Optimizing a linear function over the nondominated set of multiobjective integer programs

Banu Lokman*

Portsmouth Business School, University of Portsmouth, Richmond Building, Portland Street, Portsmouth PO1 3DE, United Kingdom
E-mail: banulokman@gmail.com

Abstract

In this paper, we develop two algorithms to optimize a linear function over the nondominated set of multi-objective integer programs. The algorithms iteratively generate nondominated points and converge to the optimal solution reducing the feasible set. The first algorithm proposes improvements to an existing algorithm employing a decomposition and search procedure in finding a new point. Differently, the second algorithm maximizes one of the criteria throughout the algorithm and generates new points by setting bounds on the linear function value. The decomposition and search procedure in the algorithms is accompanied by problem-specific mechanisms in order to explore the objective function space efficiently. The algorithms are designed to produce solutions that meet a prespecified accuracy level. We conduct experiments on multi-objective combinatorial optimization problems and show that the algorithms work well.

Keywords: multiple objective programming; integer programming; nondominated set; combinatorial optimization.

1. Introduction

Integer programming is used to model decision problems in several branches of industry and society. Since many of these problems involve multiple criteria of conflicting nature, Multi-objective Integer Programs (MOIPs) have applications in a wide variety of areas including production planning and scheduling, logistics and capital budgeting (see Ehrgott and Gandibleux, 2002, 2004). In MOIPs, there does not exist a single solution that simultaneously optimizes all objective functions, and hence the aim is to generate nondominated points. For each such point, an improvement in one of the objectives is not possible without sacrificing from at least one other objective. There exist studies that try to generate all nondominated points for MOIPs but the generation of the whole set of nondominated points is computationally very demanding in general and not useful from practical point of view (see Özlen and Azizoğlu, 2009;

*Corresponding author (e-mail: banulokman@gmail.com).

Lokman and Köksalan, 2013; Kirlik and Sayın, 2014; Zhang and Reimann, 2014; Dächert and Klamroth, 2015; Boland et al., 2016). Ehrgott et al. (2016) review exact methods for multi-objective combinatorial optimization (MOCO) problems that are special MOIPs.

MOIPs have typically many nondominated points that cannot be considered superior to each other without the preference input of a decision maker (DM). Recently, algorithms are developed to generate a compromise solution that is preferred to all other solutions in the nondominated set. Optimizing a linear function over the nondominated set of a MOIP is one approach to articulating preferences of the DM. Based on a reference point and direction specified by the DM, a compromise solution can be found without generating the whole set of nondominated points. By changing the direction, it is also possible to generate or approximate the nondominated points that lie on the boundary of the convex hull of the nondominated set. These extreme points could be used not only to improve computational efficiency of other algorithms but also to estimate the locations of the nondominated points and characterize the nondominated set.

The problem of optimizing a linear function over the nondominated set of a multi-objective linear program (MOLP) has been widely studied (see Benson, 1984; Ecker and Song, 1994; White, 1996; Benson and Sayin, 1994; Jorge, 2005) while the studies for MOIPs are limited. As discussed in Benson (1984) and Ecker and Song (1994), the problem is even difficult for MOLPs because the efficient set may be nonconvex. As an application of this problem to MOIPs, Nguyen (1992) proposes an upper bound for the optimal objective function value. Abbas and Chaabane (2006) develop a method for optimizing a linear function over the efficient set without finding all nondominated points. The method is based on the works of Benson (1984) and Ecker and Song (1994) that are developed for MOLPs. The method of Abbas and Chaabane (2006) iteratively reduces the feasible region and guarantees to obtain an improvement in the objective value by introducing different types of cuts at each iteration. It stops when the reduced feasible region contains no integer feasible solution. However, Abbas and Chaabane (2006) do not present a computational study.

Jorge (2009) develops an exact algorithm to optimize a linear function over the efficient set of a MOIP. The algorithm solves a sequence of progressively more constrained single-objective integer problems that eliminate the dominated points from further consideration. The performance of the algorithm is tested on randomly generated instances of MOIPs. Boland et al. (2017) modify the algorithm of Jorge (2009) employing a decomposition and search procedure and the computational study shows that the new algorithm outperforms the algorithm of Jorge (2009). The algorithm is also applied to find the nadir point that is composed of the worst values of each criterion over the nondominated set. Köksalan and Lokman (2015) and Kirlik and Sayın (2015) also develop exact algorithms to find the nadir point. Although the problem of finding the nadir value for a criterion is equivalent to minimizing that criterion over the nondominated set, these algorithms employ the special characteristics of the nadir point and need extensions to optimize a linear function.

In this paper, we present two algorithms that optimize any given linear function over the nondominated set of a MOIP. The algorithms iteratively generate nondominated points and reduce the feasible set by excluding not only the dominated regions but also the inferior regions with respect to the linear function. At each iteration, the reduced feasible set is decomposed into subsets and a search is conducted over all these subsets to find a new point. While the first algorithm searches for the point that maximizes the linear function as in the algorithms of Jorge (2009) and Boland et al. (2017), the second algorithm chooses one of the criteria based on the linear function and finds the point maximizing that criterion. In

contrast with the literature, the decomposition and search procedure is accompanied by problem-specific mechanisms in order to explore the objective function space efficiently. Furthermore, the algorithms are designed as approximation algorithms with performance guarantee on the quality of the solution.

The organization of the paper is as follows. In Section 2, we give definitions and provide background information. We develop the requisite theory and the algorithms in Section 3 and present the results of our computational experiments in Section 4. Lastly, we make conclusions in Section 5.

2. Background

A multi-objective integer program with p objectives is defined as:

$$
\begin{aligned}
&[MOIP]: \\
&\text{“Max”} \quad z(\mathbf{x}) = (z_1(\mathbf{x}), z_2(\mathbf{x}), \ldots, z_p(\mathbf{x})) \\
&\text{subject to} \\
&\mathbf{x} \in \mathbf{X}
\end{aligned}
$$

In $[MOIP]$, $\mathbf{x} \in \mathbb{Z}^n$ is the integer decision vector, and $\mathbf{X}$ denotes the feasible space. Point $z(\mathbf{x}) \in \mathbf{Z}$ is the image of $\mathbf{x}$ in the objective function space where $\mathbf{Z}$ denotes the feasible set in the objective function space. The quotation marks are used since there does not exist a unique solution that simultaneously maximizes all objective functions.

Definition 1. Let $\mathbf{x}^1, \mathbf{x}^2 \in \mathbf{X}$. $z(\mathbf{x}^1)$ dominates $z(\mathbf{x}^2)$ if $z_i(\mathbf{x}^1) \geq z_i(\mathbf{x}^2)$ for all $i = 1, \ldots, p$ and $z_j(\mathbf{x}^1) > z_j(\mathbf{x}^2)$ for at least one $j \in \{1, \ldots, p\}$.

Definition 2. A point $\mathbf{x} \in \mathbf{X}$ is efficient if there does not exist any $\mathbf{x}' \in \mathbf{X}$ such that $z(\mathbf{x}')$ dominates $z(\mathbf{x})$. Then, $z(\mathbf{x}) \in \mathbf{Z}$ is said to be nondominated.

The set of all nondominated points for a MOIP is called the nondominated set and denoted as $\mathbf{Z}^N$. The problem of optimizing a linear function over $\mathbf{Z}^N$ is formulated as follows:

$$
\begin{aligned}
&[MOIP(v)]: \\
&\text{Max} \quad v(\mathbf{z}) = \sum_{i=1}^{p} v_i z_i(\mathbf{x}) \\
&\text{subject to} \\
&\mathbf{x} \in \mathbf{X} \\
&z(\mathbf{x}) \in \mathbf{Z}^N
\end{aligned}
$$

In $[MOIP(v)]$, v is a linear function of the objectives. We denote the optimal objective function value and the corresponding nondominated point by v^* and $\mathbf{z}^*$, respectively. Since $\mathbf{Z}^N$ is implicitly defined and not known a priori, it is not straightforward to find v^*, when $v(\mathbf{z})$ is not a strictly positive linear combination of the objectives, $v_i \leq 0$ for some $i \in \{1, \ldots, p\}$.

3. Development of Algorithms

The algorithm proposed by Jorge (2009) optimizes a linear function over the efficient set of a MOIP and is also applicable to solving $[MOIP(v)]$. The algorithm starts with solving problem $\text{Max}_{z(\mathbf{x})\in\mathbf{Z}}\, v(\mathbf{z})$ that may yield an inefficient solution. If this is the case, a new solution dominating the optimal solution is found. In the succeeding iterations $n > 1$, the algorithm solves a model, $[P^n]$, where previously-generated nondominated points, $\mathbf{Z}_n = \{\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^{n-1}\}$, are excluded from the feasible set in order to generate a new point.

$$
\begin{aligned}
&[P^n]: \\
&\text{Max} \quad v(\mathbf{z}) = \sum_{i=1}^{p} v_i z_i(\mathbf{x}) \\
&\text{subject to} \\
&z_i(\mathbf{x}) \geq (z_i^t + \epsilon) y_i^t + M_i(1 - y_i^t) \quad i = 1, \ldots, p, \quad t = 1, \ldots, n-1 \\
&\sum_{i=1}^{p} y_i^t = 1 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad t = 1, \ldots, n-1 \\
&y_i^t \in \{0, 1\} \qquad\qquad\qquad\qquad\qquad\qquad i = 1, \ldots, p, \quad t = 1, \ldots, n-1 \\
&\mathbf{x} \in \mathbf{X}
\end{aligned}
$$

In $[P^n]$, M_i is a lower bound for criterion i and ϵ is sufficiently small positive constant. Let $\mathbf{z}'^n = (z_1'^n, z_2'^n, ..., z_p'^n)$ be the point corresponding to the optimal solution of $[P^n]$. For each previously-generated nondominated point $\mathbf{z}^t$, y_i^t is a binary variable enforced to take the value of 1 for some criterion i, $z_i(\mathbf{x}) \geq (z_i^t + \epsilon)$. Therefore, $[P^n]$ guarantees to generate a new point that is not dominated by any of $\mathbf{z}^t$, $t = 1, \ldots, n-1$. The algorithm next solves a model, $[E^n]$, to check whether $\mathbf{z}'^n$ is nondominated or not.

$$
\begin{aligned}
&[E^n]: \\
&\text{Max} \quad \sum_{i=1}^{p} s_i \\
&\text{subject to} \\
&z_i(\mathbf{x}) - s_i = z_i'^n \quad i = 1, \ldots, p \\
&\mathbf{x} \in \mathbf{X}
\end{aligned}
$$

$[E^n]$ generates a nondominated point, $\mathbf{z}^n$, that is either equal to $\mathbf{z}'^n$ or dominates $\mathbf{z}'^n$. Since $\mathbf{z}'^n, \mathbf{z}^n \in \mathbf{Z}$ and $\mathbf{z}^n \in \mathbf{Z}^N$, $v(\mathbf{z}^n) \leq v^* \leq v(\mathbf{z}'^n)$. Thus, the algorithm stops when $v(\mathbf{z}^n) = v(\mathbf{z}'^n)$. Otherwise, the algorithm defines $\mathbf{Z}_{n+1} = \mathbf{Z}_n \cup \{\mathbf{z}^n\}$ and solves $[P^{n+1}]$.

3.1. DSA

The algorithm of Jorge (2009) solves progressively more constrained models in order to generate new points that are not dominated by the existing nondominated points. To reduce the computational burden, we develop a Decomposition and Search Algorithm (DSA) that partitions the feasible set of $[P^n]$ into subsets and searches for an optimal solution over all these subsets. We also develop mechanisms and introduce cuts in order to increase the computational efficiency. We next discuss these mechanisms and then present the algorithm.

3.1.1. Decomposition Scheme of DSA

At each iteration $n > 1$, $[P^n]$ generates the point that maximizes function v and is not dominated by previously-generated points in $\mathbf{Z}_n$. In DSA, we partition the feasible criterion space into subsets that are formed by imposing bounds on each criterion. In each subset, we find the nondominated point (if any) that maximizes function v. After considering all subsets, we choose the point with maximum function v value and show that it corresponds to the optimal point of $[P^n]$.

We characterize each subset using a vector $\mathbf{k} = (k_1, \ldots, k_p)$ where $0 \leq k_i \leq n-1$ for all $i = 1, \ldots, p$. If $k_i = 0$, there is no additional lower bound for criterion i and hence we define the lower bound as M_i. Otherwise, we use i^{th} criterion value of $\mathbf{z}^{k_i} \in \mathbf{Z}_n$ to impose a lower bound for criterion i. If $\mathbf{b}^{\mathbf{k}} = (b_1^{\mathbf{k}}, \ldots, b_p^{\mathbf{k}})$ denotes the lower bound vector characterized by $\mathbf{k}$, then we define the submodel as follows:

$$
\begin{aligned}
&[P_{\mathbf{k}}]: \\
&\text{Max} \quad v(\mathbf{z}) = \sum_{i=1}^{p} v_i z_i(\mathbf{x}) \\
&\text{subject to} \\
&z_i(\mathbf{x}) \geq b_i^{\mathbf{k}} \qquad\qquad i = 1, \ldots, p \\
&\mathbf{x} \in \mathbf{X}
\end{aligned}
$$

where

$$
b_i^{\mathbf{k}} = \begin{cases} M_i & k_i = 0 \\ z_i^{k_i} + \epsilon & k_i \neq 0 \end{cases}
$$

We do not enumerate all combinations of such k_i values since many of these combinations imply the regions containing the existing nondominated points or regions that are already included in the subsets defined by other combinations. We identify a set of $\mathbf{k}$ vectors, $\mathbf{K}^n$, at each iteration so that union of the feasible sets of the corresponding submodels, $Z_{[P_{\mathbf{k}}]}$, will be equivalent to the feasible set of $[P^n]$, $Z_{[P^n]}$, i.e., $Z_{[P^n]} = \bigcup_{\mathbf{k} \in \mathbf{K}^n} Z_{[P_{\mathbf{k}}]}$. Figure 1 shows the feasible set of an example problem whose three nondominated points are available and Figure 2 shows its decomposition on the objective function space.

In order to generate all $\mathbf{k} \in \mathbf{K}^n$, we use the decomposition scheme of Lokman and Köksalan (2013).

Starting from the first criterion, the procedure iteratively sets the value of k_i and define $b_i^{\mathbf{k}}$ as in $[P_{\mathbf{k}}]$. For the following criterion $1 < j < p$, it either does not impose a lower bound (i.e. $k_j = 0$) or uses j^{th} criterion value of $\mathbf{z}^{k_j} \in R_j$ where $R_j = \{\mathbf{z}^t \in \mathbf{Z}^n : z_i^t \geq b_i^{\mathbf{k}} \quad \forall i < j \leq p\}$. That is, the procedure does not consider the nondominated points that have already been excluded from the feasible set by the current bounds. We repeat this procedure until the last criterion, p, and then set the value of k_p so that all nondominated points that are still feasible under the current bounds will be excluded. That is, if $R_p \neq \emptyset$, we set $k_p = \arg\max_{\mathbf{z}^t \in R_p} z_p^t$. Otherwise, $k_p = 0$. Lokman and Köksalan (2013) show that this procedure generates a set of $\mathbf{k}$ vectors, $\mathbf{K}^n$, satisfying $Z_{[P^n]} = \bigcup_{\mathbf{k} \in \mathbf{K}^n} Z_{[P_{\mathbf{k}}]}$. We denote the optimal point to $[P_{\mathbf{k}}]$ by $\mathbf{z}^{\mathbf{k}}$ if $Z_{[P_{\mathbf{k}}]} \neq \emptyset$. We next show that the optimal solution to $[P^n]$ could be found by solving $[P_{\mathbf{k}}]$ for all $\mathbf{k} \in \mathbf{K}^n$.

Theorem 1. *Let $[P^n]$ be feasible and $\mathbf{K}^n$ be such that $Z_{[P^n]} = \bigcup_{\mathbf{k} \in \mathbf{K}^n} Z_{[P_{\mathbf{k}}]}$. If $\mathbf{k}^* = \arg\max_{\mathbf{k} \in \mathbf{K}^n,\, Z_{[P_{\mathbf{k}}]} \neq \emptyset} v(\mathbf{z}^{\mathbf{k}})$, then $\mathbf{z}'^n = \mathbf{z}^{\mathbf{k}^*}$.*

Proof. Since $Z_{[P^n]} = \bigcup_{\mathbf{k} \in \mathbf{K}^n} Z_{[P_{\mathbf{k}}]}$, $\max_{\mathbf{k} \in \mathbf{K}^n,\, Z_{[P_{\mathbf{k}}]} \neq \emptyset} v(\mathbf{z}^{\mathbf{k}}) = \max_{\mathbf{z} \in Z_{[P^n]}} v(\mathbf{z})$. Then $\mathbf{z}'^n = \mathbf{z}^{\mathbf{k}^*}$. □

Lemma 1. *Let $\mathbf{K}^n$ be such that $Z_{[P^n]} = \bigcup_{\mathbf{k} \in \mathbf{K}^n} Z_{[P_{\mathbf{k}}]}$. $[P^n]$ is infeasible if and only if $[P_{\mathbf{k}}]$ is infeasible for all $\mathbf{k} \in \mathbf{K}^n$.*

Proof. Since $Z_{[P^n]} = \bigcup_{\mathbf{k} \in \mathbf{K}^n} Z_{[P_{\mathbf{k}}]}$, $Z_{[P^n]} = \emptyset$ if and only if $Z_{[P_{\mathbf{k}}]} = \emptyset$ for all $\mathbf{k} \in \mathbf{K}^n$. □

3.1.2. Bounding Mechanism of DSA

Different than the existing algorithms, DSA is designed in a way that it is capable of approximating v^* with a prespecified level accuracy, $g^* \geq 0$. To do this, it keeps a lower bound and an upper bound for v^* during the algorithm and stops when $g = \frac{|v^u - v^l|}{|v^l|} \leq g^*$. Different than the existing algorithms, DSA incorporates this information into the solution process and revises $[P_{\mathbf{k}}]$ for each $\mathbf{k} \in \mathbf{K}^n$ as follows:

$$
\begin{aligned}
&[S^{\mathbf{k}}]: \\
&\text{Max} \quad v(\mathbf{z}) = \sum_{i=1}^{p} v_i z_i(\mathbf{x}) \\
&\text{subject to} \\
&z_i(\mathbf{x}) \geq b_i^{\mathbf{k}} \qquad\qquad i = 1, \ldots, p \\
&\sum_{i=1}^{p} v_i z_i(\mathbf{x}) \geq v^l + g^* |v^l| \\
&\sum_{i=1}^{p} v_i z_i(\mathbf{x}) \leq v^u \\
&\mathbf{x} \in \mathbf{X}
\end{aligned}
$$

In $[S^{\mathbf{k}}]$, v^l and v^u denote the lower and upper bounds, respectively of v^*. At any iteration n, the best nondominated point found so far is called the incumbent, denoted by $\mathbf{z}^{inc}$ and the optimal point to $[P^n]$ is denoted by $\mathbf{z}'^n$. Theorem 2 shows that $v^l = v(\mathbf{z}^{inc})$ and $v^u = v(\mathbf{z}'^n)$.

Theorem 2. *Given $\mathbf{Z}_n = \{\mathbf{z}^1, \ldots, \mathbf{z}^{n-1}\}$ at iteration $n > 1$, let $\mathbf{z}^{inc}$ be the incumbent point, $v(\mathbf{z}^{inc}) = \max_{t=1,\ldots,n-1} v(\mathbf{z}^t)$, and $\mathbf{z}'^n$ be the optimal point to $[P^n]$. If $v(\mathbf{z}^*) = \max_{\mathbf{z}\in\mathbf{Z}^N} v(\mathbf{z})$ and $\mathbf{z}^* \neq \mathbf{z}^{inc}$, then $v(\mathbf{z}^{inc}) \leq v(\mathbf{z}^*) \leq v(\mathbf{z}'^n)$.*

Proof. Since $\mathbf{z}^{inc} \in \mathbf{Z}^N$ and $v(\mathbf{z}^*) = \max_{\mathbf{z}\in\mathbf{Z}^N} v(\mathbf{z})$, we have $v(\mathbf{z}^{inc}) \leq v(\mathbf{z}^*)$. In addition, $\mathbf{Z}^N \subseteq \mathbf{Z}$ and $v(\mathbf{z}'^1) = \text{Max}_{\mathbf{z}\in\mathbf{Z}}\, v(\mathbf{z})$, thus $v(\mathbf{z}^*) \leq v(\mathbf{z}'^1)$. By construction, we have $Z_{[P^n]} \subseteq Z_{[P^{n-1}]}$ that implies $v(\mathbf{z}'^n) \leq v(\mathbf{z}'^{n-1})$ for any $n > 1$. Since we also have $\mathbf{z}^* \neq \mathbf{z}^{inc}$, then $\mathbf{z}^* \in Z_{[P^n]}$ and $v(\mathbf{z}^*) \leq v(\mathbf{z}'^n)$. □

Instead of using the true lower bound, $v^l = v(\mathbf{z}^{inc})$, in submodels, DSA sets a lower bound of $v^l + g^*|v^l|$ on the objective function value of $[S^{\mathbf{k}}]$ since any point having an objective function value in between is "close" to the incumbent point in terms of the desired level of accuracy. Therefore, we rebuild each submodel in a way that it turns out to be infeasible if there does not exist a new point that is "far enough" from the incumbent point. We next formalize it in Theorem 3 and show that DSA provides performance guarantee on the quality of the resulting solution.

Theorem 3. *Given $\mathbf{Z}_n = \{\mathbf{z}^1, \ldots, \mathbf{z}^{n-1}\}$ at iteration $n > 1$, let $\mathbf{K}^n$ be such that $Z_{[P^n]} = \bigcup_{\mathbf{k}\in\mathbf{K}^n} Z_{[P_{\mathbf{k}}]}$ and $v(\mathbf{z}^{inc}) = \max_{t=1,\ldots,n-1} v(\mathbf{z}^t)$. If $v(\mathbf{z}^*) = \max_{\mathbf{z}\in\mathbf{Z}^N} v(\mathbf{z})$, $v(\mathbf{z}^*) \neq v(\mathbf{z}^{inc})$, and $[S^{\mathbf{k}}]$ is infeasible for all $\mathbf{k} \in \mathbf{K}^n$, then $\frac{|v(\mathbf{z}^*)-v(\mathbf{z}^{inc})|}{|v(\mathbf{z}^{inc})|} \leq g^*$.*

Proof. If $[S^{\mathbf{k}}]$ is infeasible for all $\mathbf{k} \in \mathbf{K}^n$ but $v(\mathbf{z}^*) \neq v(\mathbf{z}^{inc})$, then there should exist $\mathbf{k} \in \mathbf{K}^n$ for which $\mathbf{z}^* \in Z_{[P_{\mathbf{k}}]}$ but $v^l \leq v(\mathbf{z}^*) < v^l + g^*|v^l|$. Since $v(\mathbf{z}^{inc}) \leq v(\mathbf{z}^*)$ and $v^l = v(\mathbf{z}^{inc})$ by Theorem 2, $\frac{|v(\mathbf{z}^*)-v(\mathbf{z}^{inc})|}{|v(\mathbf{z}^{inc})|} \leq g^*$. □

3.1.3. Nondominance Check in DSA

Another mechanism to improve the existing algorithms applies to the nondominance check. DSA checks whether a point $\mathbf{z}'^n$ is dominated or not by solving:

$$
\begin{aligned}
&[E^n_{\mathbf{w}}]: \\
&\text{Max} \sum_{i=1}^{p} w_i z_i(\mathbf{x}) \\
&\text{subject to} \\
&z_i(\mathbf{x}) \geq z'^n_i \qquad i = 1, \ldots, p \\
&\mathbf{x} \in \mathbf{X}
\end{aligned}
$$

where $\mathbf{w} = (w_1, \ldots, w_p)$ denotes a weight vector satisfying $w_i > 0$ for all $i = 1, \ldots, p$. If $\mathbf{z}'^n$ is a dominated solution, $[E^n_{\mathbf{w}}]$ generates a nondominated point, $\mathbf{z}^n$, that dominates $\mathbf{z}'^n$. If $v(\mathbf{z}^n) > v^l$, $\mathbf{z}^{inc}$ and v^l are updated for the next iteration. In addition to the region that is dominated by $\mathbf{z}^{inc}$, the inferior regions with respect to the value function and desired level of accuracy are excluded from the feasible set. Instead of equally weighing each criterion as in the algorithm of Jorge (2009) or arbitrarily choosing a positive $\mathbf{w}$, DSA applies a heuristic method in order to generate efficient cuts. Since there may exist more than one nondominated point dominating $\mathbf{z}'^n$, DSA aims to favor the one with a better function v value and sets $w_i = v_i - \min\limits_{i=1,\ldots,p} v_i + 1$ for each $i = 1, \ldots, p$ in $[E^n_{\mathbf{w}}]$ that also satisfy $w_i > 0$ for all i. By this way, DSA aims to obtain a tighter lower bound on v^* at each iteration which helps to converge fast to the optimal solution. In Section 4, we discuss the computational contribution of this mechanism and show that it significantly improves the solution times of DSA.

3.1.4. The algorithm: DSA

We next present the steps of DSA and demonstrate it on an example problem.

Algorithm 1 DSA

Step 1: *Initialization*

$n = 1$, $w_i = v_i - \min\limits_{i=1,\ldots,p} v_i + 1 \;\; i = 1, \ldots, p$ and $\mathbf{z}'^1 = \text{Max}_{z(\mathbf{x}) \in \mathbf{Z}}\, v(\mathbf{z})$.

Step 2: *Nondominance Check*

Solve $[E^n_{\mathbf{w}}]$ and denote the optimal point as $\mathbf{z}^n$. If $\mathbf{z}^n = \mathbf{z}'^n$, then set $v^u = v^l = v(\mathbf{z}^n)$, $\mathbf{z}^{inc} = \mathbf{z}^n$ and go to Step 5. Otherwise, go to Step 3.

Step 3: *Bound Update*

Set $v^u = v(\mathbf{z}'^n)$. If $v(\mathbf{z}^n) \geq v^l$, then set $v^l = v(\mathbf{z}^n)$ and $\mathbf{z}^{inc} = \mathbf{z}^n$. If $g = \frac{|v^u - v^l|}{|v^l|} \leq g^*$, go to Step 5. Otherwise, go to Step 4.

Step 4: *New Point Generation*

Set $n = n + 1$ and update $\mathbf{K}^n$. Solve $[S^{\mathbf{k}}]$ corresponding to each $\mathbf{k} \in \mathbf{K}^n$ and denote the optimal point as $\mathbf{z}^{\mathbf{k}}$. If $[S^{\mathbf{k}}]$ is infeasible for all $\mathbf{k} \in \mathbf{K}^n$, then go to Step 5. Otherwise, find $\mathbf{k}^* = \underset{\mathbf{k} \in \mathbf{K}^n,\, Z_{[S^{\mathbf{k}}]} \neq \emptyset}{\arg\max} \; v(\mathbf{z}^{\mathbf{k}})$ and $\mathbf{z}'^n = \mathbf{z}^{\mathbf{k}^*}$. Go to Step 2.

Step 5: *Termination*

Stop. $\mathbf{z}^* = \mathbf{z}^{inc}$ maximizes (approximates) function v over the nondominated set, $v^* = v(\mathbf{z}^*)$ (with a desired level of accuracy, g^*) .

In Step 4, we store the optimal point of $[S^{\mathbf{k}}]$ corresponding to each $\mathbf{b}^{\mathbf{k}}$ in an archive. In the succeeding iterations, before solving a new model, we check whether the optimal solution will be identical to that of a previously-solved model. Furthermore, we detect the infeasibilities by utilizing the information stored. With those mechanisms, the number of submodels decreases substantially. We refer the reader to Lokman and Köksalan (2013) for the details. We next show that DSA will terminate in a finite number of iterations.

Theorem 4. *Let $\mathbf{Z}^N$ be the set of all nondominated points. If $\mathbf{Z}^N$ is a bounded set, DSA will find the*

optimal point of problem $[MOIP(v)]$ *in at most* $|\mathbf{Z}^N| + 1$ *number of iterations.*

Proof. At any iteration n, DSA finds exactly one nondominated point that is distinct from the nondominated points in $\mathbf{Z}_n$ that are generated in previous iterations. If $\mathbf{Z}^N$ is bounded, DSA will iteratively generate all these nondominated points in $|\mathbf{Z}^N|$ iterations until $Z_{[P^n]}$ becomes empty in the worst case. Then, it will terminate at iteration $n = |\mathbf{Z}^N| + 1$ when $[P^n]$ is to be detected as infeasible. □

Table 1: Demonstration of DSA on problem in Example 1

Iteration (n)	$\mathbf{z}'^n$	$v(\mathbf{z}'^n)$	$\mathbf{z}^n$	$v(\mathbf{z}^n)$	$\mathbf{z}^{inc}$	$v(\mathbf{z}^{inc})$
1	(1030, 1387, 534)	33,144	(1942, 1791, 1471)	-26,780	$\mathbf{z}^1$	-26,780
2	(1434, 1796, 985)	14,968	(1891, 1824, 1466)	-25,278	$\mathbf{z}^2$	-25,278
3	(1506, 1827, 1063)	9,788	(1875, 1831, 1549)	-33,438	$\mathbf{z}^2$	-25,278
4	(1502, 1835, 1090)	7,448	(1847, 1847, 1412)	-19,298	$\mathbf{z}^4$	-19,298
5	(1470, 1849, 1127)	4,028	(1828, 1858, 1539)	-31,692	$\mathbf{z}^4$	-19,298
6	(1528, 1861, 1163)	1,864	(1800, 1867, 1480)	-25,716	$\mathbf{z}^4$	-19,298
7	(1542, 1868, 1178)	924	(1772, 1872, 1536)	-31,448	$\mathbf{z}^4$	-19,298
8	(1415, 1874, 1199)	-2,642	(1757, 1876, 1440)	-21,850	$\mathbf{z}^4$	-19,298
9	(1570, 1877, 1232)	-3,616	(1729, 1879, 1525)	-30,586	$\mathbf{z}^4$	-19,298
10	(1536, 1885, 1249)	-5,376	(1706, 1886, 1546)	-32,644	$\mathbf{z}^4$	-19,298
11	(1944, 1651, 1218)	-8,732	(1947, 1785, 1418)	-21,722	$\mathbf{z}^4$	-19,298
12	(1949, 1642, 1217)	-9,030	(1976, 1757, 1488)	-29,772	$\mathbf{z}^4$	-19,298
13	(1491, 1887, 1294)	-10,402	(1696, 1890, 1386)	-16,576	$\mathbf{z}^{13}$	-16,576
14	(1519, 1892, 1319)	-12,250	(1641, 1899, 1392)	-17,478	$\mathbf{z}^{13}$	-16,576
15	(1852, 1841, 1359)	-14,240	(1852, 1843, 1486)	-26,836	$\mathbf{z}^{13}$	-16,576
16	(1675, 1891, 1375)	-15,718	(1675, 1891, 1375)	-15,718	$\mathbf{z}^{16}$	-15,718

Example 1. For demonstrating purposes, we present an example three-objective 50-item knapsack problem that has 540 nondominated points. Table 1 shows the progress of the algorithm, DSA, for the value function of $v(\mathbf{z}) = 14z_1 + 52z_2 - 100z_3$. If we consider the iteration, $n = 5$, at which four points are already generated, $\mathbf{Z}_5 = \{\mathbf{z}^1, \mathbf{z}^2, \mathbf{z}^3, \mathbf{z}^4\}$, $\mathbf{z}^4$ is the incumbent point, $v^l = v(\mathbf{z}^4) = -19,298$ and $v^u = v(\mathbf{z}'^4) = 7,448$. In order to generate $\mathbf{z}^5$, DSA forms the subsets as in Table 2 and finds optimal solution to $[S^{\mathbf{k}}]$ for each $\mathbf{k}$ in $\mathbf{K}^5$. Since $\mathbf{k}^* = \underset{\mathbf{k}\in\mathbf{K}^5}{\arg\max}\, v(\mathbf{z}^{\mathbf{k}}) = (0, 4, 0)$, we define $\mathbf{z}'^5 = \mathbf{z}^{\mathbf{k}^*} = (1470, 1849, 1127)$ as also shown in Table 1. Since v is not a positive linear combination of the criteria, the algorithm solves $(E^5_{\mathbf{w}})$ using $\mathbf{w} = (115, 153, 1)$ $(w_i = v_i - \min_{i=1,\ldots,p} v_i + 1 \quad i = 1, \ldots, p)$ and finds $\mathbf{z}^5 = (1828, 1858, 1539)$ that dominates $\mathbf{z}'^5$. We update $v^u = v(\mathbf{z}'^5) = 4,028$ while $\mathbf{z}^{inc}$ and v^l remain the same. The next iteration of the algorithm starts with the set of nondominated points

in $\mathbf{Z}_6 = \mathbf{Z}_5 \bigcup \{\mathbf{z}^5\}$. DSA proceeds in this way and stops in iteration 16 when $\mathbf{z}'^n = \mathbf{z}^n$. We obtain $\mathbf{z}^* = (1675, 1891, 1375)$ and $v^* = -15,718$.

Note that we do not consider all combinations of $k_i = 0, \ldots, 4$ in Table 2. Furthermore, we solve $[S^{\mathbf{k}}]$ only when it is not equivalent to the previously-solved submodels. For instance, we do not solve the submodels corresponding to $\mathbf{k} \in \{(0,1,3),(0,2,3),(4,0,3),(4,1,3),(4,2,3)\}$ since we could detect that it will be infeasible based on the solution for $\mathbf{k} = (0,0,3)$. Similarly, we also process the information obtained from submodels in previous iterations and our results show that DSA requires solving three of these 15 submodels at this iteration.

Table 2: Generated subsets and corresponding solutions in DSA at $n = 5$ for problem in Example 1

$\mathbf{k}$	$\mathbf{b}^{\mathbf{k}}$	$\mathbf{z}^{\mathbf{k}}$	$v(\mathbf{z}^{\mathbf{k}})$
(0, 0, 3)	(M_1, M_2, 1550)	infeasible	-
(0, 1, 3)	(M_1, 1792, 1550)	infeasible	-
(0, 2, 3)	(M_1, 1825, 1550)	infeasible	-
(0, 3, 4)	(M_1, 1832, 1413)	infeasible	-
(0, 4, 0)	(M_1, 1848, M_3)	**(1470, 1849, 1127)**	**4,028**
(1, 0, 0)	(1943, M_2, M_3)	(1944, 1651, 1218)	-8,732
(2, 0, 1)	(1892, M_2, 1472)	infeasible	-
(2, 1 ,0)	(1892, 1792, M_3)	(1903, 1808, 1364)	-15,742
(3, 0, 1)	(1876, M_2, 1472)	infeasible	-
(3, 1, 2)	(1876, 1792, 1467)	infeasible	-
(3, 2, 0)	(1876, 1825, M_3)	infeasible	-
(4, 0, 3)	(1848, M_2, 1550)	infeasible	-
(4, 1, 3)	(1848, 1792, 1550)	infeasible	-
(4, 2, 3)	(1848, 1825, 1550)	infeasible	-
(4, 3, 0)	(1848, 1832, M_3)	(1852, 1841, 1359)	-14,240

3.2. DSA-m

The existing studies as well as DSA maximize v during the solution process until converging to a non-dominated point. When $v_i < 0$ for some criteria i, dominated points may be found in many iterations and it will increase the computational difficulty. Based on this observation, we develop a new algorithm DSA-m that maximizes one of the criteria, m, throughout the algorithm with the aim of decreasing the number of dominated points generated. Instead of arbitrarily selecting this criterion, we choose m based

on the coefficients of the function v: $m = \underset{i=1,\ldots,p}{\arg\max}\, v_i$. We reformulate our problem at iteration at n as follows:

$$
\begin{aligned}
&[P_m^n]: \\
&\text{Max} \quad z_m(\mathbf{x}) \\
&\text{subject to} \\
&z_i(\mathbf{x}) \geq (z_i^t + \epsilon) y_i^t + M_i(1 - y_i^t) \quad i = 1, \ldots, p, \quad i \neq m \quad t = 1, \ldots, n-1 \\
&\sum_{\substack{i=1,\ldots,p \\ i \neq m}} y_i^t = 1 \qquad t = 1, \ldots, n-1 \\
&\sum_{i=1}^{p} v_i z_i(\mathbf{x}) \geq v^l + g^* |v^l| \\
&\sum_{i=1}^{p} v_i z_i(\mathbf{x}) \leq v^u \\
&y_i^t \in \{0, 1\} \qquad i = 1, \ldots, p, \quad i \neq m \quad t = 1, \ldots, n-1 \\
&\mathbf{x} \in \mathbf{X}
\end{aligned}
$$

If $[P_m^n]$ is feasible, the optimal point is denoted by $\mathbf{z}'^n$. Since $[P_m^n]$ maximizes m^{th} criterion over the feasible set in the criterion space and $Z_{(P_m^{n+1})} \subseteq Z_{[P_m^n]}$, $z_m'^{n+1} \leq z_m'^n$ for all $n \geq 1$ that allows us to decrease the number of constraints and binary variables. Similar to DSA, we partition the feasible criterion space into subsets that are identified by a $\mathbf{k}$ vector:

$$
\begin{aligned}
&[P_m^{\mathbf{k}}]: \\
&\text{Max} \quad z_m(\mathbf{x}) \\
&\text{subject to} \\
&z_i(\mathbf{x}) \geq b_i^{\mathbf{k}} \qquad i = 1, \ldots, p \quad i \neq m \\
&\sum_{i=1}^{p} v_i z_i(\mathbf{x}) \geq v^l + g^* |v^l| \\
&\sum_{i=1}^{p} v_i z_i(\mathbf{x}) \leq v^u \\
&\mathbf{x} \in \mathbf{X}
\end{aligned}
$$

$\mathbf{z}^{\mathbf{k}}$ denotes the optimal nondominated point if $[P_m^{\mathbf{k}}]$ is feasible. As described in Section 3.1.1, the bound for each criterion i is set based on the value of k_i. The set of feasible vectors $\mathbf{k}$, denoted by $\mathbf{K}_m^n$, is obtained as in DSA but with prior information of $k_m = 0$ since we do not impose a lower bound for criterion m in DSA-m.

Theorem 5. *Let* $[P_m^n]$ *be feasible and* $\mathbf{K}_m^n$ *be such that* $Z_{[P_m^n]} = \bigcup_{\mathbf{k}\in\mathbf{K}_m^n} Z_{[P_m^{\mathbf{k}}]}$. *If* $\mathbf{k}^* = \arg\max_{\mathbf{k}\in\mathbf{K}_m^n,\, Z_{[P_{\mathbf{k}}]}\neq\emptyset} z_m^{\mathbf{k}}$, *then* $\mathbf{z}'^n = \mathbf{z}^{\mathbf{k}^*}$.

Proof. Since $Z_{[P_m^n]} = \bigcup_{\mathbf{k}\in\mathbf{K}_m^n} Z_{[P_m^{\mathbf{k}}]}$, $\max_{\mathbf{k}\in\mathbf{K}_m^n,\, Z_{[P_m^{\mathbf{k}}]}\neq\emptyset} z_m^{\mathbf{k}} = \max_{\mathbf{z}\in Z_{[P^n]}} \mathbf{z}'^n_m$. Then $\mathbf{z}'^n = \mathbf{z}^{\mathbf{k}^*}$. □

We next give the outline of DSA-m and illustrate it with the following example.

Algorithm 2 DSA-m

Step 1: *Initialization*
$m = \arg\max_{i=1,\ldots,p} v_i$ and $w_i = v_i - \min_{i=1,\ldots,p} v_i + 1 \;\; i = 1,\ldots,p$.
$\mathbf{z}'^0 = \text{Max}_{z(\mathbf{x})\in\mathbf{Z}}\, v(\mathbf{z})$ and $v^u = v(\mathbf{z}'^0)$.
Solve $[E_{\mathbf{w}}^0]$ and denote the optimal point as $\mathbf{z}^0$ and set $\mathbf{z}^{inc} = \mathbf{z}^0$ and $v^l = v(\mathbf{z}^{inc})$.
$\mathbf{z}'^1 = \max_{\mathbf{x}\in\mathbf{X}} z_m(\mathbf{x})$ and $n = 1$.
Step 2: *Nondominance Check*
Solve $[E_{\mathbf{w}}^n]$ and denote the optimal point as $\mathbf{z}^n$.
Step 3: *Bound Update*
If $v(\mathbf{z}^n) > v(\mathbf{z}^{inc})$, then set $\mathbf{z}^{inc} = \mathbf{z}^n$ and $v^l = v(\mathbf{z}^{inc})$. If $g = \frac{|v^u - v(\mathbf{z}^{inc})|}{|v(\mathbf{z}^{inc})|} \leq g^*$, go to Step 5. Otherwise, go to Step 4.
Step 4: *New Point Generation*
Set $n = n+1$ and update $\mathbf{K}_m^n$. Solve $P_m^{\mathbf{k}}$ corresponding to each $\mathbf{k} \in \mathbf{K}_m^n$. If $P_m^{\mathbf{k}}$ is infeasible for all $\mathbf{k} \in \mathbf{K}_m^n$, go to Step 5. Otherwise, find $\mathbf{k}^* = \arg\max_{\mathbf{k}\in\mathbf{K}_m^n,\, Z_{[P_m^{\mathbf{k}}]}\neq\emptyset} \mathbf{z}_m^{\mathbf{k}}$, assign $\mathbf{z}'^n = \mathbf{z}^{\mathbf{k}^*}$, and go to Step 2.

Step 5. Stop. $\mathbf{z}^* = \mathbf{z}^{inc}$ maximizes (approximates) function v over the nondominated set (with the desired level of accuracy, g^*).

Example 2. In order to demonstrate the progress of the algorithm, DSA-m, we consider the problem in Example 1. As shown in Table 3, the algorithm sets $m = 2$ and generates the points in nonincreasing order of the second criterion since $v(\mathbf{z}) = 14z_1 + 52z_2 - 100z_3$ and $m = \arg\max_{i=1,2,3} v_i$. If we consider the iteration $n = 5$, $\mathbf{z}^3$ is the incumbent point, $v^l = v(\mathbf{z}^3) = -15,718$ and $v^u = v(\mathbf{z}'^0) = 33,144$. In order to find a new point, DSA-m partitions the feasible set as shown in Table 4 and solves $[P_2^{\mathbf{k}}]$ for each $\mathbf{k}$ in $\mathbf{K}_2^5$. DSA-m chooses the subset for which the optimal solution has the maximum second criterion value: $\mathbf{k}^* = \arg\max_{\mathbf{k}\in\mathbf{K}_2^5} z_2^{\mathbf{k}} = (4,0,0)$ and $\mathbf{z}'^5 = \mathbf{z}^{\mathbf{k}^*} = (1788, 1863, 1356)$. Next, DSA-$m$ solves $(E_{\mathbf{w}}^5)$ using $\mathbf{w} = (115, 153, 1)$ as in DSA and finds $\mathbf{z}^5 = (1800, 1867, 1480)$ that dominates $\mathbf{z}'^5$. The values of v^l, v^u and $\mathbf{z}^{inc}$ remain the same and the next iteration starts with $\mathbf{Z}_6 = \mathbf{Z}_5 \bigcup \{\mathbf{z}^5\}$. The algorithm proceeds in this way until the reduced feasible set becomes empty. Table 3 shows that DSA-m stops in 11 iterations with the solution of $\mathbf{z}^* = (1675, 1891, 1375)$ and $v^* = -15,718$.

Since DSA- m also checks whether the solution is identical to that of previously constructed submodels, it solves only one submodel at $n = 5$. While total number of models solved by DSA for this example

is 56, DSA-m solves 27 models in total. Furthermore, DSA first finds the optimal solution in iteration 16, DSA-m finds the optimal solution in the third iteration.

Table 3: Demonstration of DSA-m on problem in Example 2 ($m = 2$)

Iteration (n)	$\mathbf{z}'^n$	$v(\mathbf{z}'^n)$	$\mathbf{z}^n$	$v(\mathbf{z}^n)$	$\mathbf{z}^{inc}$	$v(\mathbf{z}^{inc})$
1	(1580, 1899, 1435)	-22,632	(1580, 1899, 1435)	-22,632	$\mathbf{z}^1$	-22,632
2	(1641, 1899, 1392)	-17,478	(1641, 1899, 1392)	-17,478	$\mathbf{z}^2$	-17,478
3	(1675, 1891, 1375)	-15,718	(1675, 1891, 1375)	-15,718	$\mathbf{z}^3$	-15,718
4	(1677, 1873, 1325)	-11,626	(1757, 1876, 1440)	-21,850	$\mathbf{z}^3$	-15,718
5	(1788, 1863, 1356)	-13,692	(1800, 1867, 1480)	-25,716	$\mathbf{z}^3$	-15,718
6	(1804, 1856, 1372)	-15,432	(1828, 1858, 1539)	-31,692	$\mathbf{z}^3$	-15,718
7	(1852, 1841, 1359)	-14,240	(1852, 1843, 1486)	-26,836	$\mathbf{z}^3$	-15,718
8	(1855, 1816, 1356)	-15,198	(1891, 1824, 1466)	-25,278	$\mathbf{z}^3$	-15,718
9	(1905, 1787, 1344)	-14,806	(1942, 1791, 1471)	-26,780	$\mathbf{z}^3$	-15,718
10	(1948, 1700, 1298)	-14,128	(1976, 1757, 1488)	-29,772	$\mathbf{z}^3$	-15,718
11	infeasible	-	-	-	$\mathbf{z}^3$	-15,718

Table 4: Generated subsets and corresponding solutions in DSA-m at $n = 5$ for problem in Example 2 ($m = 2$)

$\mathbf{k}$	$\mathbf{b}^{\mathbf{k}}$	$\mathbf{z}^{\mathbf{k}}$	$v(\mathbf{z}^{\mathbf{k}})$
(0, 0, 4)	(M_1, M_2, 1441)	infeasible	-
(1, 0, 4)	(1581, M_2, 1441)	infeasible	-
(2, 0, 4)	(1642, M_2, 1441)	infeasible	-
(3, 0, 4)	(1676, M_2, 1441)	infeasible	-
(4, 0, 0)	(1758, M_2, M_3)	**(1788, 1863, 1356)**	**-13,692**

4. Computational Experiments

We conduct computational experiments on randomly generated instances of multi-objective assignment problem (MOAP) and multi-objective knapsack problem (MOKP). For the three-objective case, we consider 25-, 50-, and 100-item knapsack problems and 10x10-, 20x20, and 30x30-sized assignment problems. For the four-objective case, we test the algorithms on 25-, 50-, and 100-item knapsack problems and 10x10-, and 20x20-sized assignment problems. Our random generation scheme is the same as that of Köksalan and Lokman (2015). Since all parameters are generated as positive integers, we set $\epsilon = 1$

and $M_i = 0$ for all $i = 1, \ldots p$. For each instance, we maximize a linear function, v, whose coefficients, v_i $i = 1, \ldots p$, are randomly generated as uniformly distributed integers between -100 and 100. We coded the algorithms using Microsoft Visual Studio 2010 by C++ programming language with the callable library of CPLEX 12.6. The experiments are conducted on 64-bit Microsoft Windows 7 Professional installed on an Intel (R) Core (TM) i7-4790 CPU @ 3.60GHz computer with 8.00 GB RAM. Table 5 presents computational results on the performances of the algorithms. In presenting the results in Table 5, we list the problem type, the number of models solved by the algorithms Jorge, DSA and DSA-m as well as the respective solution times (in seconds).

The computational advantage that DSA provides over Jorge (2009) is to reduce the complexity of the models solved at each iteration at the expense of solving more models. Since the complexity of the models does not increase substantially throughout the algorithm for small-sized problems, DSA is outperformed by the algorithm of Jorge (2009) for 10x10-sized MOAPs and 25-item MOKPs with three and four objectives (3MOAP10x10, 4MOAP10x10, 3MOKP25, and 4MOKP25). However, we should note that the performance of DSA relative to that of Jorge (2009) in terms of the solution time improves substantially as the problem size increases. For instance, in three-objective 30x30 MOAPs (3MOAP30x30s) the algorithm of Jorge (2009) converges to the optimal value in 2233.60 seconds on average whereas the average solution time is 343.02 seconds for DSA. Similarly, the respective solution times are 7720.44 and 246.47 seconds for four-objective 100-item MOKPs (4MOKP100s). The improvement is not only due to the decomposition mechanism but also the mechanisms applied in the nondominance check and bounding.

DSA-m evolves different than these algorithms since it selects a criterion based on the value function and finds the feasible point that maximizes that criterion at each iteration. The aim is not only to reduce the number of dominated points generated throughout the algorithm but also reduce the dimension of the search space. As seen in Table 5, these modifications decrease the solution time as well as the number of models solved. While the average solution times for 3MOKP25s and 3MOAP30x30s are almost similar for DSA and DSA-m, DSA-m outperforms DSA in the remaining instances in terms of the number of models solved and solution time. Our further analysis on 20x20-sized MOAPs shows that the average number of submodels solved is decreased by 72.69% and 66.84% for three and four objective cases, respectively when compared to those of DSA. In addition, the ratio of the number of the nondominated points to total number of points generated throughout the algorithm increases from 15.11% to 21.07% for the three-objective case and from 12.20% to 19.42% for the four-objective case. Since our bounding mechanism helps us to introduce cuts based on the nondominated points generated, DSA-m converges fast. While DSA terminates in 30.10 iterations on average in 3MOAP20x20s, DSA-m stops in 23.00 iterations. Similarly, the number of iterations required by DSA and DSA-m to solve 4MOAP20x20s are 56.10 and 48.40 on average, respectively.

In order to analyze how the performances of the algorithms change with respect to an increase in the number of the criteria, we compare the results for the three-objective case with those for the four-objective case. The results in Table 5 show that the average solution time of the algorithm of Jorge (2009) increases drastically and it becomes prohibitive to solve larger problems. This is because the nondominated set grows with the number of criteria, it takes more number of iterations to attain the optimal solution and the algorithm of Jorge (2009) solves progressively more constrained model at each iteration. Although the dimension of the search space also grows with the number of criteria and the number of subsets generated to partition the search space increases, the solution times are still quite

reasonable for DSA and DSA-m as shown in Table 5. While, out of 10 instances of 4MOAP20x20, the algorithm of Jorge (2009) could not solve one instance within 48 hours, the solution times required by DSA and DSA-m to solve this instance are 3.77 and 1.93 minutes, respectively. If we compare the results for the remaining nine instances, the algorithm of Jorge (2009) requires 93.56 iterations on average to converge to the optimal solution whereas DSA and DSA-m terminate in 47.64 and 31.67 iterations on average, respectively. Furthermore, the corresponding average solution times are 105.51 and 34.12 seconds while an average of 3.05 hours are required by the algorithm of Jorge (2009) to solve these instances. The computational efficiency mainly results from accompanying mechanisms for fast convergence, solving a number of simpler models of constant size at each iteration, and progressively processing the information obtained from these models.

Table 5: The performance results[a] on the test instances with $p = 3$ and 4

	Number of models solved			Sol. time (in seconds)		
Problem	Jorge	DSA	DSA-m	Jorge	DSA	DSA-m
3MOAP10x10 ($\bar{N} = 189.40$)	17.10	35.10	21.90	2.64	4.70	2.75
3MOAP20x20 ($\bar{N} = 1908.50$)	130.30	137.40	52.30	90.10	23.41	8.82
3MOAP30x30 ($\bar{N} = 5235.00$)	542.20	1100.50	553.90	2233.60	343.02	438.04
3MOKP25 ($\bar{N} = 58.00$)	7.20	13.20	12.20	1.15	1.78	1.79
3MOKP50 ($\bar{N} = 524.00$)	43.70	95.20	40.60	11.14	14.82	20.83
3MOKP100 ($\bar{N} = 3805.10$)	160.00	322.50	86.10	381.85	52.11	27.39
4MOAP10x10 ($\bar{N} = 962.00$)	21.60	87.00	73.10	3.76	21.68	11.11
4MOAP20x20 ($\bar{N} = 29324.50$)	187.22[b]	741.80	275.80	10994.59[b]	149.61	58.47
4MOKP25 ($\bar{N} = 165.20$)	17.40	57.00	32.20	3.18	8.84	4.63
4MOKP50 ($\bar{N} = 1576.30$)	59.50	234.90	160.30	43.36	42.15	73.23
4MOKP100	181.78[b]	1099.00	495.20	7720.44[b]	246.47	125.78

[a] Average values based on 10 instances are reported per cell. $\bar{N}$ represents the average number of nondominated points. $\bar{N}$ is not known for 4MOKP100.

[b] Average values based on 9 instances are reported. One instance that could not be solved by the algorithm of Jorge (2009) within 48 hours are excluded from the statistics.

In order to assess the efficiency of the mechanisms developed, different variants of DSA and DSA-m are built and tested on 3MOAP20x20s. In the first variant of DSA, DSA-equal $\mathbf{w}$, all objectives are equally weighted in $[E^n_{\mathbf{w}}]$ as in Jorge (2009). The results in Table 6 show that the number of models solved by DSA-equal $\mathbf{w}$ is greater than that of Jorge (2009) but the first outperforms the latter in terms of the solution time because of the efficiency of the decomposition mechanism. The comparison between the results obtained with DSA and DSA-equal $\mathbf{w}$ indicates the computational efficiency of our heuristic method that sets the weights in model $[E^n_{\mathbf{w}}]$ based on the coefficients of function v. The average number of models solved decreases from 376.40 to 137.40 that corresponds to 61.61% saving in the solution times on average. That is, while DSA and the algorithm of Jorge (2009) work in a similar way, DSA converges to the optimal point much faster than the algorithm of Jorge (2009) by the help of the mechanisms. We also revised DSA-m and built another variant, DSA-random m, that arbitrarily selects the criterion to maximize in $[P^{\mathbf{k}}_m]$. When the results obtained with DSA-random m are compared with those of DSA-m, the heuristic procedure in DSA-m that chooses the criterion, m, based on the value function provides a significant improvement by decreasing the average number of models from 170.50 to 52.30 and reducing the average solution times from 34.23 to 8.82 seconds.

Table 6: Efficiency of the Mechanisms on three-objective 20x20-sized MOAPs*

	Number of models solved		Sol. time (in seconds)	
Method	Avg.	StDev.	Avg.	StDev.
Jorge	130.30	111.86	90.10	132.52
DSA-equal $\mathbf{w}$	376.40	328.71	60.98	54.53
DSA-random m	170.50	173.18	34.23	40.79
DSA	137.40	138.16	23.41	25.23
DSA-m	52.30	28.62	8.82	5.62

* Values based on 10 problems are reported per cell

We also compare the performances of DSA and DSA-m under different values of the desired level of accuracy, g^*. Tables 7 and 8 present the results for three-objective and four-objective cases, respectively. The results show that the solution time decreases substantially as the value of g^* increases. Since DSA-m maximizes one of the criteria while setting bounds on function v value based on the desired level of accuracy, the savings in the solution times for DSA-m are larger than those of DSA. For instance, when $g^* = 0.1$ for the three-objective case, the solution time for DSA decreases by 31.64% on average while it improves by 39.98% on average for DSA-m when compared to the case of $g^* = 0$. In addition, the results show that a higher saving in the solution time is obtained for the four-criteria case when compared to the three-criteria case. The savings in the solution times when $g^* = 0.1$ increase to 43.41% and 51.53% on average for DSA and DSA-m, respectively. The results in Tables 7 and 8 also indicate that the resulting level of accuracy is much smaller than the desired level of accuracy. Even when the desired level of accuracy is set to 0.15, $g^* = 0.15$, the actual gap values turn out to be 0.0368 and 0.0274 on average for DSA and DSA-m, respectively while corresponding solution times are only 27.26 and 13.37 seconds on average. The results show that the approximation algorithms are promising for large-scale problems from practice.

We also compare the performances of DSA and DSA-m with that of Boland et al. (2017)'s algorithm on a 100-item multi-dimensional knapsack problem with three objectives, the instance available in their public library. We make 10 replications using different linear functions whose coefficients $v_i, i = 1, \ldots p$, are randomly generated as uniformly distributed integers between -100 and 100. While DSA finds the optimal nondominated point in 363.68 seconds with a standard deviation of 984.76 seconds on average, DSA-m solves the problem in 285.03 seconds with a standard deviation of 865.78 seconds. Using the source code of Boland et al. (2017), we also solved the same problems and the average solution time turns out to be 465.32 seconds with a standard deviation of 654.07 seconds. While DSA-m outperforms the algorithm of Boland et al. (2017) on the average, the results are analyzed in detail to evaluate the performances of these algorithms. The results show that the algorithm of Boland et al. (2017) work well when $v_i < 0$, for all $i = 1, \ldots p$. However, when $v_i > 0$ for at least one criterion i, DSA and DSA-m outperform the algorithm of Boland et al. (2017). For instance, when $v(\mathbf{z}) = -61z_1 - 32z_2 - 74z_3$, DSA and DSA-$m$ find the optimal point in 3132.52 seconds, and 2748.93 seconds, respectively while the algorithm of Boland et al. (2017) has a solution time of 997.15 seconds. However, when $v(\mathbf{z}) = 73z_1 - 5z_2 - 98z_3$, the solution times are 25.41, 16.12 and 1437.92 seconds, for DSA, DSA-m and the algorithm of Boland et al. (2017), respectively. This is because our algorithms are specialized with respect to the value function.

5. Conclusions

The problem of optimizing a linear function over the nondominated set of a MOIP has many applications but the solution may not be straightforward since the nondominated set is not known a priori. In this paper, we develop two algorithms to maximize a linear function over the nondominated set of any given MOIP. The first algorithm, DSA, proposes modifications to an existing algorithm by employing a set of improvement mechanisms. In order to decrease the number of models to be solved at one iteration, we develop another algorithm, DSA-m that maximizes one of the criteria throughout the algorithm. The algorithms iteratively generate nondominated points reducing the feasible set not only using the nondominated points generated so far but also introducing bounds on the value of the linear function. These algorithms, DSA and DSA-m, are both designed to work under a given desired level of accuracy such that it allows the DM to approximate the optimal solution with a performance guarantee on the quality of the solution in a reasonable solution time. Therefore, they could be applied to large-sized MOIPs for which the nondominated set is huge. We test and compare the performances of the algorithms with the existing studies on different sized multi-objective combinatorial optimization problems, which are typically hard to solve. The results show that the algorithms converge to the optimal solution in a reasonable solution time especially on large problems.

DSA is directly applicable to the problem of optimizing a linear function over the efficient sets of MOIPs. The weights to be used in nondominance check could be selected using a similar mechanism. However, DSA-m could be applied after a criterion is to be decided by the analyst. As an extension, DSA and DSA-m could be modified in order to approximate a value function over the nondominated sets of multi-objective mixed integer programs. For large-sized problems where it is hard to solve even a single model, problem-specific solution procedures could be used to generate the points. Alternatively, the interactive version of those algorithms could be applied to large problems such that the DM's preferences

Table 7: The performance results of the approximation algorithms on the test instances with $p = 3$*

Problem	g^*	Nb. of models solved		Sol. time (in seconds)		Actual Level of Accuracy	
		DSA	DSA-m	DSA	DSA-m	DSA	DSA-m
3MOAP10x10	0.00	35.10	21.90	4.70	2.75	0.0000	0.0000
	0.05	29.40	16.80	3.89	2.12	0.0055	0.0021
	0.10	21.90	13.10	2.91	1.61	0.0113	0.0059
	0.15	19.20	11.50	2.43	1.49	0.0290	0.0059
3MOAP20x20	0.00	137.40	52.30	23.41	8.82	0.0000	0.0000
	0.05	110.90	42.30	17.91	7.43	0.0083	0.0051
	0.10	88.60	32.70	14.35	5.38	0.0192	0.0131
	0.15	71.00	27.90	10.95	4.90	0.0289	0.0313
3MOAP30x30	0.00	1100.50	553.90	343.02	438.04	0.0000	0.0000
	0.05	807.20	335.30	193.99	244.93	0.0176	0.0112
	0.10	637.30	224.90	130.00	144.55	0.0314	0.0196
	0.15	557.80	147.00	112.88	75.58	0.0407	0.0342
3MOKP25	0.00	13.20	12.20	1.78	1.79	0.0000	0.0000
	0.05	7.40	7.70	0.89	1.15	0.0081	0.0052
	0.10	6.80	6.90	0.93	0.96	0.0119	0.0057
	0.15	5.70	5.90	0.65	0.76	0.0127	0.0264
3MOKP50	0.00	95.20	33.70	14.82	12.20	0.0000	0.0000
	0.05	57.90	17.90	7.52	3.31	0.0088	0.0007
	0.10	38.00	10.60	4.91	1.76	0.0374	0.0051
	0.15	31.40	7.70	4.02	1.18	0.0419	0.0155
3MOKP100	0.00	322.50	86.10	52.11	27.39	0.0000	0.0000
	0.05	178.90	42.60	25.11	8.70	0.0174	0.0131
	0.10	109.00	21.80	15.08	4.24	0.0387	0.0185
	0.15	72.00	11.00	9.88	2.04	0.0521	0.0226

* Average values based on 10 instances are reported per cell. p and g^* represents the number of criteria and the desired level of accuracy, respectively.

could be incorporated into the solution process and the algorithm could be stopped when the DM is satisfied with the incumbent point.

Table 8: The performance results of the approximation algorithms on the test instances with $p = 4$*

Problem	g^*	Nb. of models solved		Sol. time (in seconds)		Actual Level of Accuracy	
		DSA	DSA-m	DSA	DSA-m	DSA	DSA-m
4MOAP10x10	0.00	87.00	73.10	21.68	11.11	0.0000	0.0000
	0.05	74.00	44.00	15.50	6.01	0.0000	0.0048
	0.10	67.50	32.90	13.22	4.43	0.0144	0.0212
	0.15	58.40	24.90	10.63	3.34	0.0290	0.0316
4MOAP20x20	0.00	741.80	275.80	149.61	58.47	0.0000	0.0000
	0.05	606.90	165.00	118.17	33.84	0.0073	0.0081
	0.10	506.10	136.30	96.79	28.46	0.0358	0.0247
	0.15	368.60	115.10	69.51	23.93	0.0489	0.0499
4MOKP25	0.00	57.00	32.20	8.84	4.63	0.0000	0.0000
	0.05	39.90	22.40	5.88	3.31	0.0082	0.0102
	0.10	29.40	16.70	4.06	2.33	0.0252	0.0101
	0.15	22.20	11.90	3.10	1.79	0.0383	0.0103
4MOKP50	0.00	234.90	160.30	42.15	73.23	0.0000	0.0000
	0.05	122.40	62.70	21.97	11.24	0.0087	0.0129
	0.10	69.30	40.40	12.05	6.69	0.0263	0.0187
	0.15	49.80	25.60	8.55	4.15	0.0402	0.0377
4MOKP100	0.00	1099.00	495.20	246.47	125.78	0.0000	0.0000
	0.05	625.70	181.90	137.29	40.89	0.0122	0.0150
	0.10	451.90	128.80	96.81	28.49	0.0246	0.0144
	0.15	319.30	119.80	67.31	26.68	0.0426	0.0363

* Average values based on 10 instances are reported per cell. p and g^* represents the number of criteria and the desired level of accuracy, respectively.

References

Abbas, M., Chaabane, D., 2006. Optimizing a linear function over an integer efficient set. *European Journal of Operational Research* 174, 2, 1140–1161.

Benson, H., Sayin, S., 1994. Optimization over the efficient set: Four special cases. *Journal of Optimization Theory and Applications* 80, 1, 3–18.

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

Boland, N., Charkhgard, H., Savelsbergh, M., 2016. The l-shape search method for triobjective integer programming. *Mathematical Programming Computation* 8, 2, 217–251.

Boland, N., Charkhgard, H., Savelsbergh, M., 2017. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. *European journal of operational research* 260, 3, 904–919.

Dächert, K., Klamroth, K., 2015. A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems. *Journal of Global Optimization* 61, 4, 643–676.

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

Ehrgott, M., Gandibleux, X., 2002. Multiobjective Combinatorial Optimization — Theory, Methodology, and Applications, Springer US, Boston, MA. pp. 369–444.

Ehrgott, M., Gandibleux, X., 2004. Approximative solution methods for multiobjective combinatorial optimization. *Top* 12, 1, 1–63.

Ehrgott, M., Gandibleux, X., Przybylski, A., 2016. Exact methods for multi-objective combinatorial optimisation. In *Multiple Criteria Decision Analysis*. Springer, pp. 817–850.

Jorge, J.M., 2005. A bilinear algorithm for optimizing a linear function over the efficient set of a multiple objective linear programming problem. *Journal of Global Optimization* 31, 1, 1–16.

Jorge, J.M., 2009. An algorithm for optimizing a linear function over an integer efficient set. *European Journal of Operational Research* 195, 1, 98–103.

Kirlik, G., Sayın, S., 2014. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research* 232, 3, 479–488.

Kirlik, G., Sayın, S., 2015. Computing the nadir point for multiobjective discrete optimization problems. *Journal of Global Optimization* 62, 1, 79–99.

Köksalan, M., Lokman, B., 2015. Finding nadir points in multi-objective integer programs. *Journal of Global Optimization* 62, 1, 55–77.

Lokman, B., Köksalan, M., 2013. Finding all nondominated points of multi-objective integer programs. *Journal of Global Optimization* 57, 2, 347–365.

Nguyen, N.C., 1992. An algorithm for optimizing a linear function over the integer efficient set. Konrad-Zuse-Zentrum für Informationstechnik Berlin [ZIB].

Özlen, M., Azizoğlu, M., 2009. Multi-objective integer programming: a general approach for generating all non-dominated solutions. *European Journal of Operational Research* 199, 1, 25–35.

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.

Zhang, W., Reimann, M., 2014. A simple augmented epsilon-constraint method for multi-objective mathematical integer programming problems. *European Journal of Operational Research* 234, 1, 15–24.

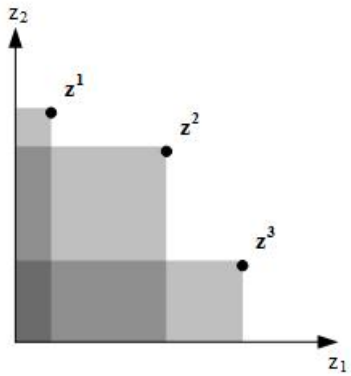

Fig. 1: The feasible set of $[P^4]$ in the criterion space, $Z_{[P^4]}$ $(n = 4, p = 2)$

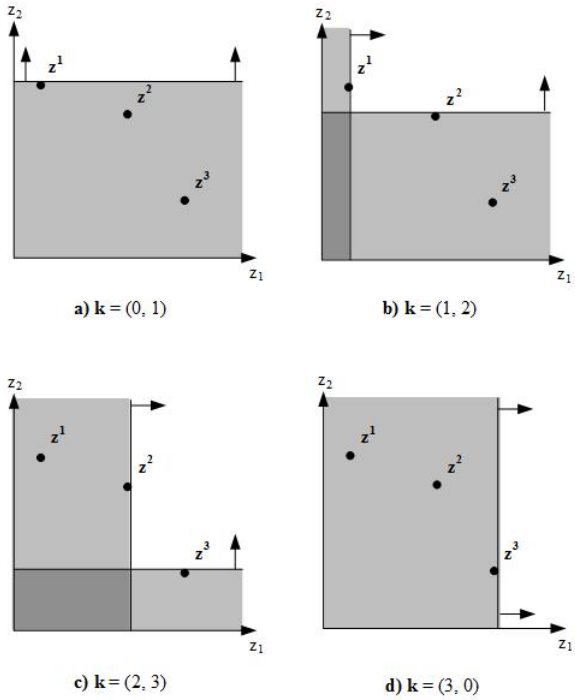

Fig. 2: The feasible sets of submodels $[P_{\mathbf{k}}]$ in the criterion space, $Z_{[P_{\mathbf{k}}]}(n = 4, p = 2)$