A BRANCH-AND-BOUND ALGORITHM FOR BIOBJECTIVE MIXED-INTEGER PROGRAMS

PIETRO BELOTTI[†], BANU SOYLU[‡], AND MARGARET M. WIECEK[†]

Abstract. We propose a branch-and-bound (BB) algorithm for biobjective mixed-integer linear programs (BOMILPs). Our approach makes no assumption on the type of problem and we prove that it returns all Pareto points of a BOMILP. We discuss two techniques upon which the BB is based: fathoming rules to eliminate those subproblems that are guaranteed not to contain Pareto points and a procedure to explore a subproblem based on the parametric simplex method.

The BB algorithm starts with an initial set of feasible points that evolves toward the Pareto set of the BOMILP. This set is gradually modified by including new points and eliminating dominated points, and, upon termination, it is identical to the Pareto set.

We provide computational results on instances of various sizes to prove the effectiveness of our algorithm.

Key words. Biobjective programming, mixed-integer linear programming, fathoming rules, branch and bound.

AMS subject classifications. 90C29, 90C11, 90C57.

1. Introduction.

Multiobjective programming finds application in a number of fields, including engineering, business, and management. Many practical problems can be represented with discrete quantities, and this gives rise to the class of multiobjective discrete programs. Often, it is possible to express relationships between variables through linear constraints, and in the most general form only a subset of variables is discrete: this is the class of *multiobjective mixed-integer linear programs* (MOMILP). Subclasses of these problems have been studied before, although the literature focuses more on the *pure* integer case, where all variables are integer.

We consider a bi-objective mixed-integer linear program (BOMILP) of the form

$$\begin{array}{lll} P: & \max_{\boldsymbol{x}} & \boldsymbol{z}(\boldsymbol{x}) = (z_1(\boldsymbol{x}) = \boldsymbol{c}_1^\top \boldsymbol{x}, z_2(\boldsymbol{x}) = \boldsymbol{c}_2^\top \boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in X, \end{array}$$

where $X = \{\boldsymbol{x} \in \mathbb{R}^p \times \mathbb{Z}^{n-p} : A\boldsymbol{x} \leq \boldsymbol{b}\}$ is the set of feasible solutions, $n > 1$, $1 \leq p < n$, $A \in \mathbb{Q}^{m \times n}$, $\boldsymbol{b} \in \mathbb{Q}^m$, and $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathbb{Q}^n$. There are p *continuous* and $n-p$ *integer* decision variables. We assume that each integer variable x_i is bounded, i.e., $-\infty < \ell_i \leq x_i \leq u_i < +\infty$ for $i = p+1, p+2, \ldots, n$, and that these bounds are contained in the inequalities $A\boldsymbol{x} \leq \boldsymbol{b}$.

Biobjective mixed-binary linear programs have been proposed to model decision making problems in various areas of human activity such as facility location [20], hub location [7], forward/reverse logistics [9, 23], and scheduling [21] among others. The proposed solution methods rely on heuristic approaches or involve interactive procedures with no intention to compute the complete set of Pareto points, and typically exploit the problem structure.

In this paper, we develop a branch and bound (BB) algorithm for computing efficient solutions and Pareto points of BOMILPs. To the best of our knowledge, this is the first attempt at proposing a general purpose algorithm for biobjective linear programs with integer and continuous variables. The algorithm computes all Pareto points of the BOMILP. The branching operations in the algorithm are similar to the single-objective case. The bounding operations, however, make use of new fathoming rules that rely on solving specially designed linear programs rather than information provided by the BOMILP or the user, a prevailing feature in the available BB algorithms.

[†]Department of Mathematical Sciences, Clemson University, Clemson, SC 29630, USA.
[‡]Department of Industrial Engineering, Erciyes University, 38039 Kayseri, Turkey.

Working with BOMILPs requires us to combine notation and methodology from two disciplines, therefore we begin by summarizing them in the next two sections: Section 2 introduces notation for multiobjective programming, while Section 3 describes *branch and bound* (BB), a well known algorithm for solving single-objective mixed-integer linear programs (SOMILPs). Sections 2 or 3, or both, can be safely skipped by a reader with expertise in the corresponding field, as they mainly provide notation.

In Section 4, previous work on BB in multiobjective programming is summarized, the main differences between BB methods in the single-objective and multiobjective case are reviewed, and an extension of implicit enumeration techniques for multiobjective problems is discussed. A key component of a BB method is the *fathoming rule*, i.e., a procedure to determine if a subset of feasible solutions can be discarded without excluding Pareto points. Previous work on this subject is presented in a unified manner in the same section. New fathoming rules and their implementation in the form of solving linear programs are introduced in Section 5. The BB algorithm with all its procedures is presented in Section 6 which is followed with an example in Section 7. Computational results are reported in Section 8 on instances from a library of BOMILPs and on a biobjective set covering problem. The paper is concluded in Section 9.

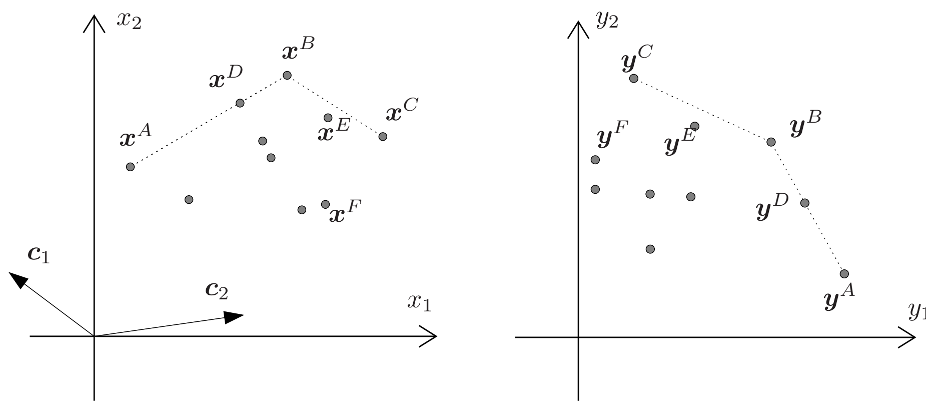
2. Multiobjective programming. The reader may refer to Steuer [26] and Ehrgott [10] for a detailed introduction to multiobjective programming. For any two vectors $\boldsymbol{y}^1, \boldsymbol{y}^2 \in \mathbb{R}^2$, we use the following notation: $\boldsymbol{y}^1 \geqq \boldsymbol{y}^2$ if $y_i^1 \geq y_i^2$ for $i = 1, 2$; $\boldsymbol{y}^1 \geq \boldsymbol{y}^2$ if $\boldsymbol{y}^1 \geqq \boldsymbol{y}^2$ and $\boldsymbol{y}^1 \neq \boldsymbol{y}^2$; and $\boldsymbol{y}^1 > \boldsymbol{y}^2$ if $y_i^1 > y_i^2$ for $i = 1, 2$. We also define the sets $\mathbb{R}^2_> = \{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} > \boldsymbol{0}\}, \mathbb{R}^2_\geq = \{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} \geq \boldsymbol{0}\}, \mathbb{R}^2_\geqq = \{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} \geqq \boldsymbol{0}\}$. We define the sets $\mathbb{R}^2_<, \mathbb{R}^2_\leq$, and $\mathbb{R}^2_\leqq$ in a similar way. For an arbitrary set $S \subseteq \mathbb{R}^2$ we define $S^> = S + \mathbb{R}^2_>, S^\geq = S + \mathbb{R}^2_\geq$, and $S^\geqq = S + \mathbb{R}^2_\geqq$, where the symbol $+$ denotes the algebraic sum of two sets. Similarly, we define the sets $S^<, S^\leq$, and $S^\leqq$. We denote by $\text{conv}(S)$ the convex hull of set S.

A point $\boldsymbol{y} \in S$ is called *isolated* provided there exists a neighborhood of $\boldsymbol{y}$ that does not include any other point in S. Let $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ be two distinct points in S such that $y_1^1 < y_1^2$ and $y_2^1 > y_2^2$. The point $\check{\boldsymbol{y}} = (y_1^1, y_2^2)$ is called the *local nadir point* w.r.t. $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$, and the point $\boldsymbol{y}^\star = (y_1^2, y_2^1)$ is called the *local ideal point* w.r.t. $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$. The *ideal point* $\boldsymbol{y}^I = (y_1^I, y_2^I)$ of problem P is defined as $y_i^I = \max\{z_i(\boldsymbol{x}) : \boldsymbol{x} \in X\}$ for $i = 1, 2$. The *nadir point* $\boldsymbol{y}^N = (y_1^N, y_2^N)$ of problem P is defined as $y_i^N = \min\{z_i(\boldsymbol{x}) : \boldsymbol{x} \in X\}$ for $i = 1, 2$.

Let $[\boldsymbol{y}^1, \boldsymbol{y}^2]$ be a line segment in S such that $y_1^1 < y_1^2$ and $y_2^1 > y_2^2$. We define the *local nadir set* of $[\boldsymbol{y}^1, \boldsymbol{y}^2]$ as the line segment itself. Similarly, let $[\boldsymbol{y}^1, \boldsymbol{y}^s]$ be a piecewise linear curve in S composed of line segments $[\boldsymbol{y}^i, \boldsymbol{y}^{i+1}]$ such that $y_1^i < y_1^{i+1}$ and $y_2^i > y_2^{i+1}, i = 1, \cdots, s-1$. We define the local nadir set of $[\boldsymbol{y}^1, \boldsymbol{y}^s]$ as the curve itself. All line segments $[\boldsymbol{y}^i, \boldsymbol{y}^{i+1}]$ composing $[\boldsymbol{y}^1, \boldsymbol{y}^s]$ are also called local nadir sets.

Consider again problem P. We assume that X is nonempty and that neither of the single objective problems $\max\{\boldsymbol{c}_1^\top \boldsymbol{x} : \boldsymbol{x} \in X\}$ and $\max\{\boldsymbol{c}_2^\top \boldsymbol{x} : \boldsymbol{x} \in X\}$ is unbounded. Let the set $Y = \{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} = (\boldsymbol{c}_1^\top \boldsymbol{x}, \boldsymbol{c}_2^\top \boldsymbol{x}), \boldsymbol{x} \in X\}$ be the set of attainable objective vectors. The spaces $\mathbb{R}^p \times \mathbb{Z}^{n-p}$ and $\mathbb{R}^2$ are referred as the *decision space* and the *objective space*, respectively. If $\boldsymbol{x}^1, \boldsymbol{x}^2 \in X$ and $\boldsymbol{z}(\boldsymbol{x}^1) \geq \boldsymbol{z}(\boldsymbol{x}^2)$, then $\boldsymbol{y}^1 = \boldsymbol{z}(\boldsymbol{x}^1)$ is said to *dominate* $\boldsymbol{y}^2 = \boldsymbol{z}(\boldsymbol{x}^2)$. The inequalities $>$ and $\geqq$ between the objective vectors define *strict* and *weak dominance*, respectively. A feasible solution $\hat{\boldsymbol{x}} \in X$ is said to be *(weakly) efficient* for problem P if there is no feasible solution $\boldsymbol{x} \in X$ such that $\boldsymbol{y} = \boldsymbol{z}(\boldsymbol{x})$ (strictly) dominates $\hat{\boldsymbol{y}} = \boldsymbol{z}(\hat{\boldsymbol{x}})$. If $\hat{\boldsymbol{x}}$ is efficient, then $\hat{\boldsymbol{y}} = \boldsymbol{z}(\hat{\boldsymbol{x}})$ is said to be a *Pareto* point. Let X_E and Y_N denote the sets of all efficient solutions and Pareto points of problem P, respectively.

For any set Q, we define $\boldsymbol{z}(Q) = \{\boldsymbol{z}(\boldsymbol{x}) : \boldsymbol{x} \in Q\}$. For an arbitrary set Q in $\mathbb{R}^n$ and the set S in $\mathbb{R}^2$ such that $S = \boldsymbol{z}(Q)$, we define two operators, $VMAX$ and EFF, as follows:



(a) Decision space (b) Objective space

FIG. 2.1. *Efficient solutions and dominated points for a pure integer biobjective program. The two objective function coefficient vectors $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are indicated in the decision space (a) for clarity. Each solution $\boldsymbol{x}^i$ in the decision space is associated with a point $\boldsymbol{y}^i$, with the same index, in the objective space. Note that $\boldsymbol{x}^A, \boldsymbol{x}^B, \ldots, \boldsymbol{x}^E$ are efficient solutions and $\boldsymbol{y}^A, \boldsymbol{y}^B, \ldots, \boldsymbol{y}^E$ are Pareto points. Also, $\boldsymbol{x}^A$, $\boldsymbol{x}^B$, and $\boldsymbol{x}^C$ are extreme supported solutions, while $\boldsymbol{x}^D$ is a nonextreme supported solution and $\boldsymbol{x}^E$ is an unsupported solution. Finally, $\boldsymbol{x}^F$ is not an efficient solution and $\boldsymbol{y}^F$ is a dominated point.*

$S_N = VMAX(S)$ and $EFF(Q) = Q_E$ where $S_N = \boldsymbol{z}(Q_E)$.

The concept of efficient solutions and Pareto points can be defined on any subset of the feasible set X and any subset S of Y, respectively. Let $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ be two Pareto points of S, and suppose w.l.o.g. that $y_1^1 < y_1^2$ and $y_2^1 > y_2^2$. The points $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ are said to be *adjacent* if their ideal point $\boldsymbol{y}^*$ and local nadir point $\check{\boldsymbol{y}}$ satisfy $(\{\check{\boldsymbol{y}}\} + \mathbb{R}^2_>) \cap (\{\boldsymbol{y}^\star\} + \mathbb{R}^2_<) \cap S = \emptyset$.

There are two types of efficient solutions: *supported* and *unsupported*. An efficient solution $\boldsymbol{x}$ is a *supported* efficient solution for problem P if it is an optimal solution of the following single objective weighted-sum problem for some $\lambda \in [0, 1]$:

$$\begin{array}{ll} \max_{\boldsymbol{x}} & \lambda \boldsymbol{c}_1^\top \boldsymbol{x} + (1-\lambda)\boldsymbol{c}_2^\top \boldsymbol{x} \\ \text{s.t.} & \boldsymbol{x} \in X. \end{array} \tag{2.1}$$

Let X_{SE} denote the set of all supported efficient solutions of problem P. The set X_{SE} has a counterpart in the objective space: $Y_{SN} = \{\boldsymbol{y} \in \mathbb{R}^2 : y_1 = \boldsymbol{c}_1^\top \boldsymbol{x}, y_2 = \boldsymbol{c}_2^\top \boldsymbol{x}, \boldsymbol{x} \in X_{SE}\}$ is the set of supported Pareto points for P. If $\boldsymbol{x}$ is a supported efficient solution and the corresponding $\boldsymbol{y}$ is an extreme point of conv(Y), then $\boldsymbol{x}$ is said to be an *extreme supported efficient solution* and $\boldsymbol{y}$ is an *extreme Pareto point*. If $\boldsymbol{x}$ is a supported efficient solution and the corresponding $\boldsymbol{y}$ is on the boundary of conv(Y) but not one of the extreme points of conv(Y), then $\boldsymbol{x}$ is said to be a *nonextreme supported efficient solution* and $\boldsymbol{y}$ is a *nonextreme supported Pareto point*. If an efficient solution $\boldsymbol{x}$ is not an optimal solution of problem (2.1) for any λ, then $\boldsymbol{x}$ and its image $\boldsymbol{y}$ are said to be *unsupported*.

Figure 2.1(a) shows a feasible set of a purely integer problem with two variables, and Figure 2.1(b) presents its objective space. Efficient solutions and Pareto points of different types are depicted. In particular, the points $\boldsymbol{y}^C$ and $\boldsymbol{y}^E$, $\boldsymbol{y}^E$ and $\boldsymbol{y}^B$, $\boldsymbol{y}^B$ and $\boldsymbol{y}^D$, $\boldsymbol{y}^D$ and $\boldsymbol{y}^A$ are adjacent. However, the points $\boldsymbol{y}^B$ and $\boldsymbol{y}^A$ are not adjacent.

The linear programming (LP) relaxation of P, denoted $\tilde{P}$, is

$$\begin{array}{lll}\tilde{P}: & \max_{\boldsymbol{x}} & \boldsymbol{z}(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in \tilde{X},\end{array}$$

where $\tilde{X} = \{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} \leq \boldsymbol{b}\}$ is the set of feasible solutions of $\tilde{P}$. As a consequence, $X = \tilde{X} \cap (\mathbb{R}^p \times \mathbb{Z}^{n-p})$. The set $\tilde{Y} = \{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} = (\boldsymbol{c}_1^\top \boldsymbol{x}, \boldsymbol{c}_2^\top \boldsymbol{x}), \boldsymbol{x} \in \tilde{X}\}$ is the set of attainable objective vectors of $\tilde{P}$. We denote as $\tilde{X}_E$ the set of efficient solutions of $\tilde{P}$ and as $\tilde{Y}_N$ the set of Pareto points of $\tilde{P}$. Problem $\tilde{P}$ is a biobjective linear program and its Pareto set is a piecewise linear concave curve.

The ideal point and the nadir point of problem $\tilde{P}$ are denoted by $\tilde{\boldsymbol{y}}^I$ and $\tilde{\boldsymbol{y}}^N$, respectively. Two *extreme points,*

$$\boldsymbol{y}^{N-W} = [\tilde{y}_1^N, \tilde{y}_2^I] \qquad \text{and} \qquad \boldsymbol{y}^{S-E} = [\tilde{y}_1^I, \tilde{y}_2^N] \tag{2.2}$$

determine the bounds in the objective space for the Pareto set of problem P. Abbreviations N-W and S-E come from North-West and South-East, respectively.

For any feasible solution $\boldsymbol{x}$ of problem P, it is easy to verify that

$$\begin{array}{ccccc} y_1^{N-W} & \leq & \boldsymbol{c}_1^\top \boldsymbol{x} & \leq & y_1^{S-E} \\ y_2^{S-E} & \leq & \boldsymbol{c}_2^\top \boldsymbol{x} & \leq & y_2^{N-W}. \end{array}$$

A *slice problem* of problem P is defined as follows:

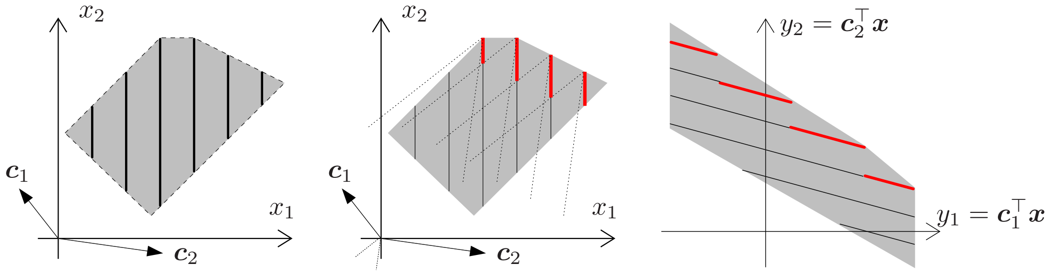
$$\begin{array}{lll}P(\bar{\boldsymbol{x}}): & \max_{\boldsymbol{x}} & \boldsymbol{z}(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in X(\bar{\boldsymbol{x}}),\end{array}$$

where the set $X(\bar{\boldsymbol{x}}) = \{\boldsymbol{x} \in \mathbb{R}^p \times \mathbb{Z}^{n-p} : A\boldsymbol{x} \leq \boldsymbol{b}, x_i = \bar{x}_i, i = p+1, p+2, \ldots, n\}$ defines a *slice* of the set X. In a slice, all integer components of $\boldsymbol{x}$ are fixed to the corresponding components of $\bar{\boldsymbol{x}} \in \mathbb{R}^p \times \mathbb{Z}^{n-p}$. Let $X_E(\bar{\boldsymbol{x}})$ denote the set of efficient solutions of problem $P(\bar{\boldsymbol{x}})$. To illustrate the idea of slice, consider the projection of X onto the subspace of integer variables, and denote this projection $\mathcal{S} \subseteq \mathbb{Z}^{n-p}$. Consider now the set $X_I = \{(\boldsymbol{0}, \boldsymbol{x}) \in \mathbb{R}^p \times \mathbb{Z}^{n-p}, \boldsymbol{x} \in \mathcal{S}\}$, i.e., the set of vectors $\tilde{\boldsymbol{x}}$ such that there exists at least one feasible solution $\boldsymbol{x} \in X$ with the same assignment of the integer variables as $\tilde{\boldsymbol{x}}$. Then $X = \bigcup_{\bar{\boldsymbol{x}} \in X_I} X(\bar{\boldsymbol{x}})$.

The set $Y(\bar{\boldsymbol{x}}) = \{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} = (\boldsymbol{c}_1^\top \boldsymbol{x}, \boldsymbol{c}_2^\top \boldsymbol{x}), \boldsymbol{x} \in X(\bar{\boldsymbol{x}})\}$ is the set of attainable objective vectors of problem $P(\bar{\boldsymbol{x}})$. Let $Y_N(\bar{\boldsymbol{x}})$ denote the set of Pareto points of problem $P(\bar{\boldsymbol{x}})$. Because $P(\bar{\boldsymbol{x}})$ is a biobjective linear program, the set $Y_N(\bar{\boldsymbol{x}})$ is also a piecewise linear concave curve. Applying Proposition 3.1 of Gardenghi et al. [12], we obtain

$$Y_N = VMAX\left(\bigcup_{\bar{\boldsymbol{x}} \in X_I} Y_N(\bar{\boldsymbol{x}})\right). \tag{2.3}$$

A multiobjective pure integer program admits a finite number of efficient solutions (and Pareto points) if the feasible set is bounded. In contrast to the pure integer case, multiobjective mixed-integer problems can admit infinitely many efficient solutions (and Pareto points) even if the feasible set of the LP relaxation of the original problem is bounded. As illustrated in Figure 2.2, the Pareto set may be a union of single points and line segments in the objective space. Each segment contains infinitely many Pareto points of a slice problem $P(\bar{\boldsymbol{x}})$, while singletons (which are not present in the example in Figure 2.2) correspond to integer solutions $\boldsymbol{x}$ whose slice $P(\boldsymbol{x})$ does not admit other Pareto points.



(a) Feasible set (b) Efficient solutions (c) Pareto points

FIG. 2.2. *Feasible, efficient, and Pareto sets for a BOMILP. Suppose $x_1 \in \mathbb{Z}$ and $x_2 \in \mathbb{R}$. In (a), an exemplified X is represented by a polyhedron intersected with the integrality constraint in x_1. The feasible solutions are hence the vertical bars within the polyhedron. In (b), the two normals to the vectors $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ form a cone (indicated with the pointed half-lines originating at $(0, 0)$) that determines the efficient solutions (marked with a bold trait). In (c), these efficient solutions are mapped into the Pareto points (marked with a bold trait).*

3. Branch and bound in single-objective programming.

In the context of single-objective programming, problems with integrality constraints on variables are solved using a variety of methods. Among the most effective general purpose methods is *branch and bound* (BB), introduced by [17]. BB algorithms are implicit enumeration techniques based on a recursive partition of the set of solutions.

The general scheme solves a convex relaxation of the problem, which yields a valid upper bound (for a maximization problem) on the optimal solution value or determines that the relaxation, and hence the problem, are infeasible. If an optimal solution to the convex relaxation is infeasible for the original problem (for instance, some of its integrally constrained variables are fractional), the algorithm identifies a valid logical disjunction that is violated by the solution and subdivides the problem into two subproblems (containing each a subset of feasible solutions), on which the BB is recursively applied. By recursively subdividing the feasible set, BB methods create a binary tree of subproblems known as *BB tree*, whose elements are interchangeably called *BB nodes* or *subproblems*.

In order to simplify notation, we define each BB node (subproblem) similarly to problem P: the generic BB node is identified with an integer k and associated with subproblem P_k. The operation of partitioning subproblem P_k, which yields two new subproblems P'_k and P''_k, is known as *branching*: problem P'_k is obtained by amending P_k with a linear inequality $\boldsymbol{a}_1^\top \boldsymbol{x} \le b_1$, whereas problem P''_k is created by adding a linear inequality $\boldsymbol{a}_2^\top \boldsymbol{x} \le b_2$. Let X_k, X'_k and X''_k denote the feasible sets of problems P_k, P'_k, and P''_k, respectively. Then $X'_k = \{\boldsymbol{x} \in X_k : \boldsymbol{a}_1^\top \boldsymbol{x} \le b_1\}$ and $X''_k = \{\boldsymbol{x} \in X_k : \boldsymbol{a}_2^\top \boldsymbol{x} \le b_2\}$. In general, $X'_k \cap X''_k = \emptyset$ as this ensures that a feasible solution is not visited twice, and $X'_k \cup X''_k = X_k$ to guarantee that no feasible solution is excluded. Rudimentary BB implementations use simple branching rules of the form $x_i \le \alpha \vee x_i \ge \alpha + 1$, with $\alpha \in \mathbb{Z}$, for an integer variable x_i, but more efficient rules can be employed. As a side note, branching rules are only one part of a more involved procedure that creates subproblems and selects which of them should be analyzed at every step. The feasible set X_k of P_k can then be rewritten as follows:

$$X_k = \{\boldsymbol{x} \in \mathbb{R}^p \times \mathbb{Z}^{n-p} : A_k \boldsymbol{x} \le \boldsymbol{b}_k\},$$

where A_k and $\boldsymbol{b}_k$ are obtained by augmenting A and $\boldsymbol{b}$, respectively, with the branching rules enforced on all nodes between the *root* node of the BB tree (i.e., the subproblem corresponding to P) and node k. In other words, X_k is a restriction of the feasible set X of problem P, and is generated through branching rules.

BB algorithms are known as implicit enumeration techniques since the entire feasible set is considered, but only its most promising regions are actually visited. Those regions that are proven not to contain any optimal solution are excluded through various mechanisms. The effectiveness of a BB relies heavily on the possibility of excluding an entire subproblem from the search when it can be proved that the subproblem does not contain any optimal solution. In order to exclude a subproblem, one usually requires knowledge of a feasible solution. A feasible solution with a very large objective function value (in maximization problems), in general, helps decrease the computation time. This is achieved, for instance, by comparing the objective function value z_k^U of an optimal solution of a convex relaxation of subproblem P_k with the objective value z^L of a feasible solution of problem P. In a maximization SOMILP, z_k^U is called an *upper bound* while z^L yields a *lower bound*. If

$$z_k^U \leq z^L, \tag{3.1}$$

then subproblem P_k does not have a feasible solution with objective function value better than z_k^U, let alone z^L, and hence can be discarded.

The portion of the BB algorithm devoted to partitioning the solution set is one of the most studied parts of single-objective mixed-integer programming solvers, and has evolved into algorithms such as *strong branching* [4], *pseudocost branching* [5], and *reliability branching* [1], which are proven to dramatically reduce the size of the BB tree.

4. Branch and bound in multiobjective programming.

In this section we first review the state-of-the-art in multiobjective B&B and then build a conceptual foundation for fathoming a BB node. We propose a generic fathoming rule and show that it encompasses other rules that are available in integer and mixed-binary multiobjective linear programming.

Literature review. The literature on single-objective programming comprises several variants of the BB method that have been developed for solving many classes of problems, especially combinatorial optimization problems. Although these methods are well-known in the single objective context, there is only a limited number of studies in the multiobjective case.

Early studies on pure integer linear multiobjective programming go back to the early nineteen eighties when Bitran and Rivera [6] and Kızıltan and Yucaoğlu [16] presented BB algorithms specifically for the binary case. Later Visée et al. [29] develop a two-phase method for the biobjective binary knapsack problem, with a BB approach being the second phase. For a scheduling problem, Sayın and Karabatı [24] present a BB algorithm enumerating all efficient solutions.

BB algorithms for multiobjective combinatorial problems with pure integer variables have also been proposed. BB procedures for biobjective problems are introduced by Ehrgott and Gandibleux [11] who address the dependence of the efficiency of a BB method on the procedure for excluding subproblems. Sourd and Spanjaard [25] develop a BB algorithm for multiobjective problems and apply it to a spanning tree problem, while Jozefowiez et al. [15] develop a generic branch and cut algorithm and apply it to a traveling salesman problem.

Multiobjective mixed-binary problems have been studied more recently. Mavrotas and Diakoulaki [18] first develop a BB algorithm and then incorporate new elements to increase its speed and reliability [19]. The techniques presented in the latter are improved upon by Vincent et al. [27, 28]. Their improvements are basically on the representation of the Pareto set, using better bounds and branching strategies. An exact algorithm for computing all extreme supported Pareto points of a multiobjective mixed-integer linear program is designed by

Özpeynirci and Köksalan [22]. Finally, Alves and Clímaco [2] provide a survey of interactive and noninteractive BB methods for multiobjective integer/mixed-integer problems.

We could not find published work on exact approaches for general BOMILPs. This is the subject of the present paper: a BB algorithm for finding the Pareto set for this class of problems. Such an algorithm needs to have two important properties: the capability of searching the Pareto set associated with a subproblem, which is not even relevant to SOMILPs because their subproblems have scalar optimal values, and the ability to eliminate a subproblem, which for SOMILPs relies on solving linear programs. The latter technique will be carried to BOMILPs but with new linear programming formulations and new criteria for exclusion. In general, a subproblem contains a (possibly empty) subset of Pareto points, and by subdividing a problem into two new subproblems one can recursively apply the search on each subproblem. In order to guarantee that the output is the complete Pareto set of the original problem, we must ensure that the process of subdividing a subproblem does not eliminate any Pareto point.

Basic concepts. In a BB algorithm for BOMILPs, branching operations are very similar to the single-objective case, as the feasible set does not change: branching creates a BB tree whose root node is the original problem P. A slice problem $X(\bar{\boldsymbol{x}})$ is hence equivalent to a leaf node k' in the BB tree, because no branching is necessary at k': all integer variables have been fixed. Subproblem P_k is defined as $\max_{\boldsymbol{x}}\{\boldsymbol{z}(\boldsymbol{x}) : \boldsymbol{x} \in X_k\}$. The set $Y_k = \{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} = (\boldsymbol{c}_1^\top \boldsymbol{x}, \boldsymbol{c}_2^\top \boldsymbol{x}), \boldsymbol{x} \in X_k\}$ is the set of attainable objective vectors of P_k and Y_{kN} is the set of Pareto points of P_k. The LP relaxation of subproblem P_k is defined as follows:

$$\begin{array}{lll} \tilde{P}_k : & \max_{\boldsymbol{x}} & \boldsymbol{z}(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in \tilde{X}_k, \end{array}$$

where $\tilde{X}_k = \{\boldsymbol{x} \in \mathbb{R}^n : A_k \boldsymbol{x} \leq \boldsymbol{b}_k\}$ is the set of feasible solutions of $\tilde{P}_k$. We denote as $\tilde{X}_{kE}$ the set of efficient solutions and as $\tilde{Y}_{kN}$ the set of Pareto points of subproblem $\tilde{P}_k$. We extend the definition of ideal and nadir points to problem $\tilde{P}_k$ by denoting them $\tilde{\boldsymbol{y}}^{kI}$ and $\tilde{\boldsymbol{y}}^{kN}$, respectively. Let $\boldsymbol{y}^{k,N-W}$ and $\boldsymbol{y}^{k,S-E}$ denote the two extreme points of problem $\tilde{P}_k$.

A BB method must use an effective procedure for identifying subproblems that can be proved not to contain Pareto points. While the problem of deciding whether a BB node contains Pareto points is NP-complete, and is as difficult as problem P itself, one can obtain sufficient conditions for fathoming subproblem P_k by comparing two sets:

- the subset $\hat{Y}$ of Y containing attainable points of problem P that have been found so far by the BB algorithm and such that $\hat{Y} = \hat{Y}_N$ (note that $\hat{Y}$ may contain none or only some of the Pareto points of problem P, and that $\hat{Y} \nsubseteq Y_k$), and
- the Pareto set $\tilde{Y}_{kN}$ of the LP relaxation of subproblem P_k.

The former set is, in general, updated continuously during the BB, and is akin to the lower bound used as a *cutoff* value in single-objective BB methods. The tighter (i.e., larger in a maximization problem) the bound, the more likely a subproblem can be safely excluded from the search. The latter set can be obtained in polynomial time with methods such as the *dichotomic search* [3] or the *parametric simplex method* [10].

Consider a set $\hat{X} \subseteq X$ of feasible solutions for P and its counterpart $\hat{Y}$ in the objective space. Note that whether or not the set $\hat{X}$ contains efficient solutions of problem P is of no interest here. Rather, during the execution of a BB algorithm it is of interest to decide whether or not a subproblem can be discarded. The ability of doing this with incomplete information is an advantage of a BB algorithm. The efficient set of problem P will only be known upon termination of the BB algorithm; until then, we may have no proof that any solution in $\hat{X}$ is efficient for problem P, but we can take advantage of feasible solutions for dominance purposes, regardless of whether they are Pareto optimal or not.

In general, due to the nature of the mixed-integer problem, the set $\hat{Y}$ may be a collection of isolated points and line segments (which will be referred to as segments). We assume that the set $\hat{Y}$ also contains the points $\boldsymbol{y}^{S-E}$ and $\boldsymbol{y}^{N-W}$ (defined in (2.2)) because they conveniently restrict this set (there cannot be attainable points $\boldsymbol{y}$ such that $y_1 > y_1^{S-E}$ or $y_2 > y_2^{N-W}$).

Given the set $\hat{Y}$, we construct local nadir points and local nadir sets implied by the points in $\hat{Y}$. In particular, we construct local nadir points for each pair of adjacent points in $\hat{Y}$, and we also construct the local nadir set for each segment or a piecewise linear curve in $\hat{Y}$. The set of all local nadir points and local nadir sets implied by the set $\hat{Y}$ is denoted by $\check{\Theta}$. Although this may be an infinite set of points, we treat it with a procedure that uses a finite number of subsets of $\check{\Theta}$. To this purpose, let $|\check{\Theta}|$ denote the cardinality of $\check{\Theta}$ intended as the number of local nadir points plus that of local nadir sets, and suppose that the latter are all line segments rather than piecewise linear sets (it is easy to show that there is a description of $\check{\Theta}$ containing a finite number of line segments). We augment the set $\check{\Theta}$ into $\check{\Theta}^{>} = \check{\Theta} + \mathbb{R}^2_{>}$. The set $\check{\Theta}^{>}$ contains all points in $\mathbb{R}^2$ such that there is no point $\hat{\boldsymbol{y}}$ in $\hat{Y}$ that strictly dominates a point in $\check{\Theta}^{>}$. Note that for all points y in $\check{\Theta}^{>}$ we have $y_1 > y_1^{N-W}$ and $y_2 > y_2^{S-E}$. Figure 4.1 illustrates these concepts: Figure 4.1(a) depicts an arbitrary set $\hat{Y} = \{\boldsymbol{y}^{N-W}, \hat{\boldsymbol{y}}^1, [\hat{\boldsymbol{y}}^2, \hat{\boldsymbol{y}}^3], [\hat{\boldsymbol{y}}^3, \hat{\boldsymbol{y}}^4], \hat{\boldsymbol{y}}^5, [\hat{\boldsymbol{y}}^6, \hat{\boldsymbol{y}}^7], [\hat{\boldsymbol{y}}^7, \hat{\boldsymbol{y}}^8], [\hat{\boldsymbol{y}}^9, \hat{\boldsymbol{y}}^{10}], \boldsymbol{y}^{S-E}\}$. Note that the mixed-integer nature of the problem allows infinitely many attainable points to lie on a segment between two attainable extreme points, for instance on the segment $[\hat{\boldsymbol{y}}^2, \hat{\boldsymbol{y}}^3]$. The sets $\check{\Theta}$ and $\check{\Theta}^{>}$ are depicted in Figure 4.1(b), where $\check{\Theta} = \{\check{\boldsymbol{y}}^1, \check{\boldsymbol{y}}^2, [\check{\boldsymbol{y}}^3 = \hat{\boldsymbol{y}}^2, \check{\boldsymbol{y}}^4 = \hat{\boldsymbol{y}}^3], [\check{\boldsymbol{y}}^4, \check{\boldsymbol{y}}^5 = \hat{\boldsymbol{y}}^4], \check{\boldsymbol{y}}^6, \check{\boldsymbol{y}}^7, [\check{\boldsymbol{y}}^8 = \hat{\boldsymbol{y}}^6, \check{\boldsymbol{y}}^9 = \hat{\boldsymbol{y}}^7], [\check{\boldsymbol{y}}^9, \check{\boldsymbol{y}}^{10} = \hat{\boldsymbol{y}}^8], \check{\boldsymbol{y}}^{11}, [\check{\boldsymbol{y}}^{12} = \hat{\boldsymbol{y}}^9, \check{\boldsymbol{y}}^{13} = \hat{\boldsymbol{y}}^{10}], \check{\boldsymbol{y}}^{14}\}$.

Necessary and sufficient condition for fathoming a BB node. A trivial fathoming rule for eliminating a subproblem P_k is $\tilde{X}_k = \emptyset$, since that implies $X_k = \emptyset$. Consider the Pareto set $\tilde{Y}_{kN}$ of the LP relaxation of subproblem P_k. Clearly, each Pareto point of P_k is weakly dominated by at least one point in $\tilde{Y}_{kN}$. Extending fathoming rule (3.1) of single objective programming to multiobjective programming, we obtain that node k can be discarded if and only if each point in $\tilde{Y}_{kN}$ is dominated by at least one point in $\hat{Y}$, that is, if and only if

$$\tilde{Y}_{kN} \subseteq \hat{Y}^{\leqq}. \tag{4.1}$$

The set $\hat{Y}^{\leqq}$ is depicted in Figure 4.1(a). The above condition is equivalent to

$$\tilde{Y}_{kN} \cap (\mathbb{R}^2 \setminus \hat{Y}^{\leqq}) = \emptyset,$$

which implies that $\tilde{Y}_{kN} \cap \check{\Theta}^{>} = \emptyset$, or equivalently,

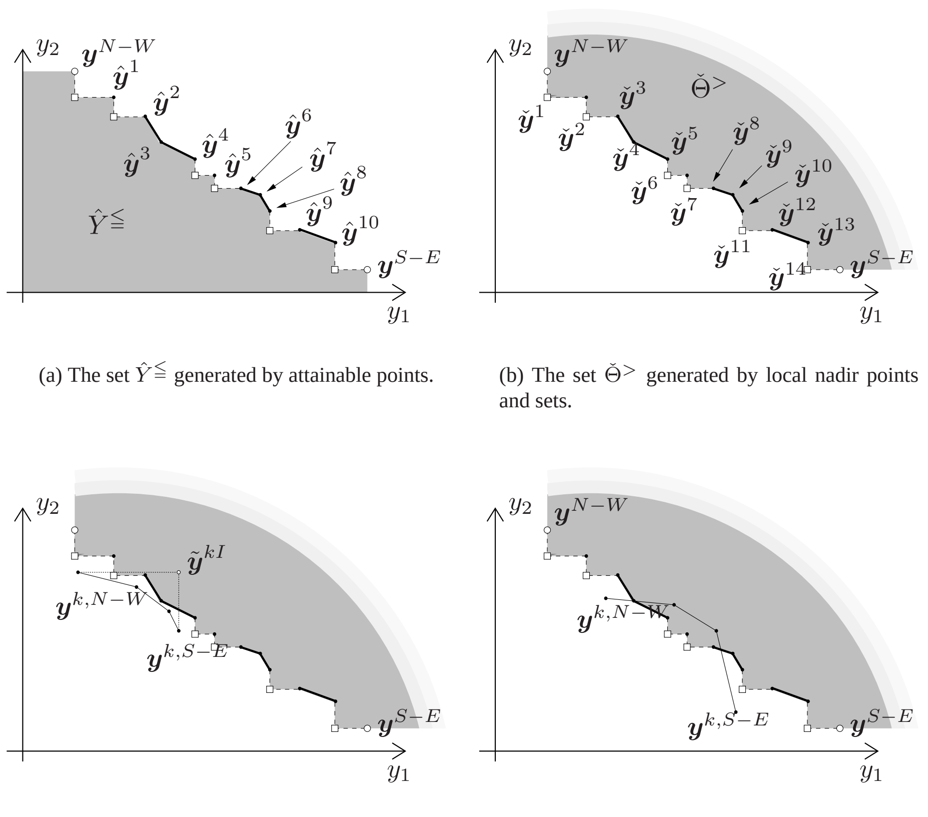
$$\tilde{Y}_{kN} \cap \bigcup_{\check{\boldsymbol{y}} \in \check{\Theta}} \left(\{\check{\boldsymbol{y}}\} + \mathbb{R}^2_{>}\right) = \bigcup_{\check{\boldsymbol{y}} \in \check{\Theta}} \left(\tilde{Y}_{kN} \cap \left(\{\check{\boldsymbol{y}}\} + \mathbb{R}^2_{>}\right)\right) = \emptyset.$$

This fathoming rule can then be decomposed: a union of a collection of sets is empty if and only if each set in the collection is empty. Hence, one needs to check whether

$$\forall \check{\boldsymbol{y}} \in \check{\Theta} \quad \tilde{Y}_{kN} \cap \left(\{\check{\boldsymbol{y}}\} + \mathbb{R}^2_{>}\right) = \emptyset. \tag{4.2}$$

Because the set $\tilde{Y}_{kN}$ is not known explicitly, checking condition (4.2) is nontrivial even for a single local nadir point or a single local nadir set. However, for each local nadir point $\check{\boldsymbol{y}} \in \check{\Theta}$, condition (4.2) is equivalent to

$$\tilde{Y}_{kN}^{\leqq} \cap \left(\{\check{\boldsymbol{y}}\} + \mathbb{R}^2_{>}\right) = \emptyset, \tag{4.3}$$


y_2
$\boldsymbol{y}^{N-W}$
$\hat{\boldsymbol{y}}^1$
$\hat{\boldsymbol{y}}^2$
$\hat{\boldsymbol{y}}^3$
$\hat{\boldsymbol{y}}^4$
$\hat{\boldsymbol{y}}^5$
$\hat{\boldsymbol{y}}^6$
$\hat{\boldsymbol{y}}^7$
$\hat{\boldsymbol{y}}^8$
$\hat{\boldsymbol{y}}^9$
$\hat{\boldsymbol{y}}^{10}$
$\hat{Y}^{\leqq}$
$\boldsymbol{y}^{S-E}$
y_1
$\check{\Theta}^{>}$
$\check{\boldsymbol{y}}^1$
$\check{\boldsymbol{y}}^2$
$\check{\boldsymbol{y}}^3$
$\check{\boldsymbol{y}}^4$
$\check{\boldsymbol{y}}^5$
$\check{\boldsymbol{y}}^6$
$\check{\boldsymbol{y}}^7$
$\check{\boldsymbol{y}}^8$
$\check{\boldsymbol{y}}^9$
$\check{\boldsymbol{y}}^{10}$
$\check{\boldsymbol{y}}^{11}$
$\check{\boldsymbol{y}}^{12}$
$\check{\boldsymbol{y}}^{13}$
$\check{\boldsymbol{y}}^{14}$
$\tilde{\boldsymbol{y}}^{kI}$
$\boldsymbol{y}^{k,N-W}$
$\boldsymbol{y}^{k,S-E}$

(a) The set $\hat{Y}^{\leqq}$ generated by attainable points.

(b) The set $\check{\Theta}^{>}$ generated by local nadir points and sets.

(c) Subproblem P_k can be fathomed.

(d) Subproblem P_k cannot be fathomed.

FIG. 4.1. *Attainable points, local nadir points and local nadir sets, and fathoming of subproblems.*

and for each local nadir set $[\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''] \in \check{\Theta}$, condition (4.2) is equivalent to

$$\tilde{Y}^{\leqq}_{kN} \cap \left([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''] + \mathbb{R}^2_{>}\right) = \emptyset. \tag{4.4}$$

Since the sets in condition (4.3) are convex, one can apply a *separation* argument: the intersection is empty if and only if there exist $a, b, c \in \mathbb{R}$ such that

$$\begin{array}{rcll} ay_1 + by_2 + c & \leq & 0 & \forall (y_1, y_2) \in \tilde{Y}^{\leqq}_{kN} \\ ay_1 + by_2 + c & > & 0 & \forall (y_1, y_2) \in \left(\{\check{\boldsymbol{y}}\} + \mathbb{R}^2_{>}\right). \end{array}$$

The augmented local nadir set in condition (4.4) may be nonconvex because a local nadir set is inherently nonconvex, although the separation argument can be applied to each line segment in this set. In general, in the mixed-integer case the set $\check{\Theta}$ can admit a very peculiar structure that can be exploited to yield a procedure for verifying condition (4.2) that allows for fathoming node k. We postpone this discussion until Section 5.

In Figures 4.1(c) and 4.1(d), a subproblem P_k with a feasible set $\tilde{X}_k$ admits a Pareto set $\tilde{Y}_{kN}$ in the form of a piecewise linear concave curve spanned between the subproblem's extreme points, $\boldsymbol{y}^{k,N-W}$ and $\boldsymbol{y}^{k,S-E}$. In Fig. 4.1(c), the set $\tilde{Y}^{\leqq}_{kN}$ has an empty intersection with the set $\check{\Theta}$ and node k can therefore be fathomed. This empty intersection can be verified by applying the separation argument to the elements of $\check{\Theta}^{>}$ including the sets $\{\check{\boldsymbol{y}}^2\} + \mathbb{R}^2_{>}$, $[\check{\boldsymbol{y}}^3, \check{\boldsymbol{y}}^4] + \mathbb{R}^2_{>}$, and $[\check{\boldsymbol{y}}^4, \check{\boldsymbol{y}}^5] + \mathbb{R}^2_{>}$. Contrary to this case, in Fig. 4.1(d), the set $\tilde{Y}^{\leqq}_{kN}$ has a nonempty intersection with $\check{\Theta}$ and subproblem P_k cannot be fathomed.

Generic and established fathoming rules. As observed above, fathoming rule (4.1) is difficult to check and it is of interest to develop practical fathoming rules that can be easily implemented. A practical fathoming rule is typically a sufficient condition that allows node k to be discarded without enumerating all the Pareto points of the corresponding subproblem P_k. The literature on integer and mixed-binary multiobjective linear programming provides fathoming rules that can be extended to the mixed-integer case. We first develop a generic fathoming rule that encompasses the existing rules. Following the single objective case, at a BB node k we define an *upper-bound* set denoted by *UB*$_k$ that is related to problem P_k. For any set of attainable points $\hat{Y}$ such that $\hat{Y} = \hat{Y}_N$ we define a *lower-bound* set denoted by *LB*. The former is an upper-bound set in the sense that its points weakly dominate all attainable points of subproblem P_k, while the latter is a lower-bound set in the sense that its points assume objective values of some feasible solutions of problem P. We also define augmented upper and lower-bound sets, *UB*$^{\leqq}_k$ and *LB*$^{>}$, and state the generic fathoming rule as

$$UB^{\leqq}_k \cap LB^{>} = \emptyset. \tag{4.5}$$

ESTABLISHED FATHOMING RULE 0. At node k, a well-known upper-bound set denoted *UB*^0_k is the ideal point of problem $\tilde{P}_k$, i.e., $UB^0_k = \{\tilde{\boldsymbol{y}}^{kI}\}$. The computation of this set is simple but the resulting rule is often ineffective. A lower-bound set is $LB^1 = \check{\Theta}$ implied by the currently known set $\hat{Y}$. Figure 4.1(c) illustrates the sets *UB*^0_k and *LB*^{1} for a biobjective problem. The ideal point $\tilde{\boldsymbol{y}}^{kI}$ is not dominated by any integer feasible solution, and fathoming rule (4.5) proves useless in this case.

ESTABLISHED FATHOMING RULE 1. Another type of upper-bound set that is tighter than the ideal point is $UB^1_k = \tilde{Y}_{kN}$. The computation of this set requires finding the extreme Pareto points of problem $\tilde{P}_k$. Using $LB^1 = \check{\Theta}$, rule (4.5) becomes

$$\tilde{Y}^{\leqq}_{kN} \cap \check{\Theta}^{>} = \emptyset, \tag{4.6}$$

which can be viewed as another version of condition (4.1).

Figure 4.1(c) also illustrates the sets $\tilde{Y}_{kN}$ and *LB*^{1} for a biobjective problem $\tilde{P}_k$. All Pareto points in $\tilde{Y}_{kN}$ are dominated by the points in $\check{\Theta}$, and based on fathoming rule (4.5) node k can be fathomed. Another problem P_k is depicted in Figure 4.1(d). In this case, not all Pareto points in $\tilde{Y}_{kN}$ are dominated by the points in $\check{\Theta}$, the rule does not apply, and node k cannot be fathomed. However, a direct implementation of condition (4.6) would require a significant computation overhead and is avoided. The literature provides other practical rules, namely the second and third type of bound sets presented below, that are an applied refinement of rule (4.6).

Bound sets *UB*^2_k and *LB*^{2} are proposed by Visée et al. [29]. The set *UB*^2_k is based on the objective value $\beta(\lambda)$ of an optimal solution of the weighted-sum problem associated with problem $\tilde{P}_k$:

$$\beta(\lambda) = \max_{\boldsymbol{x}} \{(\lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2)^\top \boldsymbol{x} : \boldsymbol{x} \in \tilde{X}_k\}, \tag{4.7}$$

where

$$\lambda = \frac{\lambda_1}{\lambda_1 + \lambda_2}, \;\; \lambda_1 = y_2^{k,N-W} - y_2^{k,S-E}, \;\; \lambda_2 = y_1^{k,S-E} - y_1^{k,N-W}. \tag{4.8}$$

The intuition for this choice of λ is that the weight vector of the objective function in (4.7) is perpendicular to the line passing through the two extreme points. The upper-bound set is

$$UB_k^2 = \{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} = (\boldsymbol{c}_1^\top \boldsymbol{x}, \boldsymbol{c}_2^\top \boldsymbol{x}) : (\lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2)^\top \boldsymbol{x} = \beta(\lambda), \boldsymbol{x} \in \tilde{X}_k\}.$$

Note that UB_k^2 is not a proper upper bound in that it does not dominate, in general, all Pareto points in $\tilde{Y}_k$. Consider now, for any $\check{\boldsymbol{y}}$ in $\check{\Theta}$, the weighted sum $\lambda\check{y}_1 + (1-\lambda)\check{y}_2$. If for all points in $\check{\Theta}$, this weighted sum is greater than that for any $\boldsymbol{y} \in UB_k^2$, node k can be fathomed. Instead of checking this condition for every point in $\check{\Theta}$, a valid lower bound is obtained by the points attaining the minimum of the weighted sum over all points in $\check{\boldsymbol{y}} \in \Theta$:

$$LB^2 = \{\boldsymbol{y} \in \mathbb{R}^2 : \lambda y_1 + (1-\lambda)y_2 = \min_{\check{\boldsymbol{y}} \in \check{\Theta}} (\lambda\check{y}_1 + (1-\lambda)\check{y}_2)\}.$$

ESTABLISHED FATHOMING RULE 2. *If*

$$\min_{\check{\boldsymbol{y}} \in \check{\Theta}} (\lambda\check{y}_1 + (1-\lambda)\check{y}_2) \geq \lambda y_1 + (1-\lambda)y_2 \tag{4.9}$$

for all $\boldsymbol{y} \in UB_k^2$, then rule (4.5) holds and node k can be fathomed.

A third type of bound sets, UB_k^3 and LB^3, is discussed by Ehrgott and Gandibleux [11] who evaluate the quality of bound sets, and by Sourd and Spanjaard [25] who construct a hyperplane separating the augmented sets $UB_k^{\leqq}$ and $LB^{>}$ to fathom a node. The bound set UB_k^3 is an extension of UB_k^2 that considers multiple values of λ. Given that these bound sets yield sufficient conditions for fathoming a node, it makes sense to try different values of λ if the one for LB^2 and UB_k^2 did not prove it. Consider a set of r weights $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_r\}$. Then define the following sets for $t \in \{1, 2, \ldots, r\}$:

$$\begin{array}{rcl} UB_k^3(\lambda_t) & = & \{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} = (\boldsymbol{c}_1^\top \boldsymbol{x}, \boldsymbol{c}_2^\top \boldsymbol{x}) : (\lambda_t \boldsymbol{c}_1 + (1-\lambda_t)\boldsymbol{c}_2)^\top \boldsymbol{x} = \beta(\lambda_t), \boldsymbol{x} \in \tilde{X}_k\} \\ LB^3(\lambda_t) & = & \{\boldsymbol{y} \in \mathbb{R}^2 : \lambda_t y_1 + (1-\lambda_t)y_2 = \min_{\check{\boldsymbol{y}} \in \check{\Theta}} (\lambda_t\check{y}_1 + (1-\lambda_t)\check{y}_2)\}. \end{array}$$

ESTABLISHED FATHOMING RULE 3. *If for all $\boldsymbol{y}$ in UB_k^3 there exists λ_t in Λ such that*

$$\min_{\check{\boldsymbol{y}} \in \check{\Theta}} (\lambda_t\check{y}_1 + (1-\lambda_t)\check{y}_2) \geq \lambda_t y_1 + (1-\lambda_t)y_2, \tag{4.10}$$

then rule (4.5) holds and node k can be fathomed.

We improve on the available fathoming rules in the next section and provide an efficient implementation of the proposed rules in the subsequent sections.

5. New Fathoming Rules.

The review of the established fathoming rules presented in the previous section implies the following sufficient condition for fathoming node k: for each element in $\check{\Theta}$ there exists a line in $\mathbb{R}^2$ separating this point from the set $\tilde{Y}_{kN}$. If this condition holds, we say that the element in $\check{\Theta}$ is *separable* from the set $\tilde{Y}_{kN}$. In this section we discuss a procedure for checking this condition. Before explaining the idea, we provide a few considerations on how to make efficient use of the set $\check{\Theta}$ with a large cardinality $|\check{\Theta}|$.

First, the bound set *LB* may include a large number of local nadir points and local nadir sets. In order to control the complexity of a fathoming procedure, one can use a set $\check{\Theta}$ induced by a subset of $\hat{Y}$. Second, at the BB node k there is no need to consider the whole set $\check{\Theta}$. Due to (4.2), one should include all local nadir points and local nadir sets that are weakly dominated by the ideal $\tilde{\boldsymbol{y}}^{kI}$. This limits the set of local nadir points and local nadir sets at node k to $\check{\Theta}_k = \check{\Theta} \cap (\{\tilde{\boldsymbol{y}}^{kI}\} + \mathbb{R}^2_{\leqq})$.

The first rule we present is obvious.

FATHOMING RULE 1. *Node k can be fathomed if $\check{\Theta}_k = \emptyset$.*

We now extend the fathoming rule proposed by Ehrgott and Gandibleux [11]. In order to satisfy (4.2), we formulate an auxiliary optimization problem. Consider a local nadir point $\check{\boldsymbol{y}}$ and local nadir set $[\check{\boldsymbol{y}}', \check{\boldsymbol{y}}'']$ in $\check{\Theta}_k$. If there exists $\lambda \in [0,1]$ such that $\lambda \check{y}_1 + (1-\lambda)\check{y}_2 \geq (\lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2)^\top \boldsymbol{x}$ for all $\boldsymbol{x} \in \tilde{X}_k$, all Pareto points $\boldsymbol{z}(\boldsymbol{x}) = \boldsymbol{y} \in \tilde{Y}_{kN}$ of problem $\tilde{P}_k$, and hence those of P_k, are weakly dominated by the local nadir point $\check{\boldsymbol{y}}$. Similarly, if there exists $\lambda \in [0,1]$ such that $\lambda \check{y}'_1 + (1-\lambda)\check{y}'_2 \geq (\lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2)^\top \boldsymbol{x}$ and $\lambda \check{y}''_1 + (1-\lambda)\check{y}''_2 \geq (\lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2)^\top \boldsymbol{x}$ for all $\boldsymbol{x} \in \tilde{X}_k$, all Pareto points $\boldsymbol{z}(\boldsymbol{x}) = \boldsymbol{y} \in \tilde{Y}_{kN}$ of problem $\tilde{P}_k$ are weakly dominated by the local nadir set $[\check{\boldsymbol{y}}', \check{\boldsymbol{y}}'']$. If this holds for all elements of $\check{\Theta}_k$, then the node k can be fathomed.

FATHOMING RULE 2. *Node k can be fathomed if each local nadir point $\check{\boldsymbol{y}}$ and each local nadir set $[\check{\boldsymbol{y}}', \check{\boldsymbol{y}}'']$ in $\check{\Theta}_k$ is separable from $\tilde{Y}_{kN}$, that is,*

$$\begin{array}{lll} \forall \check{\boldsymbol{y}} \in \check{\Theta}_k \quad \exists \lambda \in [0,1]: & & \lambda \check{y}_1 + (1-\lambda)\check{y}_2 \geq \max_{\boldsymbol{x} \in \tilde{X}_k} \{(\lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2)^\top \boldsymbol{x}\} \\ \textit{and} & & \\ \forall [\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''] \in \check{\Theta}_k \quad \exists \lambda \in [0,1]: & & \lambda \check{y}'_1 + (1-\lambda)\check{y}'_2 \geq \max_{\boldsymbol{x} \in \tilde{X}_k} \{(\lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2)^\top \boldsymbol{x}\} \\ & & \lambda \check{y}''_1 + (1-\lambda)\check{y}''_2 \geq \max_{\boldsymbol{x} \in \tilde{X}_k} \{(\lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2)^\top \boldsymbol{x}\}. \end{array} \tag{5.1}$$

Note that this fathoming rule requires that one λ exist for each local nadir point and for each local nadir set in $\check{\Theta}_k$. This rule will be enforced by solving a nonlinear optimization problem by means of a parametric linear program. As a result, we reduce condition (5.1) to checking the feasibility of two linear programs given in Theorem 5.1.

THEOREM 5.1. *Condition* (5.1) *holds if and only if for every local nadir point $\check{\boldsymbol{y}} \in \check{\Theta}_k$ and for every local nadir set $[\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''] \in \check{\Theta}_k$ there exist $\lambda \in [0,1]$ and $\boldsymbol{w} \in \mathbb{R}^m$ such that the following two linear programs are feasible:*

$$\begin{array}{lll} \hat{P}_{FR}(\check{\boldsymbol{y}}): & \min_{\lambda,\boldsymbol{w}} & 0 \\ & \textit{s.t.} & \lambda \check{y}_1 + (1-\lambda)\check{y}_2 \geq \boldsymbol{b}_k^\top \boldsymbol{w} \\ & & A_k^\top \boldsymbol{w} = \lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2 \\ & & \lambda \in [0,1] \\ & & \boldsymbol{w} \geqq \boldsymbol{0} \end{array} \qquad \begin{array}{lll} \hat{P}_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}'']): & \min_{\lambda,\boldsymbol{w}} & 0 \\ & \textit{s.t.} & \lambda \check{y}'_1 + (1-\lambda)\check{y}'_2 \geq \boldsymbol{b}_k^\top \boldsymbol{w} \\ & & \lambda \check{y}''_1 + (1-\lambda)\check{y}''_2 \geq \boldsymbol{b}_k^\top \boldsymbol{w} \\ & & A_k^\top \boldsymbol{w} = \lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2 \\ & & \lambda \in [0,1] \\ & & \boldsymbol{w} \geqq \boldsymbol{0}. \end{array}$$

Proof. Condition (5.1) can be expressed as an optimization problem P_{FR} whose objective function is zero since only the existence of a feasible λ is of interest. Because $\check{\Theta}_k$ comprises a finite set of local nadir points and a finite set of local nadir sets defined as line segments, we can rewrite it as $\check{\Theta}_k = \{\check{\boldsymbol{y}}^1, \check{\boldsymbol{y}}^2, \ldots, \check{\boldsymbol{y}}^h, [\check{\boldsymbol{v}}^1, \check{\boldsymbol{u}}^1], [\check{\boldsymbol{v}}^2, \check{\boldsymbol{u}}^2], \ldots, [\check{\boldsymbol{v}}^q, \check{\boldsymbol{u}}^q]\}$. Let $\boldsymbol{\lambda} \in [0,1]^{h+q}$ be a vector of variables λ for all local nadir points and local nadir sets in $\check{\Theta}_k$. Then for node k to be fathomed the following problem must be feasible:

$$\begin{array}{llcll} P_{FR}: & \min_{\boldsymbol{\lambda}} \ 0 & & & \\ & \text{s.t.} \ \lambda_j \check{y}^j_1 + (1-\lambda_j)\check{y}^j_2 & \geq & \max_{\boldsymbol{x} \in \tilde{X}_k} \{(\lambda_j \boldsymbol{c}_1 + (1-\lambda_j)\boldsymbol{c}_2)^\top \boldsymbol{x}\} & \forall j = 1,2,\ldots,h \\ & \lambda_{h+i} \check{v}^i_1 + (1-\lambda_{h+i})\check{v}^i_2 & \geq & \max_{\boldsymbol{x} \in \tilde{X}_k} \{(\lambda_{h+i} \boldsymbol{c}_1 + (1-\lambda_{h+i})\boldsymbol{c}_2)^\top \boldsymbol{x}\} & \forall i = 1,2,\ldots,q \\ & \lambda_{h+i} \check{u}^i_1 + (1-\lambda_{h+i})\check{u}^i_2 & \geq & \max_{\boldsymbol{x} \in \tilde{X}_k} \{(\lambda_{h+i} \boldsymbol{c}_1 + (1-\lambda_{h+i})\boldsymbol{c}_2)^\top \boldsymbol{x}\} & \\ & \boldsymbol{\lambda} \in [0,1]^{h+q}. & & & \end{array}$$

Note that a scalar λ_{h+i} is associated with the end points of the local nadir set $[\check{\boldsymbol{v}}^i, \check{\boldsymbol{u}}^i]$ in $\check{\Theta}_k$. Due to convexity, separating the end points is sufficient to separate the entire nadir set. Using

(4.7), the family of constraints of P_{FR} can be rewritten as

$$\begin{array}{rcll} \lambda_j \check{y}_1^j + (1-\lambda_j)\check{y}_2^j & \geq & \beta(\lambda_j) & \forall j = 1,2,\ldots,h \\ \lambda_{h+i}\check{v}_1^i + (1-\lambda_{h+i})\check{v}_2^i & \geq & \beta(\lambda_{h+i}) & \forall i = 1,2,\ldots,q \\ \lambda_{h+i}\check{u}_1^i + (1-\lambda_{h+i})\check{u}_2^i & \geq & \beta(\lambda_{h+i}), & \end{array} \tag{5.2}$$

where $\beta(\lambda) = \max_{\boldsymbol{x}\in\tilde{X}_k}\{(\lambda \boldsymbol{c}_1 + (1-\lambda\boldsymbol{c}_2)^\top \boldsymbol{x}\}$. Problem P_{FR} has $h+q$ variables, and can be decomposed into $h+q$ subproblems. As a consequence, P_{FR} is feasible if and only if all of the following subproblems $P_{FR}(\check{\boldsymbol{y}})$, for all local nadir points $\check{\boldsymbol{y}} \in \check{\Theta}_k$, and $P_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$, for all local nadir sets $[\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''] \in \check{\Theta}_k$, are feasible:

$$\begin{array}{lll} P_{FR}(\check{\boldsymbol{y}}): & \min_\lambda & 0 \\ & \text{s.t.} & \lambda\check{y}_1 + (1-\lambda)\check{y}_2 \geq \beta(\lambda) \\ & & \lambda \in [0,1]. \end{array}$$

$$\begin{array}{lll} P_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}'']): & \min_\lambda & 0 \\ & \text{s.t.} & \lambda\check{y}_1' + (1-\lambda)\check{y}_2' \geq \beta(\lambda) \\ & & \lambda\check{y}_1'' + (1-\lambda)\check{y}_2'' \geq \beta(\lambda) \\ & & \lambda \in [0,1]. \end{array}$$

Let $\mathcal{F}$ denote the set of all λ feasible for P_{FR} (i.e., satisfying (5.2)) and $\mathcal{F}_{\check{\boldsymbol{y}}}$ and $\mathcal{F}_{[\check{\boldsymbol{y}}', \check{\boldsymbol{y}}'']}$ denote the sets of all λ feasible for subproblems $P_{FR}(\check{\boldsymbol{y}})$ or $P_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$, respectively. Then obviously $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \times \cdots \times \mathcal{F}_{h+q}$. This suggests that verifying the feasibility of problem P_{FR} is equivalent to verifying the feasibility of each subproblem $P_{FR}(\check{\boldsymbol{y}})$ and $P_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$, i.e., if at least one of subproblems $P_{FR}(\check{\boldsymbol{y}})$ or $P_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ is infeasible, then P_{FR} is infeasible and the subproblem P_k cannot be fathomed.

The right-hand side of the constraints in the subproblems is the objective function value of an optimal solution of (4.7), that is, the weighted-sum problem of $\tilde{P}_k$ defined as $\max_{\boldsymbol{x}}\{(\lambda\boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2)^\top\boldsymbol{x} : A_k\boldsymbol{x} \leq \boldsymbol{b}_k\}$. Its dual is

$$\begin{array}{ll} \min_{\boldsymbol{w}} & \boldsymbol{b}_k^\top \boldsymbol{w} \\ \text{s.t.} & A_k^\top \boldsymbol{w} = \lambda\boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2 \\ & \boldsymbol{w} \geqq \boldsymbol{0}. \end{array} \tag{5.3}$$

Problem $P_{FR}(\check{\boldsymbol{y}})$ (or problem $P_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$) is feasible if and only if the right-hand-side $\beta(\lambda)$ is finite (in other words, if the optimal value of problem (4.7) exists and is finite) and the inequality constraint holds. If $\beta(\lambda) = -\infty$, then $\tilde{X}_k = \emptyset$ and node k can be fathomed. If $\beta(\lambda) = +\infty$, then $\max\{\boldsymbol{c}_1^\top\boldsymbol{x} : \boldsymbol{x} \in \tilde{X}\}$ or $\max\{\boldsymbol{c}_2^\top\boldsymbol{x} : \boldsymbol{x} \in \tilde{X}\}$ is unbounded, which is excluded by assumption. Using linear programming strong duality, the dual problem is feasible and assumes the finite optimal objective value $\boldsymbol{b}_k^\top\boldsymbol{w}'(\lambda)$, where $\beta(\lambda) = \boldsymbol{b}_k^\top\boldsymbol{w}'(\lambda)$ and $\boldsymbol{w}'(\lambda)$ is an optimal solution of the dual problem, clearly dependent on λ. From this point on, we ignore the dependence of $\boldsymbol{w}$ on λ because they belong to the same optimization problem and are inherently dependent on each other. We can then substitute the nonlinear right-hand sides of $P_{FR}(\check{\boldsymbol{y}})$ and $P_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ with $\boldsymbol{b}_k^\top\boldsymbol{w}'$. This substitution however requires the knowledge of $\boldsymbol{w}'$ which we would like to avoid. From weak duality, $\boldsymbol{b}_k^\top\boldsymbol{w} \geq \beta(\lambda)$, and we replace the nonlinear right-hand sides of $P_{FR}(\check{\boldsymbol{y}})$ and $P_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ with $\boldsymbol{b}_k^\top\boldsymbol{w}$ as long as the dual feasibility constraints are satisfied. Problems $P_{FR}(\check{\boldsymbol{y}})$ and $P_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ then become

$$\begin{array}{lll} \hat{P}_{FR}(\check{\boldsymbol{y}}): & \min_{\lambda,\boldsymbol{w}} & 0 \\ & \text{s.t.} & \lambda\check{y}_1 + (1-\lambda)\check{y}_2 \geq \boldsymbol{b}_k^\top \boldsymbol{w} \\ & & A_k^\top \boldsymbol{w} = \lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2 \\ & & \lambda \in [0,1] \\ & & \boldsymbol{w} \geqq \boldsymbol{0} \end{array} \qquad \begin{array}{lll} \hat{P}_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}'']): & \min_{\lambda,\boldsymbol{w}} & 0 \\ & \text{s.t.} & \lambda\check{y}'_1 + (1-\lambda)\check{y}'_2 \geq \boldsymbol{b}_k^\top \boldsymbol{w} \\ & & \lambda\check{y}''_1 + (1-\lambda)\check{y}''_2 \geq \boldsymbol{b}_k^\top \boldsymbol{w} \\ & & A_k^\top \boldsymbol{w} = \lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2 \\ & & \lambda \in [0,1] \\ & & \boldsymbol{w} \geqq \boldsymbol{0}. \end{array}$$

Naturally, we cannot guarantee that for every $\boldsymbol{w}$ feasible for problem (5.3) the constraints $\boldsymbol{b}_k^\top \boldsymbol{w} \leq \lambda\check{y}_1 + (1-\lambda)\check{y}_2$, $\boldsymbol{b}_k^\top \boldsymbol{w} \leq \lambda\check{y}'_1 + (1-\lambda)\check{y}'_2$, and $\boldsymbol{b}_k^\top \boldsymbol{w} \leq \lambda\check{y}''_1 + (1-\lambda)\check{y}''_2$ are fulfilled, or, in other words, that problems $\hat{P}_{FR}(\check{\boldsymbol{y}})$ and $\hat{P}_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ are feasible. However, there exists $\boldsymbol{w}' \geqq \boldsymbol{0}$ such that $A_k^\top \boldsymbol{w}' = \lambda\boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2$ and $\boldsymbol{b}_k^\top \boldsymbol{w}' = \beta(\lambda) \leq \lambda\check{y}_1 + (1-\lambda)\check{y}_2$ so that problems $\hat{P}_{FR}(\check{\boldsymbol{y}})$ and $\hat{P}_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ are feasible. As a result, problems $P_{FR}(\check{\boldsymbol{y}})$ and $P_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ are feasible if and only if problems $\hat{P}_{FR}(\check{\boldsymbol{y}})$ and $\hat{P}_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ are feasible respectively. □

We now reformulate the linear programs in Theorem 5.1 by moving one of the constraints to the objective function and minimizing the violation of that constraint.

COROLLARY 5.2. *Condition* (5.1) *holds if and only if for every local nadir point* $\check{\boldsymbol{y}} \in \check{\Theta}_k$ *and for every local nadir set* $[\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''] \in \check{\Theta}_k$*, the following two linear programs have a nonpositive optimal objective value:*

$$\begin{array}{lll} P^\star_{FR}(\check{\boldsymbol{y}}): & \min_{\lambda,\boldsymbol{w}} & \boldsymbol{b}_k^\top \boldsymbol{w} - (\lambda\check{y}_1 + (1-\lambda)\check{y}_2) \\ & \textit{s.t.} & A_k^\top \boldsymbol{w} = \lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2 \\ & & \lambda \in [0,1] \\ & & \boldsymbol{w} \geqq \boldsymbol{0} \end{array} \qquad \begin{array}{lll} P^\star_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}'']): & \min_{\lambda,\boldsymbol{w}} & \boldsymbol{b}_k^\top \boldsymbol{w} - (\lambda\check{y}'_1 + (1-\lambda)\check{y}'_2) \\ & \textit{s.t.} & \lambda\check{y}''_1 + (1-\lambda)\check{y}''_2 \geq \boldsymbol{b}_k^\top \boldsymbol{w} \\ & & A_k^\top \boldsymbol{w} = \lambda \boldsymbol{c}_1 + (1-\lambda)\boldsymbol{c}_2 \\ & & \lambda \in [0,1] \\ & & \boldsymbol{w} \geqq \boldsymbol{0}. \end{array}$$

Proof. Based on Theorem 5.1, condition (5.1) holds if and only if problems $\hat{P}_{FR}(\check{\boldsymbol{y}})$ and $\hat{P}_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ are feasible for each element in $\check{\Theta}_k$.

Obviously, for every local nadir point $\check{\boldsymbol{y}}$ in $\check{\Theta}_k$, $(\lambda^\star, \boldsymbol{w}^\star)$ is feasible for problem $\hat{P}_{FR}(\check{\boldsymbol{y}})$ if and only if $\boldsymbol{b}_k^\top \boldsymbol{w}^\star - (\lambda^\star\check{y}_1 + (1-\lambda^\star)\check{y}_2) \leq 0$, which holds if and only if $(\lambda^\star, \boldsymbol{w}^\star)$ is feasible for problem $\hat{P}^\star_{FR}(\check{\boldsymbol{y}})$ and this problem has a nonpositive optimal objective value.

For every local nadir set $[\check{\boldsymbol{y}}', \check{\boldsymbol{y}}'']$ in $\check{\Theta}_k$, $(\lambda^\star, \boldsymbol{w}^\star)$ is feasible for problem $\hat{P}_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ if and only if $\boldsymbol{b}_k^\top \boldsymbol{w}^\star \leq \lambda^\star y'_1 + (1-\lambda^\star)y'_2$ and $\boldsymbol{b}_k^\top \boldsymbol{w}^\star \leq \lambda^\star y''_1 + (1-\lambda^\star)y''_2$. W.l.o.g., any of these two constraints is moved to the objective function to construct problem $\hat{P}^\star_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$. Then $(\lambda^\star, \boldsymbol{w}^\star)$ is feasible for problem $P^\star_{FR}([\check{\boldsymbol{y}}', \check{\boldsymbol{y}}''])$ and this problem has a nonpositive optimal objective value. □

Fathoming Rule 2 can be applied efficiently according to a ranking of the elements in $\check{\Theta}_k$. Given that this rule fails if at least one of the linear programs given in Corollary 5.2 is infeasible, it is better to first solve the linear programs for those elements in $\check{\Theta}_k$ that are less likely to be separated from the set $\tilde{Y}_{kN}$. The lower the value $\lambda\check{y}_1 + (1-\lambda)\check{y}_2$, with λ calculated as in (4.8), the less likely is the local nadir point $\check{\boldsymbol{y}}$ to be separable.

PREPROCESSING FOR FATHOMING RULE 2. For each local nadir point $\check{\boldsymbol{y}}$ in $\check{\Theta}_k$, calculate $\lambda\check{y}_1 + (1-\lambda)\check{y}_2$, with λ obtained as in (4.8). Rank the points giving rank one to the points with the lowest value. Apply Fathoming Rule 2 to the local nadir points in the order of the ranking.

5.1. Further improvements on Fathoming Rule 2. Depending on the size of the original problem, the number $|\check{\Theta}_k|$ of problems $P^\star_{FR}$ to be solved can be very large and hence the techniques discussed above can still be inefficient. We now propose additional simple procedures to improve the fathoming rules and reduce $|\check{\Theta}_k|$.

FATHOMING RULE 2A. *Given an optimal solution $(\lambda^\star, \boldsymbol{w}^\star)$ of problem $P^\star_{FR}(\check{\boldsymbol{y}})$, eliminate from the set $\check{\Theta}_k$ all nadir points $\check{\boldsymbol{y}}^t \neq \check{\boldsymbol{y}}$ for which the associated problem $P^\star_{FR}(\check{\boldsymbol{y}}^t)$ has not yet been solved and such that $\boldsymbol{b}_k^\top \boldsymbol{w}^\star \leq \lambda^\star \check{y}_1^t + (1-\lambda^\star)\check{y}_2^t$.*

For all such local nadir points $\check{\boldsymbol{y}}^t$, $(\lambda^\star, \boldsymbol{w}^\star)$ is also a feasible solution for problem $P^\star_{FR}(\check{\boldsymbol{y}}^t)$, which therefore does not have to be solved since $\lambda^\star$ satisfies condition (5.1) for the local nadir point $\check{\boldsymbol{y}}^t$. This elimination is also valid for all local nadir sets $[\check{\boldsymbol{y}}', \check{\boldsymbol{y}}'']$ such that $\boldsymbol{b}_k^\top \boldsymbol{w}^\star \leq \lambda^\star \check{y}_1' + (1-\lambda^\star)\check{y}_2'$ and $\boldsymbol{b}_k^\top \boldsymbol{w}^\star \leq \lambda^\star \check{y}_1'' + (1-\lambda^\star)\check{y}_2''$. In other words, all such local nadir points and local nadir sets in $\check{\Theta}_k$ are separable from the set $\tilde{Y}_{kN}$.

The second improvement concerns the local nadir points $\check{\boldsymbol{y}}^s$ that cannot be eliminated from the set $\check{\Theta}_k$ using Fathoming Rule 2a. To develop Fathoming Rule 2b, we modify $\lambda^\star$ while keeping $\boldsymbol{w}^\star$ fixed. For these points, $\lambda^\star \check{y}_1^s + (1-\lambda^\star)\check{y}_2^s < \boldsymbol{b}_k^\top \boldsymbol{w}^\star$ and hence

$$\lambda^\star \check{y}_1^s + (1-\lambda^\star)\check{y}_2^s < \boldsymbol{b}_k^\top \boldsymbol{w}^\star \leq \lambda^\star \check{y}_1 + (1-\lambda^\star)\check{y}_2.$$

Two cases may occur: $\check{y}_1^s > \check{y}_1$ or $\check{y}_1^s < \check{y}_1$. The two components cannot be equal as otherwise there would be a dominated nadir point. Assume w.l.o.g. that $\check{y}_1^s > \check{y}_1$. Then $\check{y}_2^s < \check{y}_2$, since if $\check{y}_2^s \geq \check{y}_2$ then $\check{\boldsymbol{y}}^s$ would dominate $\check{\boldsymbol{y}}$. Additionally, if $y_1^s = y_2^s$ this nadir point cannot be separated with any new $\hat{\lambda}$. Given $\check{y}_1^s > \check{y}_1$ and $\check{y}_2^s < \check{y}_2$, all λ such that problem $P^\star_{FR}(\check{\boldsymbol{y}})$ is feasible must satisfy $\lambda \check{y}_1^s + (1-\lambda)\check{y}_2^s \geq \boldsymbol{b}_k^\top \boldsymbol{w}^\star$, i.e.,

$$\begin{array}{lllll} \lambda & \geq & \max & \left\{0, \frac{\boldsymbol{b}_k^\top \boldsymbol{w}^\star - \check{y}_2^s}{\check{y}_1^s - \check{y}_2^s}\right\} & \text{if } \check{y}_1^s > \check{y}_2^s; \\ \lambda & \leq & \min & \left\{1, \frac{\boldsymbol{b}_k^\top \boldsymbol{w}^\star - \check{y}_2^s}{\check{y}_1^s - \check{y}_2^s}\right\} & \text{if } \check{y}_1^s < \check{y}_2^s. \end{array} \tag{5.4}$$

Assuming $\check{y}_1^s \neq \check{y}_2^s$, define

$$\hat{\lambda} = \frac{\boldsymbol{b}_k^\top \boldsymbol{w}^\star - \check{y}_2^s}{\check{y}_1^s - \check{y}_2^s}. \tag{5.5}$$

Note that if $A_k^\top \boldsymbol{w}^* = \hat{\lambda}\boldsymbol{c}_1 + (1-\hat{\lambda})\boldsymbol{c}_2$ then $(\hat{\lambda}, \boldsymbol{w}^\star)$ is a feasible solution of problem $P^*_{FR}(\check{\boldsymbol{y}}^s)$.

FATHOMING RULE 2B. *Given an optimal solution $(\lambda^\star, \boldsymbol{w}^\star)$ of problem $P^\star_{FR}(\check{\boldsymbol{y}})$, eliminate from the set $\check{\Theta}_k$ all nadir points $\check{\boldsymbol{y}}^s \neq \check{\boldsymbol{y}}$ for which the associated problem $P^\star_{FR}(\check{\boldsymbol{y}}^s)$ has not yet been solved and such that $\boldsymbol{b}_k^\top \boldsymbol{w}^\star > \lambda^\star \check{y}_1^s + (1-\lambda^\star)\check{y}_2^s$ and $A_k^\top \boldsymbol{w}^\star = \hat{\lambda}\boldsymbol{c}_1 + (1-\hat{\lambda})\boldsymbol{c}_2$, where $\hat{\lambda}$ is given in (5.5).*

Based on Fathoming Rule 2b, no other value of λ implied by (5.4) can help resolve the separability of local nadir points from the set $\tilde{Y}_{kN}$. For elements still remaining in the set $\check{\Theta}_k$, the linear programs given in Corollary 5.2 need to be explicitly solved. The procedure is repeated until all local nadir points and all local nadir sets in $\check{\Theta}_k$ have been considered, both explicitly by solving the linear programs (i.e., applying Fathoming Rule 2) or implicitly by checking the feasibility of the linear programs with already computed solutions (i.e., applying Fathoming Rules 2a and 2b).

6. BB algorithm for BOMILPs. In this section we make use of the theory developed in the previous sections and describe a BB algorithm for finding the Pareto and efficient sets of BOMILPs. The algorithm employs four procedures that are summarized in Algorithms 1-4. Before we present the BB algorithm, we first discuss the procedures.

The first procedure COMPUTEEXTREMES computes the bounds on the objective space of problem P which are implied by the ideal and nadir points of problem $\tilde{P}$. The objective vector of any feasible solution of problem P lies in the interior of the box determined by the points $\boldsymbol{y}^{N-W}$ and $\boldsymbol{y}^{S-E}$. The pseudo-code of this procedure is given in Algorithm 1.

Algorithm 1 Computation of the bounds in the objective space of problem P.

```
1: procedure COMPUTEEXTREMES ($P$)
2:     if $\tilde{X} = \emptyset$ then
3:         $\boldsymbol{y}^{S-E} = \boldsymbol{y}^{N-W} = (-\infty, -\infty)$
4:         return $(\boldsymbol{y}^{S-E}, \boldsymbol{y}^{N-W})$
5:     Compute $y_1^{N-W} = \min\{\boldsymbol{z}_1(\boldsymbol{x}) : \boldsymbol{x} \in \tilde{X}\}$
6:     Compute $y_2^{N-W} = \max\{\boldsymbol{z}_2(\boldsymbol{x}) : \boldsymbol{x} \in \tilde{X}\}$
7:     Compute $y_1^{S-E} = \max\{\boldsymbol{z}_1(\boldsymbol{x}) : \boldsymbol{x} \in \tilde{X}\}$
8:     Compute $y_2^{S-E} = \min\{\boldsymbol{z}_2(\boldsymbol{x}) : \boldsymbol{x} \in \tilde{X}\}$
9:     return $(\boldsymbol{y}^{N-W}, \boldsymbol{y}^{S-E})$
```

The procedure COMPUTEFEASSOL-ϵ with the pseudo-code given in Algorithm 2 finds attainable points of problem P. The procedure employs the ϵ-constraint method [13] according to which the first objective function of problem P is maximized subject to the feasible set of problem P and an additional constraint that is yielded by the other objective function. The method requires that the interval $[y_2^{N-W}, y_2^{S-E}]$ be discretized into $\mathcal{N}$ subintervals, where $\mathcal{N}$ is an input to the procedure. The right-hand side ϵ of the new constraint is determined by the $\mathcal{N}$ discrete values. Although the ϵ-constraint problem is solved for each ϵ, the same optimal solution may be obtained for different values of ϵ. An optimal solution to the ϵ-constraint problem is a weakly efficient solution for problem P. In particular, if the optimal solutions $\boldsymbol{x}^{j-1}$ and $\boldsymbol{x}^j$ obtained for two subsequent values of ϵ are equal for all components $i \in \{p+1, \dots, n\}$, then all points in conv$\{\boldsymbol{x}^{j-1}, \boldsymbol{x}^j\}$ are feasible solutions for problem P, although not necessarily weakly efficient for P. The optimal solutions to the ϵ-constraint problem are stored in the set $\hat{X}$ and their images in the set $\hat{Y}$. In order to satisfy the condition $\hat{Y} = \hat{Y}_N$, we filter out the points that are not Pareto in $\hat{Y}$ by applying the $VMAX$ operator to $\hat{Y}$. We update the set $\hat{X}$ accordingly. The set $\check{\Theta}$ of local nadir points that are implied by the Pareto points are also computed.

The procedure REDUCENADIRSET, whose pseudo-code is given in Algorithm 3, applies the fathoming rules at a node k of the BB tree. The set $\check{\Theta}_k$ of local nadir points and sets associated with problem P_k is reduced to those that are not separable from the set $\tilde{Y}_{kN}$. In the procedure, the ideal point and the bounds in the objective space of problem $\tilde{P}_k$ are initially computed. Then all local nadir points and sets that are weakly dominated by $\tilde{\boldsymbol{y}}^{kI}$ are included in the set $\check{\Theta}_k$. During this process, some local nadir sets may be partitioned and reduced since only some parts of these sets might be weakly dominated by $\tilde{\boldsymbol{y}}^{kI}$. After applying the Preprocessing for Fathoming Rule 2 and Fathoming Rules 1, 2, 2a, and 2b, the final set $\check{\Theta}_k$ is returned.

If, as a result of the procedure REDUCENADIRSET, a node k cannot be fathomed, the procedure EXPLORENODE is initiated either to obtain the Pareto points of problem $\tilde{P}_k$ that are also feasible for problem P or to branch at the node k and generate two new subproblems. We use the Parametric Simplex Algorithm (PSA) [10] to solve the weighted-sum problem $\tilde{P}_k(\rho)$

Algorithm 2 The ϵ-constraint method to find feasible solutions and attainable points of problem P.

```
procedure ComputeFeasSol-ϵ ($P, \mathcal{N}, \boldsymbol{y}^{N-W}, \boldsymbol{y}^{S-E}$)
    $\hat{X} = \emptyset$, $\hat{Y} = \emptyset$
    Discretize the interval $[y_2^{S-E}, y_2^{N-W}]$ into $\mathcal{N}$ subintervals, i.e. $\tau = \frac{y_2^{N-W} - y_2^{S-E}}{\mathcal{N}}$
    Set $\epsilon_0 = y_2^{S-E}$
    for $j \in \{0, 1, 2, \ldots, \mathcal{N} - 1\}$ do
        $\max_{\boldsymbol{x}} \{\boldsymbol{c}_1^\top \boldsymbol{x} : A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{c}_2^\top \boldsymbol{x} > \epsilon_j, \boldsymbol{x} \in \mathbb{R}^p \times \mathbb{Z}^{n-p}\}$    ▷ $\epsilon$-constraint problem
        if solution $\boldsymbol{x}^j$ is found and $\boldsymbol{x}^j \notin \hat{X}$ then
            $\hat{X} \leftarrow \hat{X} \cup \{\boldsymbol{x}^j\}$
            $\hat{Y} \leftarrow \hat{Y} \cup \{\boldsymbol{z}(\boldsymbol{x}^j)\}$
        if $x_i^j = x_i^{j-1}$ for $i \in \{p+1, \ldots, n\}$ and $j > 1$ then
            $\hat{Y} \leftarrow \hat{Y} \cup [\boldsymbol{z}(\boldsymbol{x}^{j-1}), \boldsymbol{z}(\boldsymbol{x}^j)]$
        update $\epsilon_{j+1} = \epsilon_j + \tau$
    $\hat{Y} \leftarrow VMAX(\hat{Y})$
    $\hat{X} \leftarrow EFF(\hat{X})$
    Compute $\check{\Theta}$
    return $\hat{X}, \hat{Y}, \check{\Theta}$
```

associated with the subproblem $\tilde{P}_k$, where ρ is the parameter scalarizing the vector-valued objective function of problem P. The pseudo-code of the procedure ExploreNode is given in Algorithm 4. Lines 3-8 are the steps of the PSA. In every iteration of the algorithm, an optimal solution of the subproblem $\tilde{P}_k(\rho)$ is computed for the current value of ρ, starting with $\rho = 1$. For a given value of ρ the weighted-sum problem $\tilde{P}_k(\rho)$ may have multiple optimal solutions. Note that all these optimal solutions map to a unique point in the objective space of problem P. If desired, the PSA may have the capability of recording all these solutions.

In every iteration of the PSA, two mutually exclusive cases may be observed.

- The optimal solution of problem $\tilde{P}_k(\rho)$ obtained in iteration s is not feasible for problem P, i.e., $\boldsymbol{x}^s \notin \mathbb{R}^p \times \mathbb{Z}^{n-p}$. Then there exists $i \in \{p+1, p+2, \ldots, n\}$ such that $x_i^s \notin \mathbb{Z}$. At this point of the procedure, any such component x_i^s should be selected for branching so that $x_i \leq \lfloor x_i^s \rfloor$ or $x_i \geq \lceil x_i^s \rceil$;
- The optimal solution of problem $\tilde{P}_k(\rho)$ obtained in iteration s is feasible for problem P, i.e., $\boldsymbol{x}^s \in \mathbb{R}^p \times \mathbb{Z}^{n-p}$. The solution x^s is included in the set $\hat{X}$ and its image is included in the set $\hat{Y}$.

Since a solution $\boldsymbol{x}^s$ may be infeasible for problem P, we define the set U to store the infeasible solutions. Having multiple infeasible solutions gives more degrees of freedom on the variable to branch upon, which is very useful in several branch and bound algorithms given that the branching variable choice is critical to the efficiency of the search procedure.

If the optimal solution $\boldsymbol{x}^s$ obtained in iteration s is feasible for problem P, then $\boldsymbol{x}^s$ is feasible for the slice problem $P(\bar{\boldsymbol{x}}^s)$, where $\bar{x}_i^s = x_i^s$ for $i = p+1, p+2, \ldots, n$. However the optimal solutions obtained in the subsequent iteration may be feasible or not for the same slice problem. If $x_i^s = x_i^{s-1}$ for $i = p+1, p+2, \ldots, n$, then the optimal solutions $\boldsymbol{x}^{s-1}$ and $\boldsymbol{x}^s$ are feasible for the slice problems $P(\bar{\boldsymbol{x}}^{s-1})$ and $P(\bar{\boldsymbol{x}}^s)$, respectively, and $X(\boldsymbol{x}^{s-1}) = X(\boldsymbol{x}^s)$. The line segment $[\boldsymbol{z}(\boldsymbol{x}^{s-1}), \boldsymbol{z}(\boldsymbol{x}^s)]$ belongs to $Y_N(\bar{\boldsymbol{x}}^{s-1}) = Y_N(\bar{\boldsymbol{x}}^s)$. If there exists one component $i \in \{p+1, p+2, \ldots, n\}$ such that $x_i^s \neq x_i^{s-1}$, then $X(\boldsymbol{x}^{s-1}) \neq X(\boldsymbol{x}^s)$. If the latter takes place, the PSA has stopped solving the slice problem $P(\bar{\boldsymbol{x}}^{s-1})$ and switched to solving the slice problem $P(\bar{\boldsymbol{x}}^s)$. This switch is due to the fact that the Pareto set of problem

Algorithm 3 Separation of local nadir points and sets.

```
1: procedure ReduceNadirSet ($P_k, \check{\Theta}$)
2:     $(\boldsymbol{y}^{k,S-E}, \boldsymbol{y}^{k,N-W}) \leftarrow$ ComputeExtremes$(P_k)$
3:     if $(\boldsymbol{y}^{k,S-E}, \boldsymbol{y}^{k,N-W}) = (-\infty, -\infty)$ then return $\emptyset$
4:     else
5:         Compute $\boldsymbol{y}^{k*}$
6:         Set $y_1 = y_1'' - \frac{(y_1''-y_1')(y_2^{k*}-y_2'')}{(y_2'-y_2'')}$, $y_2 = y_2'' + \frac{(y_2'-y_2'')(y_1''-y_1^{k*})}{(y_1''-y_1')}$,
```

$$\check{\Theta}_k = \begin{cases} \check{\boldsymbol{y}} \in \check{\Theta} & \text{if} \quad \check{y}_1 \le y_1^{k*}, \quad \check{y}_2 \le y_2^{k*} \\ [(y_1', y_2'), (y_1'', y_2'')] \in \check{\Theta} & \text{if} \quad y_1' \le y_1^{k*}, y_1'' \le y_1^{k*}, \quad y_2' \le y_2^{k*}, y_2'' \le y_2^{k*} \\ [(y_1', y_2'), (y_1^{k*}, y_2)] \in \check{\Theta} & \text{if} \quad y_1' \le y_1^{k*} \le y_1'', \quad y_2'' \le y_2' \le y_2^{k*} \\ [(y_1, y_2^{k*}), (y_1'', y_2'')] \in \check{\Theta} & \text{if} \quad y_1' \le y_1'' \le y_1^{k*}, \quad y_2'' \le y_2^{k*} \le y_2' \\ [(y_1, y_2^{k*}), (y_1^{k*}, y_2)] \in \check{\Theta} & \text{if} \quad y_1' \le y_1^{k*} \le y_1'', \quad y_2'' \le y_2^{k*} \le y_2' \end{cases}$$

```
7:         if $\check{\Theta}_k = \emptyset$ then return $\emptyset$                    ▷ Fathoming Rule 1
8:         else
9:             Rank $\check{\Theta}_k$                     ▷ Preprocessing for Fathoming Rule 2
10:            for all elements in $\check{\Theta}_k$ do                    ▷ Fathoming Rule 2
11:                Solve $P_{FR}^{\star}$
12:                if optimum $(\lambda^\star, \boldsymbol{w}^\star)$ exists with a nonpositive objective value then
13:                    $\check{\Theta}_k \leftarrow \check{\Theta}_k \backslash \{\check{\boldsymbol{y}}\}$ or $\check{\Theta}_k \leftarrow \check{\Theta}_k \backslash \{[\boldsymbol{y}', \boldsymbol{y}'']\}$
14:                    for all $\check{\boldsymbol{y}} \in \check{\Theta}_k$ do                    ▷ Fathoming Rule 2a
15:                        if $\boldsymbol{b}_k^\top \boldsymbol{w}^\star \le \lambda^\star \check{y}_1 + (1-\lambda^\star)\check{y}_2$ then
16:                            $\check{\Theta}_k \leftarrow \check{\Theta}_k \backslash \{\check{\boldsymbol{y}}\}$
17:                    for all $[\boldsymbol{y}', \boldsymbol{y}''] \in \check{\Theta}_k$ do
18:                        if $\boldsymbol{b}_k^\top \boldsymbol{w}^\star \le \lambda^\star y_1' + (1-\lambda^\star)y_2'$ and $\boldsymbol{b}_k^\top \boldsymbol{w}^\star \le \lambda^\star y_1'' + (1-\lambda^\star)y_2''$
   then
19:                            $\check{\Theta}_k \leftarrow \check{\Theta}_k \backslash \{[\boldsymbol{y}', \boldsymbol{y}'']\}$
20:                    Update $\lambda$ and repeat Fathoming Rule 2a          ▷ Fathoming Rule 2b
21:                else
22:                    return $\check{\Theta}_k$
23:     return $\check{\Theta}_k$
```

P is composed of proper and improper subsets of $Y_N(\bar{\boldsymbol{x}}^s)$. To account for the switch, the line segment $[\boldsymbol{z}(\boldsymbol{x}^{s-1}), \boldsymbol{z}(\boldsymbol{x}^s)]$ is partitioned into two subsets by branching.

In general, Phase III of the PSA is performed for the so-called critical values of $\rho \in (0, 1]$. If the PSA reaches the stage that the index set I for computing the critical values of ρ is empty, then the entire Pareto set $\tilde{Y}_{kN}$ of problem $\tilde{P}_k$ has been explored.

The procedure stops if the PSA stopping condition is satisfied (line 8 in Algorithm 4) or any of the two additional conditions holds.

STOPPING CONDITION 1. *The optimal solution $\boldsymbol{x}^s$ is not feasible for problem P, i.e., $\boldsymbol{x}^s \notin \mathbb{R}^p \times \mathbb{Z}^{n-p}$. Then $\boldsymbol{x}^s$ is included in the set U of fractional (infeasible) solutions of problem P, and the subproblem exploration procedure stops.*

STOPPING CONDITION 2. *The optimal solution $\boldsymbol{x}^s$ is feasible for problem P, i.e., $\boldsymbol{x}^s \in \mathbb{R}^p \times \mathbb{Z}^{n-p}$, but is not feasible for the same slice problem as the optimal solution $\boldsymbol{x}^{s-1}$ found in iteration $s-1$. The solution $\boldsymbol{x}^s$ is included in the set $\hat{X}$ and its image in the set $\hat{Y}$. Since* *there exists a component $i \in \{p+1, p+2, ..., n\}$ such that $x_i^s \neq x_i^{s-1}$, a new feasible solution $\boldsymbol{x}^{new}$ for problem $\tilde{P}_k$ is constructed by setting $x_i^{new} = x_i^s + \delta(x_i^{s-1} - x_i^s)$, where $0 < \delta < 1/|x_i^{s-1} - x_i^s|$, for $i \in \{p+1, p+2, ..., n\}$ and $x_i^{new} = x_i^s$ for $i \in \{1, 2, ..., p\}$. The point $\boldsymbol{x}^{new}$ is included in the set U and the subproblem exploration procedure stops.*

The procedure returns the set of feasible solutions $\hat{X}$, the set of attainable points $\hat{Y}$, and the set U of fractional (infeasible) solutions of problem P.

Algorithm 4 Exploration of unfathomed nodes with the PSA.

```
1: procedure ExploreNode ($P_k, \hat{X}, \hat{Y}$)
2:     Set $s = 0, U = \emptyset$
3:     Phase I: Apply Phase I of the Two-phase Method to obtain initial basic feasible solution $\boldsymbol{x}^s$ to subproblem $\tilde{P}_k$. Let $\bar{\boldsymbol{c}}_1^s$ and $\bar{\boldsymbol{c}}_2^s$ be the reduced cost vectors of each objective function associated with $\boldsymbol{x}^s$.    ▷ PSA
4:     Set $\rho^s = 1$ and solve subproblem $\tilde{P}_k(\rho^s)$ for optimal $\boldsymbol{x}^s$.    ▷ Phase II
5:     loop    ▷ Phase III
6:         $s \leftarrow s + 1$
7:         if $I = \{i \in \boldsymbol{N}_k : \bar{c}_{1i}^s \leq 0, \bar{c}_{2i}^s > 0\} = \emptyset$ then
8:             Return $\hat{X}, \hat{Y}, U$    ▷ PSA stopping condition is satisfied
9:         else
10:            Compute $\rho^s$ and corresponding optimal $\boldsymbol{x}^s$.
11:            if Stopping Condition 1 is satisfied then
12:                $U \leftarrow U \cup \{\boldsymbol{x}^s\}$
13:                Return $\hat{X}, \hat{Y}, U$
14:            else if Stopping Condition 2 is satisfied then
15:                $\hat{X} \leftarrow \hat{X} \cup \{\boldsymbol{x}^s\}, \hat{Y} \leftarrow \hat{Y} \cup \{\boldsymbol{z}(\boldsymbol{x}^s)\}$
16:                $\hat{Y} \leftarrow VMAX(\hat{Y}), \hat{X} \leftarrow EFF(\hat{X})$, update $\check{\Theta}$
17:                Construct $\boldsymbol{x}^{new}$ and $U \leftarrow U \cup \{\boldsymbol{x}^{new}\}$
18:                Return $\hat{X}, \hat{Y}, U$
19:            else
20:                $\hat{Y} \leftarrow [\boldsymbol{z}(\boldsymbol{x}^{s-1}), \boldsymbol{z}(\boldsymbol{x}^s)]$
21:                $\hat{X} \leftarrow \hat{X} \cup \{\boldsymbol{x}^s\}$
22:                $\hat{Y} \leftarrow VMAX(\hat{Y}), \hat{X} \leftarrow EFF(\hat{X})$, update $\check{\Theta}$
23:    end loop
24:    Return $\hat{X}, \hat{Y}, \emptyset$
```

Algorithm 5 presents the pseudo-code of the BB algorithm for computing Pareto points and efficient solutions of BOMILPs. The data of problem P and the number $\mathcal{N}$ is the only input to the algorithm. The procedure ComputeExtremes is initially called to establish the bounds in the objective space. Problem P is assigned to the root node of the BB tree. The yet unexplored nodes of the tree are stored in the list $\mathcal{L}$. The procedure ComputeFeasSol-ϵ is then called to initialize the sets $\hat{X}$, $\hat{Y}$, and $\check{\Theta}$. The main steps of the algorithm are performed as long as the tree has at least one node. For the selected node k, the related subproblem P_k is then known to be feasible and is examined.

First, the procedure ReduceNadirSet is called to identify separable local nadir points and sets and construct the set $\check{\Theta}_k$. Next, the procedure ExploreNode is called to solve $\tilde{P}_k$ with the PSA, update the sets $\hat{X}$, $\hat{Y}$, and U. If there are some fractional solutions available in the set U, branching takes place on the current node and two new subproblems P_k' and P_k'' are attached to the BB tree. Otherwise, the current node is fathomed, and the sets $\hat{Y}$, $\hat{X}$, and $\check{\Theta}$ are updated. Once all the nodes have been visited, the final sets $\hat{X}$ and $\hat{Y}$ contain efficient solutions and Pareto points of problem P respectively.

THEOREM 6.1. *Upon termination, the BB algorithm returns a set $\hat{Y}$ that contains all Pareto points of problem P, i.e., $\hat{Y} = Y_N$.*

Proof. The proof is based on the two key operators of the BB scheme used in Algorithm 5: branching and fathoming rules. The root node of the BB tree is associated with the feasible set X of the original problem, and we prove that all Pareto points are known upon termination of the BB. The following discussion is on the generic node X_k of the BB tree, keeping in mind that the root node is but a special case.

We consider a set $\hat{X}$ of feasible solutions of P that changes during the algorithm, at each node of the BB tree and at every iteration of the PSA. An initial set $\hat{X}$ of feasible solutions and attainable points $\hat{Y}$ of problem P are generated by the procedure COMPUTEFEASSOL in Algorithm 2. During the course of the algorithm, $\hat{Y} = \hat{Y}_N$ and $\hat{Y}$ is modified by adding new elements or dropping the elements that are dominated by the newly added ones. Note that the elimination of elements occurs by applying the $VMAX$ operator to the set $\hat{Y}$ after an element has been added. The $VMAX$ operator does not eliminate any efficient solution. The set $\hat{X}$ is updated accordingly.

For each node k, fathoming rules are tested against its attainable set $\tilde{Y}_k$. If at least one of them holds, then each Pareto point of $\tilde{P}_k$ is dominated by at least one element of $\hat{Y}$, as shown in Section 5. In effect, the elements in $\tilde{Y}_k$ will not be considered by the algorithm and will never enter the set $\hat{Y}$. This is correct behavior since the fathoming rules guarantee that $\tilde{Y}_k^{\leqq}$ cannot contain any Pareto point of problem P, and hence neither can Y_k.

If no fathoming rule succeeds, node k is explored by Algorithm 4, that relies on the PSA. Starting from an initial solution $\boldsymbol{x}^0$, the algorithm finds, at each iteration s, a new efficient solution $\boldsymbol{x}^s$ to $\tilde{P}_k$, for $s = 1, 2, \dots, \sigma$ where σ is the number of iterations performed until either the PSA comes to completion or a stopping condition is satisfied.

If $\boldsymbol{x}^s$ is not feasible for P, i.e., $\boldsymbol{x}^s \notin \mathbb{R}^p \times \mathbb{Z}^{n-p}$, a branching rule $x_i \leq \lfloor x_i^s \rfloor \vee x_i \geq \lceil x_i^s \rceil$ is applied on a variable x_i for $i \in \{p+1, p+2, \dots, n\}$ such that $x_i^s \notin \mathbb{Z}$ and $i \in \{p+1, p+2, \dots, n\}$ and two new BB nodes are created. By branching, we do not exclude any Pareto point of P from $\hat{Y}$ because we do not modify $\hat{Y}$ at all. Also, any Pareto point of P_k is a Pareto point of either P'_k or P''_k, which will be examined in later iterations.

If $\boldsymbol{x}^s$ is feasible for P, i.e., $\boldsymbol{x}^s \in \mathbb{R}^p \times \mathbb{Z}^{n-p}$, then $\boldsymbol{x}^{s-1}$ is integer feasible, that is, $\boldsymbol{x}^{s-1} \in \mathbb{R}^p \times \mathbb{Z}^{n-p}$, otherwise a branching rule would have been applied in step $s-1$. The solution $\boldsymbol{x}^s$ is added to $\hat{X}$ and its image is added to $\hat{Y}$.

Two cases arise: in the first case, $\boldsymbol{x}^s$ and $\boldsymbol{x}^{s-1}$ do not have the same integer components, i.e., there exists $i \in \{p+1, p+2, \dots, n\}$ such that $x_i^s \neq x_i^{s-1}$. Suppose w.l.o.g. that $x_i^{s-1} < x_i^s$. Similar to the fractional case above, a branching rule $x_i \leq b \vee x_i \geq b+1$ is applied, where $b \in (x_i^{s-1}, x_i^s) \cap \mathbb{Z}$. This generates two new BB subproblems, which will be visited in a subsequent step of the algorithm.

In the second case, $\boldsymbol{x}^s$ and $\boldsymbol{x}^{s-1}$ have the same integer components, i.e., $x_i^s = x_i^{s-1}$ for $i \in \{p+1, p+2, \dots, n\}$. Because these solutions have been found by the PSA, conv$\{\boldsymbol{x}^s, \boldsymbol{x}^{s-1}\}$ is in the set of efficient solutions of $\tilde{P}_k$. Since conv$\{\boldsymbol{x}^s, \boldsymbol{x}^{s-1}\}$ contains only feasible solutions for P, conv$\{\boldsymbol{x}^s, \boldsymbol{x}^{s-1}\}$ is also in the set of efficient solutions of X_k. Hence conv$\{\boldsymbol{x}^s, \boldsymbol{x}^{s-1}\}$ is added to $\hat{X}$ and its image, which is in the set of Pareto points of P_k, is added to $\hat{Y}$.

Since at each iteration of the PSA the set of Pareto points is either saved into $\hat{Y}$ or partitioned by branching, no Pareto point is excluded from the search. This proves that the exploration of a single subproblem does not exclude any Pareto point of P_k. Since every node is generated through branching, which is an operator that does not eliminate any feasible solution of P, all Pareto points of P and at least one efficient solution for each Pareto point are output by the BB algorithm. This concludes the proof. □

Algorithm 5 The branch-and-bound algorithm for BOMILPs.

```
1: procedure BicriMILP-BB ($P, \mathcal{N}$)
2:     $(\boldsymbol{y}^{N-W}, \boldsymbol{y}^{S-E}) = $ ComputeExtremes$(P)$    ▷ Two LPs of size $(m, n)$
3:     if $(\boldsymbol{y}^{k,S-E}, \boldsymbol{y}^{k,N-W}) = (-\infty, -\infty)$ then return $\emptyset, \emptyset$
4:     else
5:         $\mathcal{L} \leftarrow \{P\}$    ▷ Problem at the root node
6:         $\hat{X}, \hat{Y}, \check{\Theta} \leftarrow$ ComputeFeasSol-$\epsilon(P, \mathcal{N}, \boldsymbol{y}^{N-W}, \boldsymbol{y}^{S-E})$    ▷ $\check{\Theta}, \hat{Y}$ initialized
7:         while $\mathcal{L} \neq \emptyset$ do
8:             Select $P_k \in \mathcal{L}$; $\mathcal{L} \leftarrow \mathcal{L} \backslash \{P_k\}$    ▷ Select w.r.t. a selection rule
9:             $\check{\Theta}_k \leftarrow$ ReduceNadirSet$(P_k, \check{\Theta})$
10:            if $\check{\Theta}_k \neq \emptyset$ then    ▷ This node could not be fathomed
11:                $\hat{X}, \hat{Y}, U \leftarrow$ ExploreNode$(P_k, \hat{X}, \hat{Y})$    ▷ $U$: set of fractional solutions
12:                if $U \neq \emptyset$ then
13:                    Select $\boldsymbol{x}^\star \in U$
14:                    Select $i \in \{p+1, \ldots, n\}$ such that $x_i^\star \notin \mathbb{Z}$ ▷ Select w.r.t. a branching
    rule
15:                    $P_k' \leftarrow P_k \cap \{\boldsymbol{x} \in \mathbb{R}^p \times \mathbb{Z}^{n-p} : x_i \leq \lfloor x_i^\star \rfloor\}$
16:                    $P_k'' \leftarrow P_k \cap \{\boldsymbol{x} \in \mathbb{R}^p \times \mathbb{Z}^{n-p} : x_i \geq \lceil x_i^\star \rceil\}$
17:                    $\mathcal{L} \leftarrow \mathcal{L} \cup \{P_k', P_k''\}$    ▷ Branch: $\mathcal{L}$ is augmented
18:        return $\hat{X}, \hat{Y}$
```

7. Example.

7. Example. Consider an example given as the BOMILP (7.1). Figure 7.1 illustrates the Pareto sets $Y_N(\bar{\boldsymbol{x}})$ of ten slice problems for this example. Some sets intersect each other while others do not. All sets but one are in the form of a piecewise linear concave curve while one is a singleton. The Pareto set Y_N of the overall problem can be found by applying (2.3). It can be observed from the figure that Y_N is a nonconvex and disconnected set which makes the problem difficult to solve.

$$
\begin{array}{llllllll}
\max & \boldsymbol{z}_1(\boldsymbol{x}) = & -3x_1 & -6x_2 & +3x_3 & +5x_4 & & \\
\max & \boldsymbol{z}_2(\boldsymbol{x}) = & 15x_1 & +4x_2 & +x_3 & +2x_4 & & \\
\text{s.t.} & & -x_1 & & & & +3x_5 & & \leq 0 \\
 & & x_1 & & & & -6x_5 & & \leq 0 \\
 & & & -x_2 & & & +3x_5 & & \leq 0 \\
 & & & x_2 & & & -6x_5 & & \leq 0 \\
 & & & & -x_3 & & & +4x_6 & \leq 0 \\
 & & & & x_3 & & & -4.5x_6 & \leq 0 \\
 & & & & & -x_4 & & +4x_6 & \leq 0 \\
 & & & & & x_4 & & -4.5x_6 & \leq 0 \\
 & & & & & & x_5 & +x_6 & \leq 5 \\
 & & x_1, & x_2, & x_3, & x_4 \in \mathbb{R}_+ & & & \\
 & & & & & & x_5, & x_6 \in \mathbb{Z}_+. &
\end{array}
\tag{7.1}
$$

We illustrate the basic steps of the BB algorithm on this example. Initially, the procedure ComputeExtremes is called to compute the extreme points, $\boldsymbol{y}^{N-W} = (-270, 570)$ and $\boldsymbol{y}^{S-E} = (180, 67.5)$, of problem (7.1) and establish the bounds in the objective space. Then the procedure ComputeFeasSol-ϵ is used to compute five feasible solutions and the cor-

Attainable points $\hat{Y}$	Nadir points $\check{\Theta}$
$\hat{\boldsymbol{y}}^1 = (-191.3, 517.5)$	$\check{\boldsymbol{y}}^1 = (-270, 517.5)$
$\hat{\boldsymbol{y}}^2 = (-107.2, 417.5)$	$\check{\boldsymbol{y}}^2 = (-191.3, 417.5)$
$\hat{\boldsymbol{y}}^3 = (-32.9, 317.5)$	$\check{\boldsymbol{y}}^3 = (-107.2, 317.5)$
$\hat{\boldsymbol{y}}^4 = (41.4, 217.5)$	$\check{\boldsymbol{y}}^4 = (-32.9, 217.5)$
$\hat{\boldsymbol{y}}^5 = (115.7, 117.5)$	$\check{\boldsymbol{y}}^5 = (41.4, 117.5)$
	$\check{\boldsymbol{y}}^6 = (115.7, 67.5)$

TABLE 7.1
Initial attainable points and nadir points of problem (7.1).

responding attainable criterion vectors of problem (7.1). The initial attainable points and the associated nadir points are listed in Table 7.1.

We now discuss the steps of the BB algorithm on two arbitrarily selected node problems. Let P_{k_1} and P_{k_2} be two different node problems. Problem P_{k_1} is problem (7.1) with the additional constraint $x_6 \geq 4$, and problem P_{k_2} is problem (7.1) with the additional constraints $x_5 \leq 3$ and $x_6 = 0$. The upper-bound sets and the ideal points of problems $\tilde{P}_{k_1}$ and $\tilde{P}_{k_2}$ are illustrated in Figure 7.2.

Assume problem P_{k_1} is selected from the list $\mathcal{L}$. Since the procedure REDUCENADIRSET is called, the set $\check{\Theta}_{k_1} = \{\check{\boldsymbol{y}}^5, \check{\boldsymbol{y}}^6\}$ is constructed and the fathoming rules are checked. Fathoming Rule 1 fails, since $\check{\Theta}_{k_1} \neq \emptyset$. Therefore Fathoming Rule 2 is applied starting with the preprocessing which assigns rank 1 to the nadir point $\check{\boldsymbol{y}}^5$. In effect, problem $P^{\star}_{FR}(\check{\boldsymbol{y}}^5)$ is solved:

$$
\begin{array}{lllllllll}
P^{\star}_{FR}(\check{\boldsymbol{y}}^5): & \min & 5w_9 & -4w_{10} & +76.1\lambda & -117.5 & & & \\
& \text{s.t.} & -w_1 & +w_2 & -18\lambda & & & & \geq 15 \\
& & -w_3 & +w_4 & -10\lambda & & & & \geq 4 \\
& & -w_5 & +w_6 & +2\lambda & & & & \geq 1 \\
& & -w_7 & +w_8 & +3\lambda & & & & \geq 2 \\
& & 3w_1 & -6w_2 & +3w_3 & -6w_4 & +w_9 & & \geq 0 \\
& & 4w_5 & -4.5w_6 & +4w_7 & -4.5w_8 & +w_9 & -w_{10} & \geq 0 \\
& & & & & & & 0 \leq \lambda & \leq 1 \\
& & & & & & & w_1, w_2, \ldots, w_{10} & \geq 0.
\end{array}
$$

Since the optimal objective value of this linear program is equal to 50.5, the nadir point $\check{\boldsymbol{y}}^5$ is not separable and, by Fathoming Rule 2, node k_1 cannot be fathomed. Since the rule fails on $\check{\boldsymbol{y}}^5$, the nadir point $\check{\boldsymbol{y}}^6$ does not have to be checked for separability. The procedure EXPLORENODE begins to search for possible Pareto points at this node by solving problem $\tilde{P}_{k_1}$ with the PSA. As soon as a solution with noninteger x_5 or x_6 components has been found in an iteration of the PSA, branching is applied on the integer variable having maximum infeasibility.

In the first iteration of the PSA, a solution to $\tilde{P}_{k_1}$ is found that is also feasible to problem (7.1). The integer components of this solution are $x_5 = 0$ and $x_6 = 5$ and the corresponding Pareto point is $(180, 67.5)$. In the second iteration, the PSA yields another solution to $\tilde{P}_{k_1}$ (and feasible to (7.1)) with $x_5 = 1, x_6 = 4$, and the image point $(108, 156)$. Since these two optimal solutions have different values of the two integer components, they belong to two different slice problems. Therefore one can say that the PSA "jumps" from one slice problem to another and therefore any convex combination of $(180, 67.5)$ and $(108, 156)$ is not attainable for problem (7.1). Since the maximum infeasibility is equal for the two integer variables, branching may occur on either x_5 or x_6.

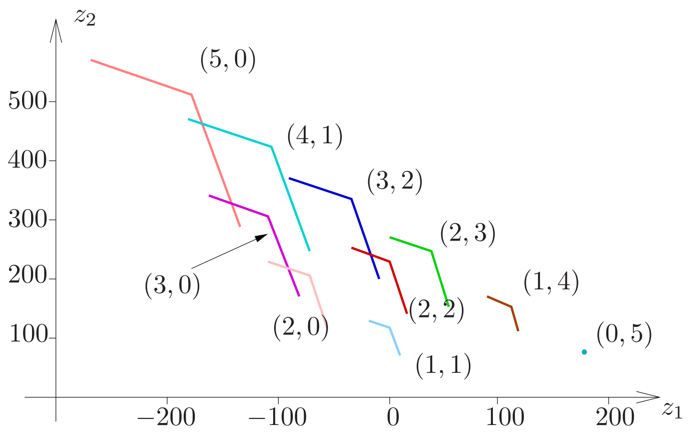


FIG. 7.1. *Pareto sets of slice problems of BOMILP* (7.1)*. Each Pareto set is indicated in a different shade. Next to each Pareto set is the pair* (x_5, x_6) *of values of the integer variables the slice problem is associated with.*

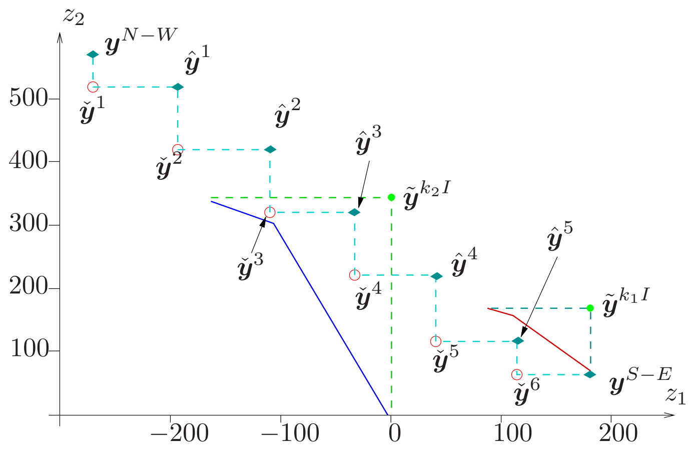


FIG. 7.2. *The initial set* $\hat{Y}$ *and the associated set* $\check{\Theta}$ *for BOMILP* (7.1)*. The ideal points and Pareto sets of problems* $\tilde{P}_{k_1}$ *and* $\tilde{P}_{k_2}$ *are also depicted.*

Assume now that problem P_{k_2} is selected from the list $\mathcal{L}$ during the execution of the BB algorithm. Here $\check{\Theta}_{k_2} = \{\check{\boldsymbol{y}}^3, \check{\boldsymbol{y}}^4\} \neq \emptyset$. After applying the preprocessing for Fathoming Rule 2, problem $P^{\star}_{FR}(\check{\boldsymbol{y}}^3)$ is solved and its optimal objective value is equal to -7.22 with $\lambda^{\star} = 0.4$ and $\boldsymbol{w}^{\star} = (w_1^{\star}, w_2^{\star}, \ldots, w_{11}^{\star}) = (0, 7.8, 0, 0, 0,\ 1.8,\ 0,\ 3.2,\ 0,\ 46.8,\ 22.5)$. This means that the nadir point $\check{\boldsymbol{y}}^3$ is separable. At this stage, Fathoming Rule 2a is applied and the feasibility of the current optimal solution $(\lambda^{\star}, \boldsymbol{w}^{\star})$ is checked for the nadir point $\check{\boldsymbol{y}}^4$. The condition $\boldsymbol{b}_{k_2}^{\top}\boldsymbol{w}^{\star} \leq \lambda^{\star}\check{y}_1^4 + (1-\lambda^{\star})\check{y}_2^4$ is not satisfied, and the set $\check{\Theta}_{k_2}$ is reduced to the singleton $\{\check{\boldsymbol{y}}_4\}$.

Fathoming Rule 2b is then applied. Using (5.5), $\hat{\lambda} = 0.308$ is computed to check that the condition $A_{k_2}^{\top}\boldsymbol{w}^{\star} = \hat{\lambda}\boldsymbol{c}_1 + (1-\hat{\lambda})\boldsymbol{c}_2$ does not hold. Hence, node k_2 cannot be fathomed.

8. Computational Results.

The proposed BB algorithm was coded in C using the Cplex 12.1 solver library [8]. This library allows to implement a branch-and-bound for a single-objective problem, therefore adapting it to solve BOMILPs required to turn off the check for the *cutoff* value to fathom a node, given that there is none in the BOMILP case, and to construct a separate data structure to hold the sets $\hat{X}$ and $\hat{Y}$. The algorithm was run on a 3.2 GHz workstation with 4 GB of RAM memory and tested on instances of BOMILPs of different size. The problems were generated using the interval ranges of Mavrotas and Diakoulaki [18, 19]. The elements of the matrix A and the vectors $\boldsymbol{b}$ and $\boldsymbol{c}$ were randomly generated within the following intervals:

- $[-10, 10]$ for the elements of $\boldsymbol{c}$ associated with continuous variables;
- $[-200, 200]$ for the elements of $\boldsymbol{c}$ associated with 0-1 variables;
- $[-50, 50]$ for the elements of $\boldsymbol{c}$ associated with integer variables;
- $[50, 100]$ for the elements of $\boldsymbol{b}$;
- $[-1, 20]$ for the elements of A.

In all test problems, the number of variables is equal to the number of constraints ($n = m$). The size of the problems gradually increases to 80 variables and 80 constraints. Different types of problems have been generated: problems with all types of variables (continuous, integer, and binary), problems with no integer variables, and problems with no binary variables. Five instances were generated for each type and each size of the problem. In the current implementation, the set of local nadir points and local nadir sets $\check{\Theta}$ is created using the initial points in $\hat{Y}$ and remains unchanged throughout the BB. Therefore, one can expect a better performance with larger values of $\mathcal{N}$, because more local nadir points and local nadir sets allow us to fathom more BB nodes. A more efficient implementation requires a dynamic data structure that is currently under development.

The experiments were performed in four stages. In the first stage, the set $\hat{Y}$ was initialized by setting $\mathcal{N}$ to 100 which means that the ϵ-constraint method was used for 100 subintervals (which does not guarantee to obtain 100 points in Y since the same points can be computed for different subintervals). Table 8.1 presents the results obtained for test problems with all types of variables, while Tables 8.2 and 8.3 present the results for test problems with no integer and no binary variables, respectively. In the first column of the tables, the following notation is used for the problem size:

n_c**:** the number of continuous variables ($n_c = p$);
n_b**:** the number of binary variables ($0 \leq n_b \leq n - p$);
n_i**:** the number of integer variables ($0 \leq n_i \leq n - p, n_b + n_i = n - p$).

For each group of five problems the following statistics related to the effectiveness of the BB algorithm were calculated and are reported in the tables in column two through column ten:

CPU: the CPU time measured in seconds;
nodes: the number of nodes processed;
leaf: the number of leaf nodes processed;
FR1: the number of nodes fathomed using *Fathoming Rule 1*;
FR2: the number of nodes fathomed using *Fathoming Rule 2*;
inf: the number of nodes fathomed due to infeasibility;
full exp.: the number of nodes fully explored by the Parametric Simplex Algorithm (PSA);
Pareto: the percentage of Pareto extreme points among all attainable extreme points computed;
preproc.: the number of problems $P^{\star}_{FR}(\check{\boldsymbol{y}})$ that the algorithm did not have to solve because preprocessing allowed to discard them.

According to the results obtained, as the problem size increases the number of nodes processed by and the CPU time of the BB algorithm increase as expected. The number of nodes processed grows not only with the size of the problem but also with the problem type. Since the leaf nodes statistic is zero, the algorithm constructs the set Y_N before reaching a leaf node. On average, 45% of the nodes opened are fathomed by Fathoming Rule 1 or 2

$n \times m \times n_c \times n_b \times n_i$	CPU (s)	nodes	leaf	FR1	FR2	inf	full exp.	Pareto	preproc.
$20 \times 20 \times 10 \times 5 \times 5$	0.60	78.0	0	22.0	6.0	0	23.6	0.54	2.0
$40 \times 40 \times 20 \times 10 \times 10$	3.32	238.0	0	67.6	34.6	0	30.0	0.53	28.8
$60 \times 60 \times 30 \times 15 \times 15$	10.96	458.4	0	108.4	74.6	22.6	32.8	0.45	91.6
$80 \times 80 \times 40 \times 20 \times 20$	22.34	616.8	0	156.0	71.2	158.4	27.2	0.35	45.8

TABLE 8.1
Performance statistics of the BB algorithm for instances with all types of variables.

$n \times m \times n_c \times n_b \times n_i$	CPU (s)	nodes	leaf	FR1	FR2	inf	full exp.	Pareto	preproc.
$20 \times 20 \times 10 \times 10 \times 0$	0.54	70.0	0	21.0	5.4	0	18.0	0.65	1.4
$40 \times 40 \times 20 \times 20 \times 0$	2.40	164.4	0	38.2	30.0	0.8	23.2	0.48	17.6
$60 \times 60 \times 30 \times 30 \times 0$	14.40	638.4	0	171.0	89.8	31.4	28.8	0.39	94.4
$80 \times 80 \times 40 \times 40 \times 0$	37.60	1026.8	0	203.6	165.6	110.8	40.4	0.27	199.4

TABLE 8.2
Performance statistics of the BB algorithm for instances with continuous and binary variables.

or due to infeasibility. Note that the number of nodes fully explored by the PSA obviously increases with the problem size but does not significantly differ for problems of the same size and different type. The *Pareto* column indicates that as the problem size increases, the probability of an attainable extreme point to be a Pareto point decreases.

The last column shows the savings due to the preprocessing (at line 9 of Algorithm 3), i.e., the number of linear programs that were not solved explicitly. It is apparent that the preprocessing stage contributes to the performance of the algorithm.

At the second stage, experiments were conducted for three different numbers of subintervals, $\mathcal{N} = 10, 50, 100$, to examine the effect of this number on the performance of the BB algorithm, and the results are reported in Table 8.4. For each $\mathcal{N}$, the statistics including the CPU time, the number of nodes processed (‘nodes’), and the number of all nodes fathomed due to Fathoming Rule 1 or 2 or infeasibility (‘fathomed’) are reported. As expected, as $\mathcal{N}$ increases, all performance measures improve but the improvement between $\mathcal{N} = 10$ and $\mathcal{N} = 50$ is better than between $\mathcal{N} = 50$ and $\mathcal{N} = 100$. The choice of $\mathcal{N}$ may depend on the problem complexity and the method for computing the initial set $\hat{Y}$. The tradeoff between the CPU time spent in the initial $\mathcal{N}$ mixed-integer linear programs and the benefit of a tighter set of local nadir points/sets may differ from one problem class to the other. Given that the algorithm has to solve $\mathcal{N}$ mixed-integer linear programs, for large scale instances it is desirable to keep $\mathcal{N}$ limited. Although we used the exact procedure, a heuristic method can also be applied.

The goal of the experiments performed in the third stage was to compare the effectiveness of the fathoming rules already established in the literature and the new fathoming rules proposed in this paper. Table 8.6 reports the results obtained when the established and proposed fathoming rules are embedded in the BB algorithm. The results are presented for four combinations of the BB algorithm: (i) the complete BB algorithm, Fathoming Rule 1 (FR1) and 2 (FR2); (ii) the BB algorithm equipped only with FR1; (iii) the BB algorithm equipped with FR1 and Established Fathoming Rule 2 (EFR2); (iv) the BB algorithm equipped with FR1 and Established Fathoming Rule 3 (EFR3). The implementation of EFR2 is straightforward. In order to implement EFR3, we use the set Λ of uniformly distributed weight vectors with $|\Lambda| = 11$.

For every combination we report four statistics: the CPU time, the number of nodes processed (‘nodes’), the number of nodes fathomed by the applied rule(s) (‘fathomed by’),

$n \times m \times n_c \times n_b \times n_i$	CPU (s)	nodes	leaf	FR1	FR2	inf	full exp.	Pareto	preproc.
$20 \times 20 \times 10 \times 0 \times 10$	0.54	66.0	0	21.4	5.6	0	11.6	0.59	1.8
$40 \times 40 \times 20 \times 0 \times 20$	4.06	281.6	0	61.6	55.0	0.2	38.4	0.29	65.0
$60 \times 60 \times 30 \times 0 \times 30$	16.04	738.8	0	212.4	70.4	45.0	40.4	0.21	139.6
$80 \times 80 \times 40 \times 0 \times 40$	53.48	1414.8	0	279.4	219.2	150.8	42.4	0.20	244.4

TABLE 8.3
Performance statistics of the proposed BB algorithm for instances with continuous and integer variables.

	$\mathcal{N}$=10			$\mathcal{N}$=50			$\mathcal{N}$=100		
	CPU	nodes	fathomed	CPU	nodes	fathomed	CPU	nodes	fathomed
$80 \times 80 \times 40 \times 20 \times 20$	31.96	935.2	416.2	23.38	635.6	293.4	22.34	616.8	385.6
$80 \times 80 \times 40 \times 40 \times 0$	72.44	2150.8	952.0	39.64	1096.8	508.8	37.58	1026.8	480.0
$80 \times 80 \times 40 \times 0 \times 40$	99.96	2902.8	1246.6	54.80	1496.4	669.6	53.12	1414.8	649.4

TABLE 8.4
Performance statistics of the BB algorithm for different number of subintervals, $\mathcal{N}$.

and the number of times the PSA is run (‘PSA run’). The $*$ sign marks the best combination of the BB algorithm for each statistic. According to the obtained results, in terms of the number of nodes processed, the proposed BB algorithm (combinations (i) and (ii)) is better than the other two algorithmic combinations ((iii) and (iv)). The performance of combination (iv) is close to that of (i). In terms of CPU time, the BB algorithm takes longer time due to solving the auxiliary linear programs and combination (iv) is better than others especially for larger instances. Note that all rules (FR1+EFR2+EFR3+FR2) can be used together in a BB. However, we wanted to measure the impact of each rule rather than build a fast BOMILP solver.

In the final experimental stage, we solved the biobjective set covering problem (BOSCP) with the BB algorithm using the data in Jaszkiewicz [14]. The BOSCP is a well-known problem in which two linear objective functions are maximized subject to covering constraints and all variables are constrained to be binary. Using m and n to denote the number of items and sets, respectively, we report the obtained results in Table 8.5 for problems of three sizes, which are specified in column 1. In columns two to four we report the CPU time, the number of nodes processed (‘nodes’), and the numbers of nodes fathomed due to Fathoming Rule 1 or 2 (‘fathomed by FR1 + FR2’), respectively. Observe that, as the problem size increases, the BB tree grows dramatically. Despite the fact that the algorithm is designed for BOMILPs, it can be applied to purely integer programs and its performance on the BOSCP, a difficult combinatorial optimization problem, is acceptable.

9. Conclusions.

In this study, we investigated the properties of biobjective mixed-integer linear programs (BOMILPs) and their Pareto sets, proposed a BB algorithm for finding all Pareto points of BOMILPs, and tested it on a variety of instances.

The algorithm is equipped with new fathoming rules that are more general and powerful than those already available in the literature. In general, a fathoming rule separates the nadir points associated with the currently available attainable points of the original problem from the Pareto set of the linear relaxation of the problem at a node of the BB tree. The rules we have found in the literature use a simple separating inequality obtained from the two extreme points of the node problem or apply a separating hyperplane provided by the user. We propose rules that model this separation problem as a linear program and, by using strong duality, prove that a node problem does not contain Pareto points. As a consequence, more

$m \times n$	CPU(s)	nodes	fathomed by FR1 + FR2
5×25	0.1	4	1+0
10×50	2.1	466	128+59
10×100	49.3	7450	2096+1171

TABLE 8.5
Performance statistics of the proposed BB algorithm for biobjective SCP instances.

	Proposed BB				BB with FR1			
$n \times m \times n_c \times n_b \times n_i$	CPU (s)	nodes	Fathomed by FR1+FR2	PSA run	CPU	nodes	Fathomed by FR1	PSA run
$20 \times 20 \times 10 \times 5 \times 5$	0.62	78.0*	22.0+6.0	52.8	0.30*	102.4	37.2	66.0
$40 \times 40 \times 20 \times 10 \times 10$	3.32	238.0*	67.6+34.6	136.8	2.60	589.2	271.6	313.2
$60 \times 60 \times 30 \times 15 \times 15$	10.86	458.4*	108.4+74.6	251.2	10.62	1514.8	657.8	779.6
$80 \times 80 \times 40 \times 20 \times 20$	22.34	616.8*	156.0+71.2	328.4	15.28	1453.2	579.8	746.4
$20 \times 20 \times 10 \times 10 \times 0$	0.54	70.0*	21.0+5.4	46.2	0.32*	88.0	35.4	55.2
$40 \times 40 \times 20 \times 20 \times 0$	2.38	164.4*	38.2+30.0	97.0	2.18	499.6	229.8	264.4
$60 \times 60 \times 30 \times 30 \times 0$	14.38	638.4*	171.0+89.8	338.2	11.42	1638.8	717.0	838.8
$80 \times 80 \times 40 \times 40 \times 0$	37.58	1026.8*	203.6+165.6	538.2	26.40	2482.5	962.5	1267.8
$20 \times 20 \times 10 \times 0 \times 10$	0.56	66.0	21.4+5.6	41.4	0.26*	87.6	37.4	52.4
$40 \times 40 \times 20 \times 0 \times 20$	4.00	281.6*	61.6+55.0	162.2	4.10	954.4	435.2	502.0
$60 \times 60 \times 30 \times 0 \times 30$	15.90	738.8*	212.4+70.4	392.8	11.40	1640.4	683.4	843.8
$80 \times 80 \times 40 \times 0 \times 40$	53.12	1414.8*	279.4+219.2	738.4	35.20	3217.1	1214.7	1609.3

	BB with FR1 $+$ Established FR2				BB with FR1 $+$ Established FR3			
$n \times m \times n_c \times n_b \times n_i$	CPU	nodes	Fathomed by FR1+EFR2	PSA run	CPU	#node	Fathomed by FR1+EFR3	PSA run
$20 \times 20 \times 10 \times 5 \times 5$	0.52	92.4	30.8+4.2	60.2	0.48	85.6	26.8+5.0	56.6
$40 \times 40 \times 20 \times 10 \times 10$	2.30	273.2	82.4+37.0	154.6	2.22*	248.4	72.0+35.2	142.0
$60 \times 60 \times 30 \times 15 \times 15$	7.58	568.4	150.0+83.8	304.8	6.44*	464.8	114.8+70.4	253.0
$80 \times 80 \times 40 \times 20 \times 20$	14.28	708.8	196.6+72.0	373.8	12.96*	629.2	161.4+72.2	333.8
$20 \times 20 \times 10 \times 10 \times 0$	0.42	76.8	24.2+5.6	49.6	0.44	72.0	21.6+5.8	47.2
$40 \times 40 \times 20 \times 20 \times 0$	1.80	203.2	51.0+36.2	116.6	1.58*	172.0	40.6+31.2	101.0
$60 \times 60 \times 30 \times 30 \times 0$	11.04	822.4	239.2+110.4	431.0	9.12*	660.4	179.4+93.4	350.2
$80 \times 80 \times 40 \times 40 \times 0$	26.32	1312.4	306.8+183.8	681.4	21.42*	1029.2	214.2+155.8	539.8
$20 \times 20 \times 10 \times 0 \times 10$	0.40	68.0	26.0+2.0	42.6	0.40	66.0	25.0+2.2	41.4
$40 \times 40 \times 20 \times 0 \times 20$	3.08	363.2	88.4+65.8	203.4	2.52*	294.0	67.8+54.4	168.4
$60 \times 60 \times 30 \times 0 \times 30$	11.96	909.6	279.4+78.0	477.8	10.46*	773.2	222.0+75.4	409.4
$80 \times 80 \times 40 \times 0 \times 40$	36.40	1814.1	325.1+98.2	739.2	30.28*	1470.0	296.4+222.6	764.0

TABLE 8.6
Comparison of established and proposed fathoming rules (a "" indicates the best of all fathoming rules for the corresponding instance and measure).*

nodes are discarded from the BB tree, thus rendering the procedure more effective.

The emerging behavior of the BB algorithm is that an initial set of attainable points, $\hat{Y}$, evolves toward the Pareto set Y_N of problem P: $\hat{Y}$ is gradually modified by including new points and eliminating dominated points, and, upon termination, $\hat{Y}$ is identical to Y_N.

Our implementation is, however, preliminary. There are multiple improvements that are needed to make our solver capable of solving large-scale BOMILPs, such as a procedure to store the information gathered during the PSA in a node problem and use this information in descendent nodes. Also, an efficient data structure for the sets $\hat{X}$ and $\hat{Y}$, which store candidates for the efficient solutions and Pareto points, modeled as a special binary tree, would make the algorithm run faster.

Acknowledgement. The support from the Scientific and Technological Research Council of Turkey (TÜBITAK) for B. Soylu, who was a visiting assistant professor for a year at the Department of Mathematical Sciences, Clemson University, Clemson, SC, when working on this paper, is gratefully acknowledged.

References.

[1] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2004.

[2] M. J. Alves and J. Clímaco. A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, 180(1): 99–115, 2007.

[3] Y. P. Aneja and K. P. K. Nair. Bicriteria transportation problem. *Management Science*, 25(1):73–78, 1979.

[4] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem, A Computational Study*. Princeton, 2006.

[5] M. Benichou, J. M. Gauthier, P. Girodet, G. Hentges, G. Ribiere, and O. Vincent. Experiments in mixed–integer linear programming. *Mathematical Programming*, 1:76–94, 1971.

[6] G. R. Bitran and J. M. Rivera. A combined approach to solve binary multicriteria problems. *Naval Research Logistics Quarterly*, 29(2):181–201, 1982.

[7] M. d. G. Costa, M. E. Captivo, and J. Clímaco. Capacitated single allocation hub location problem-a bi-criteria approach. *Computers & Operations Research*, 35(11):3671–3695, 2008.

[8] Cplex. IBM ILOG Cplex 12.1 optimizer user's manual. *http://www-01.ibm.com/software/integration/optimization/cplex-optimizer*, 2010.

[9] F. Du and G. W. Evans. A bi-objective reverse logistics network analysis for post-sale service. *Computers & Operations Research*, 35(8):2617–2634, 2007.

[10] M. Ehrgott. *Multicriteria Optimization*. Springer-Verlag, Berlin, second edition, 2005.

[11] M. Ehrgott and X. Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Computers & Operations Research*, 34(9):2674–2694, 2007.

[12] M. Gardenghi, T. Gómez, F. Miguel, and M. M. Wiecek. Algebra of efficient sets for complex systems. *Journal of Optimization Theory and Applications*, 149:385–410, 2011.

[13] Y. Y. Haimes, L. Lasdon, and D. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Trans. Syst. Man. Cybern*, 1(3):296–297, 1971.

[14] A. Jaszkiewicz. A comparative study of multiple objective metaheuristics on the biobjective set covering problem and the Pareto memetic algorithm. *Annals of Operations Research*, 131:135–158, 2004.

[15] N. Jozefowiez, G. Laporte, and F. Semet. A generic branch-and-cut algorithm for multiobjective optimization problems: Application to the multilabel traveling salesman problem. *INFORMS Journal on Computing*, 24:554–564, 2012. doi: 10.1287/ijoc.1110.0476.

[16] G. Kızıltan and E. Yucaoğlu. An algorithm for multiobjective zero-one linear programming. *Management Science*, 29(12):1444–1453, 1983.

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

[18] G. Mavrotas and D. Diakoulaki. A branch and bound algorithm for mixed zero-one

multiple objective linear programming. *European Journal of Operational Research*, 107(3):530–541, 1998.

[19] G. Mavrotas and D. Diakoulaki. 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, 2005.

[20] Y.-S. Myung, H.-g. Kim, and D.-w. Tcha. A bi-objective uncapacitated facility location problem. *European Journal of Operational Research*, 100(3):608–616, 1997.

[21] B. Naderi, M. Aminnayeri, M. Piri, and H. Y. M.H. Multi-objective no-wait flowshop scheduling problems: models and algorithms. *International Journal of Production Research*, 50(10):2592–2608, 2012.

[22] Ö. Özpeynirci and M. Köksalan. An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs. *Management Science*, 56(12):2302–2315, 2010.

[23] M. S. Pishvaee, R. Z. Farahani, and W. Dullaert. A memetic algorithm for bi-objective integrated forward/reverse logistics network design. *Computers & Operations Research*, 37(6):1100–1112, 2010.

[24] S. Sayın and S. Karabatı. A bicriteria approach to the two-machine flowshop scheduling problem. *European Journal of Operational Research*, 113(2):435–449, 1999.

[25] F. Sourd and O. Spanjaard. A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem. *INFORMS Journal on Computing*, 20(3):472–484, 2008.

[26] R. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley, New York, 1986.

[27] T. Vincent, F. Seipp, S. Ruzika, A. Przybylski, and X. Gandibleux. Mavrotas and Diakoulaki's algorithm for multiobjective mixed 0-1 linear programming revisited. In *MOPGP10*, Sousse, Tunisie, May 2010.

[28] T. Vincent, F. Seipp, S. Ruzika, A. Przybylski, and X. Gandibleux. Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers & Operations Research*, 40(1):498–509, 2013.

[29] M. Visée, J. Teghem, M. Pirlot, and E. Ulungu. Two-phases method and branch and bound procedures to solve the biobjective knapsack problem. *Journal of Global Optimization*, 12:139–155, 1998.