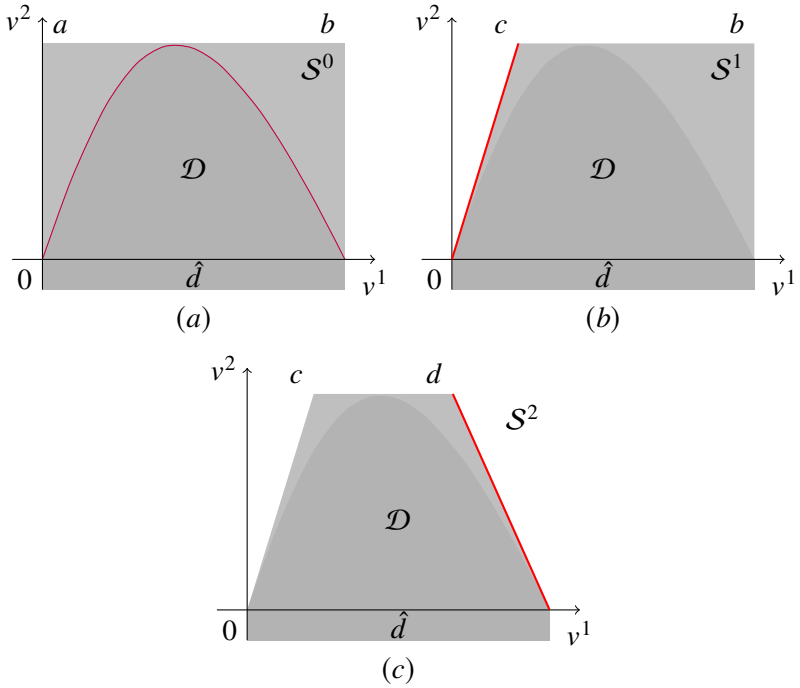

Figure 9. Iterations of Algorithm 5 in Example 7.

In Figure 9 (a), an initial polyhedron $\mathcal{S}^0$ is constructed with two vertices $a, b \in W_D$. In the first iteration, vertex a is selected to generate a supporting hyperplane to $\mathcal{D}$ at $(0, 0)^T$ as shown in Figure 9 (b). Now a new vertex c is found. In Figure 9 (c) vertex b is employed

to generate another supporting hyperplane to $\mathcal{D}$ at $(1,0)^T$ resulting in vertex d. Since $c, d \notin W_D$, there is no infeasible vertex belonging to W_D and the algorithm terminates with the approximate solution $y^* = (3,0)^T$ with value $\mu^T y^* = 15$. This is the optimal solution in this example.

7. Computational Results

7.1. *The linear case*

In this section, we use randomly generated instances to compare some of the algorithms for solving (8). The method proposed by [7] is used to generate instances the coefficients of which 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 a computer with an Intel i7 processor (3.40GHz and 16GB RAM). We solved three instances of the same size for MOLPs with p objectives, m constraints and n variables. Note that $m = n$ for all instances. Table 3 below shows the average CPU times (in seconds). We tested six algorithms, namely

- the brute force algorithm (Algorithm 1), labelled A1
- the bi-objective branch and bound algorithm of Section 4.1, labelled A2
- the conical branch and bound algorithm of Section 4.2, labelled A3
- Benson's branch and bound algorithm (Section 4.3), labelled A4
- the primal algorithm (Algorithm 2), labelled A5
- the dual algorithm (Algorithm 4), labelled A6

Table 3. CPU times of six different algorithms in the linear case.

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 minute 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. The dimension factor maybe plays a less important role when p is small. Other factors such as the number of variables and constraints may have more influential impact on CPU times. 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. The brute force algorithm (A1) performs better than the branch and bound algorithms (A3 and A4) because A1 only solves one LP in each iteration whereas A3 and A4 solve multiple LPs. Solving LPs is the most time-consuming step in the algorithms. Table 3 also shows the dual algorithm performs better than the primal algorithm in solving instances of large scale. In Figure 7.1, 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.

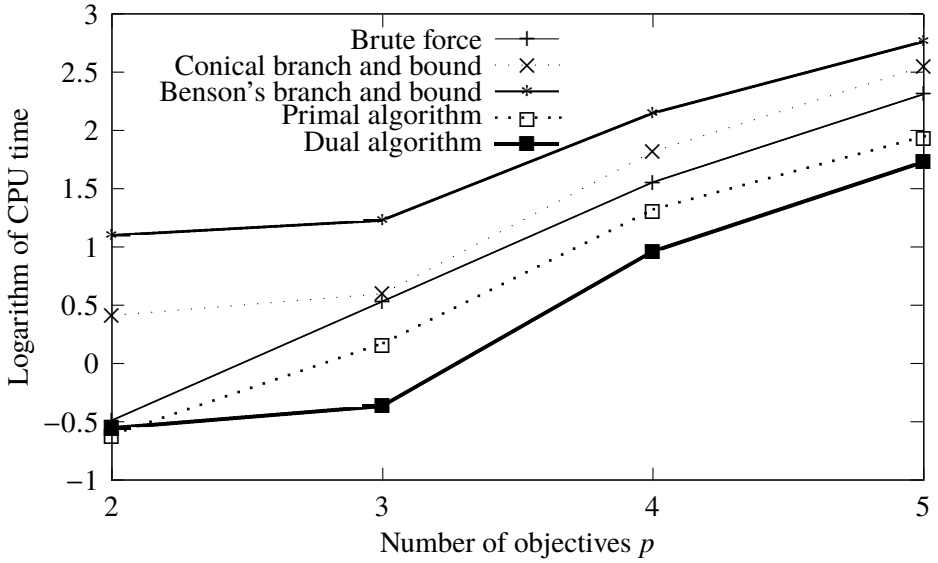

Figure 10. Log-transformed CPU times for instances with 500 variables and constraints.

7.2. *The convex case*

In this section, we use randomly generated instances to compare some of the algorithms discussed in Section 4 and the primal and the dual algorithms for (10). The method proposed by [7] 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 value of ϵ was set to 10^{-4}. The experiments were run on a computer with Intel i7 processor (3.40GHz, 16GB RAM). Table 4 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, labelled A1
- the extended bi-objective branch and bound algorithm in Section 4.1, labelled A2
- the conical branch and bound algorithm of Section 4.2, labelled A3
- the primal algorithm (Algorithm 3), labelled A4
- the dual algorithm (Algorithm 5), labelled A5

Table 4. CPU times of five different algorithms in the convex case.

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 to the chosen accuracy. We notice that the number of objective functions is a crucial factor. All of the instances with $p = 2$ objectives can be solved within 30 seconds. The most efficient algorithm is the extended bi-objective branch and bound algorithm (A2) proposed by [21], which solves the largest instances with 100 variables and constraints within 2 seconds. Unfortunately it is specific to the bi-objective case. 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 substantially so that even instances with 5 variables and constraints take a few minutes to solve. Furthermore, the largest instances with 100 variables and constraints, take a few hours. The dual algorithm solves the largest instances in half of the time used by the conical branch and bound algorithm (A3). We also notice that the dual algorithm is faster than the primal one 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 algorithm enumerates a large number of vertices which are redundant. This also proves the advantage of the techniques employed in our new algorithms. Additionally, in the implementation of the algorithms, we notice that time required to solve instances is sensitive to the approximation accuracy (reflected by ϵ as stated in the algorithms). In this test the level of accuracy is 10^{-4}. It is expected that a lower level of accuracy results in faster solution time.

8. Conclusion

Optimisation over the efficient set is a problem of concern when decision makers have to select a preferred point among an infinite number of efficient solutions in multiple criteria decision making. We have addressed the case that this selection is based on the optimisation of a linear function over the non-dominated set of a linear multi-objective optimisation problem. 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 to reduce the need for complete enumeration of all non-dominated extreme points. 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.

We also extended the primal and the dual algorithms to optimise a linear function over the non-dominated set of a convex multi-objective optimisation problem. We employed the techniques of the linear primal and dual methods to facilitate the non-linear ones. Comparing with other algorithms from the literature, our primal and dual algorithms are the fastest. In the future we plan to address more challenging versions of this problem, e.g. when the function to be optimised is nonlinear and when the underlying MOP is nonconvex, where we are particularly interested in the discrete case.

Acknowledgments and Data Statement

We express our gratitude to two anonymous referees, whose careful reading and comments helped us improve the paper. This research was supported by grant FA8655-13-1-3053 of the US Airforce Office for Scientific Research. More information on the data can be found at doi `https://dx.doi.org/10.17635/lancaster/researchdata/224`.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] Benson, H. P. (1991). An all-linear programming relaxation algorithm for optimizing over the efficient set. *Journal of Global Optimization*, 1(1):83–104.

[7] 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.

[8] 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.

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

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

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

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

[13] 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.

[14] Ehrgott, M. and Wiecek, M. (2005). Mutiobjective programming. In *Multiple Criteria*

Decision Analysis: State of the Art Surveys, volume 78 of *International Series in Operations Research and Management Science*, pages 667–708. Springer New York.

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

[16] Fülöp, J. (1993). On the equivalency between a linear bilevel programming problem and linear optimization over the efficient set. Technical report, Hungarian Academy of Sciences.

[17] 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.

[18] 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.

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

[20] Horst, R. and Tuy, H. (1993). *Global Optimization: Deterministic Approaches*. Springer, Berlin.

[21] 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.

[22] 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.

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

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

[25] Yamamoto, Y. (2002). Optimization over the efficient set: overview. *Journal of Global Optimization*, 22(1-4):285–317.

[26] 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.