European Journal of Operational Research 271 (2018) 436–449

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Solving discrete multi-objective optimization problems using modified augmented weighted Tchebychev scalarizations

Tim Holzmann, J. C. Smith*

Department of Industrial Engineering, Clemson University, 100B Freeman Hall, Clemson SC, 29634-0020, United States

ARTICLE INFO

Article history:
Received 10 November 2017
Accepted 17 May 2018
Available online 25 May 2018

Keywords:
Multiple objective programming
Tchebychev norm
Computational optimization
Generating methods

ABSTRACT

In this paper we present the modified augmented weighted Tchebychev norm, which can be used to generate a complete efficient set of solutions to a discrete multi-objective optimization problem. We contribute a generating algorithm that will, without supervision, generate the entire non-dominated set for any number of objectives. To our knowledge, this is the first generating method for general discrete multi-objective problems that uses a variant of the Tchebychev norm. In a computational study, our algorithm's running times are comparable to previously proposed algorithms.

1. Introduction

In this paper we examine discrete multi-objective optimization problems in which there can exist any finite number of objective functions. The goal of this study is to provide an algorithm that generates a complete set of efficient solutions to the problem without user intervention. Section 1.1 presents concepts and notation used in this paper. Section 1.2 discusses literature most closely related to our study. Section 1.3 states our primary contributions and gives the organization of the paper.

1.1. Preliminaries

Consider the multi-objective problem:

$$\min\ [f_1(\mathbf{x}), \ldots, f_p(\mathbf{x})]$$
$$\text{subject to}: \mathbf{x} \in X,$$

where $p \in \mathbb{N}$, $X \subseteq \mathbb{R}^n$, and $f_k : X \to \mathbb{R}$ for each $k \in P = \{1, \ldots, p\}$. We call X the *decision set* and $\mathbb{R}^n$ the *decision space*. For any $X' \subseteq X$, let $\mathbf{f}(X') = \{\mathbf{z} \in \mathbb{R}^p : \exists \mathbf{x} \in X' \text{ such that } \mathbf{f}(\mathbf{x}) = \mathbf{z}\}$ be the image set of X' under the vector-valued function $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_p(\mathbf{x})]$. We call $Z = \mathbf{f}(X)$ the *solution set* and $\mathbb{R}^p$ the *objective space* or *solution space*.

We apply the standard Pareto preference ordering to the vectors in $\mathbb{R}^p$, as given in the following definitions.

Definition 1. Let $\mathbf{z}^a, \mathbf{z}^b \in \mathbb{R}^p$. Then

$\mathbf{z}^a \leqq \mathbf{z}^b \Leftrightarrow z^a_k \leq z^b_k, \ \forall k \in P$. We say $\mathbf{z}^a$ *weakly dominates* $\mathbf{z}^b$.

$\mathbf{z}^a \leq \mathbf{z}^b \Leftrightarrow \mathbf{z}^a \leqq \mathbf{z}^b$ and $\mathbf{z}^a \neq \mathbf{z}^b$. We say $\mathbf{z}^a$ *dominates* $\mathbf{z}^b$.

$\mathbf{z}^a < \mathbf{z}^b \Leftrightarrow z^a_k < z^b_k, \ \forall k \in P$. We say $\mathbf{z}^a$ *strongly dominates* $\mathbf{z}^b$.

Definition 2. $\mathbf{z} \in Z$ is

non-dominated $\Leftrightarrow \nexists \mathbf{z}' \in Z : \mathbf{z}' \leq \mathbf{z}$, and

weakly non-dominated $\Leftrightarrow \nexists \mathbf{z}' \in Z : \mathbf{z}' < \mathbf{z}$.

Definition 3. The nondominated set is $Z_N = \{\mathbf{z} \in Z : \mathbf{z}$ is non-dominated$\}$.

The necessary and sufficient conditions for our algorithm to operate correctly are given in Assumptions 1 and 2. However, since both conditions refer to Z_N, which is unknown at the outset, it may be difficult to verify that these assumptions hold. Therefore, Lemma 1 states a condition on Z that is sufficient for Assumptions 1 and 2 to hold.

Assumption 1. $|Z_N| < \infty$.

Assumption 2. $\forall \mathbf{z} \in Z$, there exists $\mathbf{z}^* \in Z_N$ such that $\mathbf{z}^* \leqq \mathbf{z}$.

Lemma 1. *If $|Z| < \infty$, then Assumptions 1 and 2 hold.*

Proof. Assume $|Z| < \infty$. By construction, $Z_N \subseteq Z$ so $|Z_N| \leq |Z| < \infty$, and Assumption 1 holds. To show Assumption 2 holds, consider $\mathbf{z}_1 \in Z$. If $\mathbf{z}_1$ is non-dominated, then $\mathbf{z}_1 \in Z_N$. Otherwise it is dominated, and there exists $\mathbf{z}_2 \in Z$: $\mathbf{z}_2 \leq \mathbf{z}_1$. If $\mathbf{z}_2$ is non-dominated, then $\mathbf{z}_2 \in Z_N$. Otherwise, $\exists \mathbf{z}_3 \in Z$: $\mathbf{z}_3 \leq \mathbf{z}_2$. Continuing in this manner, we can construct the sequence $\mathbf{z}_1 \geq \mathbf{z}_2 \geq \cdots \geq \mathbf{z}_d$ where $\{\mathbf{z}_i : 1 \leq i \leq d \leq |Z|\} \subseteq Z$

* Corresponding author.
E-mail address: jcsmith@clemson.edu (J.C. Smith).

https://doi.org/10.1016/j.ejor.2018.05.036
0377-2217/

and $\nexists \mathbf{z} \in Z : \mathbf{z} \leq \mathbf{z}_d$. Thus, $\mathbf{z}_d \in Z_N$, and Assumption 2 holds. This completes the proof. □

Definition 4. For $Z' \subseteq \mathbb{R}^p$, $\mathbf{z}^{\text{lb}} \in \mathbb{R}^p$ is a *(weak/·/strong) lower bound* if $(\mathbf{z}^{\text{lb}} \leqq \mathbf{z} / \mathbf{z}^{\text{lb}} \leq \mathbf{z} / \mathbf{z}^{\text{lb}} < \mathbf{z}), \forall \mathbf{z} \in Z'$.

Definition 5. For $Z' \subseteq \mathbb{R}^p$, $\mathbf{z}^{\text{ub}} \in \mathbb{R}^p$ is a *(weak/·/strong) upper bound* if $(\mathbf{z} \leqq \mathbf{z}^{\text{ub}} / \mathbf{z} \leq \mathbf{z}^{\text{ub}} / \mathbf{z} < \mathbf{z}^{\text{ub}}), \forall \mathbf{z} \in Z'$.

Lemma 2. *(See Sutherland, 1975.) Let $Z' \subseteq \mathbb{R}^p$. Then $|Z'| < \infty \Leftrightarrow Z'$ is compact and discrete.*

Corollary 3. *Z_N has a weak lower bound, $\mathbf{z}^{\text{lb}}$, and a strong upper bound, $\mathbf{z}^{\text{ub}}$.*

Proof. The result follows from the boundedness of Z_N given by Assumption 1 and Lemma 2. □

Definition 6. The *step size*, s, over Z_N is given by $\min_{\{\mathbf{z}^a, \mathbf{z}^b \in Z_N,\ k \in P\}} \{|z_k^a - z_k^b| : z_k^a \neq z_k^b\}$.

Lemma 4. *Let $r = \max_{k \in P}\{z_k^{\text{ub}} - z_k^{\text{lb}}\}$ where $\mathbf{z}^{\text{lb}}$ and $\mathbf{z}^{\text{ub}}$ are weak lower and strong upper bounds on Z_N, respectively, and let s be the step size over Z_N. Then, $|Z_N| > 1 \Rightarrow 0 < s < r$.*

Proof. Assume $|Z_N| > 1$. We note $0 < s$, since $s = \min_{\{\mathbf{z}^a, \mathbf{z}^b \in Z_N,\ k \in P\}} \{|z_k^a - z_k^b| : z_k^a \neq z_k^b\} > 0$. To show $s < r$, let $\mathbf{z}^a, \mathbf{z}^b \in Z_N$ where $\mathbf{z}^a \neq \mathbf{z}^b$. Then we have some $k \in P$ such that $z_k^a < z_k^b$, which gives us:

$$
\begin{aligned}
& z_k^{\text{lb}} + s \leq z_k^a + s \leq z_k^b < z_k^{\text{ub}} \\
\Rightarrow\ & z_k^{\text{lb}} + s < z_k^{\text{ub}} \\
\Leftrightarrow\ & s < z_k^{\text{ub}} - z_k^{\text{lb}} \leq r.
\end{aligned}
$$

Thus, $0 < s < r$, and the proof is complete. □

Definition 7. A point $\mathbf{z}^* \in Z$ is *k-preferred* for some $k \in P$ if $z_k^* \leq z_k$, $\forall \mathbf{z} \in Z$. A set $Z' \subseteq Z$ of preferred points is a *complete set of preferred points* if $\forall k \in P$, $\exists \mathbf{z}' \in Z'$ such that $\mathbf{z}'$ is k-preferred.

In addition to the above definitions for the solution space, we present the following definitions regarding efficiency in the decision space.

Definition 8. Point $\mathbf{x} \in X$ is *efficient* if $\mathbf{f(x)} \in Z_N$. The set $X_E = \{\mathbf{x} \in X : \mathbf{x} \text{is efficient}\}$ is the efficient set.

Definition 9. Any subset $X' \subseteq X_E$ such that $\mathbf{f}(X') = Z_N$ is a *complete set of efficient points*.

1.2. Literature review

Hwang and Masud (1979) classify techniques to find non-dominated points into three categories, depending on the level of decision maker interaction: a-priori methods, interactive, and a-posteriori (or generating) methods. Our approach falls into the a-posteriori category. These methods seek to generate the full set of non-dominated solutions, and then allow the decision maker (a-posteriori) to select any one among those solutions. Our review of relevant literature examines five families of generating methods.

1.2.1. ϵ-constraint methods

First presented by Haimes, Lasdon, and Wismer (1971), the ϵ-constraint method constrains all objectives but one. Those constraints are initially loose, and they are iteratively tightened. At each iteration, the algorithm minimizes the unconstrained objective, subject to the other objectives' constraints. The algorithm terminates when no feasible solutions remain under the tightened constraints.

Several improvements for the ϵ-constraint method have been proposed. Notably, while Haimes et al.'s algorithm may return points that are only weakly non-dominated, Ehrgott and Ruzika (2008) develop the elastic ϵ-constraint method, which avoids generating such weak solutions. Their method adds slack variables to the ϵ-constraints and then appends a small (possibly zero) weighted contribution of those slack variables to the objective. Both Hamacher, Pederson, and Ruzika (2007) and Özlen and Azizoğlu (2009) improve the algorithm using a weighting scheme that produces a lexicographical ordering for tightening the ϵ-constraints. Laumanns, Thiele, and Eckart (2006) and Kirlik and Sayin (2014) suggest methods to reduce the number of search regions, which reduces computation time considerably. Zhang and Reimann (2014) provide further improvements via early exits and bouncing steps. Mavrotas (2009) and Mavrotas and Florios (2013) implement the augmented ϵ-constraint method, combining the elastic method with lexicographically-ordered searches.

1.2.2. Klein and Hannan's family of methods

Klein and Hannan (1982) develop the dominated region exclusion approach, which iteratively identifies efficient solutions and then adds constraints to the problem that exclude those solutions and the regions dominated by them. Sylva and Crema (2004) enhance the method by using a sum of the objective functions, but they observe that the approach is impractical for large non-dominated sets, because the constraint set grows each iteration. Lokman and Köksalan (2013) improve the method with two algorithms. Their first algorithm reduces the number of constraints, thus improving the results in Sylva and Crema (2004), but still suffering for large non-dominated sets. Their second algorithm replaces a single subproblem with many variables and constraints, with multiple subproblems with fewer variables and constraints. Numerical tests suggest this second algorithm seems to overcome the computational limitations of previous efforts to generate Z_N by other methods in this family.

1.2.3. Two-phase methods

The two-phase method was first suggested by Ulungu and Teghem (1994). The concept is to find solutions on the convex hull of the non-dominated set by means of a weighted sum of the objective functions. Then a different method, e.g., branch-and-bound as given in Ulungu and Teghem (1994), is used to find the solutions in the interior of the hull. This algorithm is applied to various types of bi-objective problems, including knapsack (Visée, Teghem, Pirlot, & Ulungu, 1998), minimum spanning tree (Ramos, Alonso, Sicilia, & Gonzáles, 1998; Steiner & Radzik, 2008), minimum cost flow (Raith & Ehrgott, 2009) and assignment (Przybylski, Gandibleux, & Ehrgott, 2008; 2010; Ulungu & Teghem, 1994). While all use the weighted sum for the first phase, the authors vary their search algorithms for the second. Przybylski, Gandibleux, and Ehrgott (2010) demonstrate an application of this method to a problem having more than two objectives.

1.2.4. Branch-and-bound methods

White (1982) conceives of applying branch-and-bound algorithms to multi-objective combinatorial optimization problems. This approach conducts the search in the decision space rather than the solution space. Thus, it is the components of $\mathbf{x} \in X \subseteq \mathbb{R}^n$ that are branched, fathomed, and pruned. The benefit of this approach, as shown in Mavrotas and Diakoulaki (1998, 2005) and Vincent, Florian, Ruzika, and Przybylski (2013), is that it is capable of generating a full representation of Z_N even for combinatorial multi-objective mixed-integer programs (MOMIPs) where $|Z_N| \neq \infty$. Stidsen, Andersen, and Damman (2014) compare the performance of a bi-objective branch-and-bound algorithm against a two-phase algorithm on six types of combinatorial MOMIPs, demonstrating that the branch-and-bound method performed better than the two-phase method in five of the six types of problems.

Algorithm 1. Search algorithm using the MAWT norm.

Input : $\mathcal{I}$: Multi-objective problem instance, $\min\{[f_1(\mathbf{x}),\ldots,f_p(\mathbf{x})] : \mathbf{x} \in X\}$
s: Positive lower bound on single-objective step sizes in Z
$\mathbf{z}^{\text{ub}}$: Strong upper bound for Z_N, (i.e., $\mathbf{z}^{\text{ub}} > \mathbf{z},\ \forall z \in Z_N$)
Output: $\widehat{X}_E$: A complete efficient set
$\widehat{Z}_N$: The non-dominated set

Initialize

1 **for** $k \in P$ **do**
2 $\quad z_k^{\text{lb}} \leftarrow \min_{\mathbf{x}\in X} f_k(\mathbf{x})$ // At the end $\mathbf{z}^{\text{lb}} = [z_1^{\text{lb}},\ldots,z_p^{\text{lb}}]$
3 Choose $\epsilon \in (0, s/(p(r-s)))$ where $r = \max_{k\in P}\{z_k^{\text{ub}} - z_k^{\text{lb}}\}$ // e.g., $\epsilon \leftarrow s/(2p(r-s))$
4 $\widehat{X}_E \leftarrow \emptyset$, $\widehat{Z}_N \leftarrow \emptyset$, $j \leftarrow 0$; and $L \leftarrow \{\mathbf{z}^{\text{ub}}\}$

Main Loop

5 **while** $L \neq \emptyset$ **do**
6 $\quad j \leftarrow j+1$
7 $\quad$ Choose $\mathbf{z}^{(j)} \in L$; $L \leftarrow L \setminus \{\mathbf{z}^{(j)}\}$
8 $\quad$ **for** $k \in P$ **do**
9 $\quad\quad w_k \leftarrow 1/\left(\max\left\{s, z_k^{(j)} - z_k^{\text{lb}}\right\}\right)$ // At the end $\mathbf{w} = [w_1,\ldots,w_p]$
10 $\quad g(\mathbf{f}(\mathbf{x})) \leftarrow \|\mathbf{f}(\mathbf{x}) - \mathbf{z}^{\text{lb}}\|^{\mathbf{w},\epsilon}$
11 $\quad \mathbf{x}^{(j)} \leftarrow \text{argmin}_{\mathbf{x}\in X}\, g(\mathbf{f}(\mathbf{x}))$
12 $\quad$ **if** $g(\mathbf{f}(\mathbf{x}^{(j)})) < 1$ **then**
13 $\quad\quad$ **if** $\mathbf{f}(\mathbf{x}^{(j)}) \notin \widehat{Z}_N$ **then** $\widehat{X}_E \leftarrow \widehat{X}_E \cup \{\mathbf{x}^{(j)}\}$ and $\widehat{Z}_N \leftarrow \widehat{Z}_N \cup \{\mathbf{f}(\mathbf{x}^{(j)})\}$.
14 $\quad\quad$ **for** $k \in P$ **do**
15 $\quad\quad\quad$ **if** $f_k(\mathbf{x}^{(j)}) > z_k^{\text{lb}}$ **then**
16 $\quad\quad\quad\quad \mathbf{z}^{(j),-k} \leftarrow [z_1^{(j)},\ldots,z_{k-1}^{(j)}, f_k(\mathbf{x}^{(j)}), z_{k+1}^{(j)},\ldots,z_p^{(j)}]$
17 $\quad\quad\quad\quad L \leftarrow L \cup \{\mathbf{z}^{(j),-k}\}$

Terminate

18 **return** $\widehat{X}_E, \widehat{Z}_N$

Belotti, Soylu, and Wiecek (2013) demonstrate a multi-objective branch-and-bound algorithm for general integer problems.

1.2.5. Tchebychev norm methods

Bowman Jr. (1976) suggests the application of the Tchebychev norm for finding the non-dominated set of a multi-objective problem. Since the norm is known to only provide weakly efficient solutions, Steuer and Choo (1983) propose augmenting the base (Tchebychev) norm with a weighted sum of all the objectives multiplied by some constant coefficient. This augmented norm is applied successfully in interactive methods (Alves & Clímaco, 2000; Steuer, 1986; Wierzbicki, 1980) as well as in generating approximations to the non-dominated set (Klamroth, Tind, & Wiecek, 2002; Schandl, Klamroth, & Wiecek, 2002). We are unaware of any complete generating methods using the Tchebychev norm until the work of Dächert, Gorski, and Klamroth (2012). Dächert et al. demonstrate that, in general, no single coefficient will generate the entire non-dominated set for a problem instance. However, by adaptively adjusting the coefficient based on the search region, they use the augmented Tchebychev norm to generate Z_N for bi-objective problems. Their results exhibit comparable computation time to the ϵ-constraint method as implemented by Özpeynirci and Köksalan (2010), with about half the standard deviation.

1.3. Contribution and overview

Similar to the work of Dächert et al., our approach employs a variant of the Tchebychev norm, adaptively adjusting the search parameters without human intervention to generate Z_N. Our contribution is to propose a modification to the augmented Tchebychev norm that affords a simpler computation of the coefficient for the augmentation term. We also give an algorithm to generate the non-dominated set for any number of objectives. To our knowledge, this is the first generating method that uses a variant of the Tchebychev norm to solve the non-dominated set for discrete multi-objective optimization problems with any number of objectives. By contrast, the approach in Dächert et al. (2012) employs the (more traditional) augmented weighted Tchebychev norm with a sophisticated parameter scheme in order to generate a complete efficient set of solutions. The authors additionally provide a rigorous analysis of the parameters employed in the augmented weighted Tchebychev norm. The goal in their work is to find the largest possible augmentation parameter that still guarantees that all nondominated points can be found, noting that larger values of the augmentation term tend to avoid numerical difficulties in optimization. By contrast, the structure of the variant of the Tchebychev norm that we propose in this paper will allow us to state a simpler weighting scheme that can be used to automatically generate a complete efficient set of solutions with any number of criteria.

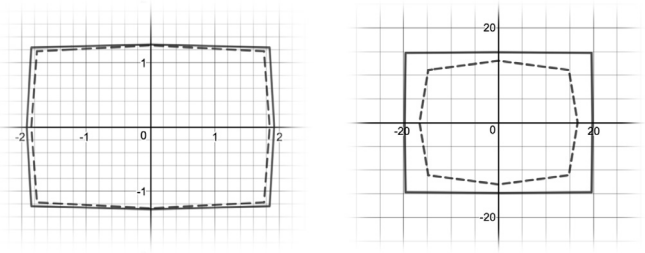
In Section 2 we present our proposed modification and our algorithm. We prove the correctness and finiteness of our algorithm in Section 3, and give an example in Section 4. In Section 5 we present computational results. The paper concludes in Section 6 with consideration of future research areas.

2. Algorithm

We provide Algorithm 1 in this section, which generates a complete efficient set without requiring user intervention. Our approach uses a modification of the augmented weighted Tchebychev norm (Dächert et al., 2012; Steuer & Choo, 1983). The augmented weighted Tchebychev norm is given by:

$$\|\mathbf{z}\|_{\epsilon}^{\mathbf{w}} = \|\mathbf{z}\|_{\infty}^{\mathbf{w}} + \epsilon\|\mathbf{z}\|_1 = \max_{k\in P}\{w_k|z_k|\} + \epsilon\sum_{k\in P}|z_k|, \tag{1}$$

where $\mathbf{w} \geq \mathbf{0}$ and $\epsilon > 0$. Our proposed modification incorporates the same weighting into the augmentation term as is used on the base



(a) $\mathbf{w} = \left[\frac{1}{2}, \frac{3}{4}\right]$ and $\epsilon = 0.04$. (b) $\mathbf{w} = \left[\frac{1}{20}, \frac{1}{15}\right]$ and $\epsilon = 0.01$.

Fig. 1. Level curves for $\|\mathbf{z}\|_{\epsilon}^{\mathbf{w}} = 1$ (dashed) and $\|\mathbf{z}\|^{\mathbf{w},\epsilon} = 1$ (solid).

Tchebychev norm. Thus, the modified augmented weighted Tchebychev (MAWT) norm we propose is:

$$\|\mathbf{z}\|^{\mathbf{w},\epsilon} = \|\mathbf{z}\|_{\infty}^{\mathbf{w}} + \epsilon\|\mathbf{z}\|_{1}^{\mathbf{w}} = \max_{k\in P}\{w_k|z_k|\} + \epsilon\sum_{k\in P} w_k|z_k|. \tag{2}$$

Kaliszewski (1987) also examines a modification of the Tchebychev norm similar to (2) in some respects; however, there are distinct differences. Kaliszewski (2000) considers a more general Tchebychev-norm variant which, with the appropriate selection of the parameters, may be reduced to (2). These papers aim at interactive (a-posteriori) multi-objective methods and do not provide the contribution that is the focus of our paper. We distinguish between the norms (1) and (2) by placing the ϵ parameter in the superscript in (2) rather than the subscript as in (1). Fig. 1 compares level curves for (1) and (2).

In the initialization phase of Algorithm 1, we determine a weak lower bound, $\mathbf{z}^{\text{lb}}$, by minimizing each objective individually to give a complete set of preferred points, and selecting their respective preferred objective values (lines 1–2). Thus $\mathbf{z}^{\text{lb}} \leqq \mathbf{z} < \mathbf{z}^{\text{ub}}, \forall \mathbf{z} \in Z_N$, where $\mathbf{z}^{\text{ub}}$ is a strong upper bound over Z_N given as input to the algorithm. We choose not to compute $\mathbf{z}^{\text{ub}}$ because it is generally non-trivial to solve for a upper bound on Z_N (Ehrgott, 2005). However, a strong upper bound for Z_N is often apparent for specific problem classes (e.g., $(-1, \ldots, -1)$ for a multi-objective max flow problem).

Next, with $\mathbf{z}^{\text{lb}}$ and $\mathbf{z}^{\text{ub}}$, we solve $r = \max_{k\in P}\{z_k^{\text{ub}} - z_k^{\text{lb}}\}$, and we select a parameter ϵ such that:

$$0 < \epsilon < \frac{s}{p(r-s)}, \tag{3}$$

where $s > 0$ is a lower bound on the step size (given as an input parameter). Recall from Lemma 4 that either $s < r$, or $|Z_N| \leq 1$ (moreover, $|Z_N| = 0 \Rightarrow X = \emptyset$ by Assumption 2, and $|Z_N| = 1 \Rightarrow Z_N = \{\mathbf{z}^{\text{lb}}\}$ by construction of $\mathbf{z}^{\text{lb}}$). Finally we initialize our lists of efficient and non-dominated solutions $\widehat{X}_E$ and $\widehat{Z}_N$ as empty sets, our iteration counter $j = 0$, and list of search regions, L, to contain a single entry.

Each iteration j of the main loop begins in line 5 by choosing a vector, $\mathbf{z}^{(j)} \in L$, and removing it from L. The vector $\mathbf{z}^{(j)}$ uniquely defines our search region for that iteration, $B^{(j)} = \{\mathbf{z} \in \mathbb{R}^p : \mathbf{z}^{\text{lb}} \leqq \mathbf{z} < \mathbf{z}^{(j)}\}$, a hyperbox with weak lower and strong upper bounds given by the points $\mathbf{z}^{\text{lb}}$ and $\mathbf{z}^{(j)}$, respectively. To determine whether $B^{(j)} \cap Z_N = \emptyset$, we will use (2), for which we require parameters $\mathbf{w}$ and ϵ. We have the ϵ parameter from line 3. To find $\mathbf{w}$, we consider the side-lengths of $B^{(j)}$, given by $(z_k^{(j)} - z_k^{\text{lb}})$ for each $k \in P$. Then we take:

$$\mathbf{w} = \left[\frac{1}{\max\{s, z_1^{(j)} - z_1^{\text{lb}}\}}, \ldots, \frac{1}{\max\{s, z_p^{(j)} - z_p^{\text{lb}}\}}\right]. \tag{4}$$

Intuitively, Eq. (4) scales the side lengths to unit values, with the caveat that all side lengths are at least s to prevent division by zero.

In line 10, we use the MAWT norm of (2) to define our scalarizing function, $g : \mathbb{R}^p \to \mathbb{R}$, and in line 11 we use g to solve

$$\min_{\mathbf{x}\in X} g(\mathbf{f}(\mathbf{x})) = \|\mathbf{f}(\mathbf{x}) - \mathbf{z}^{\text{lb}}\|^{\mathbf{w},\epsilon}, \tag{5}$$

where ϵ and $\mathbf{w}$ are given by (3) and (4), respectively. Note that (5) is feasible as long as $X \neq \emptyset$ due to Assumption 2, and it is bounded because the objective is non-negative. If the solution, $\mathbf{x}^{(j)}$, has objective value $g(\mathbf{f}(\mathbf{x}^{(j)})) < 1$, then we claim $\mathbf{f}(\mathbf{x}^{(j)}) \in B^{(j)} \cap Z_N$. In that case, line 13 adds the solutions $\mathbf{x}^{(j)}$ and $\mathbf{f}(\mathbf{x}^{(j)})$ to $\widehat{X}_E$ and $\widehat{Z}_N$, respectively, unless $\mathbf{f}(\mathbf{x}^{(j)})$ already belongs to $\widehat{Z}_N$. Lines 14–17 then spawn "child" upper bounds, $\{\mathbf{z}^{(j),-k} \in \mathbb{R}^p : \mathbf{z}^{(j),-k} = [z_1^{(j)}, \ldots, z_{k-1}^{(j)}, f_k(\mathbf{x}^{(j)}), z_{k+1}^{(j)}, \ldots, z_p^{(j)}], k \in P\}$ and add them to L. Otherwise, if $g(\mathbf{f}(\mathbf{x}^{(j)})) \geq 1$, we claim $B^{(j)} \cap Z_N = \emptyset$, and Algorithm 1 proceeds to the next iteration. We prove both of the foregoing claims in Section 3.

Algorithm 1 terminates when no remaining vectors are in L, and we claim (and prove in Section 3) that $\widehat{X}_E$ is a complete efficient set.

3. Algorithm proof

We begin this section by demonstrating that Algorithm 1 generates only efficient solutions.

Proposition 1. *The solution, $\mathbf{x}^{(j)}$, obtained in line 11 of Algorithm 1 is efficient.*

Proof. Let $\mathbf{x}^{(j)}$ optimize (5) with ϵ and $\mathbf{w}$ given as in lines 3 and 9, respectively. By contradiction, assume $\exists \mathbf{z} \in Z$ such that $\mathbf{z} \leq \mathbf{f}(\mathbf{x}^{(j)})$. Then

$$\begin{aligned}\|\mathbf{f}(\mathbf{x}^{(j)}) - \mathbf{z}^{\text{lb}}\|^{\mathbf{w},\epsilon} &= \max_{k\in P}\left\{w_k(f_k(\mathbf{x}^{(j)}) - z_k^{\text{lb}})\right\} + \epsilon\sum_{k\in P} w_k(f_k(\mathbf{x}^{(j)}) - z_k^{\text{lb}}) \\ &> \max_{k\in P}\left\{w_k(z_k - z_k^{\text{lb}})\right\} + \epsilon\sum_{k\in P} w_k(z_k - z_k^{\text{lb}}) \\ &= \|\mathbf{z} - \mathbf{z}^{\text{lb}}\|^{\mathbf{w},\epsilon},\end{aligned}$$

which contradicts the optimality of $\mathbf{x}^{(j)}$. This completes the proof. □

Remark 1. The inequality in the proof of Proposition 1 is valid because $\mathbf{z}^{\text{lb}} \leq \mathbf{f}(\mathbf{x}^{(j)})$. However, if $\epsilon w_k(f_k(\mathbf{x}^{(j)}) - z_k^{\text{lb}})$ becomes very small for all $k \in P$, then round-off errors may allow a non-dominated solution to be returned. If we assume $s < r/2$ and $p \geq 2$, then recalling $\epsilon < s/(p(r-s))$ from (3), we can mitigate the potential for numerical issues by maintaining ϵ^2 to be sufficiently large, since $w_k(f_k(\mathbf{x}^{(j)}) - z_k^{\text{lb}}) \geq s/r \geq s/(p(r-s)) > \epsilon$. We therefore suggest monitoring the condition that $\epsilon s/r$ should be significantly larger than machine epsilon, and since $\epsilon s/r < s^2/(pr^2)$, this also bounds the granularity achievable using the MAWT norm.

The following theorem states that the evaluation of line 12 properly determines whether $B^{(j)} \cap Z_N = \emptyset$.

Theorem 5. *Let $\mathbf{z}^{\text{lb}} < \mathbf{z}^{(j)} \leqq \mathbf{z}^{\text{ub}}$ and $\mathbf{x} \in X$. Then $\mathbf{f}(\mathbf{x}) \in B^{(j)}$ if and only if $g(\mathbf{f}(\mathbf{x})) < 1$.*

Proof. Assume $g(\mathbf{f}(\mathbf{x})) < 1$. We know $\mathbf{z}^{\text{lb}} \leqq \mathbf{f}(\mathbf{x})$ by our construction of $\mathbf{z}^{\text{lb}}$, so we have:

$$\begin{aligned}1 &> \max_{k\in P}\left\{\frac{(f_k(\mathbf{x}) - z_k^{\text{lb}})}{(z_k^{(j)} - z_k^{\text{lb}})}\right\} + \epsilon\sum_{k\in P}\frac{(f_k(\mathbf{x}) - z_k^{\text{lb}})}{(z_k^{(j)} - z_k^{\text{lb}})} \\ &\geq \max_{k\in P}\left\{\frac{(f_k(\mathbf{x}) - z_k^{\text{lb}})}{(z_k^{(j)} - z_k^{\text{lb}})}\right\} \\ \Rightarrow 1 &> \frac{f_k(\mathbf{x}) - z_k^{\text{lb}}}{z_k^{(j)} - z_k^{\text{lb}}} \quad \forall k \in P\end{aligned}$$

$$\Leftrightarrow z_k^{(j)} - z_k^{\text{lb}} > f_k(\mathbf{x}) - z_k^{\text{lb}} \quad \forall k \in P$$
$$\Leftrightarrow \mathbf{z}^{(j)} > \mathbf{f}(\mathbf{x}), \text{ which guarantees } \mathbf{f}(\mathbf{x}) \in B^{(j)}.$$

Conversely, assume $\mathbf{f}(\mathbf{x}) \in B^{(j)}$ for some $\mathbf{x} \in X$. Then, since $\mathbf{f}(\mathbf{x}) \in B^{(j)}$ we know $\mathbf{f}(\mathbf{x}) < \mathbf{z}^{(j)}$. This gives us $\mathbf{z}^{\text{lb}} \leqq \mathbf{f}(\mathbf{x}) < \mathbf{z}^{(j)} \leqq \mathbf{z}^{\text{ub}}$, and so

$$\begin{aligned} g(\mathbf{f}(\mathbf{x})) &= \max_{k\in P}\left\{\frac{(f_k(\mathbf{x}) - z_k^{\text{lb}})}{(z_k^{(j)} - z_k^{\text{lb}})}\right\} + \epsilon \sum_{k\in P} \frac{(f_k(\mathbf{x}) - z_k^{\text{lb}})}{(z_k^{(j)} - z_k^{\text{lb}})} \\ &\leq \max_{k\in P}\left\{\frac{(z_k^{(j)} - z_k^{\text{lb}}) - s}{(z_k^{(j)} - z_k^{\text{lb}})}\right\} + \epsilon \sum_{k\in P} \frac{(z_k^{(j)} - z_k^{\text{lb}}) - s}{(z_k^{(j)} - z_k^{\text{lb}})} \\ &\leq \max_{k\in P}\left\{\frac{(z_k^{\text{ub}} - z_k^{\text{lb}}) - s}{(z_k^{\text{ub}} - z_k^{\text{lb}})}\right\} + \epsilon \sum_{k\in P} \frac{(z_k^{\text{ub}} - z_k^{\text{lb}}) - s}{(z_k^{\text{ub}} - z_k^{\text{lb}})} \\ &\leq \frac{r-s}{r} + \epsilon \ \frac{p(r-s)}{r} \\ &< \frac{r-s}{r} + \left(\frac{s}{p(r-s)}\right)\frac{p(r-s)}{r} = \frac{r-s}{r} + \frac{s}{r} = 1. \end{aligned}$$

Thus $g(\mathbf{f}(\mathbf{x})) < 1$, which completes the proof. □

Corollary 6. *If $g(\mathbf{f}(\mathbf{x}^{(j)})) < 1$ then $\mathbf{f}(\mathbf{x}^{(j)}) \in B^{(j)} \cap Z_N$. Otherwise $B^{(j)} \cap Z_N = \emptyset$.*

Proof. We note that $g(\mathbf{f}(\mathbf{x}^{(j)})) \geq 1 \Leftrightarrow g(\mathbf{f}(\mathbf{x})) \geq 1, \forall \mathbf{x} \in X$. Proposition 1 and Theorem 5 then establish the corollary. □

Our next result considers how Algorithm 1 continues the search over $(B^{(j)} \cap Z_N) \backslash \{\mathbf{f}(\mathbf{x}^{(j)})\}$.

Lemma 7. *Let $\mathbf{z}^{\text{lb}} < \mathbf{z}^{(j)} \leqq \mathbf{z}^{\text{ub}}$ for some iteration j, and $\mathbf{z} \in (B^{(j)} \cap Z_N) \backslash \{\mathbf{f}(\mathbf{x}^{(j)})\}$. Then $\exists \mathbf{z}^{(j),-k} = [z_1^{(j)}, \ldots, z_{k-1}^{(j)}, f_k(\mathbf{x}^{(j)}), z_{k+1}^{(j)}, \ldots, z_p^{(j)}]$ for some $k \in P$, such that $\mathbf{z}^{\text{lb}} < \mathbf{z}^{(j),-k} \leq \mathbf{z}^{\text{ub}}$ and $\mathbf{z}^{(j),-k}$ is added to L in line 17.*

Proof. Because $\mathbf{z} \in B^{(j)}$, we have $\mathbf{z} < \mathbf{z}^{(j)}$, and $\mathbf{z} \in Z_N$ implies $\exists k \in P$ such that $z_k^{\text{lb}} \leq z_k < f_k(\mathbf{x}^{(j)}) < z_k^{(j)}$. This satisfies the condition of line 15, so the point $\mathbf{z}^{(j),-k}$ is added to L. Moreover, $\mathbf{z}^{\text{lb}} \leq \mathbf{z} < \mathbf{z}^{(j),-k} \leq \mathbf{z}^{(j)} \leqq \mathbf{z}^{\text{ub}}$, and the proof is complete. □

As a result of Lemma 7, when a new efficient solution is identified in the algorithm, we generate "child" upper bounds of the form $\mathbf{z}^{(j),-k}$ and add them to L. We skip in line 17 any k for which $\mathbf{f}(\mathbf{x}^{(j)})$ is a k-preferred point, because that child, $\mathbf{z}^{(j')} = \mathbf{z}^{(j),-k}$ yields an empty corresponding search region, $B^{(j')} = \emptyset$. The requirement that $\mathbf{z}^{\text{lb}} < \mathbf{z}^{(j),-k} \leq \mathbf{z}^{\text{ub}}$ provides the sufficient conditions to apply Theorem 5 when the child is chosen in a future iteration.

We conclude this section with arguments on the finiteness and completeness of the returned sets $\widehat{Z}_N$ and $\widehat{X}_E$.

Proposition 2. *Algorithm 1 terminates finitely, having the number of iterations bounded by $O(p^{|Z_N|})$.*

Proof. Consider a tree graph structure, where each node corresponds to one iteration. The tree branches correspond to the child upper bounds $z^{(j),-k}$, for $k \in P$. Each node may have up to p branches; thus, the number of nodes at level of i of the tree is no more than p^i. To bound the tree depth, consider one $\mathbf{z}^{(j),-k} = \mathbf{z}^{(j')}$, and its associated search region $B^{(j')}$. Note $\mathbf{z}^{(j')} \leq \mathbf{z}^{(j)} \Rightarrow B^{(j')} \subset B^{(j)}$. Moreover, $\mathbf{f}(\mathbf{x}^{(j)}) \neq \mathbf{z}^{(j')} \Rightarrow \mathbf{f}(\mathbf{x}^{(j)}) \notin B^{(j')} \Rightarrow (B^{(j+1)} \cap Z_N) \subset (B^{(j)} \cap Z_N)$. Therefore, on every path from the root, the search regions diminish by at least one non-dominated point at each node. Hence, no path from the root node to a leaf can contain more than $|Z_N| + 1$ nodes, and so the tree depth is bounded by $|Z_N| + 1$. The total number of nodes in the full tree is then bounded by $\sum_{i=0}^{|Z_N|} p^i$, i.e., $O(p^{|Z_N|})$ nodes. This completes the proof. □

Remark 2. The bounds generated in Proposition 2 are loose and do not reflect the state-of-the-art. This is due to the simple decomposition of the search elements in Algorithm 1. As we find in our computational experiments, Algorithm 1 solves many more IPs than other recent generating algorithms. Better results may be obtained using more advanced decomposition schemes such as those of Klamroth, Lacour, and Vanderpooten (2015) and Dächert, Klamroth, Lacour, and Vanderpooten (2017). We discuss these improvements further in Section 5.3. The decomposition proposed in Algorithm 1, however, is simpler for our initial exposition and sufficient for the proof of finiteness.

Theorem 8. *At termination, Algorithm 1 returns $\widehat{Z}_N = Z_N$.*

Proof. To show $\widehat{Z}_N \subseteq Z_N$, note that only solutions to (5) are added to $\widehat{Z}_N$ in line 13, and by Proposition 1, those solutions are non-dominated. To show that $Z_N \subseteq \widehat{Z}_N$, consider some $\mathbf{z} \in Z_N$, and assume that $\mathbf{z} \in B^{(j)}$ for some $j \in \mathbb{N}$. Since $B^{(j)} \cap Z_N \neq \emptyset$, by Corollary 6, $g(\mathbf{f}(\mathbf{x}^{(j)})) < 1$, meeting the condition of line 12. Now we have two cases:

Case 1: $\mathbf{z} = \mathbf{f}(\mathbf{x}^{(j)})$: Then line 13 guarantees $\mathbf{z} \in \widehat{Z}_N$.

Case 2: $\mathbf{z} \neq \mathbf{f}(\mathbf{x}^{(j)})$: Then from Lemma 7, we know some $\mathbf{z}^{(j')} = \mathbf{z}^{(j),-k}$ is added to L in line 17, where $\mathbf{z} < \mathbf{z}^{(j')} \leq \mathbf{z}^{\text{ub}} \Rightarrow \mathbf{z} \in B^{(j')}$ and the algorithm continues.

Observe from these cases that Algorithm 1 cannot terminate unless $\mathbf{z} = \mathbf{f}(\mathbf{x}^{(j)})$ for some j. Then because Proposition 2 shows the algorithm must terminate, we have $\mathbf{z} \in \widehat{Z}_N$ at termination, and the proof is complete. □

Note that every time an element $\mathbf{f}(\mathbf{x}^{(j)})$ is added to $\widehat{Z}_N$ in line 13, its pre-image $\mathbf{x}^{(j)}$ is also added to $\widehat{X}_E$, which guarantees by Theorem 8 that $\widehat{X}_E$ is a complete efficient set at termination.

4. Example

In this section, we illustrate Algorithm 1 on the following bounded, non-convex bi-objective problem:

$$\begin{aligned} &\min \quad [f_1(\mathbf{x}) = x_1, \ \ f_2(\mathbf{x}) = x_2] \\ &\text{subject to} : (x_1, x_2) \in (X_1 \cup X_2) \cap \mathbb{Z}^2, \text{ where} \end{aligned}$$
$$X_1 = \left\{(x_1, x_2) \in \mathbb{R}^2 : \sqrt{\frac{(x_1-7)^2}{3} + \frac{(x_2-2)^2}{1.5}} \leq 1\right\}$$
$$X_2 = \left\{(x_1, x_2) \in \mathbb{R}^2 : \frac{3}{2}x_1 + x_2 \geq 9, \ \ -\frac{2}{5}x_1 - x_2 \geq -8, \ \ x_2 \geq 4\right\}.$$

A lower bound on the step size is $s = 1$. A strict upper bound on Z_N is $\mathbf{z}^{\text{ub}} = [11, 8]$.

In Fig. 2 we show the regions X_1 and X_2 as shaded. The point $\mathbf{z}^{\text{lb}}$ is marked with a star. Dominated points are dull circles, non-dominated points are bold circles, and the non-dominated solution to (5) in each iteration, $\mathbf{x}^{(j)}$, is a filled circle. Also for each iteration, the upper bound, $\mathbf{z}^{(j)}$, is a filled diamond, and the child upper bounds spawned from that iteration, $\mathbf{z}^{(j),-k}$, are empty diamonds. Finally, for each iteration we show the level curves $\{\mathbf{z} \in \mathbb{R}^p : g(\mathbf{z}) = g(\mathbf{f}(\mathbf{x}^{(j)}))\}$ (solid line) and $\{\mathbf{z} \in \mathbb{R}^p : g(\mathbf{z}) = 1\}$ (dashed line).

Initialization: (See Fig. 2(a).) We initialize by first solving for the lower bound, $\mathbf{z}^{\text{lb}} = (2, 1)$. Then, we choose $\epsilon = 0.05 < 1/16 = s/p(r-s)$. Finally, we initialize $L = \{(11, 8)\}$.

Iteration 1: (See Fig. 2(b).) We choose $\mathbf{z}^{(1)} = (11, 8)$ and remove this vector from L. Solving (5), we get $\mathbf{x}^{(1)} = \mathbf{f}(\mathbf{x}^{(1)}) = (4, 2)$ with $g(\mathbf{f}(\mathbf{x}^{(1)})) \approx 0.2405$. Since $g(\mathbf{f}(\mathbf{x}^{(1)})) < 1$, we add $\mathbf{x}^{(1)}$ to our solution set, so that $\widehat{X}_E = \widehat{Z}_N = \{(4, 2)\}$. Next, we find the child upper bounds $\mathbf{z}^{(1),-1} = (4, 8)$ and $\mathbf{z}^{(1),-2} = (11, 2)$ and add them to $L = \{(4, 8), (11, 2)\}$.

Iteration 2: (See Fig. 2(c).) We choose $\mathbf{z}^{(2)} = (4, 8)$ and remove this vector from L. Solving (5), we get $\mathbf{x}^{(2)} = \mathbf{f}(\mathbf{x}^{(2)}) = (3, 5)$ with $g(\mathbf{f}(\mathbf{x}^{(2)})) = 0.625$. Since $g(\mathbf{f}(\mathbf{x}^{(2)})) < 1$, we add $\mathbf{x}^{(2)}$ to our solution

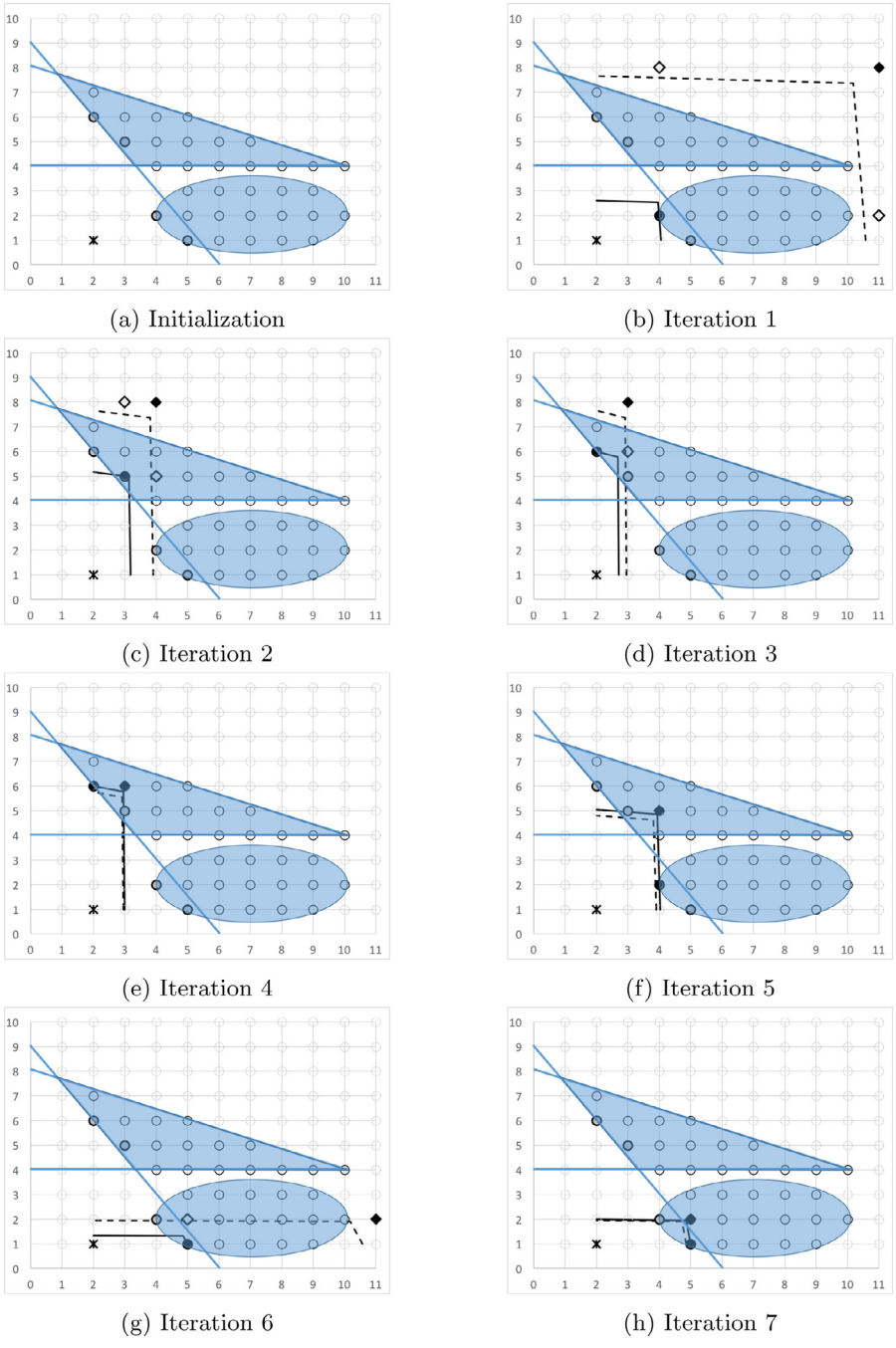


Fig. 2. Illustration of Algorithm 1 iterations for Section 4 example problem.

set, so that $\widehat{X}_E = \widehat{Z}_N = \{(3,5),(4,2)\}$. Next, we find the child upper bounds $\mathbf{z}^{(2),-1} = (3,8)$ and $\mathbf{z}^{(2),-2} = (4,5)$ and add them to $L = \{(11,2),(3,8),(4,5)\}$.

Iteration 3: (See Fig. 2(d).) We choose $\mathbf{z}^{(3)} = (3,8)$ and remove this vector from L. Solving (5), we get $\mathbf{x}^{(3)} = \mathbf{f}(\mathbf{x}^{(3)}) = (2,6)$ with $g(\mathbf{f}(\mathbf{x}^{(3)})) = 0.75$. Since $g(\mathbf{f}(\mathbf{x}^{(3)})) < 1$, we add $\mathbf{x}^{(3)}$ to our solution set, so that $\widehat{X}_E = \widehat{Z}_N = \{(2,6),(3,5),(4,2)\}$. Note $\mathbf{f}(\mathbf{x}^{(3)})$ is 1-preferred because $f_1(\mathbf{x}^{(3)}) = z_1^{\text{lb}}$, so we do not generate a 1-child for $\mathbf{z}^{(3)}$. However, we still add the 2-child, $\mathbf{z}^{(3),-2} = (3,6)$, to $L = \{(11,2),(4,5),(3,6)\}$.

Iteration 4: (See Fig. 2(e).) We choose $\mathbf{z}^{(4)} = (3,6)$ and remove this vector from L. Solving (5), we get $\mathbf{x}^{(4)} = \mathbf{f}(\mathbf{x}^{(4)}) = (2,6)$ with $g(\mathbf{f}(\mathbf{x}^{(4)})) = 1.05$. Since $g(\mathbf{f}(\mathbf{x}^{(4)})) > 1$, we move on to the next iteration, selecting a new upper bound from $L = \{(11,2),(4,5)\}$.

Iteration 5: (See Fig. 2(f).) We choose $\mathbf{z}^{(5)} = (4,5)$ and remove this vector from L. Solving (5), we get $\mathbf{x}^{(5)} = \mathbf{f}(\mathbf{x}^{(5)}) = (4,2)$ with $g(\mathbf{f}(\mathbf{x}^{(5)})) = 1.0625$. Since $g(\mathbf{f}(\mathbf{x}^{(5)})) > 1$, we move on to the next iteration, selecting a new upper bound from $L = \{(11,2)\}$.

Iteration 6: (See Fig. 2(g).) We choose $\mathbf{z}^{(6)} = (11,2)$ and remove this vector from L. Solving (5), we get $\mathbf{x}^{(6)} = \mathbf{f}(\mathbf{x}^{(6)}) = (5,1)$ with $g(\mathbf{f}(\mathbf{x}^{(6)})) = 0.35$. Since $g(\mathbf{f}(\mathbf{x}^{(6)})) < 1$, we add $\mathbf{x}^{(6)}$ to our solution set, so that $\widehat{X}_E = \widehat{Z}_N = \{(2,6),(3,5),(4,2),(5,1)\}$. Because $f_2(\mathbf{x}^{(6)}) = z_2^{\text{lb}}$, we add only $\mathbf{z}^{(6),-1} = (5,2)$ to $L = \{(5,2)\}$.

Iteration 7: (See Fig. 2(h).) We choose $\mathbf{z}^{(7)} = (5,2)$ and remove this vector from L. Solving (5), we get $\mathbf{x}^{(7)} = \mathbf{f}(\mathbf{x}^{(7)}) = (5,1)$ with $g(\mathbf{f}(\mathbf{x}^{(7)})) = 1.05$. Since $g(\mathbf{f}(\mathbf{x}^{(7)})) > 1$ and L is empty, the algorithm terminates, returning $\widehat{X}_E = \widehat{Z}_N = \{(2,6),(3,5),(4,2),(5,1)\}$.

Table 1
$|Z_N|$ for instances in our testbed.

		Instance number									
p	n	1	2	3	4	5	6	7	8	9	10
3	10 (KP)	10	19	18	22	6	27	15	14	8	14
	25 (AP)	20	22	12	10	10	22	11	6	9	15
4	8 (KP)	16	22	12	13	54	17	21	11	19	13
	9 (AP)	3	3	3	2	6	5	5	4	6	4
	10 (KP)	27	45	22	57	47	77	15	31	32	64
	20 (KP)	110	160	664	611	482	708	451	328	197	372
	25 (AP)	31	26	43	25	35	23	32	28	16	17
	64(AP)	185	212	292	389	264	310	362	190	254	141
	100 (AP)	1359	1186	1043	310	524	546	999	1378	500	837
5	10 (KP)	76	96	57	175	48	86	72	49	82	59
	25 (AP)	24	36	73	60	52	35	81	18	35	41
6	10 (KP)	101	115	97	166	153	91	124	102	45	86
	25 (AP)	67	66	49	76	38	75	48	50	46	67

5. Computational study

5.1. Implementation

We conducted an empirical study to compare the efficiency of searches using the MAWT norm versus those performed by recently published algorithms. For this study, we refer to our implementation of Algorithm 1 as "BOX." For comparison, we implemented the ϵ-constraint algorithm of Kirlik and Sayin (2014), which we refer to as "KS," and we implemented the recursive ϵ-constraint algorithm of Özlen, Burton, and MacRae (2014), which we refer to as "OBM." Our implementations were written in Python 2.7, and we used Gurobi 7.5 to solve the scalarized sub-problems, $\min_{\mathbf{x}\in X} g(\mathbf{f}(\mathbf{x}))$, in each iteration.

For the study we ran 60 multi-objective cardinality-constrained knapsack and 70 multi-objective assignment problem instances. The number of non-dominated solutions for these 130 instances is given in Table 1. All source code and problem instances are available as supplemental data in Holzmann and Smith (2017).

We generated the knapsack problems by varying the number of binary decision variables $n \in \{8, 10, 20\}$ while holding the number of objectives $p = 4$, and then we varied the number of objectives $p \in \{3, 5, 6\}$ while holding decision variables $n = 10$. For each of the six (n, p) combinations we generated 10 instances, giving 60 instances in total. Our knapsack problems were formulated as:

$$\begin{aligned} \min \quad & [-\mathbf{w}_1^T\mathbf{x}, \ldots, -\mathbf{w}_p^T\mathbf{x}] \\ \text{subject to :} \quad & \sum_{i=1}^{n} x_i \le n/2 \\ & \mathbf{x} \in \{0,1\}^n, \end{aligned}$$

where each $\mathbf{w}_k \in \{0, \ldots, 10\}^n$ for $k \in P$ was randomly generated using a discrete uniform distribution. To initialize BOX, we used $\mathbf{z}^{\text{ub}} = 1\mathbf{e}$, and $s = 1$, and we chose $\epsilon = 1/pr < s/(p(r-s))$. Then observing that the minimum possible objective in any component is $-10n/2$, our ϵ parameter was smallest on the instances where $p = 4$, $n = 20$, which give $\epsilon \ge 1/(4 * (-1 - (-10 * 10))) = 1/396$. Thus, we easily avoided numerical stability issues for our knapsack problems.

For our assignment problems we used complete bipartite graphs having node set $N_1 \cup N_2$, with $|N_1| = |N_2|$. We varied the number of nodes over $|N_1| \in \{3, 5, 8, 10\}$ while holding $p = 4$. Since the number of arcs in a complete bipartite graph is given by $|N_1|^2$, this varied the number of decision variables over $n \in \{9, 25, 64, 100\}$ with $p = 4$. We then varied the number of objectives $p \in \{3, 5, 6\}$ while holding $n = 25$ (i.e., $|N_1| = 5$). Again, for each (n, p) combination, we generated 10 instances, resulting in a total of 70 instances. Our assignment problems were formulated as:

$$\begin{aligned} \min \quad & [\mathbf{w}_1^T\mathbf{x}, \ldots, \mathbf{w}_p^T\mathbf{x}] \\ \text{subject to :} \quad & \sum_{j\in N_2} x_{ij} = 1 \quad \forall i \in N_1 \\ & \sum_{i\in N_1} x_{ij} = 1 \quad \forall j \in N_2 \\ & x_{ij} \in \{0,1\} \quad \forall (i,j) \in N_1 \times N_2, \end{aligned}$$

where each $\mathbf{w}_k \in \{0, \ldots, 10\}^n$ for $k \in P$ was randomly generated using a discrete uniform distribution. For the assignment problems, our upper bound input was $\mathbf{z}^{ub} = 10np\mathbf{e}$, and $s = 1$. As before, we chose $\epsilon = 1/pr$. Then since zero is a lower bound on any component objective, we can form a lower theoretical bound on ϵ over our test suite using instances where $p = 6$ and $n = 10$, which give $\epsilon \ge 1/(6 * (10 * 10 * 6 - 0)) = 1/3600$, which again was large enough to ensure that we did not encounter numerical stability issues.

We executed each algorithm five times on each instance and recorded data on the fastest total computational time observed. The study was run on an Intel Core i5 1.8 gigahertz processor with 4 gigabytes of 1600 megahertz DDR3 RAM. We imposed an 1800 seconds time limit, after which the algorithm terminated and returned the partial nondominated set, $\widehat{Z}_N$, obtained at termination.

5.2. Computational results

The first measure of merit in our experiment was how quickly Gurobi solved the integer programming (IP) sub-problems for each algorithm. Since KS and OBM employed lexicographic preferences to find each non-dominated point, each IP required multiple calls to Gurobi: OBM required p calls per IP, and KS required two. We scored these multiple calls as solving a single IP, summing Gurobi's solving time over the multiple calls to get the total time to solve the IP sub-problem. Our experiment indicates that Gurobi consistently solved KS's sub-problems faster than either BOX's or OBM's (see Fig. 3). The times were comparable, however, with differences usually less than one order of magnitude. Usually, Gurobi solved OBM's sub-problems in less time than it did for BOX, but as n grew large this trend reversed. Gurobi's time to solve BOX's sub-problems became increasingly comparatively slower than the other two as p increased.

Surprisingly, however, the faster computation of sub-problems did not always translate to faster time to termination (see Table 2). Fig. 4 shows the performance profiles for each algorithm, depicting the cumulative percentage of problems solved by each algorithm within time limits appearing on the horizontal axis. Fig. 4(a) depicts these performance profiles on a linear time axis, which

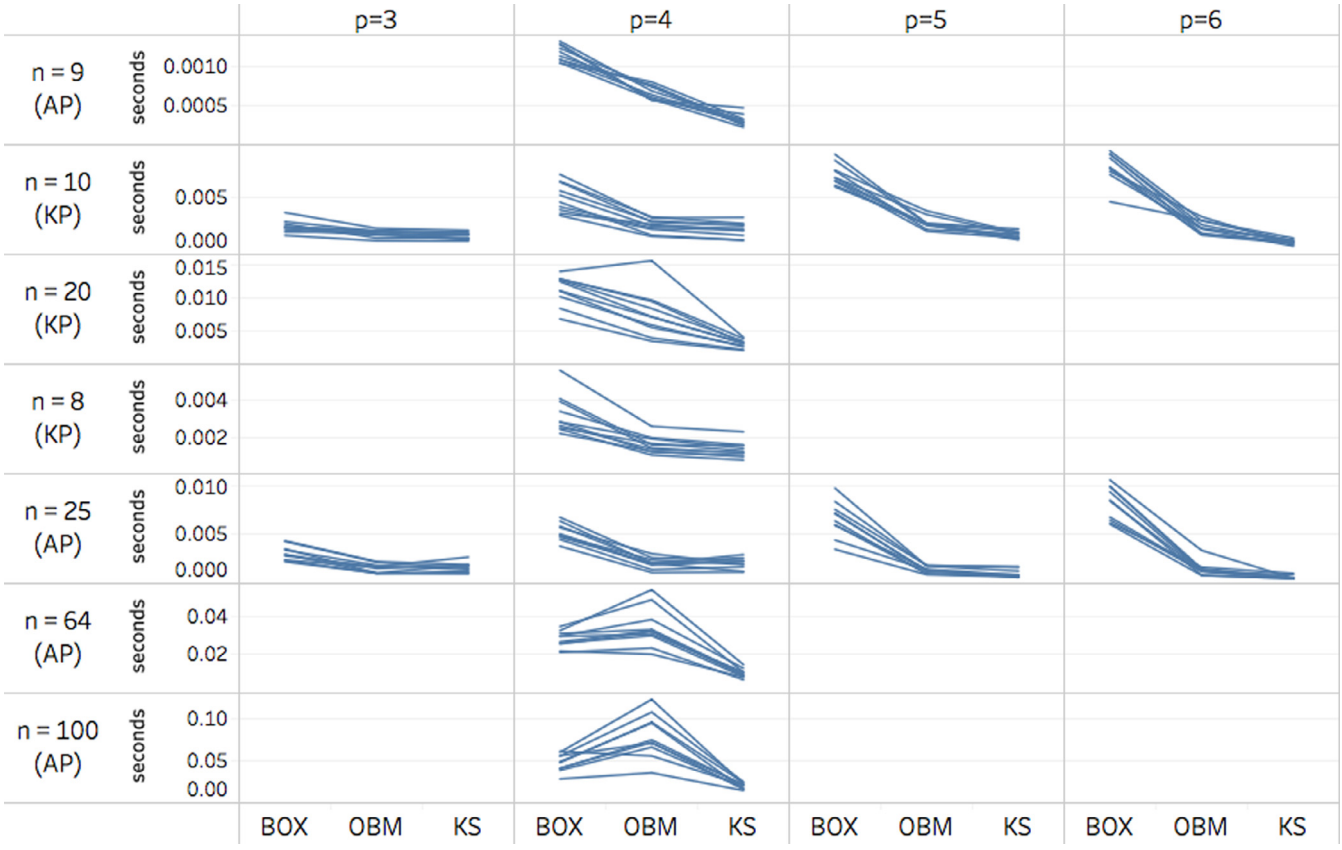
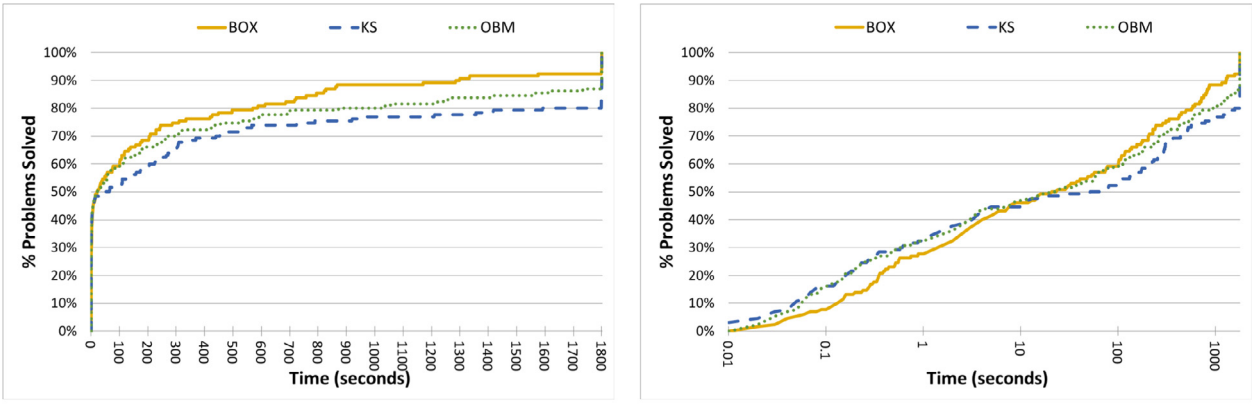


Fig. 3. Average time for Gurobi to solve one IP sub-problem in our testbed. Each line represents an average of solver times over all IP sub-problems for one problem instance.

Table 2
Summary of computation times. All times are in seconds and significant to the nearest second.

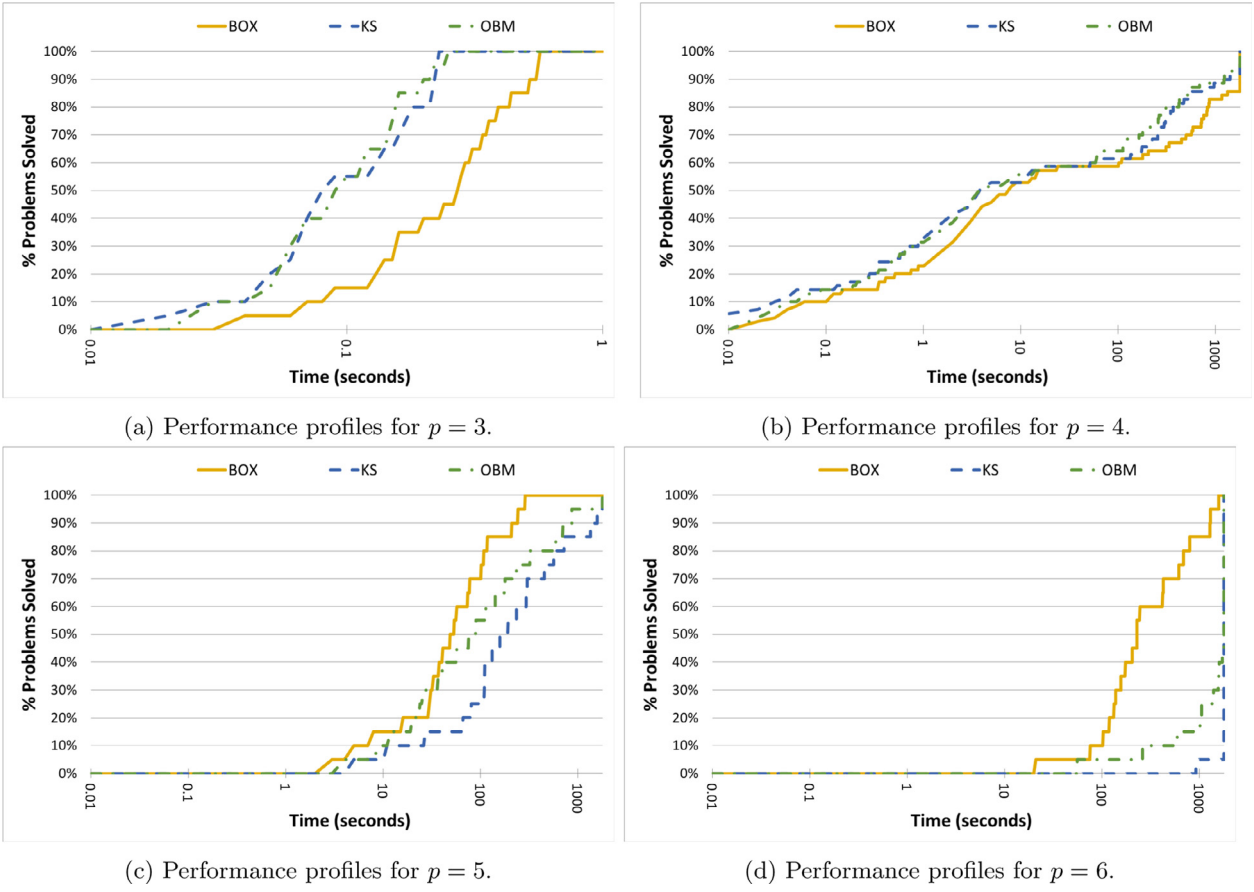
		BOX			KS			OBM		
p	*n*	Min	Median	Max	Min	Median	Max	Min	Median	Max
3	10 (KP)	< 0.5	< 0.5	1	< 0.5	< 0.5	< 0.5	< 0.5	< 0.5	< 0.5
	25 (AP)	< 0.5	< 0.5	1	< 0.5	< 0.5	< 0.5	< 0.5	< 0.5	< 0.5
4	8 (KP)	< 0.5	1	8	< 0.5	< 0.5	10	< 0.5	< 0.5	2
	9 (AP)	0.51	< 0.5	< 0.5	< 0.5	< 0.5	< 0.5	< 0.5	< 0.5	< 0.5
	10 (KP)	1	4	15	< 0.5	2	13	< 0.5	2	13
	20 (KP)	23	392	867	15	281	474	16	116	572
	25 (AP)	1	4	9	1	2	4	< 0.5	2	8
	64(AP)	204	790	1800	134	262	790	44	206	457
	100 (AP)	862	1800	1800	307	1800	1800	531	1530	1800
5	10 (KP)	8	55	286	27	213	1356	12	82	899
	25 (AP)	2	36	241	5	109	1800	3	85	873
6	10 (KP)	20	236	1576	920	1800	1800	55	1766	1800
	25 (AP)	75	184	1299	1800	1800	1800	261	1800	1800



(a) Performance profiles with linear time axis.

(b) Performance profiles with logarithmic time axis.

Fig. 4. Performance profiles for algorithms.



(a) Performance profiles for $p = 3$.

(b) Performance profiles for $p = 4$.

(c) Performance profiles for $p = 5$.

(d) Performance profiles for $p = 6$.

Fig. 5. Performance profiles for algorithms by p (logarithmic time axis).

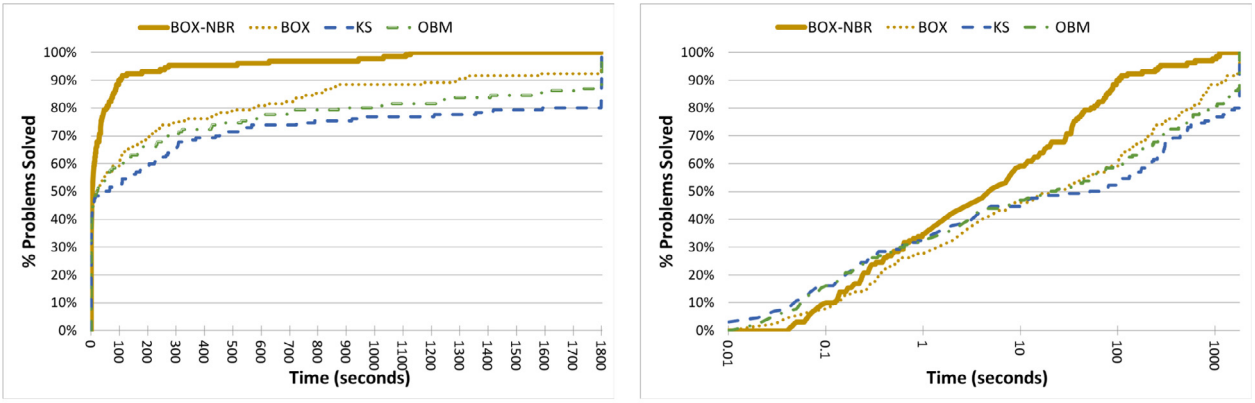
Table 3
Number of IPs solved for each algorithm on completed problem instances in our testbed. "N/A" indicates the algorithm failed to complete a sufficient number of problem instances to compute the summary statistic.

		BOX			KS			OBM		
p	n	Min	Median	Max	Min	Median	Max	Min	Median	Max
3	10 (KP)	14	69	138	8	35	66	12	34	54
	25 (AP)	19	57	106	12	28	66	14	25	43
4	8 (KP)	91	185	1064	36	69	414	51	91	252
	9 (AP)	7	21	64	5	14	27	9	18	32
	10 (KP)	235	572	1758	57	169	497	77	186	399
	20 (KP)	2522	24,815	44,057	403	2335	3586	462	1605	2751
	25 (AP)	214	449	962	85	182	306	90	185	301
	64(AP)	5936	17,844	27,755	1226	1999	3420	786	1534	2127
	100 (AP)	17,265	N/A	N/A	1480	4336	4918	1815	2822	3430
5	10 (KP)	1036	6666	26,855	370	998	3302	585	1412	3152
	25 (AP)	419	3781	19,242	159	735	2973	294	1296	4082
6	10 (KP)	3189	26,131	147,968	745	N/A	N/A	1117	6891	7632
	25 (AP)	9091	17,944	96,117	N/A	N/A	N/A	2799	5306	6560

demonstrates that our proposed BOX algorithm outperforms both KS and OBM over the aggregated set of test instances we generated. However, this plot hides the behavior of the algorithms when they terminate relatively quickly. Fig. 4(b) shows the performance profiles on a logarithmic time axis instead, revealing that these both outperform BOX on instances that require about ten seconds or less of computational time. Fig. 5 provides further insight to this trend, showing that BOX underperformed when $p \leq 4$, but tended to be faster when $p \geq 5$.

5.3. Running time improvement

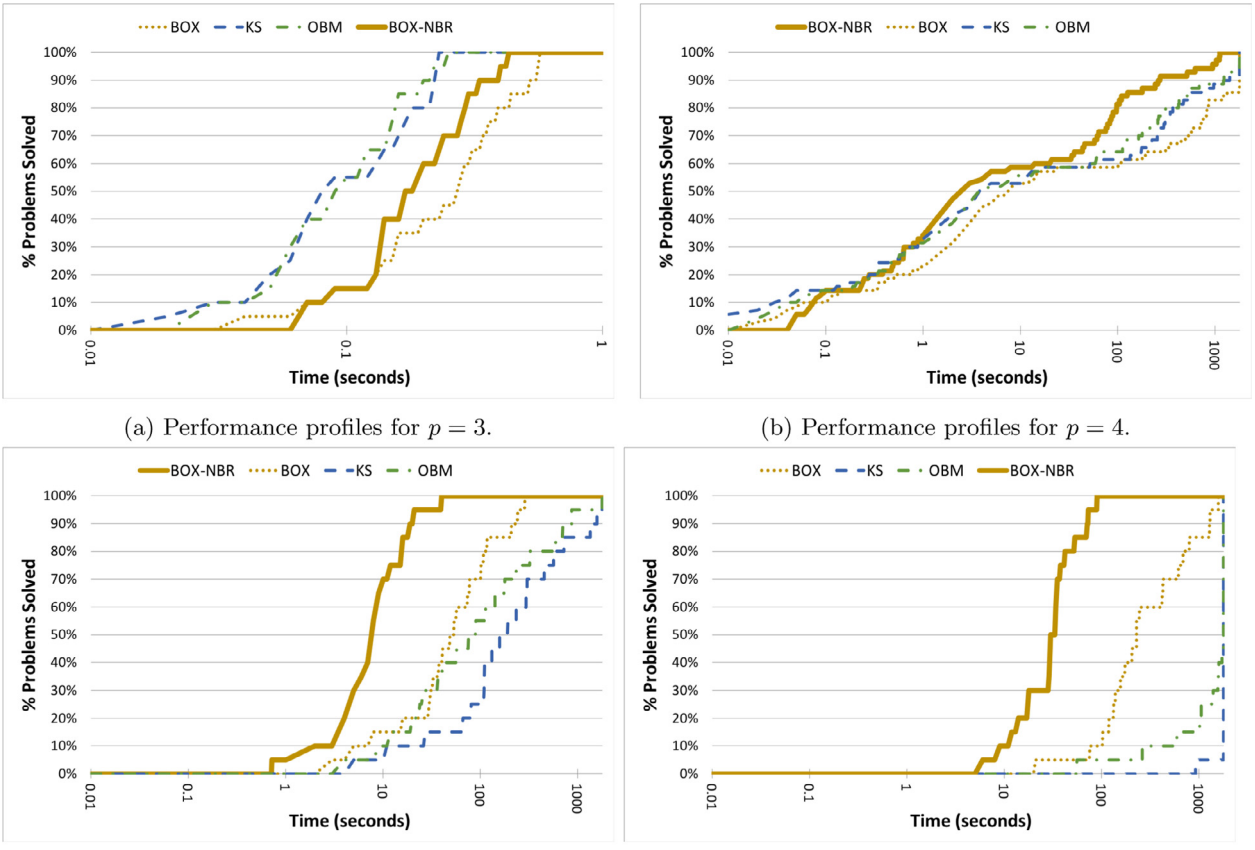
Our study demonstrated that using the MAWT norm as implemented in Algorithm 1 is comparable with two recently published algorithms and that it shows improved performance as the number of objectives increases. However, in comparing the number of IPs solved (see Table 3), it is clear that the BOX algorithm still has not reached its fullest potential, since it solves many more IPs than necessary (see Remark 2). Klamroth et al. (2015) describe two techniques to avoid such redundant searches: redundancy elimination (RE) and redundancy avoidance (RA). Upon inspection, it may be seen that both KS and OBM implement RE variants. Dächert et al. provide an RA routine for managing search regions using a neighbor relationship between the search region upper bounds. Dächert et al. prove that if all solutions are in general position (i.e., no two solutions have objectives equal in any component) their routine complexity is bounded in the order of the size of the upper bounding set over Z_N. In our testbed, the general position



(a) Performance profiles with linear time axis.

(b) Performance profiles with logarithmic time axis.

Fig. 6. Performance profiles for BOX-NBR (previous algorithm profiles are given as dotted lines for comparison).



(a) Performance profiles for $p = 3$.

(b) Performance profiles for $p = 4$.

(c) Performance profiles for $p = 5$.

(d) Performance profiles for $p = 6$.

Fig. 7. Performance profiles for BOX-NBR by p (logarithmic time axis, previous algorithm profiles are given for comparison).

assumption does not hold, and while Dächert et al. describe how to modify their routine for the non-general-position case, they provide no complexity results for such a scenario. We implemented the list management routine of Dächert et al., still using the MAWT norm to identify non-dominated solutions in the search regions. We call resulting algorithm "BOX-NBR."

After implementing these improvements, we tested BOX-NBR on the same instances from our computational study. The results demonstrate marked improvement over the base routine (see Table 4). Over the entire testbed, BOX-NBR exhibited an average of 64.0% improvement over BOX running times; moreover, the running time reductions tended to increase as p increased. Overall, BOX-NBR significantly outperformed all three of the previous algorithms over the comprehensive testbed, but was slower than KS and OBM on instances taking less than 1 second (see Fig. 6). As before, the strength of the MAWT norm becomes most apparent as the number of objectives is increased (see Fig. 7). Notably, of the four algorithms tested, BOX-NBR was the only one to successfully solve all testbed instances within 1800 seconds. Nevertheless, we also observe that BOX-NBR still required more IP sub-problems than OBM on a number of problems (see Table 4). This is likely because the OBM routine, as an ϵ-constraint method, conducts its

Table 4
Running times (in seconds) and number of IPs solved by BOX-NBR along with respective average percent improvement over BOX.

		Running time				IP's solved			
p	n	Min	Median	Max	% Improved	Min	Median	Max	% Improved
3	10 (KP)	<0.5	<0.5	<0.5	−19.5	29	51	90	−19.7
	25 (AP)	<0.5	<0.5	<0.5	−30.9	25	44	78	−25.3
4	8 (KP)	<0.5	<0.5	3	−55.7	67	98	321	−57.6
	9 (AP)	<0.5	<0.5	<0.5	+8.3	28	36	49	+25.8
	10 (KP)	<0.5	2	5	−65.5	88	212	450	−70.8
	20 (KP)	8	51	104	−87.0	563	2549	4571	−89.2
	25 (AP)	1	1	3	−65.7	96	166	243	−69.0
	64(AP)	36	88	180	−89.4	792	1583	2766	−91.8
	100 (AP)	110	572	1128	−67.0	1815	5844	9017	−82.8
5	10 (KP)	1	5	20	−88.6	479	825	2934	−88.9
	25 (AP)	1	5	20	−88.6	143	424	1196	−90.9
6	10 (KP)	5	35	89	−91.5	546	2724	6097	−93.7
	25 (AP)	9	23	70	−93.1	751	1593	3843	−94.7

search in $\mathbb{R}^{p-1}$ space rather than $\mathbb{R}^p$, and it is reasonable to expect that fewer search regions are required for a full search in the lower-dimensional space.

6. Conclusion

We contribute the MAWT norm and an associated algorithm, which, under fairly general assumptions, generates the entire non-dominated set for a multi-objective optimization problem having any number of objectives. Numerical experiments suggest that the MAWT norm yields scalarized sub-problems that may be more difficult to solve than comparable ϵ-constraint sub-problems. However, the algorithm still produced comparable computational results to two recently published algorithms. In particular, our computational investigation showed that the proposed BOX algorithm is preferable to the ϵ-constraint algorithms when the number of objectives was relatively large (more than four). However, our testbed includes only a limited set of problems, none of which are NP–hard in the single linear objective case. We also consider an improvement to our algorithm leveraging the work of Dächert et al. (2017) and Klamroth et al. (2015) and show significant reductions in running time by avoiding redundant searches.

Our observations motivate further computational study of various scalarization-based generating methods, including ϵ-constraint methods, Klein and Hannan's family of methods, and the MAWT norm. Each of these methods can be integrated with advanced approaches for managing the list of search regions and avoiding redundant searches. The integration of various scalarization methods with the diverse list management and redundancy mitigation schemes in the literature warrants further attention. Future studies may also examine the performance of these combinations on a variety of problem classes, including nonlinear, non-convex, and ill-conditioned problems.

Acknowledgments and disclaimer

The authors are grateful to three anonymous referees for their thorough and insightful comments, which led to an improved exposition of the paper. This work was supported by the Air Force Office of Scientific Research (grant no. FA9550-12-1-0353); and the Office of Naval Research (grant nos. N00014-13-1-0036, N00014-17-1-2421). The views expressed in this article are those of the authors and do not necessarily reflect the official policy or position of the Air Force, the Department of Defense or the U.S. Government.

Appendix A. Algorithms

In this appendix we present pseudo-code for the BOX and KS implementations used in our computational study described in Section 5. For our implementation, the functionality for each algorithm is separated among three modules, using an object-oriented framework common to both algorithms. Such a separation was not apparent for OBM, since that algorithm uses a recursive structure, as opposed to the looping structure of BOX and KS. Fig. 8 provides a graphical summary of Algorithm 2's search process and the data flows among the modules (initialization processes are not depicted). Module 1 is the controller, which directs the iterative search in Z and stores non-dominated solutions as they are discovered. The list of search regions is maintained by Module 2, and the searches are performed by Module 3. Algorithm 3 uses a similar division of functions.

The modular design depicted in Fig. 8 provides several benefits. First, it allows the algorithm to be easily reconfigured by varying the combination of search and list management routines. For example, we used the modular design to implement Dächert et al.'s neighbor routine as our list manager for the tests in Section 5.3. Depending on search and list management routines, the resulting algorithm could also implement other ϵ-constraint methods, Tchebychev search methods, and dominated region exclusion methods. Secondly, the modular design allows us to easily examine the effects of the search algorithm separate from the management of L, and control the influence of exogenous variables, such as timing commands by using the common controller. In our study, the timers were managed by the control module. The controller started a timer, then called a command from either the list manager or search routine, and stopped the timer when the result was returned. By using a common controller module, we limit the variance due to the controller's coding and we can easily examine how the algorithms perform updating and searching routines separately.

In Algorithm 2 we give pseudocode in an object-oriented framework that is equivalent to Algorithm 1. The objects are initialized by calling the Controller module's *init* procedure. Inputs to the *init* function are the same as the inputs to Algorithm 1. In turn, the Controller defines the list manager and search objects by calling their respective *init* procedures. In this pseudocode we assume the initialize procedures are class methods which return the created object. The main algorithm is then run by calling the Controller's *FindNonDom* procedure.

Algorithm 3 presents pseudocode for KS, our object-oriented implementation of Kirlik and Sayin's ϵ-constraint algorithm (Kirlik & Sayin, 2014). It uses essentially the same controller module as

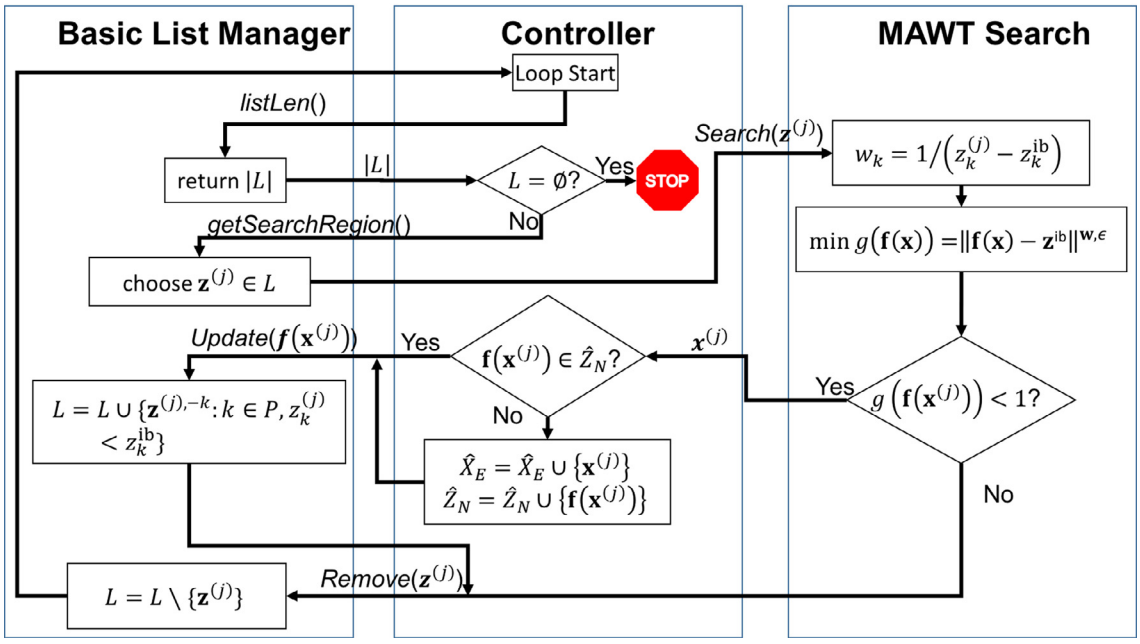


Fig. 8. Algorithm 1 search functions separated into three modules.

Algorithm 2. BOX: An object-oriented version of Algorithm 1.

Module 1: *Controller*

Procedure $init(\mathcal{I}, s, \mathbf{z}^{\text{ub}})$:
  for $k \in P$ **do**
    $z_k^{\text{lb}} \leftarrow \min\{f_k(\mathbf{x}) : x \in X\}$
  ListMgr $\leftarrow$ BasicListMgr.$init(\mathbf{z}^{\text{ub}}, \mathbf{z}^{\text{lb}})$
  Search $\leftarrow$ MAWTSrch.$init(\mathbf{z}^{\text{ub}}, \mathbf{z}^{\text{lb}}, s)$
  $\widehat{X}_E \leftarrow \emptyset$; $\widehat{Z}_N \leftarrow \emptyset$; $j \leftarrow 0$
Procedure *FindNonDom*():
  while ListMgr.*listLen*() > 0 **do**
    $j \leftarrow j + 1$
    $\mathbf{z}^{(j)} \leftarrow$ListMgr.*getSearchRegion*()
    $\mathbf{x}^{(j)} \leftarrow$Search.*search*$(\mathbf{z}^{(j)})$
    if $\mathbf{x}^{(j)} =$ Null **then**
      ListMgr.*remove*$(\mathbf{z}^{(j)})$
    else
      $\widehat{X}_E \leftarrow \widehat{X}_E \cup \{\mathbf{x}^{(j)}\}$
      $\widehat{Z}_N \leftarrow \widehat{Z}_N \cup \{\mathbf{f}(\mathbf{x}^{(j)})\}$
      ListMgr.*update*$(\mathbf{z}^{(j)}, \mathbf{f}(\mathbf{x}^{(j)}))$
  return $\widehat{X}_E$, $\widehat{Z}_N$

Module 2: *BasicListMgr*

Procedure $init(\mathbf{z}^{\text{ub}}, \mathbf{z}^{\text{lb}})$:
  $L \leftarrow \{\mathbf{z}^{\text{ub}}\}$
Procedure *getSearchRegion*():
  return $\mathbf{z}^{(j)} \in L$
Procedure *listLen*():
  return $|L|$
Procedure $remove(\mathbf{z}^{(j)})$:
  $L \leftarrow L \setminus \{\mathbf{z}^{(j)}\}$
Procedure $update(\mathbf{z}^{(j)}, \mathbf{f}(\mathbf{x}^{(j)}))$:
  remove$(\mathbf{z}^{(j)})$
  $L \leftarrow L \cup \{\mathbf{z}^{(j),-k} : k \in P, z_k^{(j)} < z_k^{\text{lb}}\}$

Module 3: *MAWTSrch*

Procedure $init(\mathbf{z}^{\text{ub}}, \mathbf{z}^{\text{lb}}, s)$:
  $r \leftarrow \max_{k \in P}\{z_k^{\text{ub}} - z_k^{\text{lb}}\}$
  $\epsilon \leftarrow (0, s/(2p(r-s)))$
Procedure $search(\mathbf{z}^{(j)})$:
  for $k \in P$ **do** $w_k \leftarrow 1/(z_k^{(j)} - z_k^{\text{lb}})$
  $(\mathbf{x}^*, g(\mathbf{f}(\mathbf{x}^*)) \leftarrow \min_{\mathbf{x} \in X}\{\|\mathbf{f}(\mathbf{x}) - \mathbf{z}^{\text{lb}}\|^{\mathbf{w},\epsilon}\}$
  if $g(\mathbf{f}(\mathbf{x}^*)) < 1$ **then**
    return $\mathbf{x}^*$
  else
    return Null

Algorithm 3. KS: an object-oriented implementation of Kirlik and Sayin's ϵ-constraint method.

Module 4: *KSController*

Procedure $init(P, s, \mathbf{z}^{\text{ub}})$**:**
  for $k \in P$ **do**
    $z_k^{\text{lb}} \leftarrow \min\{f_k(\mathbf{x}) : x \in X\}$
  ListMgr $\leftarrow$ KSListMgr.$init(\mathbf{z}^{\text{ub}},\ \mathbf{z}^{\text{lb}})$
  Search $\leftarrow$ KSSrch.$init(\mathbf{z}^{\text{ub}}, \mathbf{z}^{\text{lb}}, s)$
  $\widehat{X}_E \leftarrow \emptyset$; $\widehat{Z}_N \leftarrow \emptyset$; $j \leftarrow 0$
Procedure *FindNonDom*()**:**
  See Module 1

Module 5: *KSSearch*

Procedure $init(\mathbf{z}^{\text{ub}}, \mathbf{z}^{\text{lb}}, s)$**:**
  return
Procedure $run((\mathbf{z}^i, \mathbf{z}^n))$**:**
  for $k \in \{1, \ldots, p-1\}$ **do**
    Add constraint $f_k(\mathbf{x}) \leq z_k^n$
  $(\mathbf{x}^*, g(\mathbf{f}(\mathbf{x}^*))) \leftarrow \min_{\mathbf{x}\in X}\{f_p(\mathbf{x})\}$
  if *Problem infeasible* **then**
    Remove added constraints
    return Null
  else
    Add constraint $f_p(\mathbf{x}) = f_p(\mathbf{x}^*)$
    $(\mathbf{x}^*, g(\mathbf{f}(\mathbf{x}^*))) \leftarrow \min_{\mathbf{x}\in X}\{\sum_{k=1}^{p} f_k(\mathbf{x})\}$
    Remove added constraints
    return $\mathbf{x}^*$

Module 6: *KSListMgr*

Procedure $init(\mathbf{z}^{\text{ub}}, \mathbf{z}^{\text{lb}})$**:**
  $L \leftarrow \{([\mathbf{z}_1^{\text{lb}}, \ldots, \mathbf{z}_{p-1}^{\text{lb}}], [\mathbf{z}_1^{\text{ub}}, \ldots, \mathbf{z}_{p-1}^{\text{ub}}])\}$
Procedure *getSearchRegion*()**:**
  return $(\mathbf{z}^i, \mathbf{z}^n) \leftarrow \text{argmax}_{(\mathbf{z}^i,\mathbf{z}^n)\in L}\{\Pi_{k=1}^{p-1}(z_k^{\text{ub}} - z_k^{\text{lb}})\}$
Procedure *remove*$([\mathbf{f}_1(\mathbf{x}^{(j)}), \ldots, \mathbf{f}_{p-1}(\mathbf{x}^{(j)})])$**:**
  for $(\mathbf{i}, \mathbf{n}) \in L$ **do**
    if $\mathbf{i} \geqq [\mathbf{f}_1(\mathbf{x}^{(j)}), \ldots, \mathbf{f}_{p-1}(\mathbf{x}^{(j)})]$ AND $\mathbf{n} \leqq \mathbf{z}^n$ **then** // $\mathbf{z}^n$ is from line 4
      $L \leftarrow L \setminus \{(\mathbf{i}, \mathbf{n})\}$
Procedure *listLen*()**:**
  return $|L|$
Procedure *update*$((\mathbf{z}^i, \mathbf{z}^n), \mathbf{f}(\mathbf{x}^{(j)}))$**:**
  for $(\mathbf{i}, \mathbf{n}) \in L$ **do** // $(\mathbf{i}, \mathbf{n})$ is a candidate parent
    $L \leftarrow L \setminus \{(\mathbf{i}, \mathbf{n})\}$
    $T \leftarrow \{(\mathbf{i}, \mathbf{n})\}$ // T is the container for children of $(\mathbf{i}, \mathbf{n})$
    for $k \in \{1, \ldots, p-1\}$ **do** // Iterate through objectives
      if $i_k < f_k(\mathbf{x}^{(j)}) < n_k$ **then** // Check if we need k-children
        for $(\mathbf{i}, \mathbf{n}) \in T$ **do** // Loop divides each $(\mathbf{i}, \mathbf{n}) \in T$ into 2 k-children
          $T \leftarrow T \setminus \{(\mathbf{i}, \mathbf{n})\}$
          $\mathbf{newN} \leftarrow [n_1, \ldots, n_{k-1}, f_k(\mathbf{x}^{(j)}), n_{k+1}, \ldots, n_p]$
          $\mathbf{newI} \leftarrow [i_1, \ldots, i_{k-1}, f_k(\mathbf{x}^{(j)}), i_{k+1}, \ldots, i_p]$
          $T \leftarrow T \cup \{(\mathbf{i}, \mathbf{newN}), (\mathbf{newI}, \mathbf{n})\}$
    Set $L = L \cup T$
  remove$([f_1(\mathbf{x}), \ldots, f_{p-1}(\mathbf{x})])$

Algorithm 2. Modules 5 and 6 function as the search and list manager modules, respectively.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2018.05.036.

References

Alves, M., & Clímaco, J. (2000). An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. *European Journal of Operational Research, 124*(3), 478–494.

Belotti, P., Soylu, B., & Wiecek, M. M. (2013). A branch-and-bound algorithm for biobjective mixed integer programs. *Optimization Online*. http://www.optimization-online.org/DB_HTML/2013/01/3719.html.

Bowman, V., Jr (1976). On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In H. Thiriez, & S. Zionts (Eds.), *Multiple criteria decision making: Lecture notes in economics and mathematical systems: Vol. 130* (pp. 76–86). Berlin, Heidelberg: Springer.

Dächert, K., Gorski, J., & Klamroth, K. (2012). An augmented weighted Tchebycheff method with adaptively chosen parameters for discrete bicriteria optimization problems. *Computers and Operations Research, 39*(12), 2929–2943.

Dächert, K., Klamroth, K., Lacour, R., & Vanderpooten, D. (2017). Efficient computation of the search region in multi-objective optimization. *European Journal of Operational Research, 260*(3), 841–855.

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

Ehrgott, M., & Ruzika, S. (2008). Improved ϵ-constraint method for multiobjective programming. *Journal of Optimization Theory and Applications, 138*(3), 375–396.

Haimes, Y., Lasdon, L., & Wismer, D. (1971). On a bicriterion formulation of the problem of integrated system identification and system optimizations. *IEEE Transactions on Systems, Man, and Cybernetics, 1*, 296–297.

Hamacher, H. W., Pederson, C. R., & Ruzika, S. (2007). Finding representative systems for discrete bicriterion optimization problems. *Operations Research Letters, 35*(3), 336–344.

Holzmann, T., & Smith, J. C. (2017). Code and problem instances for testing the modified augmented weighted tchebychev norm. Data in Brief.

Hwang, C.-L., & Masud, A. (1979). Multiple objective decision making – methods and applications: A state-of-the-art survey. In M. Beckman, & H. Künzi (Eds.), *Lecture notes in economics and mathematical systems: Vol. 164*. New York: Springer-Verlag.

Kaliszewski, I. (1987). A modified weighted Tchebycheff metric for multiple objective programming. *Computers and Operations Research, 14*(4), 315–323.

Kaliszewski, I. (2000). Using trade-off information in decision-making algorithms. *Computers and Operations Research, 27*(2), 161–182.

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

Klamroth, K., Lacour, R., & Vanderpooten, D. (2015). On the representation of the search region in multi-objective optimization. *European Journal of Operational Research, 245*(3), 767–778.
Klamroth, K., Tind, J., & Wiecek, M. M. (2002). Unbiased approximation in multicriteria optimization. *Mathematical Methods of Operations Research, 56*(3), 413–437.
Klein, D., & Hannan, E. (1982). An algorithm for the multiple objective linear programming problem. *European Journal of Operational Research, 9*(4), 378–385.
Laumanns, M., Thiele, L., & Eckart, Z. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research, 169*(3), 932–942.
Lokman, B., & Köksalan, M. (2013). Finding all nondominated points of multi-objective integer programs. *Journal of Global Optimization, 57*(2), 347–365.
Mavrotas, G. (2009). Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation, 213*(2), 455–465.
Mavrotas, G., & Diakoulaki, D. (1998). A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research, 107*(3), 530–541.
Mavrotas, G., & Diakoulaki, D. (2005). Multi-criteria branch and bound: A vector maximization algorithm for mixed 0–1 multiple objective linear programming. *Applied Mathematics and Computation, 171*(1), 53–71.
Mavrotas, G., & Florios, K. (2013). An improved version of the augmented ϵ-constraint method (AUGMECON2) for finding the exact pareto set in multi-objective integer programming problems. *Applied Mathematics and Computation, 219*(18), 9652–9669.
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.
Özlen, M., Burton, B. A., & MacRae, C. A. G. (2014). Multi-objective integer programming: An improved recursive algorithm. *Journal of Optimization Theory and Applications, 160*(2), 470–482.
Özpeynirci, O., & Köksalan, M. (2010). An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs. *Management Science, 56*(12), 2302–2315.
Przybylski, A., Gandibleux, X., & Ehrgott, M. (2008). Two-phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research, 185*(2), 509–533.
Przybylski, A., Gandibleux, X., & Ehrgott, M. (2010). A two-phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization, 7*(3), 149–165.
Raith, A., & Ehrgott, M. (2009). A two-phase algorithm for the biobjective integer minimum cost flow problem. *Computers and Operations Research, 36*(6), 1945–1954.
Ramos, R., Alonso, S., Sicilia, J., & Gonzáles, C. (1998). The problem of the optimal biobjective spanning tree. *European Journal of Operational Research, 111*(3), 617–628.
Schandl, B., Klamroth, K., & Wiecek, M. M. (2002). Norm-based approximation in multicriteria programming. *Computers and Mathematics with Applications, 44*(7), 925–942.
Steiner, S., & Radzik, T. (2008). Computing all efficient solutions of the biobjective minimum spanning tree problem. *Computers and Operations Research, 35*(1), 198–211.
Steuer, R. (1986). *Multiple criteria optimization: Theory computation and application*. New York: John Wiley and Sons.
Steuer, R., & Choo, E. (1983). An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming, 26*(3), 326–344.
Stidsen, T., Andersen, K. A., & Damman, B. (2014). A branch and bound algorithm for a class of biobjective mixed integer programs. *Management Science, 60*(4), 1009–1032.
Sutherland, W. A. (1975). *Introduction to metric and topological spaces*. New York: Oxford University Press.
Sylva, J., & Crema, A. (2004). A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research, 158*(1), 46–55.
Ulungu, E., & Teghem, J. (1994). Multi-objective combinatorial optimization problems: A survey. *Journal of Multi-Criteria Decision Analysis, 3*(2), 83–104.
Vincent, T., Florian, S., Ruzika, S., & Przybylski, A. (2013). Multiple objective branch and bound for mixed 0–1 linear programming: Corrections and improvements for the biobjective case. *Computers and Operations Research, 40*(1), 498–509.
Visée, M., Teghem, J., Pirlot, M., & Ulungu, E. (1998). Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization, 12*(2), 139–155.
White, D. (1982). The set of efficient solutions for multiple objective shortest path problems. *Computers and Operations Research, 9*(2), 101–107.
Wierzbicki, A. (1980). The use of reference objectives in multiobjective optimization. In G. Fandel, & T. Gal (Eds.), *Multiple criteria decision making theory and application* (pp. 468–486). Berlin: Springer.
Zhang, W., & Reimann, M. (2014). A simple augmented ϵ-constraint method for multi-objective mathematical integer programming problems. *European Journal of Operational Research, 234*(1), 15–24.