The Triangle Splitting Method for Biobjective Mixed Integer Programming

Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh

School of Mathematical and Physical Sciences,
The University of Newcastle, NSW 2308, Australia
{Natashia.Boland,Martin.Savelsbergh}@newcastle.edu.au,
Hadi.Charkhgard@uon.edu.au

Abstract. We present the first criterion space search algorithm, the triangle splitting method, for finding the efficient frontier of a biobjective mixed integer program. The algorithm is relatively easy to implement and converges quickly to the complete set of nondominated points. A computational study demonstrates the efficacy of the triangle splitting method.

Keywords: biobjective mixed integer program, triangle splitting method, efficient frontier.

1 Introduction

Multiobjective optimization, one of the earliest fields of study in operations research, has been experiencing a resurgence of interest in the last decade. This is due, in part, to the ever increasing adoption of optimization-based decision support tools in industry. Since most real-world problems involve multiple, often conflicting, goals, the want for multiobjective optimization decision support tools is not surprising. The development of effective, and relatively easy to use, evolutionary algorithms for multiobjective optimization is another contributing factor. Finally, the availability of cheap computing power has played a role. Solving multiobjective optimization problems is highly computationally intensive (more so than solving single-objective optimization problems) and thus the availability of cheap computing power has acted as an enabler.

Exact algorithms for multiobjective optimization can be divided into solution space search algorithms, i.e., methods that search in the space of feasible solutions, and criterion space search algorithms, i.e., methods that search in the space of objective function values. It has long been argued (see for example [3]) that criterion space search algorithms have advantages over solution space search algorithms and are likely to be more successful. Our motivation for focusing on criterion space search algorithms is that we want to exploit the power of commercial (single-objective) integer programming solvers. Several extremely powerful commercial integer programming solvers exist, e.g., the IBM ILOG CPLEX Optimizer, the FICO Xpress Optimizer, and the Gurobi Optimizer, and criterion space search algorithms can take full advantage of their features. Embedding

J. Lee and J. Vygen (Eds.): IPCO 2014, LNCS 8494, pp. 162–173, 2014.

a commercial integer programming solver in any algorithm has the additional advantage that enhancements made to the commercial solver immediately result in improved performance of the algorithm in which it is embedded.

In this study, we focus on biobjective mixed integer programs (BOMIPs). Computing the efficient frontier of a BOMIP is challenging because of two reasons. First, the existence of unsupported nondominated points, i.e., nondominated points that cannot be obtained by optimizing a convex combination of the objective functions. Secondly, the existence of continuous parts in the efficient frontier, i.e., parts where all points of a line segment are nondominated. Figure 1 shows an example of a nondominated frontier of a BOMIP. Observe that this nondominated frontier contains isolated points as well as closed, half open, and open line segments.

There are only a few studies that present algorithms for finding the efficient frontier of a BOMIP (several of these algorithms were later shown to be incomplete or incorrect). All these algorithms are solution space search algorithms and are based on the following observation (which is made more precise in Section 2). If S_I is the projection of the set of feasible solutions to a BOMIP on to the space of the integer variables, then fixing the integer variables to the values of s for any $s \in S_I$ changes the BOMIP to a biobjective linear program (BOLP). Furthermore, if $\mathcal{Y}_N(s)$ for $s \in S_I$ denotes the nondominated frontier of the resulting BOLP, then the nondominated frontier of the BOMIP is the set of nondominated points in $\bigcup_{s \in S_I} \mathcal{Y}_N(s)$. This observation suggests a natural algorithm for computing the nondominated frontier: enumerate the solutions in S_I, for each of these solutions find the nondominated frontier of the associated BOLP, take the union of these nondominated frontiers, and, finally, eliminate any dominated points from this set.

Unfortunately, this natural algorithm has a number of drawbacks in practice:

– The set $\bigcup_{s \in S_I} \mathcal{Y}_N(s)$ can become prohibitively large, which implies that storing and maintaining this set of points may require an excessive amount of memory;

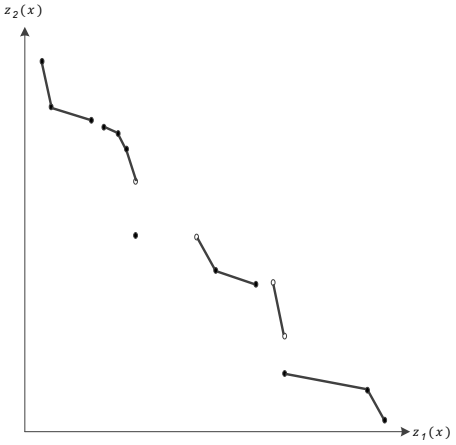

Fig. 1. Example of a nondominated frontier of a BOMIP

- Eliminating dominated points from the set $\bigcup_{s \in S_I} \mathcal{Y}_N(s)$ can become prohibitively time-consuming; and
- The nondominated frontier is only available upon completion, i.e., during the course of the algorithm the set of points maintained contains both dominated and nondominated points.

Most of the research on algorithms for BOMIPs has focused on addressing the first two drawbacks, either by developing specialized data structures and methods for efficiently storing and maintaining the set $\bigcup_{s \in S_I} \mathcal{Y}_N(s)$, or by curtailing the enumeration of $s \in S_I$, i.e., by recognizing or determining that for a given $s \in S_I$ all the points in $\mathcal{Y}_N(s)$ will (eventually) be eliminated. The third drawback, unfortunately, is an inherent feature of this algorithm and cannot be avoided.

We have developed a completely different algorithm, the triangle splitting algorithm, which does not suffer from any of these drawbacks. To the best of our knowledge, it is the first criterion space search algorithm for finding the efficient frontier of a BOMIP. The algorithm recursively explores smaller and smaller rectangles and right triangles that may contain as yet unknown nondominated points. The triangle splitting algorithm has the following advantages:

- It is a criterion space search method that relies on a small number of ideas and techniques, which makes it both easy to understand and easy to implement.
- It has minimal requirements in terms of information storage.
- It maintains at any time during its execution a set of points which are guaranteed to be part of the nondominated frontier.
- It relies on solving single objective mixed integer programs (MIPs) and benefits automatically from any advances in single objective MIP solvers.

A computational study demonstrates the efficacy of the triangle splitting algorithm. Its performance is as good or better than the best known algorithm for BOMIPs [2] and it can handle instance sizes that far exceed the size that solution space algorithms can handle.

In the remainder of this extended abstract, we describe the basic version of the algorithm. In Section 2, we give preliminaries, and in Section 3, we introduce the triangle splitting method and present computational results.

2 Preliminaries

A multiobjective mixed integer programming problem (MOMIP) can be stated as follows

$$\min_{x \in \mathcal{X}} \; z(x) := \{z_1(x), ..., z_p(x)\},$$

where $\mathcal{X} \subseteq \mathbb{Z}^n \times \mathbb{R}^m$ is defined by a set of linear constraints and represents the *feasible set in the decision space* and the image $\mathcal{Y}$ of $\mathcal{X}$ under vector-valued function $z = \{z_1, ..., z_p\}$ represents the *feasible set in the criterion space*, i.e., $\mathcal{Y} := z(\mathcal{X}) := \{y \in \mathbb{R}^p : y = z(x) \text{ for some } x \in \mathcal{X}\}$. We will sometimes use

$x = (x_I, x_C)$, for $x \in \mathcal{X}$, to distinguish the integer and continuous variables in a feasible solution. For convenience, we also use the notation $\mathbb{R}^p_{\geq} := \{y \in \mathbb{R}^p : y \geq 0\}$ for the nonnegative orthant of $\mathbb{R}^p$, and $\mathbb{R}^p_{>} := \{y \in \mathbb{R}^p : y > 0\}$ for the positive orthant of $\mathbb{R}^p$.

Definition 1. *A feasible solution $x' \in \mathcal{X}$ is called efficient or Pareto optimal, if there is no other $x \in \mathcal{X}$ such that $z_k(x) \leq z_k(x')$ for $k = 1, ..., p$ and $z(x) \neq z(x')$. If x' is efficient, then $z(x')$ is called a nondominated point. The set of all efficient solutions $x' \in \mathcal{X}$ is denoted by $\mathcal{X}_E$. The set of all nondominated points $y' = z(x') \in \mathcal{Y}$ for some $x' \in \mathcal{X}_E$ is denoted by $\mathcal{Y}_N$ and referred to as the nondominated frontier or the efficient frontier.*

Definition 2. *Let $x' \in \mathcal{X}_E$. If there is a $\lambda \in \mathbb{R}^{n+m}_{>}$ such that x' is an optimal solution to $\min_{x \in \mathcal{X}} \lambda^T z(x)$, then x' is called a supported efficient solution and $y = z(x')$ is called a supported nondominated point.*

Definition 3. *Let $\mathcal{Y}^e$ be the set of extreme points of $Conv(\mathcal{Y})$. A point $y \in \mathcal{Y}$ is called an extreme supported nondominated point if $y \in \mathcal{Y}^e \cap \mathcal{Y}_N$.*

Theorem 1. *For a multiobjective linear program (MOLP), we have that $\mathcal{Y}$ is closed and convex if $\mathcal{X}$ is closed.*

Corollary 1. *For a MOLP, the nondominated points in $\mathcal{Y}_N$ are supported and connected, i.e., between any pair of nondominated points there exists a sequence of nondominated points with the property that all points on the line segment between consecutive points in the sequence are also nondominated.*

Let $\mathcal{X}_I = proj_I(\mathcal{X})$, where $proj_I(\mathcal{X}) := \{x \in \mathbb{Z}^n :$ there exists a $x_C \in \mathbb{R}^m$ such that $(x, x_C) \in \mathcal{X}\}$, and for $\bar{x} \in \mathcal{X}_I$ let $\mathcal{Y}_N(\bar{x})$ denote the nondominated frontier of the MOLP defined by

$$\min_{(x_I, x_C) \in \mathcal{X} : x_I = \bar{x}} z(x) := \{z_1(x), ..., z_p(x)\}.$$

Finally, let the function $\mathcal{ND} : \mathbb{R}^p \to \mathbb{R}^p$ be one that takes a set of points $\mathcal{P}$ in the criterion space and removes any point $p \in \mathcal{P}$ that is dominated by any other point $p' \in \mathcal{P}$. We have the following theorem (see for example Gardenghi et al. [4]).

Theorem 2. *The nondominated frontier $\mathcal{Y}_N = \mathcal{ND}(\bigcup_{\bar{x} \in \mathcal{X}_I} \mathcal{Y}_N(\bar{x}))$.*

Theorem 2 forms the basis of most solution space search algorithms for MOMIPs. These algorithms essentially enumerate $\mathcal{X}_I$, compute $\mathcal{Y}_N(\bar{x})$ for all $\bar{x} \in \mathcal{X}_I$, form the union of the resulting nondominated frontiers, and eliminate any dominated points, i.e., set $\mathcal{Y}_N = \mathcal{ND}(\bigcup_{\bar{x} \in \mathcal{X}_I} \mathcal{Y}_N(\bar{x}))$.

The first "branch-and-bound" algorithm for solving multiobjective mixed 0-1 integer programs was proposed by Mavrotas and Diakoulaki [6]. The enumeration algorithm ensures that at the leaf nodes of the search tree, the values of the 0-1 variables are fixed at either zero or one. The algorithm maintains a list of

potential nondominated points, which is updated at each leaf node of the search tree, i.e., potential nondominated points are added to the list and dominated points are removed from the list. In a follow-up paper, Mavrotas and Diakoulaki [7] show that their initial scheme for updating the list was incomplete in the sense that some dominated points might erroneously remain in the list. They propose an a posteriori filtering method to remedy the situation. Vincent et al. [8] show another deficiency of the branch-and-bound algorithm of Mavrotas and Diakoulaki, namely that it may not identify all nondominated points, i.e., some nondominated points are missed. They show that this issue can be corrected for biobjective mixed 0-1 integer programs. More recently, Belotti et al. [2] propose a different branch-and-bound algorithm to compute the nondominated frontier of a biobjective mixed integer program. It more closely resembles the traditional branch-and-bound for single objective mixed integer programs, in the sense that bounding strategies are employed to fathom nodes during the search.

Next, we introduce concepts and notation that will facilitate the presentation and discussion of the triangle splitting method. For the remainder of the paper, we restrict ourselves to BOMIPs. Let $z^1 = (z_1^1, z_2^1)$ and $z^2 = (z_1^2, z_2^2)$ be two points in the criterion space with $z_1^1 \leqslant z_1^2$ and $z_2^2 \leqslant z_2^1$. We denote with $R(z^1, z^2)$ the rectangle in the criterion space defined by the points z^1 and z^2. Furthermore, we denote with $T(z^1, z^2)$ the right triangle in the criterion space defined by the points z^1, (z_1^2, z_2^1), and z^2. Finally, we denote with $H(z^1, z^2)$ the line segment in the criterion space defined by the points z^1 and z^2, i.e., the hypotenuse of triangle $T(z^1, z^2)$.

A point $\bar{z}$ in criterion space corresponding to a solution with smallest value for $z_2(x)$ among all solutions with smallest value for $z_1(x)$ among all feasible solutions with objective function values in $T(z^1, z^2)$, if one exists, can be found by solving two mixed integer programs in sequence, namely

$$\begin{aligned} \bar{z}_1 = \min_{x \in \mathcal{X}} \; & z_1(x) \\ \text{subject to } & z(x) \in T(z^1, z^2), \end{aligned}$$

followed by

$$\begin{aligned} \bar{z}_2 = \min_{x \in \mathcal{X}} \; & z_2(x) \\ \text{subject to } & z(x) \in T(z^1, z^2) \text{ and } z_1(x) \leq \bar{z}_1. \end{aligned}$$

As this is an operation that will be performed frequently in our criterion space search algorithm, we introduce the following notation to represent the process:

$$\bar{z} = \operatorname*{lex\,min}_{x \in \mathcal{X}} \{z_1(x), z_2(x) : z(x) \in T(z^1, z^2)\}.$$

Finding a point $\bar{z}$ in criterion space corresponding to a solution with smallest value for $z_1(x)$ among all solutions with smallest value for $z_2(x)$ among all feasible

solutions with objective function values in $T(z^1, z^2)$, if one exists, can be done similarly, and we introduce the following notation to represent that process:

$$\bar{z} = \operatorname*{lex\,min}_{x \in \mathcal{X}} \{z_2(x), z_1(x) : z(x) \in T(z^1, z^2)\}.$$

It is often convenient to assume that the points of the efficient frontier are listed in order of nondecreasing value of the first objective function. In that case, the first and last point of the efficient frontier can be found by solving

$$z^T := \operatorname*{lex\,min}_{x \in \mathcal{X}} \{z_1(x), z_2(x) : z(x) \in R((-\infty, \infty), (-\infty, \infty))\}.$$

and

$$z^B := \operatorname*{lex\,min}_{x \in \mathcal{X}} \{z_2(x), z_1(x) : z(x) \in R((-\infty, \infty), (-\infty, \infty))\},$$

respectively (where the feasible region is defined by a rectangle instead of a triangle).

The next propositions and their corollaries provide the basis for the development of the triangle splitting method.

Proposition 1. *Let z^1 and z^2 be two points in the criterion space with $z_2^2 < z_2^1$ and let v be such that $z_2^2 < v < z_2^1$. If $\{(z - \mathbb{R}^2_>) \cap \mathcal{Y}_N : z \in H(z^1, z^2)\} = \emptyset$, then $\operatorname{lex\,min}_{x \in \mathcal{X}} \{z_1(x), z_2(x) : z_2(x) \leq v,\ z(x) \in T(z^1, z^2)\}$ returns a nondominated point if it has a solution.*

Corollary 2. *Let z^1 and z^2 be two points in the criterion space with $z_2^2 < z_2^1$ and let v be such that $z_2^2 < v < z_2^1$. If $\{(z - \mathbb{R}^2_>) \cap \mathcal{Y}_N : z \in H(z^1, z^2)\} = \emptyset$ and $z^2 \in \mathcal{Y}_N$, then $\bar{z}^1 = \operatorname{lex\,min}_{x \in \mathcal{X}} \{z_1(x), z_2(x) : z_2(x) \leq v,\ z(x) \in T(z^1, z^2)\} \in \mathcal{Y}_N$. Furthermore, if $\bar{z}^1 = z^2$, then z^2 is the only nondominated point in $T(z^1, z^2)$ with $z_2(x) \leq v$.*

Proposition 2. *Let z^1 and z^2 be two points in the criterion space with $z_1^1 < z_1^2$ and let v be such that $z_1^1 < v < z_1^2$. If $\{(z - \mathbb{R}^2_>) \cap \mathcal{Y}_N : z \in H(z^1, z^2)\} = \emptyset$, then $\operatorname{lex\,min}_{x \in \mathcal{X}} \{z_2(x), z_1(x) : z_1(x) \leq v,\ z(x) \in T(z^1, z^2)\}$ returns a nondominated point if it has a solution.*

Corollary 3. *Let z^1 and z^2 be two points in the criterion space with $z_1^1 < z_1^2$ and let v be such that $z_1^1 < v < z_1^2$. If $\{(z - \mathbb{R}^2_>) \cap \mathcal{Y}_N : z \in H(z^1, z^2)\} = \emptyset$ and $z^1 \in \mathcal{Y}_N$, then $\bar{z}^2 = \operatorname{lex\,min}_{x \in \mathcal{X}} \{z_2(x), z_1(x) : z_1(x) \leq v,\ z(x) \in T(z^1, z^2)\} \in \mathcal{Y}_N$. Furthermore, if $\bar{z}^2 = z^1$, then z^1 is the only nondominated point in $T(z^1, z^2)$ with $z_1(x) \leq v$.*

Theorem 3. *Let z^1 and z^2 be two nondominated points in the criterion space. If $\{(z - \mathbb{R}^2_>) \cap \mathcal{Y}_N : z \in H(z^1, z^2)\} = \emptyset$ and there exists an $x \in \mathcal{X}_I$ and x_C^1 and $x_C^2 \in \mathbb{R}^m$ such that $(x, x_C^1), (x, x_C^2) \in \mathcal{X}$, $z_1((x, x_C^1)) \leq z_1^1$, $z_2((x, x_C^1)) \leq z_2^1$, $z_1((x, x_C^2)) \leq z_1^2$, and $z_2((x, x_C^2)) \leq z_2^2$, then $H(z^1, z^2) \subset \mathcal{Y}_N$.*

Theorem 3 implies that for a triangle $T(z^1, z^2)$ such that $\{(z - \mathbb{R}^2_{\geq}) \cap \mathcal{Y}_N : z \in H(z^1, z^2)\} = \emptyset$, the following *line detection* mixed integer program (MIP) establishes whether $H(z^1, z^2) \subset \mathcal{Y}_N$:

$$\begin{aligned} & \max z_1((x_I, x_C^2)) \\ \text{subject to} & \\ & z_1((x_I, x_C^1)) \leq z_1^1 \\ & z_2((x_I, x_C^1)) \leq z_2^1 \\ & \lambda_1 z_1((x_I, x_C^2)) + \lambda_2 z_2((x_I, x_C^2)) = \lambda_1 z_1^1 + \lambda_2 z_2^1 \\ & (x_I, x_C^1) \in \mathcal{X}, (x_I, x_C^2) \in \mathcal{X} \end{aligned}$$

where $\lambda_1 = z_2^1 - z_2^2$ and $\lambda_2 = z_1^2 - z_1^1$. The constraint $\lambda_1 z_1((x_I, x_C^2)) + \lambda_2 z_2((x_I, x_C^2)) = \lambda_1 z_1^1 + \lambda_2 z_2^1$ expresses that the ratio of the horizontal distance and the vertical distance is the same as $\frac{z_1^2 - z_1^1}{z_2^1 - z_2^2}$, which implies that the point is on the imaginary line connecting z^1 and z^2. Note that if the optimal value is greater than or equal to z_1^2, then z^1 and z^2 satisfy the conditions of Theorem 3.

Determining whether the points on the hypotenuse of a triangle are all nondominated, i.e., whether the hypotenuse of the triangle is part of the nondominated frontier, is a core component of the triangle splitting algorithm. Another core component of the triangle splitting method is the weighted sum method [1]. The weighted sum method is used to find all locally extreme supported nondominated points in a rectangle defined by two nondominated points z^1 and z^2. The weighted sum method uses the following optimization problem to search for extreme supported nondominated points in rectangle $R(z^1, z^2)$:

$$\begin{aligned} z^* = \min_{x \in \mathcal{X}} \ & \lambda_1 z_1(x) + \lambda_2 z_2(x) \\ \text{subject to } & z(x) \in R(z^1, z^2) \end{aligned}$$

with $\lambda_1 = z_2^1 - z_2^2$ and $\lambda_2 = z_1^2 - z_1^1$, i.e., the objective function is parallel to the line that connects z^1 and z^2 in the criterion space. Figure 2 shows an example with $z^1 = z^T$ and $z^2 = z^B$. It is easy to see that the optimum point z^* is an as yet unknown locally supported nondominated point if $\lambda_1 z_1^* + \lambda_2 z_2^* < \lambda_1 z_1^1 + \lambda_2 z_2^1$. That is, the optimization either returns a new locally supported nondominated point z^* or a convex combination of z^1 and z^2. When an as yet unknown locally supported nondominated point z^* is returned, the method is applied recursively to search $R(z^1, z^*)$ and $R(z^*, z^2)$ for additional as yet unknown locally supported nondominated points. Note that the set of locally supported nondominated points returned by the weighted sum method is guaranteed to include *all* locally extreme supported nondominated points (but it may also include locally supported nondominated points that are not extreme).

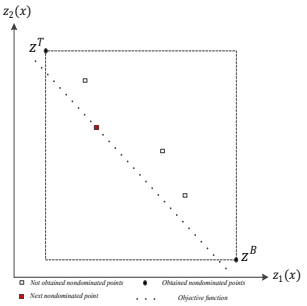

Fig. 2. Searching for a nondominated point using the weighted sum optimization problem

3 The Triangle Splitting Method

The triangle splitting method maintains a priority queue with rectangles and triangles, each of which still has to be explored, i.e., may still contain as yet unknown nondominated points. Each element of the priority queue is characterized by two nondominated points z^1 and z^2, a shape, ***rectangle*** or ***triangle***, and a splitting direction, ***horizontal*** or ***vertical***. The algorithm also maintains an ordered list of nondominated points, which is updated after finding a new nondominated point or after detecting that all points on the hypotenuse of a triangle are nondominated. The nondominated points are maintained in order of nondecreasing value of their first objective value. In addition to the nondominated point itself, there is an indicator that specifies whether the nondominated point is connected to the next nondominated point in the list (indicator value 1) or not (indicator value 0), i.e., whether all points on the line segment defined by the two nondominated points are also nondominated. The list is initialized with $(z^T, 0)$ and $(z^B, 0)$. The priority queue is initialized with $(z^T, z^B, \textbf{\textit{rectangle}}, \textbf{\textit{horizontal}})$.

Next, we discuss how rectangles and triangles are explored. A rectangle is explored by applying the weighted sum method to find locally extreme supported nondominated points and divide the rectangle into one or more triangles. The locally supported nondominated points are added to the list of nondominated points and the triangles are added to the priority queue (with the same splitting direction). See Figure 3 for an example of the exploration of a rectangle. Exploring a triangle $T(z^1, z^2)$ starts by determining whether all the points on the hypotenuse of the triangle are nondominated. (Note that by construction, there are no nondominated points "below" the hypotenuse, i.e., $\{(z - \mathbb{R}^2_>) \cap \mathcal{Y}_N : z \in H(z^1, z^2)\} = \emptyset$.) If so, the list of nondominated points is updated accordingly, i.e., element $(z^1, 0)$ is changed to $(z^1, 1)$. (Note that z^1 and z^2 appear consecutively in the list of nondominated points). Otherwise, the triangle will be split into two rectangles, either by splitting the triangle horizontally at $\frac{z_2^1 + z_2^2}{2}$ or by splitting the triangle vertically at $\frac{z_1^1 + z_1^2}{2}$, which will then be added to the priority queue unless they cannot contain as yet unknown nondominated points.

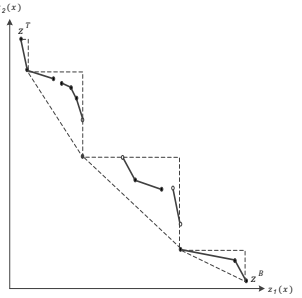

Fig. 3. The triangles determined by the exploration of initial rectangle $R(z^T, z^B)$ for the BOMIP that gives rise to the nondominated frontier of Figure 1

More specifically, when splitting triangle $T(z^1, z^2)$ horizontally at height $\frac{z_2^1+z_2^2}{2}$, we start by computing

$$\bar{z}^1 = \operatorname*{lex\,min}_{x\in\mathcal{X}}\{z_1(x), z_2(x) : z_2(x) \leq \frac{z_2^1 + z_2^2}{2},\ z(x) \in T(z^1, z^2)\}.$$

If $\bar{z}_2^1 = \frac{z_2^1+z_2^2}{2}$, i.e., if the resulting point is on the cut, then we set $\bar{z}^2 = \bar{z}^1$. If not, then we compute

$$\bar{z}^2 = \operatorname*{lex\,min}_{x\in\mathcal{X}}\{z_2(x), z_1(x) : z_1(x) \leq \bar{z}_1^1,\ z(x) \in T(z^1, z^2)\}.$$

By construction of $\bar{z}^1$ and $\bar{z}^2$, all as yet unknown nondominated points in triangle $T(z^1, z^2)$ must be in rectangles $R(z^1, \bar{z}^2)$ and $R(\bar{z}^1, z^2)$. Note that when $\bar{z}^2 = z^1$, rectangle $R(z^1, \bar{z}^2)$ consists of a single nondominated point and does not need to be explored further (and, similarly, when $\bar{z}^1 = z^2$ rectangle $R(\bar{z}^1, z^2)$ consists of a single nondominated point and does not need to be explored further). The two situations that can occur during horizontal splitting at height $\frac{z_2^1+z_2^2}{2}$ are illustrated in Figures 4 and 5. Note that if $\bar{z}^1$ is on the cut, then it is easy to see that $\bar{z}^2 = \bar{z}^1$ and we do not need to compute $\bar{z}^2$. (In this case, $\bar{z}^1 = \bar{z}^2$ is almost surely on a line segment of the nondominated frontier.) Splitting a triangle vertically at height $\frac{z_1^1+z_1^2}{2}$ proceeds analogously.

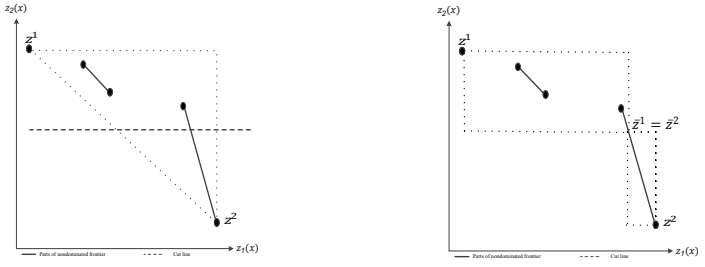

Fig. 4. Horizontal splitting of triangle $T(z^1, z^2)$ when $\bar{z}^1$ is on the cut

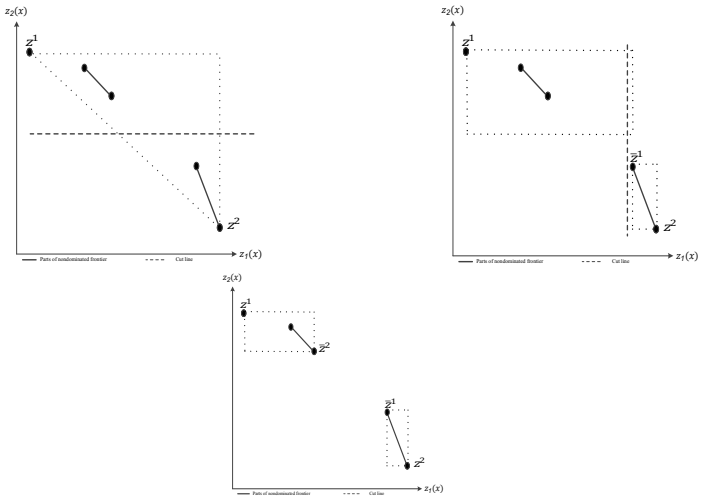

Fig. 5. Horizontal splitting of triangle $T(z^1, z^2)$ when $\bar{z}^1$ is not on the cut

The splitting direction plays an important role in the triangle splitting method. Recall that a nondominated frontier may contain horizontal and vertical discontinuities or gaps. If the same splitting direction is used throughout the algorithm, finding these gaps can be unnecessarily time-consuming or even impossible.

Figure 6 shows the progression of the triangle splitting method for a nondominated frontier with a horizontal gap when only horizontal cutting is employed to split triangles. In this situation, the triangle splitting methods continuous to split the continuous segment of the nondominated frontier until the area of the remaining rectangles is less than a pre-specified tolerance. To avoid these situations, an *alternating splitting direction* strategy is employed, i.e., the splitting

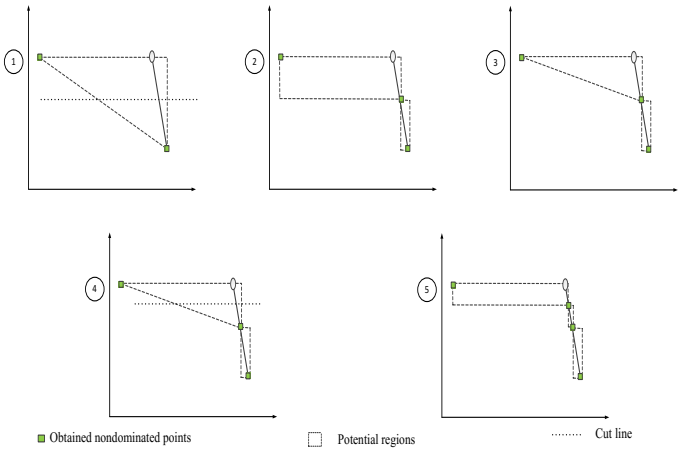

Fig. 6. A horizontal gap may not be detected when splitting horizontally only

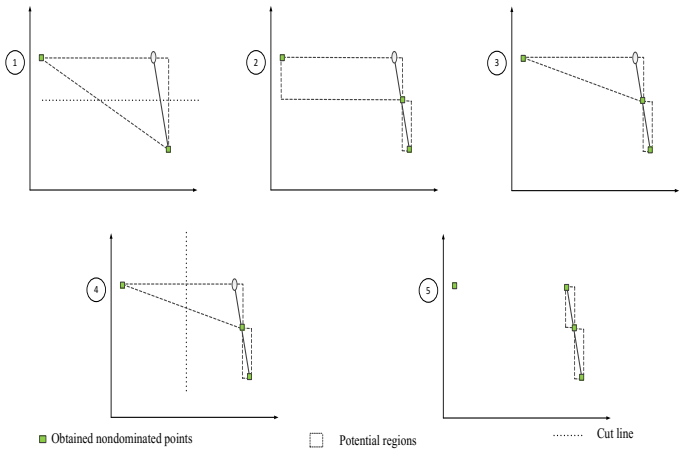

Fig. 7. A horizontal gap is easily detected using alternating splitting directions

direction for newly generated rectangles is set to the opposite of the direction that was used to create these rectangles. Figure 7 shows the progression of the triangle splitting method for the same nondominated frontier when an alternating splitting direction strategy is employed. We see that the horizontal gap is detected easily.

Thus, rectangles are added to the priority queue with the opposite splitting direction.

To investigate the efficacy of the triangle splitting method, we used the class of biobjective 0-1 mixed integer programs introduced by Mavrotas et al. [6]. This class has been used in all previous computational studies of algorithms for BOMIPs. We generated five sets of instances for our computational study, each consisting of five instances and identified by the number of constraints m in the instance, i.e., S20, S40, S80, S160, and S320. The number of variables is equal to the number of constraints, and half of the variables are binary and half of the variables are continuous. The sets S160 and S320 contain instances that are much larger than any instances solved in previous studies. For example, Vincent et al. [8] solve instances of up to size 70 and Belotti et al. [2] solve instances up to size 80. The average solution time for the instances in each set is shown in Table 1. We observe that the triangle splitting algorithm is able to solve large instances in a relatively short amount of time. Even the largest instances (in Set S320) are solved in little over an hour.

Table 1. Runtime of the triangle splitting method

Set	**S20**	**S40**	**S80**	**S160**	**S320**
Time (secs.)	0.60	2.76	29.08	274.54	3852.63

References

1. Aneja, Y.P., Nair, K.P.K.: Bicriteria transportation problem. Management Science 27, 73–78 (1979)
2. Belotti, P., Soylu, B., Wiecek, M.M.: A branch-and-bound algorithm for biobjective mixed-integer programs, http://www.optimization-online.org/DB_HTML/2013/01/3719.html
3. Benson, H.P., Sun, E.: A weight set decomposition algorithm for finding all efficient extreme points in the outcome set of multiple objective linear program. European Journal of Operational Research 139, 26–41 (2002)
4. Gardenghi, M., Gómez, T., Miguel, F., Wiecek, M.M.: Algebra of efficient sets for multiobjective complex systems. Journal of Optimization Theory and Applications 149, 385–410 (2011)
5. Isermann, H.: The enumeration of the set of all efficient solutions for a linear multiple objective program. Operational Research Quarterly 28(3), 711–725 (1977)
6. Mavrotas, G., Diakoulaki, D.: A branch and bound algorithm for mixed zero-one multiple objective linear programming. European Journal of Operational Research 107, 530–541 (1998)
7. Mavrotas, G., Diakoulaki, D.: Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming. Applied Mathematics and Computation 171, 53–71 (2005)
8. Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., Gandibleux, X.: Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for biobjective case. Computers & Operations Research 40(1), 498–509 (2013)