European Journal of Operational Research 260 (2017) 856–872

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Multi-objective branch and bound

Anthony Przybylski*, Xavier Gandibleux

University of Nantes — IRCCyN UMR CNRS 6597, Faculty of Sciences and Technologies, Building 11, 2 Rue de la Houssinière BP 92208, Nantes Cedex 03 F44322, France

ARTICLE INFO

Article history:
Received 2 September 2015
Accepted 19 January 2017
Available online 25 January 2017

Keywords:
Multiple objective programming
Branch and bound
Bound sets

ABSTRACT

Branch and bound is a well-known generic method for computing an optimal solution of a single-objective optimization problem. Based on the idea “divide to conquer”, it consists in an implicit enumeration principle viewed as a tree search. Although the branch and bound was first suggested by Land and Doig (1960), the first complete algorithm introduced as a multi-objective branch and bound that we identified was proposed by Kiziltan and Yucaoglu (1983). Rather few multi-objective branch and bound algorithms have been proposed. This situation is not surprising as the contributions on the extensions of the components of branch and bound for multi-objective optimization are recent. For example, the concept of bound sets, which extends the classic notion of bounds, has been mentioned by Villarreal and Karwan (1981). But it was only developed for the first time in 2001 by Ehrgott and Gandibleux, and fully defined in 2007.

This paper describes a state-of-the-art of multi-objective branch and bound, which reviews concepts, components and published algorithms. It mainly focuses on the contributions belonging to the class of optimization problems who has received the most of attention in this context from 1983 until 2015: the linear optimization problems with zero-one variables and mixed 0–1/continuous variables. Only papers aiming to compute a complete set of efficient solutions are discussed.

1. Introduction

A *multi-objective integer linear program* (MOILP) is written as

$$\min\{(z_1(x),\dots,z_p(x)) = Cx : x \in X\}. \tag{MOILP}$$

where $p \geqq 2$ and $C \in \mathbb{R}^{p\times n}$.

X denotes the set of feasible solutions of the problem and is defined by

$$X := \{x \in \mathbb{Z}^n : Ax \leqq b, x \geqq 0\} \tag{1}$$

Here $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $C \in \mathbb{R}^{p\times n}$. The *outcome set* Y is defined by $Y := \{Cx\colon x \in X\}$. We will denote the kth objective vector by c^k for $k \in \{1,\dots,p\}$, the objective costs associated to x_j by c_j for $j \in \{1,\dots,n\}$, and c_j^k the objective costs associated to the kth objective.

By dropping the integrality requirement on a subset of variables in X, we obtain a *multi-objective mixed-integer linear program* (MOMILP). If we replace integer variables by binary variables for respectively a MOILP and a MOMILP, we obtain respectively a *multi-objective zero-one linear program* (MO01LP) and a *multi-objective mixed zero-one linear program* (MOM01LP). *Multi-objective combinatorial optimization problems* (MOCO) are a particular case of MO01LP that are characterized by a special constraint structure (Ehrgott & Gandibleux, 2000). In this class of problems, we find knapsack, assignment, spanning tree, shortest path problems, etc. Finally, we will replace the letters MO by BO in all these acronyms to make a distinction for the particular bi-objective case.

1.1. Solutions and performance vectors

We assume that no feasible solution minimizes all objectives simultaneously and use the following notation for component-wise orders in $\mathbb{R}^p$. Let $y^1, y^2 \in \mathbb{R}^p$. We write $y^1 \leqq y^2$ (y^1 *weakly dominates* y^2) if $y^1_k \leqq y^2_k$ for $k = 1,\dots,p$, $y^1 \leq y^2$ (y^1 *dominates* y^2) if $y^1 \leqq y^2$ and $y^1 \neq y^2$, and $y^1 < y^2$ (y^1 *strictly dominates* y^2) if $y^1_k < y^2_k, k = 1,\dots,p$. We define $\mathbb{R}^p_{\geqq} := \{x \in \mathbb{R}^p : x \geqq 0\}$ and analogously $\mathbb{R}^p_{\geq}$ and $\mathbb{R}^p_{>}$.

A feasible solution $\hat{x} \in X$ is called *efficient (weakly efficient)* if there does not exist $x \in X$ such that $Cx \leq C\hat{x}$ $\left(Cx < C\hat{x}\right)$. If $\hat{x}$ is (weakly) efficient, then $C\hat{x}$ is called *(weakly) nondominated*. The

* Corresponding author.
E-mail addresses: anthony.przybylski@univ-nantes.fr (A. Przybylski), xavier.gandibleux@univ-nantes.fr (X. Gandibleux).
URL: http://www.univ-nantes.fr/przybylski-a (A. Przybylski), http://www.univ-nantes.fr/gandibleux-x (X. Gandibleux)

http://dx.doi.org/10.1016/j.ejor.2017.01.032
0377-2217/© 2017 Elsevier B.V. All rights reserved.

efficient set $X_E \subseteq X$ is defined as

$$X_E := \{x \in X : \nexists\ \bar{x} \in X : C\bar{x} \leq Cx\},$$

and its image under the vector-valued linear mapping C is referred to as the *nondominated set* $Y_N := CX_E$. Equivalently, Y_N can be defined by $Y_N := \{y \in Y : (y - \mathbb{R}^p_{\geqq}) \cap Y = \{y\}\}$. This concept is extended by defining $S_N := \{s \in S : (s - \mathbb{R}^p_{\geqq}) \cap S = \{s\}\}$ for an arbitrary set $S \in \mathbb{R}^p$. In this paper, to solve a multiple objective program is understood as computing a *complete set* of efficient solutions, i.e. a subset of the feasible set containing at least one efficient solution for each nondominated point.

As we consider here problems with integer variables, several classes of efficient solutions need to be distinguished.

- *Supported* efficient solutions are optimal solutions of a weighted sum single objective problem

  $$\min\{\lambda_1 z_1(x) + \cdots + \lambda_p z_p(x) : x \in X\} \qquad (P_\lambda)$$

  for some $\lambda \in \mathbb{R}^p_>$. Their images in objective space are supported nondominated points. We use the notations X_{SE} and Y_{SN}, respectively. All supported nondominated points are located on the boundary of the convex hull of Y (convY), i.e., they are nondominated points of (convY) $+ \mathbb{R}^p_{\geqq}$.
- *Nonsupported* efficient solutions are efficient solutions that are not optimal solutions of (P_λ) for any $\lambda \in \mathbb{R}^p_>$. Nonsupported nondominated points are located in the interior of (convY) $+ \mathbb{R}^p_{\geqq}$.

In addition we can distinguish two classes of supported efficient solutions. The $x \in X_{SE}$ for which the objective vector $z(x)$ is located:

- on the vertex set of convY. They are called *extremal supported efficient solutions*, X_{SE1}, and *nondominated extreme points*, Y_{SN1}, respectively.
- in the relative interior of a face of convY. For such a solution x there exist supported efficient solutions $x^1, \ldots, x^p$ such that $z(x^i) \neq z(x^j)$ for all $i, j \in \{1, \ldots, p\}$ verifying $i \neq j$, and $\alpha \in \mathbb{R}^p_{\geq}$ (with at least two strict inequalities) such that $\sum_{i=1}^p \alpha_i = 1$ with $z(x) = \sum_{i=1}^p \alpha_i z(x^i)$. The sets of *non-extremal supported efficient solutions* and their outcome vectors are denoted by X_{SE2} and Y_{SN2}, respectively.

The main methodologies for the exact solution of multi-objective linear problems with (mixed-)integer variables are the two phase method, ε-constraint method, dynamic programming and branch and bound (see Ehrgott & Gandibleux, 2000). This paper is devoted to a review of the branch and bound methodology, restricted to MO01LP and MOM01LP. The next subsection reminds the main components of a branch and bound algorithm.

1.2. Main lines of a single-objective branch and bound

Branch and bound is a well-known generic method for computing an optimal solution of a single objective optimization problem. Based on the "divide to conquer" idea, it consists in an implicit enumeration principle, viewed as a tree search. The feasible set of the problem to optimize is iteratively partitioned to form subproblems of the original one. Each subproblem is evaluated to obtain a lower bound on the subproblem objective value. The lower bounds on subproblem objective values are used to construct a proof of optimality without exhaustive search. Uninteresting and infeasible subproblems are pruned, promising subproblems are selected and instantiated.

The strategic stages defining a branch and bound algorithm are: branching procedure; upper bound on the optimal value and its update; lower bound on the optimal value of a subproblem; choice of the subproblem to investigate.

- *Branching procedure*: In order to enumerate the feasible solutions of the problem, the feasible set X is iteratively partitioned into several subsets, defining subproblems. A subproblem is defined by its set of *fixed* variables taking value 0 or 1. The other variables are called *free* variables. By associating a node to each subproblem, we obtain the *branch and bound tree*, where the root corresponds to the initial problem, and the leaves define the current partition of the feasible set of the problem. Some of the subproblems defined by the leaves may be infeasible. There is thus nothing to enumerate in these subproblems, the corresponding node is *fathomed by infeasibility*. A branching procedure requires a *partitioning procedure* adapted to the problem structure and a *branching strategy*. For example, a basic partitioning procedure can be to fix a free variable to 0 or 1 in order to obtain a complete partition of the feasible set, and the branching strategy is next to choose a suitable variable x_j (called *branching variable*) on which to apply this partitioning procedure. There are two main kinds of branching strategy. In a *static* strategy, a permutation of the index of variables is defined before starting the enumeration and we always choose the first free variable in this permutation. In a *dynamic* strategy, the choice of the branching variable is done using available informations during the enumeration.
- *Upper bound on the optimal value*: A branch and bound algorithm stores the best known feasible solution, in order to provide an optimal solution at the end of the enumeration. The value of this solution is called *incumbent* and gives an upper bound on the optimal value of the problem. Generally, the incumbent is initialized by applying a (meta)heuristic before starting the enumeration. Next, this value is potentially updated when a new feasible solution is obtained. Generation of feasible solutions is thus a key point here. A new feasible solution can be obtained by considering the subproblem associated to any node of the branch and bound tree, and by solving it either exactly or approximately (use of a heuristic).
- *Lower bound on the optimal value of a subproblem*: When creating a node, a lower bound on objective values of all feasible solutions of the subproblem it defines is computed. This is generally done using a relaxation (linear, lagreangean, combinatorial...) that does not guarantee to obtain a feasible solution. If this happens, the incumbent may be updated. Moreover, if the objective value associated to this solution is equal to the lower bound, an optimal solution of the subproblem is obtained. Hence, it will not be necessary to enumerate any other feasible solution of this subproblem. The corresponding node is thus *fathomed by optimality*. Even if this does not happen, a further branching of the considered node can be avoided if the lower bound is larger than or equal to the incumbent. Indeed, this implies that there is no solution giving a strictly better value than the incumbent among the feasible solutions of the considered subproblem. The node is thus *fathomed by dominance*. The quality of both lower and upper bounds is a crucial point in order to fathom nodes by dominance.
- *Choice of the active node*: The leaves of the branch and bound tree that are not fathomed are called *active nodes*. We can potentially find solutions giving a better value than the incumbent by investigating these nodes, and we must therefore choose one of these nodes in order to apply the branching procedure. The order of consideration of these nodes may have an important impact on the size of the branch and bound tree. Indeed, to consider an appropriate node early in the algorithm may allow the early enumeration of a very good feasible solution that would allow next to fathom more nodes by dominance. The same way as the branching strategy, the choice of the active node can be done according to a *static* strategy (depth-first search, breadth-first search...) or a *dynamic* strategy (the most

basic is the best bound strategy that chooses the node the associated subproblem of which has the best lower bound).

As underlined in Section 1.1, the main difference between single and multi-objective optimization comes from the notion of *optimal solution* that is replaced by the notion of *complete set of efficient solutions*. Thus the expected output of a multi-objective branch and bound algorithm (MOB&B) is not any longer a single solution but a set of solutions. Consequently, the design of such an algorithm requires to reconsider its components: the notion of bounds, the branching procedure, and the choice of active node. According to the literature, the notion of bounds has received much more attention than the other components. This observation influences the structure of this paper, which is organized as follow.

Section 2 is devoted to the definitions of the notion of bounds in the multi-objective case. Section 3 describes the extension of the other components of a branch and bound to the multi-objective case. Section 4 presents the practical application of dominance tests in recent branch and bound algorithms. Section 5 introduces components of a MOB&B which are specific to the multi-objective case. In the light of Sections 2 –5, a guideline destined to a reader who is faced to the design of a MOB&B is presented in Section 6. Next, a review of published MOB&B algorithms is reported in Sections 7 and 8. For each algorithm, a brief standardized description is given, outlining the main components and the specificities when they exist. Section 9 cites briefly some others so called MOB&B found in the literature, while being out of the scope of this paper. Finally, open questions in terms of the remaining difficulties and main challenges are presented in Section 10.

2. From bounds to bound sets

For a single-objective optimization problem, there is at most one optimal value and the definition of upper and lower bound on this value is therefore natural.

Definition 1. Given the optimal value $\tilde{y}$ of a single-objective problem, lower and upper bound l and u on $\tilde{y}$ are real values verifying $l \leqq \tilde{y} \leqq u$.

For a multi-objective optimization problem, bounds are considered on the set of nondominated points Y_N. Well-known bounds are given by the ideal point y^I and the nadir point y^N that are defined respectively by

$$y^I_k := \min_{y \in Y} y_k; \quad k = 1, \ldots, p$$

$$y^N_k := \max_{y \in Y_N} y_k; \quad k = 1, \ldots, p.$$

Using only two points, the ideal and nadir points are the tightest possible bounds such that $y^I \leqq y \leqq y^N$ for all $y \in Y_N$, as depicted by Fig. 1. Hence, these points seem to be a perfect generalization of the notion of bound. Nevertheless, the general assumption that there is no common optimal solution among all objectives implies that $y^I \neq y^N$. Moreover, the number of nondominated points is generally large and this implies that y^I and y^N are generally located far away from the nondominated set. Consequently, y^I and y^N are practically very weak bounds and their use in MOB&B has therefore been inefficient.

Finally, the computation of y^I is already hard if the single-objective problem cannot be solved efficiently in practice. Also the computation of y^N is difficult for problems with three objectives and more even if the single-objective version of the problem is polynomially solvable (Ehrgott & Tenfelde-Podehl, 2003; Kirlik & Sayin, 2015; Köksalan & Lokman, 2015).

Rather than using a unique point as a bound, the use of particular sets of points to bound Y_N more tightly has been proposed. In the following, the term *bound set* (Ehrgott & Gandibleux, 2001)

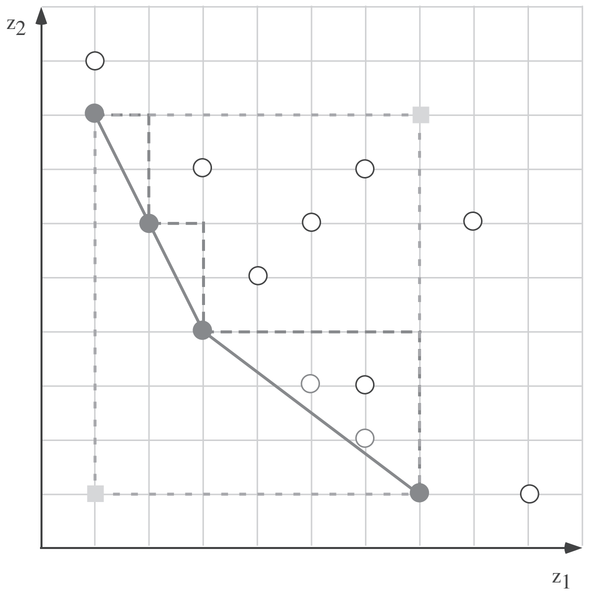

Fig. 1. Illustration of some bound sets: ideal and nadir points (denoted by squares), convex relaxation (denoted by a dark grey line), and corner points deduces from Y_{SN} (endpoints of doted lines).

which is now commonly accepted to describe such sets of points, is used. Note that the idea of handling a collection of points as a bound is found implicitly in many enumerative methods for multi-objective optimization published several years ago. This is the case for example when the set of all known feasible points are used in a dominance test. In the coming subsection, we give a discussion about the properties of bounds that should be preserved in a natural and correct definition of bound sets, and we review the most commonly used bound sets. Next, we present the main formal definitions of bound sets that have been proposed explicitly by authors in the literature, for a general application in enumerative algorithms. These definitions are not necessarily equivalent. Moreover, the definition of lower and upper bound sets are not necessarily symmetric. In order to highlight the differences between these definitions, all are presented in the minimization case. Finally, note that the notion of bound set has evolved independently into several methodologies, and have thus not been necessarily proposed initially for an application in a MOB&B. However, these propositions remain valid for a MOB&B and are thus of interest.

2.1. Desirable properties and most common bound sets

Starting from the definition of a lower bound, a basic definition of a lower bound set L for Y_N should verify at least the following condition: for all $y \in Y_N$, there is $l \in L$ such that $l \leqq y$. In other words, $Y_N \subset L + \mathbb{R}^p_{\geqq}$. All formal definitions of lower bound set include this condition. Next, restrictive conditions are added in order to exclude (ideally) only sets of points that do not define a "correct" bound set. Here, correctness is related to the fact that the purpose of a bound set is to be composed of several points, some of which could be unnecessary (for example, points dominated by other points of the bound set, or points dominated by feasible points of the problem).

Generally, lower bound sets are obtained by solving exactly a relaxation of the problem. Their determination is therefore a natural extension of the single-objective case. However, a relaxation of a multi-objective problem remains a multi-objective problem, and can thus be difficult to solve. This is why for almost all particular lower bound sets L proposed in the literature, $L + \mathbb{R}^p_{\geqq}$

is convex. We can note that this is also true for the ideal point. Among these particular lower bound sets, the most used are the *convex relaxation* and the linear relaxation. The convex relaxation is defined by $\min\{z(x) : x \in \text{conv}X\}$, and its nondominated set is given by $(\text{conv}Y_{SN1})_N$. Its computation can be done using a dichotomic scheme (see Aneja & Nair, 1979; Cohon, 1978 for the bi-objective case and Przybylski, Gandibleux, and Ehrgott (2010a), Özpeynirci and Köksalan (2010) for the multi-objective case). It requires to be able to solve the single-objective version of the problem in reasonable time. Hence, it is mainly used for problems the single-objective version of which belongs to $\mathcal{P}$. In other cases, the linear relaxation has been often used.

Finally, the multi-objective version of some other relaxations that are well-known in the single-objective case, has been used in (old) published branch and bound algorithms to determine only their ideal point. Given a relaxation $(\tilde{P})$ of a multi-objective problem (P), Ehrgott and Gandibleux (2001) have proposed to consider the weighted sum problem $(\tilde{P}_\lambda)$ for any $\lambda \in \mathbb{R}^p_>$. This last problem is indeed a relaxation of the single-objective problem (P_λ) and can thus be solved directly using single-objective methods. Next, using a set of weights $\{\lambda^1, \dots, \lambda^k\}$ and computing an optimal solution $\tilde{x}^\lambda$ of $(\tilde{P}_\lambda)$, we obtain $Y_N \subset \bigcap_{i=1}^k (\lambda^{i^T} z(\tilde{x}^{\lambda^i}) + \mathbb{R}^p_{\geqq})$. Thus, $(\bigcap_{i=1}^k (\lambda^{i^T} z(\tilde{x}^{\lambda^i}) + \mathbb{R}^p_{\geqq}))_N$ defines a lower bound set for Y_N. It is of course possible to obtain better lower bound sets based on the use of relaxations. However, very few thorough studies are available in the literature. A main difficulty comes from the use of parameters (generally called *multipliers*) in some of these relaxations. Given a problem (P) and a multipliers vector μ, a relaxation $(\tilde{P}(\mu))$ is obtained. In the single-objective case, a lower bound $l(\mu)$ is obtained by its solution. As the quality of the lower bound $l(\mu)$ relies on the choice of μ, the problem $l^* := \max_\mu l(\mu)$ can be solved (approximately or exactly) to obtain a tight bound. The final value of the bound is defined by one choice for the multipliers vector μ. The extension of such a relaxation to the multi-objective case is immediate in its definition. It is possible to obtain a lower bound set $L(\mu)$ by solving a relaxation $\tilde{P}(\mu)$. Nevertheless, to solve the problem $L^* := \max_\mu L(\mu)$ becomes far more difficult, as several multipliers vector μ are necessary to define the lower bound set L^*, and the time required to solve such a problem with a guarantee of quality is significantly large given the current state of knowledge. Such a study has been done recently for the surrogate relaxation applied to bi-objective bi-dimensional knapsack problem (Cerqueus, Przybylski, & Gandibleux, 2015b).

In order to define upper bound sets, two families of definitions have been proposed following two different points of view. Starting from the single-objective definition of an upper bound, a basic definition of an upper bound set U for Y_N should verify at least the following condition: for all $y \in Y_N$, there is $u \in U$ such that $y \leqq u$. However, this condition is not included in every formal definition of upper bound set.

Practically, an upper bound is just defined by the incumbent, at any step of the application of a single-objective branch and bound algorithm. The extension of the notion of incumbent to the multi-objective case is straightforward. Any algorithm for the solution of a multi-objective problem maintains a list of known feasible points filtered by dominance, that we will denote by *incumbent list* in the following. It is interesting to note that the incumbent list U does not necessarily verify the condition: for all $y \in Y_N$, there is $u \in U$ such that $y \leqq u$. This is illustrated using the example $U = Y_{SN}$ in Fig. 1.

Next, we can deduce from Y_{SN}, a set of *corner points* that delimit the area that is not dominated by Y_{SN}. More generally, any subset U of feasible points filtered by dominance allows to deduce such a set $D(U)$ of corner points. In the particular bi-objective case, these corner points are usually called *local nadir points*: given two feasible consecutive points y^1 and y^2 with respect to z^1, their local nadir point is defined by (y^2_1, y^1_2). However, the term local nadir point is not appropriate in the multi-objective case as this set of points is not defined locally (see Klamroth, Lacour, & Vanderpooten, 2015; Przybylski, Gandibleux, & Ehrgott, 2010b for its computation). Finally, the set of corner points gives the best possible set U (using all known feasible points) satisfying the condition: for all $y \in Y_N$, there is $u \in U$ such that $y \leqq u$. Hence, we ends with two natural ways to define upper bound sets.

Finally, a strong property of bounds on $\tilde{y}$ in the single-objective case is that if the lower bound l is equal to the upper bound u, then both are equal to $\tilde{y}$. The natural extension of this property for a lower bound set L and an upper bound set U for Y_N is that, $L \cap U \subseteq Y_N$ and if $L = U$ then $L = U = Y_N$. This property is verified by the incumbent list but not by the set of corner points. Hence, the incumbent list appears to be a more natural extension of the notion of upper bound, if we wish to extend all single-objective properties. However, corner points will remain useful in order to test if nodes should be fathomed, as it will be developed in Section 4.

2.2. Main formal definitions of bound sets

To our knowledge, the first authors that have formally defined the notion of bound sets are Villarreal and Karwan (1981), for a dynamic programming algorithm to solve MOILP. However, their definition of lower bound set is very restrictive and excludes for example the linear and the convex relaxations, and more generally all continuous lower bound sets. This definition is therefore no longer used.

The term bound sets and a related formal definition have been proposed in Ehrgott and Gandibleux (2001). In this definition, lower and upper bound sets are proposed to bound a subset $\bar{Y} \subset Y$ of feasible points. In the context of a branch and bound, $\bar{Y}$ should be a subset of nondominated points of the problem, or of one subproblem defined by a node of the branch and bound tree. This definition has been initially proposed for MOCO problems but can be applied to any MOMILP or MOILP.

Definition 2 (Bound sets according to Ehrgott and Gandibleux, 2001)**.** A lower bound set for $\bar{Y}$ is a subset $L \subseteq \mathbb{R}^p$ such that

(i) For each $y \in \bar{Y}$ there is some $l \in L$ such that $l \leqq y$,
(ii) There is no pair $y \in \bar{Y}$, $l \in L$ such that y dominates l.

An upper bound set for $\bar{Y}$ is a subset $U \subset \mathbb{R}^p$ such that

(iii) For each $y \in \bar{Y}$ there is some $u \in U$ such that $y \leqq u$,
(iv) There is no pair $y \in \bar{Y}$, $u \in U$ such that u dominates y.

All particular lower bound sets (for Y_N) described in the preceding subsection verify items (i) and (ii) of Definition 2. Indeed, no point of these lower bound sets is dominated by a nondominated point. The incumbent list U is generally not an upper bound set (for Y_N) according to this definition. However, the set of corner points $D(U)$ that is deduced from U is a valid upper bound set for Y_N according to Ehrgott and Gandibleux (2001).

Simplified versions of Definition 2 are sometimes stated by some authors to give a formalization for the particular bound sets that they use next in the solution method they propose. For example, a definition containing only items (i) and (iii) of Definition 2 is stated in Delort and Spanjaard (2010), Rong and Figueira (2014)

Ehrgott and Gandibleux (2007) have refined the main proposals discussed in Ehrgott and Gandibleux (2001), with a different definition of bound sets, theoretical results on their computation, and comparison of bound sets. We introduce some additional terminology before the next definition. A set $S \subset \mathbb{R}^p_{\geqq}$ is called $\mathbb{R}^p_{\geqq}$-closed if the set $S + \mathbb{R}^p_{\geqq}$ is closed, and $\mathbb{R}^p_{\geqq}$-bounded is there exists $s_0 \in \mathbb{R}^p$ such that $S \subset s_0 + \mathbb{R}^p_{\geqq}$.

Definition 3 (Bound sets according to Ehrgott & Gandibleux, 2007)**.** A lower bound set L for $\bar{Y}$ is an $\mathbb{R}^p_{\geq}$-closed and $\mathbb{R}^p_{\geqq}$-bounded set $L \subset \mathbb{R}^p$ such that $\bar{Y} \subset L + \mathbb{R}^p_{\geqq}$ and $L \subset (L + \mathbb{R}^p_{\geq})_N$.

An upper bound set U for $\bar{Y}$ is an $\mathbb{R}^p_{\geq}$-closed and $\mathbb{R}^p_{\geqq}$-bounded set $U \subset \mathbb{R}^p$ such that $\bar{Y} \subset cl[(U + \mathbb{R}^p_{\geq})^c]$ and $U \subset (U + \mathbb{R}^p_{\geq})_N$.

In this last definition, $L \subset (L + \mathbb{R}^p_{\geq})_N$ and $U \subset (U + \mathbb{R}^p_{\geq})_N$ means that in (upper and lower) bound sets, there is no point that dominates another one. This idea quite natural was not formalized in the preceding definitions, but applied to any particular proposed bound sets. $\bar{Y} \subset L + \mathbb{R}^p_{\geqq}$ is equivalent to item (i) of Definition 2 and implies item (ii) (with $L \subset (L + \mathbb{R}^p_{\geq})_N$). However, the definitions of upper bound set on $\bar{Y}$ according to respectively Ehrgott and Gandibleux (2001) and Ehrgott and Gandibleux (2007) are not equivalent, as this later definition is motivated by the natural idea that the incumbent list should be an upper bound set for Y_N. Using such a particular upper bound set U for Y_N, we deduce that no nondominated point is included in $U + \mathbb{R}^p_{\geq}$, in other words that $Y_N \subset (U + \mathbb{R}^p_{\geq})^c$. Finally, in order to obtain a $\mathbb{R}^p_{\geqq}$-closed set, the closure $cl[(U + \mathbb{R}^p_{\geq})^c] = cl[(U + \mathbb{R}^p_{\geqq})^c]$ is considered. This slight modification in the definition implies that considering a general upper bound set U on Y_N according to Definition 3, there is no nondominated point that is strictly dominated by points in U. This is just a minor difference as the area containing Y_N defined by dominance using a set of feasible points has the same interior and closure. Finally, equality of lower and upper bound sets L and U for Y_N according to Ehrgott and Gandibleux (2007) implies that $L = U = Y_N$. Obviously, we cannot expect to obtain this equality using a connected lower bound set L on Y_N (except if $y^I = y^N$).

Several notions and properties about bound sets have been proposed in (Ehrgott & Gandibleux, 2007). We will be next interested by the notion of dominance between bound sets.

Definition 4 (Ehrgott & Gandibleux, 2007)**.** Let L_1, L_2 be two lower bound sets for $\bar{Y}$. We say that L_1 *dominates* L_2 if $L_1 \subset L_2 + \mathbb{R}^p_{\geqq}$ and $L_1 + \mathbb{R}^p_{\geqq} \neq L_2 + \mathbb{R}^p_{\geqq}$.

This notion generalizes the notion of tightness of a lower bound in the single-objective case. However, a main difference is that dominance does not always occur between two different lower bound sets on Y_N. Ehrgott and Gandibleux (2007) have therefore proposed to merge several lower bound sets. To our knowledge, this last result has not been yet applied to any branch and bound algorithms of the literature.

In the rest of the paper, all considered bound sets (for any sets $\bar{Y}$) will implicitly be based on Definition 3.

3. Architecture of a multi-objective branch and bound

Now that the notion of bound sets is stated, this section presents the extension of the other components of branch and bound algorithms to the multi-objective case, and on which the developments have been far more scarce in the literature.

3.1. Branching procedure

The branching procedure concerns only the feasible set of the problem. Consequently, there is no difference here in the partitioning procedure between the single and the multi-objective case. In particular, nodes of the branch and bound tree can be fathomed by infeasibility in the same way as the single-objective case. However, the strategies used in the single-objective case for the branching strategy are not transposed so easily to the multi-objective case. Difficulties appear even with the most basic static strategies. To illustrate this, we consider the well known uni-dimensional knapsack problem with binary variables defined by the following model,

$$\begin{aligned} \max z_k(x) \quad &= \quad \sum_{j=1}^{n} c_j^k x_j \qquad k = 1, \ldots, p \\ &\quad\;\; \sum_{j=1}^{n} w_j x_j \leqq \omega \\ &\quad\;\; x_j \in \{0, 1\} \end{aligned}$$

where c_j^k and w_j are non-negative integer numbers. For this problem, several good solution methods for the single-objective case start by sorting the items by decreasing order of the ratio c_j^1/w_j (Kellerer, Pferschy, & Pisinger, 2004). Following this order, we consider first the items that are individually the most promising. A common static strategy for this problem is thus to branch on the first free variable according to this order.

The extension of this static strategy to the multi-objective case can be done in many different ways. Indeed, we can consider the same ratio according to the objective k for any $k \in \{1, \ldots, p\}$ and the obtained orders $\mathcal{O}_k$ may vary considerably since we make the assumption that the objectives are conflicting. It seems thus important to consider all objectives simultaneously to define such an order. Several propositions have been done.

- Ulungu and Teghem (1997) have proposed to consider the rank r_j^k of each item j according to $\mathcal{O}_k$ for $k \in \{1, \ldots, p\}$, and have next proposed the order $\mathcal{O}^{sumrank}$ obtained by sorting the items by decreasing value of the sum of the ranks $\sum_{k=1,\ldots,p} r_j^k$.
- Bazgan, Hugot, and Vanderpooten (2009) have proposed the order $\mathcal{O}_{sum}$ obtained by sorting by decreasing order the sum of the ratio $\sum_{k=1,\ldots,p} c_j^k/\omega_j$. Next, Bazgan et al. (2009) have proposed the order $\mathcal{O}_{max}$ and $\mathcal{O}_{min}$ that sort the items by increasing value of respectively the worst rank $\max_{k=1,\ldots,p} r_j^k$ and the best rank $\min_{k=1,\ldots,p} r_j^k$, and next in case of equality following the order $\mathcal{O}_{sum}$.
- Jorge (2010) has proposed to consider the dominance between the vector of ratios $(c_j^1/w_j, \ldots, c_j^p/w_j)$. Clearly, if $(c_i^1/w_i, \ldots, c_i^p/w_i) \geq (c_j^1/w_j, \ldots, c_j^p/w_j)$ then item i should be considered before item j. The order $\mathcal{O}_{dom}$ applies this principle, with many possible permutations as there are generally incomparable vector of ratios. Next (Jorge, 2010) has proposed to count for each item j the number of items $|dom(j)|$ dominating it, the order $\mathcal{O}_{\#}$ consists in sorting the items j by increasing value of $|dom(i)|$. Finally, Jorge (2010) has proposed the order $\mathcal{O}_{rg}$ consisting in sorting the items by their increasing rank following the valuation principle proposed by Goldberg (1989) and implemented in several Evolutionary Multi-Objective algorithms (Deb, 2001) .

In most of the published papers, a static strategy is used for the choice of the branching variable. Generally, only one order is defined and is rarely justified. Consequently, there is a general lack of comparisons between static strategies. Jorge (2010) has applied all orders described above (except $\mathcal{O}^{sumrank}$) in his branch and bound algorithm for three-objective uni-dimensional knapsack problem. His experiments show that, if the choice of order may have a considerable impact, the order allowing the shortest computational time is variable even for instances from a same series.

The case of a dynamic branching strategy is obviously more complex, as it arises already difficult questions in the single-objective case. In this context, these strategies are generally guided by the unique solution returned by a given relaxation. For example if we use the linear relaxation of the problem, a possible strategy is to choose to branch on the most fractional variable. In the multi-objective case, the computation of a lower bound set generally returns several solutions and these solutions can be very different due to the conflicting nature of the objectives. Hence, there are several ways to transpose this kind of dynamic strategy to the multi-objective case. Belotti, Soylu, and Wiecek (2013) have proposed to choose the most fractional variable among the solutions obtained in the computation of the linear relaxation (Belotti, 2014). Some other very specific dynamic strategies have been proposed and are thus not detailed here (see Sections 7 and 8). We can note that there is no available comparison between dynamic and static strategies for the choice of the branching variable.

3.2. Choice of the active node

There is no modification of the static strategies for the choice of the active node. The depth-first and breadth-first search strategies can be directly applied. Most of the published branch and bound algorithms simply follow a depth-first search strategy. However, the application of dynamic strategies to the multi-objective case arises again many questions. For example, the application of the best bound strategy is not straight-forward. Indeed, lower bound sets are generally composed of several points and thus there is generally no lower bound set that dominates (Definition 4, Page 10) all other ones, and thus several ways to define a best lower bound strategy could be proposed. Among the very rare dynamic strategies that have been applied, Stidsen, Andersen, and Dammann (2014) have proposed to rely on a MIP solver (ILOG CPLEX 12.1.1) for the choice of the active node (Belotti et al., 2013; Stidsen, 2014) have chosen to sum integer infeasibility for all variables in all solutions computed in the linear relaxation and to choose the active node with maximal number of integer infeasibility (Belotti, 2014). Again, there is no available comparison between dynamic and static strategies for the choice of the active node in published algorithms. It is therefore difficult to claim that one strategy should be preferred in a given context.

3.3. Fathoming nodes using bound sets in a multi-objective branch and bound algorithm

All branch and bound algorithms of the literature maintains a set of feasible points filtered by dominance, that is finally equal to Y_N at the end of the enumeration. This upper bound set on Y_N is initialized with the unique (infeasible) point $\{(\infty, \ldots, \infty)\}$ or better using a (meta)heuristic to obtain an initialization with a set of feasible points filtered by dominance. This upper bound set will be possibly updated anytime a new feasible point is known. The generation of (good) feasible points is therefore a key component in a multi-objective branch and bound as it is in the single-objective case. The same way as the single-objective case, a direct possibility is to apply a heuristic to the subproblem associated to any active node. Another possibility comes with the computation of the lower bound set, that may generate feasible points.

In all published branch and bound algorithms of the literature, for each considered node in the branch and bound tree, a lower bound set L on the set of nondominated points $\bar{Y}_N$ of the subproblem it defines is computed. This lower bound set can be used to fathom nodes. We consider first the possibility to fathom nodes by optimality. This fathoming test is straightforward in the single-objective case, as it is simply done by testing if the unique solution returned by the relaxation is feasible, and if its objective value is equal to the lower bound value. A main difference appears immediately in the multi-objective case. Indeed, several solutions are generally computed in the determination of the lower bound set on $\bar{Y}_N$. The same way as the single-objective case, we can conclude that a feasible solution which corresponding point belongs to the lower bound set, also belongs to $\bar{Y}_N$. Nevertheless, this does not allow to claim that $\bar{Y}_N$ has been found. In order to fathom a node by optimality, it would be necessary not only to find the set $\bar{Y}_N$ but also to prove that this set is found. A way to perform this proof would be to compute an upper bound set U on $\bar{Y}_N$ and to obtain the equality between U and L. Nevertheless for most of the lower bound sets L used in the literature, L is connected and it is therefore not possible to fathom a node by optimality. The only exception (that does not require the computation of an upper bound set on $\bar{Y}_N$) happens when a lower bound set is composed of a unique ideal feasible point.

To fathom a node by dominance, the lower bound set L on $\bar{Y}_N$ must be compared with the current upper bound set U on Y_N. If for all $l \in L$, there is $u \in U$ such that $u \leqq l$ then all feasible points of the considered subproblem are weakly dominated by feasible points in U. Consequently, no point in $\bar{Y}_N$ would allow an improvement of the upper bound set U on Y_N and the corresponding node can thus be fathomed by dominance. Hence, this dominance test does not consist in a comparison between two scalars (as in the single-objective case), and is therefore no longer a basic step. We describe the main ways to apply practically this dominance test in the next section.

4. Pruning by dominance: practical application

The dominance test for fathoming nodes is not a straightforward operation. This section reports several propositions found in the literature for the practical application of this test, with a considerable impact on the performances. We suppose first that we consider MO01LP (U is therefore a finite set of points) before to consider the case of MOM01LP.

4.1. Practical application of the dominance test in the case of MO01LP

In the case where L is also a finite set of points, we can apply a classical pairwise comparison between the points of L and U to test the weak dominance of points in L. This can be applied if L is composed of one ideal point for example. However, this pairwise comparison is not possible if L is a connected lower bound set composed of several points, like the convex or the linear relaxation.

In the following, we consider lower bound sets L such that $L + \mathbb{R}^p_{\geqq}$ is convex, and more precisely such that $L + \mathbb{R}^p_{\geqq}$ is a convex polyhedra. In this case, L is in particular composed of edges. One key point in the weak dominance test of a lower bound set, is to test its weak dominance during its computation. This has been done first for bound sets composed of one edge. Indeed, Visée, Teghem, Pirlot, and Ulungu (1998) have noticed that if an ideal point is sufficient to fathom a node, there is no need to determine a tighter lower bound set. Next, they have proposed the first use of a lower bound set L composed of an edge (with a different terminology at that time) in the specific bi-objective case. They have observed that this edge is weakly dominated by points in U if and only if no corner point is dominated by points of L. The only corner points that may be dominated by the lower bound set are the one that were dominated by the ideal point. Finally, the node can be fathomed by dominance if $\lambda^T u \leqq \lambda^T l$ for all such corner point u, where $\lambda \in \mathbb{R}^2_{>}$ is the normal to the edge and l is any point of L ($\lambda^T l$ has the same value for any $l \in L$).

Next, Sourd and Spanjaard (2008) have proposed a theoretical formalization to test the weak dominance of more general lower bound sets, with a particular application to lower bound sets L such that $L + \mathbb{R}^p_{\geqq}$ is a convex polyhedra. Various computational improvements have been proposed and used in methods proposed later (Delort, 2011; Jorge, 2010). It is important to highlight that the method that Sourd and Spanjaard (2008) have proposed for the solution of bi-objective minimum weight spanning tree

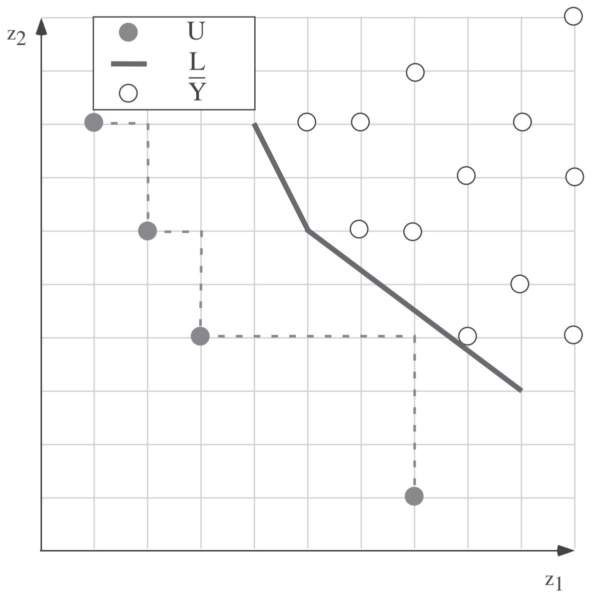

Fig. 2. The node can be fathomed by dominance.

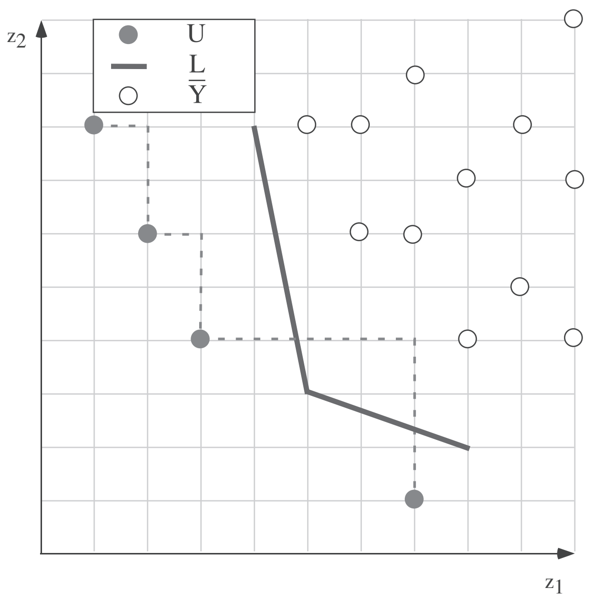

Fig. 3. The node cannot be fathomed by dominance with the considered lower bound set on $\bar{Y}_N$.

problem, has been the first published multi-objective branch and bound algorithm that has been competitive with other methodologies on a same problem. The ideas proposed to perform this weak dominance test have been the foundations (with the use of the convex relaxation) to design competitive multi-objective branch and bound algorithms for problems the single-objective version of which belongs to $\mathcal{P}$.

The initial idea is an extension of the propositions by Visée et al. (1998): all points in L are weakly dominated by points in U if and only if all corner points $D(U)$ are not dominated by any points in L (Figs. 2 and 3). Sourd and Spanjaard (2008) have shown that this last condition is verified if there is a function $h : \mathbb{R}^p \to \mathbb{R}$ strictly monotone with respect to strict dominance (i.e. for $y^1, y^2 \in \mathbb{R}^p$, $y^1 > y^2 \Rightarrow h(y^1) > h(y^2)$) such that $h(l) \geqq 0$ for all $l \in L$ and $h(u) \leqq 0$ for all $u \in D(U)$. This implies in particular that $h(y) \geqq 0$ for all $y \in \bar{Y}$. The level curve $\{y \in \mathbb{R}^p \mid h(y) = 0\}$ plays the role of *separating hypersurface* between $\bar{Y}$ and $D(U)$. Intuitively, we have for $y \in \mathbb{R}^p$ that $h(y) > 0$ if y is located "above" the level curve, $h(y) = 0$ if y belongs to the level curve and $h(y) < 0$ if y is located "below" the level curve. The function h must of course be constructed in function of the used lower bound set L.

These authors have proposed a family of functions h for a practical application to the particular case for which $L + \mathbb{R}^p_{\geqq}$ is a convex polyhedra. It can therefore be applied on most of the proposed lower bound sets (like the convex or the linear relaxation). However, despite a presentation in the general multi-objective case, we should note as often that the provided description is complete and correct only for the bi-objective case. This is due to the fact that the considered lower bound sets can be fully described by a set of extreme points and a set of maximal faces (i.e. faces not included into another one) of dimension $1, \dots, p-1$. Hence, its maximal faces are facets (edges) only if $p = 2$.

It is supposed that the lower bound set L is computed by the solution of weighted sum scalarizations (for example using a dichotomic scheme) of either the problem or one relaxation. Suppose that during this computation, a subset of points $\{l_1, \dots, l_k\}$ (note: it is possible that $l_i = l_j$ for $i \neq j$), a subset of weights $\Lambda := \{\lambda_1, \dots, \lambda_k\}$ are known, and we have for all $i \in \{1, \dots, k\}$, $\lambda^{i^T} l^i = \min_{j=1,\dots,k} \lambda^{i^T} l^j$. A lower bound set $L' := (\bigcap_{i=1,\dots,k} (\lambda^{i^T} l^i + \mathbb{R}^p_{\geqq}))_N$ is therefore known, and this lower bound set is dominated by L. Sourd and Spanjaard (2008) have proposed to test if L' is enough to fathom a node in order to avoid a useless complete computation of L. The function $h_\Lambda : \mathbb{R}^p \to \mathbb{R}$ is defined by

$$h_\Lambda(y) := \min_{\lambda \in \Lambda} \{\lambda^T y - \min_{l \in L'} \lambda^T l\}.$$

By construction, we have $h_\Lambda(l) = 0$ for all $l \in L'$ and it remains thus to check that $h_\Lambda(u) \leq 0$ for all $u \in D(U)$ in order to fathom the corresponding node by dominance. If there is $u \in D(U)$ such that $h_\Lambda(u) > 0$ and if $L = L'$, we conclude that the node cannot be fathomed by dominance. If there is $u \in D(U)$ such that $h_\Lambda(u) > 0$ and if $L \neq L'$, the computation of L continues using a new weight in order to update L', Λ and therefore h_Λ. In this update, the values of h_Λ can only decrease. Hence, it is only necessary to consider $u \in D(U)$ for which we had previously $h_\Lambda(u) > 0$.

However, the conclusion $L = L'$ can be achieved this way only in the bi-objective case, and then $\Lambda := \{\lambda \in \mathbb{R}^2_> \mid \lambda \text{ is the normal to an edge of } L'\}$. If the number of objectives p is greater than or equal to 3, then L is composed of faces of maximal dimension $1, \dots, p-1$. L' is only composed of maximal faces of dimension $p-1$. Even if L and L' have the same set of extreme points, the function h_Λ as defined above is constructed using only facets of L'. As a consequence, even if L is completely computed, such a function h_Λ may be unable to fathom a node that should be fathomed.

Finally, Sourd and Spanjaard (2008) have proposed to apply some reoptimizations from the father node to a child node of the branch and bound tree. First, some of the solutions corresponding to the extreme points of the lower bound set L_f of the father node generally remain feasible for the child nodes, and thus the corresponding points belong to the lower bound set L_c of the child node. It is therefore useless to determine again these points. Second, the lower bound set L_f on the nondominated points of the father node is necessarily dominated (see Definition 4, Page 10) by the lower bound set L_c on the nondominated points of the child node. This implies that the corner points $u \in D(U)$ that are not weakly dominated by points of L_f, are also not weakly dominated by points of L_c. For all such corner points $u \in D(U)$, it will therefore not be necessary to test that $h_\Lambda(u) \leq 0$ using the corresponding functions h_Λ of any of the child nodes.

Sourd and Spanjaard (2008) have also proposed to improve this dominance test in the case of a MO01LP with integer cost

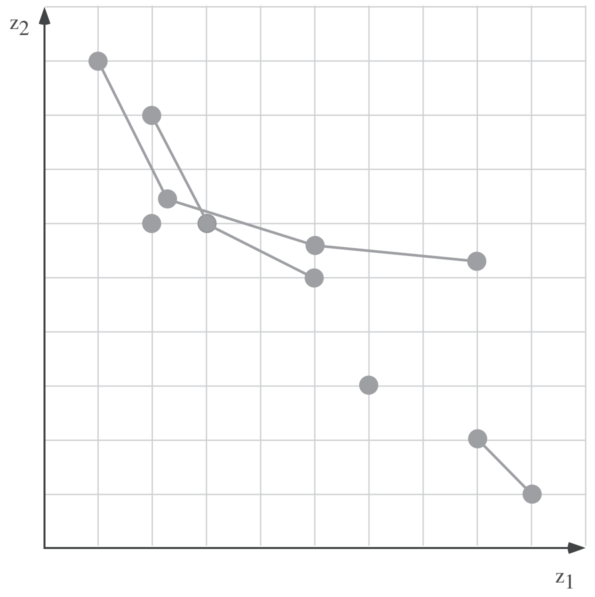

Fig. 4. Illustration of the nondominated sets of some BOLP.

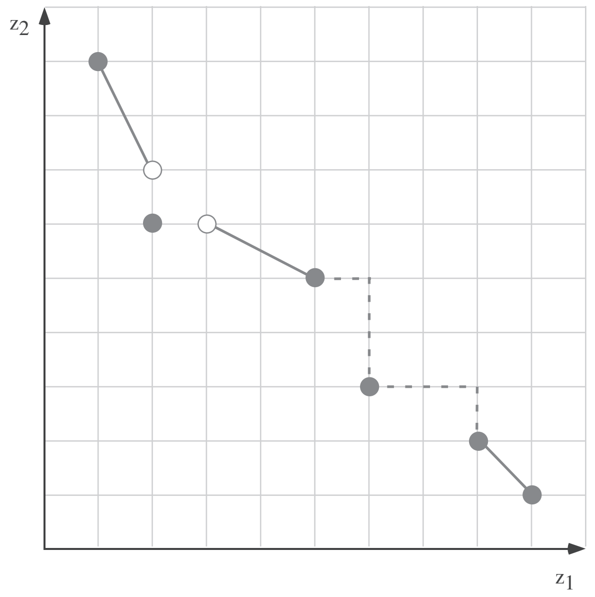

Fig. 5. After a filtering by dominance, it remains isolated points and edges.

coefficients, exploiting the fact that the feasible points of the problem belong to $\mathbb{Z}^p$.

4.2. Practical application of the dominance test in the case of BOM01LP

We consider now the case of BOM01LP. The main difference is that here the incumbent list is no longer a discrete set of points.

Theorem 1 (Vincent et al. (2013)). *The nondominated set of a BOM01LP is composed of edges (that can be closed, half-open, open or reduced to a point).*

A thorough update procedure is necessary to maintain only feasible points that are not dominated during the course of any enumerated algorithms (see Figs. 4 and 5). Comparison between edges are in particular necessary. We refer to Vincent et al. (2013) for a detailed description of this update procedure. Belotti et al. (2013) noticed that the nondominated set of a BOMILP shares the same characteristics as a BOM01LP. Vincent et al. (2013) and Belotti et al. (2013) have proposed a different specific way to compare the incumbent list of a BOM01LP/BOMILP to a lower bound set defined by the linear relaxation of a subproblem defined by a node of the branch and bound tree. It is interesting to note that the ideas they have proposed can also be applied to the particular case of BO01LP.

A bi-objective linear program can be solved using the parametric simplex algorithm (see Ehrgott, 2005). This algorithm generates the extreme points of the linear relaxation by increasing value of the first objective. Hence, edges of the linear relaxation are also obtained by increasing value of the first objective. Contrary to the dichotomic scheme that could also be applied, edges of the linear relaxation are immediately known during the execution of the parametric simplex algorithm. Vincent et al. (2013) have noticed that comparing one edge of the linear relaxation to the incumbent list, there are three possible cases: either all its points are weakly dominated, or none of its points are weakly dominated, or some points are weakly dominated and some are not. This can be easily checked using the update procedure proposed by Vincent et al. (2013). In the last two cases, the corresponding node cannot be fathomed. Consequently, as soon as it is known that one edge of the linear relaxation is not completely weakly dominated, it becomes useless to determine the other edges. Consequently, the linear relaxation is entirely computed only if the node can be fathomed.

The propositions of Belotti et al. (2013) can be seen as an extension of the propositions of Sourd and Spanjaard (2008) to the context of BOMILP with a different formulation, and with a restriction to the use of the linear relaxation as a lower bound set. First, the ideal point $\tilde{y}^l$ of the linear relaxation is used to fathom possibly the considered node. If it is not possible, it is necessary to test if the linear relaxation is weakly dominated by points and edges of the incumbent list, and only the subset of points and edges that are not dominated by $\tilde{y}^l$ are required to perform this comparison. Belotti et al. (2013) have shown that it is equivalent to test if for each local nadir point y' deduced from this reduced incumbent list, and for each edge $[y', y'']$ of this same list, there exists a weight $\lambda \in \mathbb{R}^2_{\geqq}$ such that $\min_{y \in L} \lambda^T y \geqq \lambda^T y'$ in the case of a local nadir point and also $\min_{y \in L} \lambda^T y \geqq \lambda^T y''$ in the case of an edge. Belotti et al. (2013) have shown that, for each local nadir point and for each edge, the set of weights $\lambda \in \mathbb{R}^2_{\geqq}$ and of points y of L that verifies this condition is defined by a set of linear constraints. Hence, feasibility problems are obtained and can be solved as linear programs (we refer to Belotti et al., 2013 for technical details). The linear relaxation of the bi-objective problem is therefore not explicitly computed. If one feasibility problem (defined either by a local nadir point or an edge) has no solution, the node cannot be fathomed and the other feasibility problems do not need to be considered. Hence, the same way as Sourd and Spanjaard (2008) (and contrary to Vincent et al. (2013), the computational cost of the weak dominance test of the lower bound set is maximal only if the node cannot be fathomed.

5. Global branch and bound, local branch and bound, preprocessing

In spite of all the ideas proposed in the literature to extend each component of a branch and bound algorithm to the multi-objective case, the number of nodes that can be fathomed by dominance remains limited. Indeed, if only a minor portion of the lower bound set on the nondominated points of a subproblem is not weakly dominated, the node cannot be fathomed. Moreover, it happens practically very often. To have the possibility to fathom

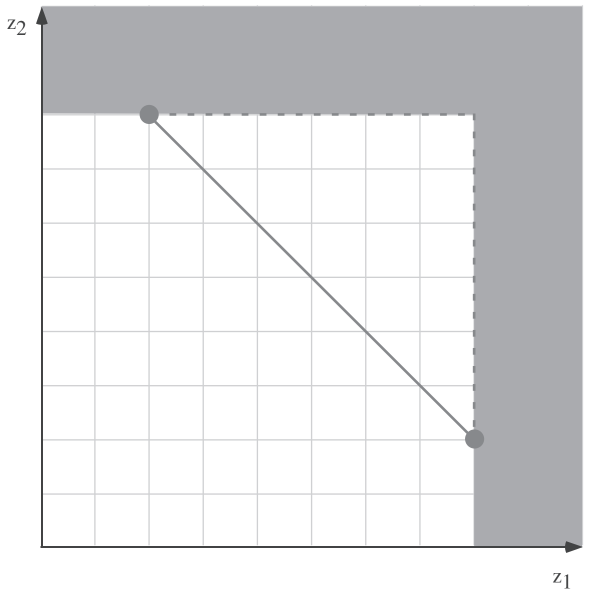

Fig. 6. Rectangle containing the triangle to explore and defined using its corner point.

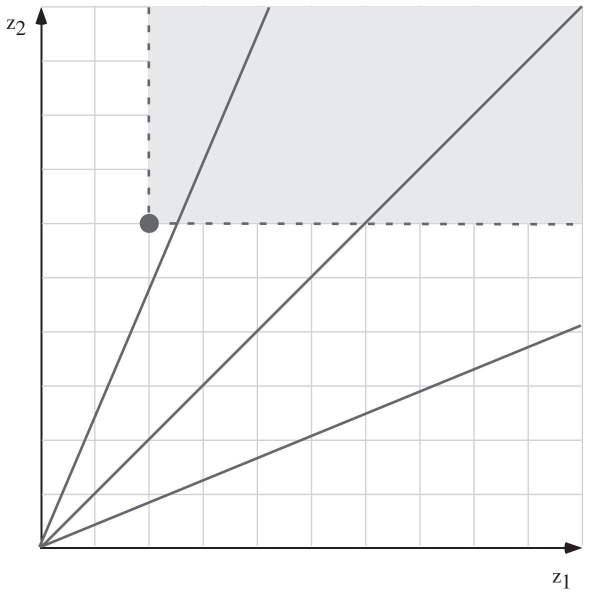

Fig. 7. Illustration of the partitioning of objective space using slicing, we can observe that the dominance cone of one point is generally not limited to one partition.

more nodes, it has been proposed to divide the exploration with a point of view seen in objective space.

The first proposition for this has been to embed a branch and bound algorithm in a two phase method. It has been initially proposed by Visée et al. (1998) for a specific application to the bi-objective uni-dimensional knapsack problem. As the two phase method can be applied to more than two objectives (see Przybylski et al., 2010b), this idea is not limited to the bi-objective context. However, its application with branch and bound algorithms has been done only on bi-objective problems in the literature. In the first phase of the method, a set Y_{SN1} is determined. In the second phase of the method, the nondominated points that remain to find are computed using problem-dedicated enumerative techniques. For this, a search area containing all nondominated points is defined, using all known feasible points. In the particular bi-objective case, the search area to explore when starting Phase 2 is composed of triangles the hypothenuse of which is defined by consecutive supported points with respect to one objective. Generally, each triangle is explored individually and the proposition of Visée et al. (1998) has been to apply a branch and bound algorithm locally to each triangle. This idea can be seen as the repeated application of several *local* branch and bound algorithms (by opposition to the application of a unique *global* branch and bound algorithm). This requires very few adaptations: the exploration is technically done in a rectangle defined with the origin as the lower-left point and the corner point of the explored triangle as the upper-right point (see Fig. 6), the incumbent list as well as any computed lower bound sets are restricted to this rectangle. These restrictions are applied without modification of the structure problem. The main advantage is that far more nodes can be fathomed. The main drawback is that some nodes may be repeated between different triangles. Next, to work in a relatively small portion of objective space gives the possibility to apply a preprocessing at the root node of the branch and bound tree in order to reduce the size of the problem. For this, it is tested if the nodes obtained by fixing only one single variable x_j to 0 or 1 can be fathomed. Finally, the enumeration starts and the variables fixed with the preprocessing do not need to be considered again. This last idea has been initially proposed by Kiziltan and Yucaoglu (1983) for a global branch and bound algorithm. However, the conflicting nature of objective functions generally implies that there are very few variables that share the same value in all efficient solutions of the problem. Consequently, such a preprocessing may fix very few variables in a global branch and bound algorithm. On the other hand, it can reduce the size of the problem significantly in a local branch and bound algorithm.

Two other propositions have been formulated to partition the exploration in objective space, both assume the use of the linear relaxation as a lower bound set, and both are currently only described for the bi-objective context. First, Stidsen et al. (2014) have proposed the use of *slicing*. Its principle is to partition objective space, by using cutting lines passing through the point (0, 0). The cutting lines have an equation of the kind $z_2 = \tan(\alpha)z_1$ where α is the angle between the z_1-axis and the cutting line. Each partition can therefore be explored individually by adding two constraints: $z_2 \leqq \tan(\alpha_1)z_1$ and $z_2 \geqq \tan(\alpha_2)z_1$ where α_1 and α_2 defines the cutting lines delimiting the partition. Note that the additional constraints does not increase significantly the computational difficulty of the determination of the lower bound set, as it still requires the solution of a bi-objective linear program. A main difference with the application of a two phase method, is that each partition is not independent. A part of a partition can be dominated by a point located in another partition (see Fig. 7). The same way as a local branch and bound algorithm embedded in a two phase method, more nodes can be fathomed. The same drawback is that several applications of a branch and bound algorithm (with some common nodes) are necessary.

Contrary to the two phase method and the slicing method, Parragh and Tricoire (2014) have proposed to apply a global branch and bound algorithm with what we call here a *two stage* branching procedure. If a node of the branch and bound tree cannot be fathomed by dominance, the authors noticed that the linear relaxation is often partially dominated, i.e. some parts of the edges are dominated and some other are not (see Fig. 8). By identifying the sub-edges that are not dominated, it is possible to determine a search area that may contain new nondominated points, and that is defined by the area weakly dominated by the linear relaxation and not dominated by the incumbent list. This search area is composed of several connected components (see

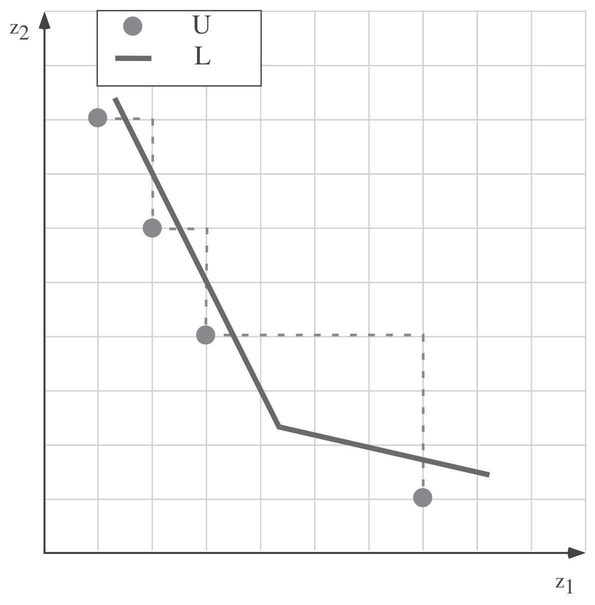

Fig. 8. The node cannot be fathomed by dominance.

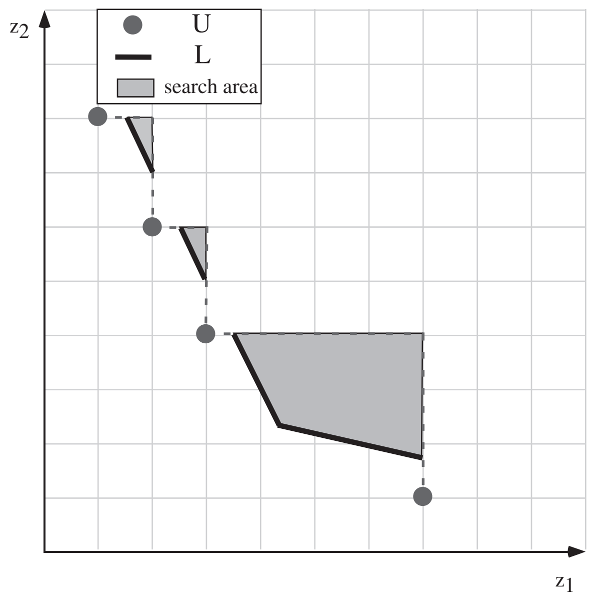

Fig. 9. Keeping only nondominated sub-edges of the linear relaxation, a search area is defined.

Fig. 9), that the authors propose to explore separately. For this, a corner point μ is associated to each connected component and is used in a new original branching strategy: one intermediate node is defined by adding the constraints $z_1(x) \leqq \mu_1$ and $z_2(x) \leqq \mu_2$. However, there is no evaluation of such a node, as by construction, it cannot be fathomed. A classical branching procedure is thus immediately applied to continue the exploration. For the same reason as for the slicing, the addition of these constraints does not increase significantly the computational cost of the linear relaxation, as no more than two linear constraints are added.

6. Guidelines

We make a distinction between problems the single-objective version of which can be solved in polynomial or pseudo-polynomial time, and also between structured and unstructured problems. As for other methodologies, there has been far more published methods for problems the single-objective version belongs to $\mathcal{P}$. Consequently, we have the confirmation that some ingredients work well or better than other ones for this class of problems. The feedback about other classes of problems is less precise, but some promising ideas have been proposed recently and should/could be confirmed/refined in the future.

For each class of problem, we do not detail here a particular choice for the active node and the branching strategy. Indeed, there is no evidence that a particular strategy works better for a particular context. However, the depth-first search strategy and the basic static strategies surely have the advantage of simplicity and are not expensive to apply.

For problems the single-objective version of which belongs to $\mathcal{P}$ or that can be solved in pseudo-polynomial time, it is natural to apply a two phase method. After the determination of a set of extremal supported solutions, the application of a local search allows generally to obtain a very good initialization of the upper bound set on Y_N. Next, the exploration of each triangle can start. The lower bound set to use preferably is here the convex relaxation. Its main advantages are that it is the best possible bound set L such that $L + \mathbb{R}^p_{\geqq}$ is convex, and that feasible solutions are generated with its computation, with a possible update of the incumbent list. Moreover, it can be computed in reasonable time in this context. The tightness of the convex relaxation as a lower bound set is also a strong point for the efficiency of the preprocessing. The published algorithms based on these elements have obtained very good performances. These algorithms perform very well on almost all kinds of instances and are very few sensitive to the shape of the nondominated set. However, they seem to suffer from the range of data. Indeed, instances with a larger range of data imply generally larger triangles to explore.

For structured problems for which the single-objective version cannot be solved in pseudo-polynomial time, the convex relaxation becomes expensive and can no longer be used intensively. Hence, another lower bound set should be used. However, to apply a two phase method (i.e. compute the nondominated extreme points only for the root node) may remain a reasonable choice for the initialization of the upper bound set (and a local search can again be applied next) and to divide the exploration in objective space. Next, there are two possible choices for the lower bound set: either the linear relaxation (cheaper than the convex relaxation, but surely not tight enough) or to use another relaxation that is well-adapted for the single-objective case. This last possibility has not yet been used in published branch and bound algorithms (if we exclude the use of the ideal point), but papers on bound sets have shown that using a number of weights (to tune for finding a good trade-off) could give interesting results. On the other hand, the linear relaxation has been used recently to specifically divide the exploration in objective space (using slicing or a two stage branching procedure), which increases its interest.

Finally for unstructured problems, the linear relaxation becomes the immediate choice as a lower bound set. An initialization using some scalarization methods with well-dispersed weights (to tune again for finding a good trade-off) remains essential to start the enumeration with a good initialization of the upper bound set for Y_N.

7. Multi-objective zero-one branch and bound

In this section, we give a description of some of the published multi-objective branch and bound algorithms from the literature. Indeed even if we have tried to consider as much published papers as possible, we can never claim to be exhaustive with the literature. All descriptions are done according to the same format, by instantiating the components of a multi-objective branch and bound algorithm. In particular, we will denote by $\bar{Y}$ the set of feasible points of the subproblem defined by a considered node. Next, some additional notes are provided.

7.1. Kiziltan and Yucaoglu (1983)

This method has been proposed to solve any MO01ILP. Contrary to the usual assumption for MOCO problems, the authors make the assumption that objective coefficients are not restricted in sign. We keep here the presentation in the maximization sense as originally described.

- Lower bound set on Y_N: Incumbent list.
- Upper bound set on $\bar{Y}_N$: For each objective z_k, one solution x^k is constructed by fixing each free variable x_j to 0 (if $c_j^k < 0$) or 1 (if $c_j^k \geqq 0$). Constraints of the problem are ignored in this process. Finally, a single point $\bar{y}$ such that $\bar{y}_k := z_k(x^k)$ is obtained as an upper bound set.
- Generation of new feasible points: At each considered node, one solution is constructed by fixing all free variables to 0 and it is checked for feasibility. Feasible solutions may also be obtained in the computation of the upper bound set.
- Choice of the active node: Depth-first search.
- Branching procedure: One variable x_j is fixed at the value 1 first and next to 0. The choice of the variable to be fixed is done in a dynamic way depending on the constructed solution (by fixing all free variable to 0) at each node. If this solution is feasible, then any free variable x_j such that $c_j \nleqq 0$ can be chosen (otherwise the quality of the constructed solution will decrease). Otherwise, the solution constructed in each child node (i.e. by fixing one free variable x_j to 1 and the others free ones to 0) is considered. If it is possible to obtain a feasible solution by fixing $x_j = 1$ then we branch on x_j. If it is not possible to obtain a feasible solution, then the infeasibility of each solution is measured by summing the slack variable of each unsatisfied constraint, and we branch next on the variable that is the nearest to feasibility according to this measure.

One additional dominance test is proposed next in a variant of this method and are applied in each considered node. Before to branch on a new variable, it has been proposed to preprocess the considered subproblem by testing (using the proposed upper and lower bound sets) if any solution considered in further completion by fixing any free variable x_j to 1 or 0 will lead to a dominated solution or not. The application of this idea was of course limited by the quality of the used upper bound set.

7.2. Ulungu and Teghem (1997)

This method has been proposed for the solution of bi-objective uni-dimensional binary knapsack problem. This problem has already been described in Section 3.1.

- Lower bound set on Y_N: Incumbent list.
- Upper bound set on $\bar{Y}_N$: For each objective function z_k, one single-objective upper bound (Martello & Toth, 1990) is computed. One single point is obtained as an upper bound set by taking the values of both single-objective bounds.
- Generation of new feasible points: At each considered node, one feasible solution is constructed by fixing all free variables to 0.
- Choice of the active node: Depth-first search.
- Branching procedure: One variable x_j is fixed at the value 1 first and next to 0. The choice of the free variable to be fixed is done according to a static strategy. The order $\mathcal{O}^{sumrank}$ described in Section 3.1 is used.

Nodes of the branch and bound tree are fathomed by infeasibility if the residual capacity of the knapsack is strictly negative.

7.3. Visée et al. (1998)

This method is in its main lines, an application of the branch and bound algorithm proposed by Ulungu and Teghem (1997) in a two phase method. In the following, the extreme points of the considered triangle will be denoted by y^r and y^s and the weight $\lambda = (y_2^r - y_2^s, y_1^s - y_1^r)^T \in \mathbb{R}^2_>$ denotes the normal to the line $\overline{y^r y^s}$. We denote the considered triangle $\Delta(y^r, y^s)$.

- Lower bound set on $Y_N \cap \Delta(y^r, y^s)$: Incumbent list, restricted to the considered triangle.
- Upper bound set on $\bar{Y}_N \cap \Delta(y^r, y^s)$: For each objective z_k, one single-objective upper bound v^k is computed (Martello & Toth, 1990). The same way as the method by Ulungu and Teghem (1997), a single point $u = (v^1, v^2)$ is considered as an upper bound set to apply the dominance test. Next, an upper bound v^λ on the single-objective problem defined by $\lambda^T z$ is determined if the node has not been fathomed. An upper bound set defined by the edge $\{y \in \mathbb{R}^2 \ : \ \lambda^T y = \lambda^T v^\lambda, y_1 \leqq \min\{v^1, y_1^s\}, y_2 \leqq \min\{v^2, y_2^r\}\}$ is therefore obtained.
- Generation of new feasible points: At each considered node, one feasible solution is constructed by fixing all free variables to 0. However, it is necessary to test that this feasible solution belongs to the considered triangle, before a possible update of the lower bound set.
- Choice of the active node: Depth-first search.
- Branching procedure: One variable x_j is fixed at the value 1 first and next to 0. The choice of the free variable to be fixed is done according to a static strategy. For this, the variables x_j are sorted by decreasing value of $\lambda^T(\frac{c_j^1}{w_j}, \frac{c_j^2}{w_j})^T$.

For each triangle $\Delta(y^r, y^s)$, a preprocessing is applied at the root node of the branch and bound tree in order to reduce the size of the problem.

7.4. Ramos, Alonso, Sicilia, and González (1998)

This method has been proposed for the solution of bi-objective minimum weight spanning tree problem. The set of vertices is denoted by $V = \{1, \ldots, n\}$ in the following.

- Upper bound set on Y_N: Incumbent list.
- Choice of the active node: Depth-first search.
- Branching procedure: One edge e is mandatory (first) or (implicitly) forbidden (next). The choice of the edge to fix is done according to a static strategy. At root node, only edges that are incident to the vertex 1 are considered. The other edges that are incident to the vertex 2, 3, ..., $n-1$ are considered respectively in the stages 2,3,..., $n-1$ of the branch and bound tree. For any vertex i, the incident edges are chosen in any order.
- Lower bound set on $\bar{Y}_N$: For all vertex $i \in \{2, \ldots, n-1\}$, the minimum cost e_i^k with respect to objective k for any edge that is incident to vertex i is initially computed (i.e. before starting the enumeration). Let j be the last vertex considered in the branching procedure, a lower bound $\bar{z}_k$ is computed according to each objective by summing the weights of the mandatory edges and by adding $\sum_{k=j+1}^{n-1} e_i^k$. From these two single-objective lower bounds, a single point is deduced and used as a lower bound set on $\bar{Y}_N$.
- Generation of new feasible points: The set of all supported non-dominated points is initially computed to initialize the upper bound set on Y_N. Next feasible solutions are obtained at the leaves of the branch and bound tree.

Despite an initialization using a set of supported points, this remains a global branch and bound algorithm.

7.5. Sourd and Spanjaard (2008)

This method has been proposed for the solution of bi-objective minimum weight spanning tree problem. In the following, the weight of an edge e according to objective k is denoted by w_e^k.

- Lower bound set on Y_N: Convex relaxation.
- Upper bound set on $\bar{Y}_N$: Incumbent list.
- Choice of the active node : Depth-first search.
- Branching procedure: One edge e is mandatory (first) or forbidden (next). The choice of the edge is done according to a static strategy (free edge e such that e gives the minimal value for $\min\{w_e^1, w_e^2\}$).
- Generation of feasible points: The algorithm is initialized with the computation of extreme supported solutions, followed by an application of a local search. Next, feasible points are obtained by computing (possibly partially) the convex relaxation at each node.

Before starting the enumeration, the size of the problem is reduced by a preprocessing of the graph. The principle is to find edges that are mandatory or forbidden in at least one efficient solution corresponding to each nondominated point of the problem. This has been done using adaptations of the cut optimality condition and of the cycle optimality condition.

7.6. Florios, Mavrotas, and Diakoulaki (2010)

This method has been proposed for the solution of multi-dimensional multi-objective knapsack problem. This problem is defined by the following model,

$$\begin{aligned} \max z_k(x) &= \sum_{j=1}^{n} c_j^k x_j && k = 1, \ldots, p \\ & \sum_{j=1}^{n} w_j^i x_j \leqq \omega^i && i = 1, \ldots, m \\ & x_j \in \{0, 1\} \end{aligned}$$

where c_j^k and w_j^i are non-negative integer numbers.

- Lower bound set on Y_N: Incumbent list.
- Upper bound set on $\bar{Y}_N$: Ideal point of the linear relaxation.
- Choice of the active node: Depth-first search.
- Branching procedure: One binary variable x_j is fixed at the value 1 first and next 0. The choice of the free binary variable to be fixed is done according to a static strategy. Several orders have been proposed. The first consist in considering the variables in the order of their index. Next, the order AVG (average) sorts the variables x_j by decreasing order of $\sum_{i=1}^{m} \sum_{k=1}^{p} \frac{c_j^k}{w_j^i}$, the order MAX sorts the variables x_j by decreasing order of $\max_{i=1,\ldots,m;k=1,\ldots,p} \frac{c_j^k}{w_j^i}$, and the order LEX sorts the variables by decreasing order (in the lexicographic sense) of $(\frac{c_j^1}{w_j^1}, \ldots, \frac{c_j^1}{w_j^m}, \frac{c_j^2}{w_j^1}, \ldots, \frac{c_j^2}{w_j^m}, \ldots, \frac{c_j^p}{w_j^1}, \ldots, \frac{c_j^p}{w_j^m})$. Finally, the composite order is obtained by solving the linear relaxation of the problem, in the sense of computing the efficient extreme solutions of the problem. Next for each variable x_j, a sum S_j is done by cumulating the corresponding values of each computed solution. The variables x_j are next sorted by decreasing value of S_j.
- Generation of feasible points: An initialization is done by using the solution of the linear relaxation of the problem. For each obtained efficient extreme solution, the fractional variables are rounded to the nearest value (0 or 1) and these modified solutions are tested for feasibility. During the execution of the algorithm, feasible solutions are obtained when all variables are fixed (i.e. in leaves of the branch and bound tree).

The best results are generally obtained with the composite order on the small instances. For the largest instances, the order MAX or the order LEX generally gives better computational times.

7.7. Jorge (2010)

Two branch and bound methods for the three-objective unidimensional binary knapsack problem have been proposed in Jorge (2010). The first proposed method is based on a depth-first search exploration using a static strategy for the choice of the branching variable.

- Lower bound set on Y_N: Incumbent list.
- Upper bound set on $\bar{Y}_N$: Similarly to the method by Ulungu and Teghem (1997) in the bi-objective case, one single-objective upper bound is computed for each objective z_k. One single point y^U is obtained as an upper bound set by taking the values of all computed single-objective bounds. If this upper bound set is not tight enough to fathom the node, a single-objective problem defined by a weighted sum with weight $\lambda = y^U$ is solved exactly and an upper bound set defined by $\{y \in \mathbb{R}^p \mid y_1 \leqq y_1^U, y_2 \leqq y_2^U, \lambda^T y \leqq \max_{\bar{y} \in \bar{Y}} \lambda^T \bar{y}\}_N$ is used for the dominance test.
- Branching procedure: One variable x_j is fixed at the value 1 first and next to 0. The choice of the free variable to be fixed is done according to a static strategy. The orders $\mathcal{O}_k$ for $k \in \{1, 2, 3\}$, $\mathcal{O}_{sum}$, $\mathcal{O}_{max}$, $\mathcal{O}_{min}$, $\mathcal{O}_{dom}$, $\mathcal{O}_{\#}$ and $\mathcal{O}_{rg}$ described in Section 3.1, have been used.
- Choice of the active node: Depth-first search.
- Generation of feasible points: One feasible point is obtained with the computation of the upper bound set (when the considered node is not fathomed by dominance), and at the leaves of the branch and bound tree.

The second proposed method is one of the only proposed branch and bound algorithms for which the choice of the active node as well as the branching strategy are done in a dynamic way.

- Lower bound set on Y_N: Incumbent list.
- Upper bound set on $\bar{Y}_N$: The convex relaxation is computed (possibly partially) and defines an upper bound set on $\bar{Y}_N$. However, only facets of this convex relaxation are used to fathom nodes (see Section 4.1).
- Branching procedure: One variable x_j is fixed at the value 1 or 0. A mixed strategy for the choice of the branching variable is proposed here. First, a dynamic strategy is applied considering all supported solutions obtained in the determination of the lower bound set. If there are regular items, i.e. items j such that the corresponding variable x_j has the same value for all supported solutions composing the lower bound set, the branching variable is any of the corresponding variables. If the dynamic strategy cannot be applied (if there is no regular item), the static strategy proposed in the first version of the branch and bound algorithm is used.
- Choice of the active node: All active nodes are sorted lexicographically according to the five following criteria.
  - Tightness ratio: The active nodes are sorted by decreasing value of $|0,5 - \sum_{i \in F} \frac{w_i}{\bar{\omega}}|$ where F denotes the subset of free variables and $\bar{\omega}$ denotes the residual capacity (subproblems with a tightness ratio near 0.5 are generally harder to solve).
  - Quality of the computed lower bound set: The number of supported solutions associated to an active node, that are not dominated so far in the enumeration are counted. The active nodes with the largest set of nondominated solutions are preferred.
  - Generation of feasible solutions: The active nodes the lower bound set of which contains the largest set of supported solutions are preferred.

  - Residual capacity: The active nodes with the smallest residual capacity are preferred.
  - Number of free variables: The active nodes with the smallest number of free variables are preferred.
- Generation of feasible points: Feasible points are obtained at each node of the branch and bound tree with the (possibly partial) computation of the convex relaxation.

In both proposed branch and bound algorithms, Jorge (2010) has proposed an improvement in the branching procedure based on a notion of dominance between items. Item i is said to dominate item j if $(c^1_i, \dots, c^p_i, -w_i) \geq (c^1_j, \dots, c^p_j, -w_j)$. For all item $j \in \{1, \dots, n\}$, the sets $Pref(j) = \{i \in \{1, \dots, n\} : (c^1_i, \dots, c^p_i, -w_i) \geq (c^1_j, \dots, c^p_j, -w_j)\}$ and $Dom(j) = \{i \in \{1, \dots, n\} : (c^1_i, \dots, c^p_i, -w_i) \leq (c^1_j, \dots, c^p_j, -w_j)\}$ are initially computed, and represents the sets of items that are respectively preferred to and dominated by j. Consequently, when fixing x_j at 0, x_i can also be fixed at 0 for all items $i \in Dom(j)$. The same way, when fixing x_j at 1, x_i can also be fixed at 1 for all items $i \in Pref(j)$.

7.8. Delort (2011)

This method has been proposed for the solution of bi-objective linear assignment problem. This problem is defined by the following model,

$$\begin{aligned} \min z_k(x) \;=\; & \sum_{i=1}^{n}\sum_{j=1}^{n} c^k_{ij} x_{ij} && k = 1, \dots, p \\ & \sum_{j=1}^{n} x_{ij} = 1 && i = 1, \dots, n \\ & \sum_{i=1}^{n} x_{ij} = 1 && j = 1, \dots, n \\ & x_{ij} \in \{0, 1\} \end{aligned}$$

where c^k_{ij} are non-negative integer numbers. The proposed algorithm is based on the two phase method. In the following, the extreme points of the considered triangle will be denoted by y^r and y^s and the weight $\lambda = (y^r_2 - y^s_2, y^s_1 - y^r_1)^T \in \mathbb{R}^2_>$ denotes the normal to the line $\overline{y^r y^s}$. We denote the considered triangle $\Delta(y^r, y^s)$.

- Upper bound set on $Y_N \cap \Delta(y^r, y^s)$: Incumbent list, restricted to the considered triangle.
- Lower bound set on $\bar{Y}_N \cap \Delta(y^r, y^s)$: Convex relaxation restricted to the considered triangle.
- Choice of the active node: Depth-first search.
- Branching procedure: One variable x_{ij} is fixed to 1 (first) or 0 (next). The choice of the free variable to fix is done according to a static strategy. The variables x_{ij} are considered by increasing order of $\lambda^T(c^1_{ij}, c^2_{ij})$. Of course if a variable x_{ij} is fixed at the value 1, all variable $x_{i'j}$ or $x_{ij'}$ for any $i' \neq i$, $j' \neq j$, are fixed at the value 0.
- Generation of feasible points: The algorithm is globally initialized with a heuristic based on the computation of extreme supported solutions, followed by a local search (based on a simple swap). This global initialization is used to define next an upper bound set restricted to each triangle. Next, feasible solutions are obtained by computing the convex relaxation at each node (if its restriction to the considered triangle is not empty).

For each triangle $\Delta(y^r, y^s)$, a preprocessing is applied at the root node of the branch and bound tree in order to reduce the size of the problem. This can be done as usually by testing if the nodes obtained by fixing one single variable x_{ij} to 0 or 1 should be fathomed by dominance or infeasibility. However, Delort (2011) noticed that the problem reduction is far more significant if we can fix a variable x_{ij} definitively at the value 1 (as $x_{ij'}$ and $x_{i'j}$ are consequently fixed at 0 for any $i' \neq i$, $j' \neq j$) rather than definitively at the value 0. Delort (2011) also noticed that this preprocessing can be expensive as there are n^2 variables to consider. Consequently to these observations, Delort (2011) has proposed a restriction of this preprocessing to the variables x_{ij} that are fixed at the value 1 in the solution x^r and x^s corresponding to the extreme points y^r and y^s of the triangle, and it is only tested if these variables should be fixed definitively at the value 1.

7.9. Parragh and Tricoire (2014)

Parragh and Tricoire (2014) have proposed a method to solve the bi-objective team orienteering problem with time windows. Basically, this problem consists in defining m paths from a source node to a destination node. Each path will pass along a subset of nodes called control points, by respecting some time windows. The first objective consists in minimizing the total cost of the paths. The second objective consists in maximizing the profit generated by visiting control points. This branch and bound method is the first application of the two stage branching procedure (described in Section 5) that divides dynamically the exploration in objective space.

- Upper bound set on Y_N: Incumbent list.
- Lower bound set set on $\bar{Y}_N$: Linear relaxation. It is interesting to note that it is computed here using a dichotomic scheme for which the single-objective linear problem is solved using column generation.
- Choice of the active node: Breadth-first search.
- Generation of new feasible points: The upper bound set on Y_N is initialized with a multi-directional local search method. Next, feasible solutions are sometimes obtained with the linear relaxation.
- Branching procedure: A two stage branching procedure is applied. The classical branching procedure to apply in the second stage is an adaptation of the most fractional value criteria. The authors propose to consider the extreme efficient solutions of the linear relaxation and to compute the average value for the binary variables associated to the visit of control points. The control point with the most fractional average value is chosen to define the branching variable. If all have a binary value, the authors propose to branch on the arc that is traversed in average with the most fractional value.

8. Multi-objective mixed 0–1 linear branch and bound

There is a recent interest for multi-objective linear programs mixing continuous and binary variables. It is in particular interesting to note that the branch and bound methodology has been often chosen for these recent development (with the exception of Boland, Charkhgard, & Savelsbergh 2015).

8.1. Mavrotas and Diakoulaki (1998, 2005)

Mavrotas and Diakoulaki (1998, 2005) have proposed a branch and bound algorithm for MOM01LP. The description given below is in the minimization case (as the original presentations).

- Upper bound set on Y_N: Incumbent list.
- Lower bound set on $\bar{Y}_N$: Ideal point of the linear relaxation.
- Choice of the active node: Depth-first search.
- Branching procedure: One binary variable x_j is fixed at the value 0 first and next 1. The choice of the free binary variable to be fixed is done according to a static strategy, by increasing index of the variables.

- Generation of new feasible points: When all binary variables are fixed, there remains only continuous variables in the problem. Thus, we obtain a Multi-Objective Linear Program (MOLP) that is computed with a multicriteria simplex algorithm (see Ehrgott, 2005). The nondominated set of a MOLP is connected and composed of maximal faces of dimension $1, \dots, p-1$. Most of the algorithms for the solutions of MOLP computes only nondominated extreme points of these faces (that are implicitly deduced). Mavrotas and Diakoulaki (1998, 2005) only consider these extreme points to update the incumbent list.

The main differences between the algorithms in Mavrotas and Diakoulaki (1998, 2005) are computational improvements regarding the solution of MOLP and a final dominance test for the incumbent list U. Indeed, Mavrotas and Diakoulaki (2005) have noticed that considering the nondominated extreme points of MOLP is not sufficient to discard all dominated points, and as a consequence, the incumbent list may contain dominated points at the end of enumeration. Nevertheless, Vincent et al. (2013) have shown that this final dominance test remains incomplete, due to the information lost (faces of MOLP) during the enumeration, and that consequently the incumbent list may still contain dominated points.

8.2. *Vincent et al. (2013)*

Vincent et al. (2013) have proposed a branch and bound algorithm for bi-objective zero-one mixed-integer linear programs (BOM01LP). This algorithm is a correction, including also several improvements, of the algorithm proposed in Mavrotas and Diakoulaki (1998, 2005), but with a restricted application to the bi-objective context. The following presentation is done in the minimization case.

- Upper Bound Set on Y_N: Consequently to Theorem 1 stated in Section 4.2, an extended incumbent list with additional informations is used. The idea is not only to store extreme points of edges, but also additional informations (connectivity between points, weakly dominated points). The objective is twofolds: it ensures a correct (and complete) result as output of the algorithm, it improves the considered upper bound set on Y_N (see Fig. 5) and consequently strengthen any dominance test.
- Lower bound Set on $\bar{Y}_N$: Four bound sets have been used (Ideal point of the linear relaxation, ideal point of the convex relaxation, linear relaxation, convex relaxation). According to the experimental results, the linear relaxation is the best compromise.
- Choice of the active node: Depth-first search.
- Branching procedure: One binary variable x_j takes the value 0 or 1. The choice of the free binary variable to branch on is done according to a static strategy. For each variable x_j, the coefficient $e(j) := (c^1_j - \mu_1) + (c^2_j - \mu_2)$ where μ_1 and μ_2 are the mean of the cost coefficients of respectively the first and the second objective are computed. The branching variable is x_j such that $e(j)$ takes the maximal absolute value (variable fixed to 0 first if $e(k) > 0$ and 1 otherwise).
- Generation of feasible points: The upper bound set on Y_N is initialized by computing the set of nondominated extreme points of the problem. As the combinations of binary variables of the corresponding supported solutions are promising, the BOLP obtained by fixing the same combinations of binary variables are solved. Next anytime a leaf node is reached during the enumeration of the branch and bound tree, a BOLP is solved and the extended incumbent list is possibly updated.

8.3. *Stidsen et al. (2014)*

This method has been proposed for the solution of a particular class of BOM01LP: continuous variables are present only in one objective function. The nondominated set Y_N is discrete in this case. The presentation is done in the minimization case.

This method starts with the computation of lexicographic optimal points y^1 and y^2 with respect to the permutations (1, 2) and (2, 1). The ideal and nadir points are therefore known, and all nondominated points are located in the rectangle defined by y^1, y^N, y^2, y^I. The objective functions are next rescaled (without change for the efficient and dominated solutions) so that this box becomes the square $[0, 1] \times [0, 1]$. This is done by using the following objective functions: $\bar{z}_1(x) = \frac{z_1(x)-y^1_1}{y^2_1-y^1_1}$, $\bar{z}_2(x) = \frac{z_2(x)-y^2_2}{y^1_2-y^2_2}$. Next, the weighted sum objective function $\bar{z} := \bar{z}_1 + \bar{z}_2$ is considered.

- Upper bound set on Y_N: Incumbent list.
- Lower bound set on $\bar{Y}_N$: The linear relaxation of the single-objective problem with objective function $\bar{z}$ is solved, by adding constraints (called no-good constraints Hooker, 2000) excluding already enumerated feasible solutions, and a solution $\tilde{x}$ is obtained. $\{y \in [0,1]^2 \ : \ \frac{y_1-y^1_1}{y^2_1-y^1_1} + \frac{y_2-y^2_2}{y^1_2-y^2_2} = \bar{z}(\tilde{x})\}$ defines next the lower bound set on $\bar{Y}_N$.
- Generation of new feasible points: The solution $\tilde{x}$ obtained with the computation of the lower bound set is tested for feasibility.
- Choice of the active node: This component is controlled by the MIP solver used for the implementation (ILOG CPLEX 12.1.1) (Stidsen, 2014).
- Branching procedure: There are two cases here. If the solution $\tilde{x}$ is infeasible, then two nodes are created by fixing one fractional variable to respectively 0 and 1. The choice of the branching variable is controlled by the MIP solver used for the implementation (Stidsen, 2014). If the solution $\tilde{x}$ is feasible, one new node is created by adding a constraint excluding the enumeration of $\tilde{x}$ (this branching procedure is called integer fathoming).

Some refinements are proposed next. The first one is the slicing (Section 5), that partitions objective space before starting the exploration. Depending on the considered problem, the number of partitions to consider to obtain good computational results is varying (from 2 to 32). The second refinement is called Pareto branching and replaces integer fathoming. It is therefore applied when a feasible solution $\tilde{x}$ is generated. The idea is not only to generate a solution that is different than already generated feasible solutions, but that is also not strictly dominated by a subset of them (the feasible solutions $\tilde{x}$ such that $\bar{z}(x) \leqq \bar{z}(\tilde{x})$). This is done by creating two nodes (rather than one with integer fathoming) by adding respectively the constraints $\bar{z}_1(x) \leqq \bar{z}_1(\tilde{x}^1)$ and $\bar{z}_2(x) \leqq \bar{z}_2(\tilde{x}^2)$, where $\tilde{x}^1$ and $\tilde{x}^2$ are already enumerated feasible solutions such that $\bar{z}(\tilde{x}^1) \leqq \bar{z}(\tilde{x})$ and $\bar{z}(\tilde{x}^2) \leqq \bar{z}(\tilde{x})$ and that minimize respectively the first and the second objective. This branching procedure performs better than integer fathoming.

8.4. *Belotti et al. (2013)*

The method has been proposed for the solution of general bi-objective mixed-integer linear programs. This method is thus proposed for a more general context than the method by Vincent et al. (2013), binary variables are here replaced by general integer variables. This is the only method in this survey that considers explicitly this last kind of variables. At any node of the branch and bound tree, the same way as in the single-objective case, the integer variables are bounded by particular values rather than to be fixed. We present the main elements of this branch and bound algorithm in the maximization case (as it is stated in the original technical report).

- Lower Bound set on Y_N: Incumbent list defined by isolated points and edges.

- Upper Bound set on $\bar{Y}_N$: The linear relaxation is used as an upper bound set but is not explicitly computed as described in Section 4.2.
- Branching procedure: The parametric simplex algorithm is used to solve (partially) the linear relaxation. Indeed, the computation of the linear relaxation stops as soon as either one of its computed extreme solutions is infeasible for the mixed-integer initial problem, or consecutive extreme solutions have different values for the integer variables (an infeasible point between them is next deduced).
  The most fractional variable x_j in the obtained infeasible solutions is chosen and its value is noted α (Belotti, 2014). Two nodes are created by adding respectively the constraints $x_j \leqq \lfloor \alpha \rfloor$ and $x_j \geqq \lceil \alpha \rceil$.
- Choice of the active node: For all solutions computed in the (partial) linear relaxation, the integer infeasibility is summed for all integer variables. The node with maximum sum of integer infeasibility is selected (Belotti, 2014).
- Generation of feasible points: The algorithm is initialized by computing minimal and maximal feasible values of the linear relaxation of both objectives. These values are used to define a box containing all feasible points of the linear relaxation of the bi-objective problem, and thus containing all feasible points of the bi-objective problem. The ϵ-constraint method is next used with a parameterized number of fixed steps to generate weakly nondominated points inside this box.
  During the execution of the algorithm, when a node cannot be fathomed by dominance, a partial computation of the linear relaxation is performed and the feasibility of extreme solutions of this linear relaxation is tested.

9. Other multi-objective branch and bound algorithms

In this section, we enumerate some other multi-objective branch and bound algorithms that does not fit in the two preceding sections. Some of these algorithms may use very specific components (for example, scheduling problems), or address a class of problem that is outside of the scope of this paper (non linear problems). Hence, the purpose of this section is not to give a complete description of these algorithms.

9.1. Specific discrete optimization problems

Facility location: Gandibleux, Przybylski, Bourougaa, Derrien, and Grimault (2012) introduces a solution method for the solution of the zero-one version of the bi-objective uncapacitated facility location problem. The first step of this method consists in determining a set of refined boxes containing all nondominated points. Equivalently, a particular pair of lower and upper bound sets on Y_N is obtained. The main originality in comparison to the literature is that the lower bound set L is a discrete set of points, i.e. $(L + \mathbb{R}^2_{\geqq})$ is not convex. The construction of these boxes is performed following a branch and bound principle, each box corresponding to a node of the search tree. The second step of the algorithm (exploration of the boxes) is done according to a label setting principle.

Others papers related to this class of problems: Bouibede-Hocine (2007), Reiter and Gutjahr (2012).

9.2. Non linear problems

For multi-objective non linear problems (MONLP), solution methods based on specific techniques, such as interval programming or continuous constraint programming, have been proposed. The authors of these methods classify them as branch and bound method. However, these methods are generally far more similar to the literature about multi-objective approximation (Ruzika & Wiecek, 2005) than the branch and bound architecture described in this survey.

Fernández and Tóth (2007, 2009): Inspired of theories and techniques of global optimization, this paper presents an interval branch-and-bound algorithm aiming at obtaining a tight outer approximation of the nondominated set of nonlinear biobjective problems. The general idea is, starting from a initial box containing the set of feasible points, to select and split a box into several sub-boxes, until a series of final boxes contains a sufficiently tight description of the nondominated set. Numerical results are reported for facility location problems.

Martin, Goldsztejn, Granvilliers, and Jermann (2014), Martin (2014): The proposed algorithm solves bi-objective nonlinear programming problems. Thanks to the assumption of differentiability of the functions, the algorithm makes use of continuation methods. Starting from an initial solution, a continuation method produces the corresponding connected part of the nondominated set. One initial solution is thus required in each connected part of the efficient frontier. The branch and bound algorithmic embeds a continuous constraint programming solver, which is based on interval analysis principle. The output is a certified computation of the nondominated set.

Cacchiani and D'Ambrosio (2013): The paper proposes a general-purpose branch and bound algorithm for solving convex multi-objective mixed integer non-linear programming problems. The proposed algorithm follows a standard branch and bound methodology. Here, the upper bound set for Y_N is the incumbent list (initialized by an application of a scalarization method) and the lower bound set for the nondominated set of any nodes is given by the ideal point of the corresponding subproblem. The performances of the algorithm are discussed on base of a hydro unit commitment problem simplified, and restricted to binary variables and 2 objectives.

Others papers related to this class of problems: Dellnitz, Schütze, and Hestermeyer (2005), Kubica and Woźniak (2008), Scholz (2010).

9.3. Related domains

Constraint Programming (CP) Milano (2004), Freuder and Wallace (2005) grew out of the logic programming community as part of an effort to embed constraints in a programming language. It is an optimization paradigm based on declarative formulations of the problem to solve in terms of variables-constraints-domains, a graph-based representation of search space and principles of the reduction of domain's variables. Constraint technology got success in some problem solving areas such as logistics, scheduling, design, or bio-informatics. In CP, search principles are mainly based on a search tree. It is then not surprising to notate papers proposing multi-objective branch and bound approaches based on CP techniques (Marinescu, 2009; Rollon & Larrosa, 2009).

Parallel computing, distributed computing, grid computing, GP-GPU architecture or again multicore programming are examples of useful computing paradigms with the hope of speeding up calculation of optimal solution(s) as much as possible. In particular, the use of parallel computing in the context of branch and bound is well established since many years (Gendron & Crainic, 1994). Those paradigms are more and more considered in the design and implementation of multiobjective optimization algorithms (Ismail, EL-Wahed, & el Raoof, 2010; Mezmaz, Melab, & Talbi, 2007; Talbi et al., 2008; Zhang, 2008).

10. Open questions

Several open questions are raised at the end of this review. Here, three of them are suggested.

Most of the recently proposed branch and bound algorithms use as a lower bound set (case of minimization) either the convex relaxation or the linear relaxation. Other relaxations should therefore be used, and the linear relaxation is generally not tight enough. Hence, there are two possible ways to investigate: improve the linear relaxation using cutting planes, or use other relaxations. The first way is usual in the single-objective case and is called branch and cut algorithms. A first attempt has been recently proposed in the literature for bi-objective problems for which one objective has a very small range. It consists in the application of an ε-constraint method where the single-objective problems are solved using a dedicated single-objective branch and cut algorithm (Jozefowiez, Laporte, & Semet, 2012). In recent workshops and conferences, on-going works about multi-objective branch and cut algorithms (Cerqueus, Gandibleux, Przybylski, Ruzika, & Saubion, 2015; Gadegaard, Ehrgott, & Nielsen, 2015; Stidsen & Andersen, 2015) have been presented and appears as a new trend for the years to come. The second way has been very scarcely investigated until now. A promising solution could be to merge several lower bound sets as it is proposed in Ehrgott and Gandibleux (2007). This idea is used in Cerqueus et al. (2015b) to compute tight bound sets using surrogate relaxation for bi-objective bi-dimensional knapsack problem. Practical use in a branch and bound algorithms remains to be done.

Next, most of the methods use a static strategy for the choice of the active node (depth-first search) and for the choice of the branching variable. Moreover, these strategies are straight-forward extensions of the single-objective case. Dynamic strategies remains to be investigated.

Finally, most of the published branch and bound algorithms are specifically designed for the bi-objective case. Indeed, the case of $p > 2$ is far more difficult whatever the used methodology. In particular, it remains to propose a way to apply a complete weak dominance test for the case of a lower bound set L such that $L + \mathbb{R}^p_{\geqq}$ is a convex polyhedra.

Acknowledgment

This work is supported by the project ANR/DFG-14-CE35-0034-01 “Exact Efficient Solution of Mixed Integer Programming Problems with Multiple Objective Functions (vOpt)”.

References

Aneja, Y. P., & Nair, K. P. K. (1979). Bicriteria transportation problem. *Management Science, 25*, 73–78.

Bazgan, C., Hugot, H., & Vanderpooten, D. (2009). Solving efficiently the 0–1 multi--objective knapsack problem. *Computers & Operations Research, 36*, 260–279.

Belotti, P. (2014). Personal communication.

Belotti, P., Soylu, B., & Wiecek, M. M. (2013). A branch-and-bound algorithm for biobjective mixed-integer programs. *Technical report*. www.optimization-online.org/DB_FILE/2013/01/3719.pdf

Boland, N., Charkhgard, H., & Savelsbergh, M. (2015). A criterion space search algorithm for biobjective mixed integer programming: The triangle search method. *INFORMS Journal on Computing, 27*, 597–618.

Bouibede-Hocine, K. (2007). *Enumération en ordonnancement multicritère : application à un problème bicritère à machines parallèles (In French)* Ph.D. thesis. Polytech'Tours.

Cacchiani, V., & D'Ambrosio, C. (2013). A branch-and-bound algorithm for convex multi-objective mixed integer non-linear programming problems. *Technical report*. http://www.optimization-online.org/DB_HTML/2013/12/4183.html

Cerqueus, A., Gandibleux, X., Przybylski, A., Ruzika, S., & Saubion, F. (2015). A branch-and-cut method for the bi-objective bi-dimensional knapsack problem. *Proceedings of the 23rd international conference on multiple criteria decision making, Hamburg, Germany*.

Cerqueus, A., Przybylski, A., & Gandibleux, X. (2015b). Surrogate upper bound sets for bi-objective bi-dimensional binary knapsack problems. *European Journal of Operational Research, 244*(2), 417–433.

Cohon, J. L. (1978). *Multiobjective programming and planning*. Academic Press, New York.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Wiley.

Dellnitz, M., Schütze, O., & Hestermeyer, T. (2005). Covering pareto sets by multilevel subdivision techniques. *Journal of Optimization Theory and Applications, 124*(1), 113–136.

Delort, C. (2011). *Algorithmes d'énumération implicite pour l'optimisation multi-objectifs : exploitation d'ensembles bornant et application aux problèmes de sac à dos et d'affectation (in French)* Ph.D. thesis. France: Université Pierre et Marie Curie (Paris VI).

Delort, C., & Spanjaard, O. (2010). Using bound sets in multiobjective optimization: Application to the biobjective binary knapsack problem. In *Proceedings of the 9th international symposium on experimental algorithms (SEA 2010)*. In *Lecture Notes in Computer Science: Vol. 6049* (pp. 253–265).

Ehrgott, M. (2005). *Multicriteria optimization* (2nd). Berlin - Heidelberg: Springer.

Ehrgott, M., & Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum, 22*, 425–460.

Ehrgott, M., & Gandibleux, X. (2001). Bounds and bound sets for biobjective combinatorial optimization problems. In M. Koksalan, & S. Zionts (Eds.), *Multiple criteria decision making in the new millennium*. In *Lectures Notes in Economics and Mathematical Systems: Vol. 507* (pp. 241–253). Springer.

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

Ehrgott, M., & Tenfelde-Podehl, D. (2003). Computation of ideal and nadir values and implications for their use in MCDM methods. *European Journal of Operational Research, 151*, 119–139.

Fernández, J., & Tóth, B. (2007). Obtaining an outer approximation of the efficient set of nonlinear biobjective problems. *Journal of Global Optimization, 38*(2), 315–331.

Fernández, J., & Tóth, B. (2009). Obtaining the efficient set of nonlinear biobjective optimization problems via interval branch-and-bound methods. *Computational Optimization and Applications, 42*(3), 393–419.

Florios, K., Mavrotas, G., & Diakoulaki, D. (2010). Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms. *European Journal of Operational Research, 203*(1), 14–21.

Freuder, E. C., & Wallace, M. (2005). Constraint programming. In E. K. Burke, & G. Kendall (Eds.), *Search methodologies chapter 9* (pp. 239–272). Springer.

Gadegaard, S. L., Ehrgott, M., & Nielsen, L. R. (2015). A cut and branch approach for a class of bi-objective combinatorial optimization problems. *Proceedings of the 23rd international conference on multiple criteria decision making, Hamburg, Germany*.

Gandibleux, X., Przybylski, A., Bourougaa, S., Derrien, A., & Grimault, A. (2012). Computing the efficient frontier for the 0/1 biobjective uncapacitated facility location problem. cors/mopgp'12, june 11–13, niagara falls (canada).

Gendron, B., & Crainic, T. G. (1994). Parallel branch-and-bound algorithms: Survey and synthesis. *Operations Research, 42*(6), 1042–1066.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing Co.

Hooker, J. (2000). *Logic-based methods for optimization: Combining optimization and constraint satisfaction.*. John Wiley & Sons, New York.

Ismail, M. M., EL-Wahed, W. F. A., & el Raoof, O. A. (2010). A parallel branch and bound algorithm for solving large scale integer multiobjective problems. In *Proceedings of the 2010 7th international conference on informatics and systems (infos)* (pp. 1–7).

Jorge, J. (2010). *Nouvelles propositions pour la résolution exacte du sac à dos multi-objectif unidimensionnel en variables binaires (in French)* Ph.D. thesis. Université de Nantes.

Jozefowiez, N., Laporte, G., & Semet, F. (2012). A generic branch-and-cut algorithm for multiobjective optimization problems: Application to the multilabel traveling salesman problem. *INFORMS Journal on Computing, 24*, 554–564.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack problems*. Springer.

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

Kiziltan, G., & Yucaoglu, E. (1983). An algorithm for multiobjective zero-one linear programming. *Management Science, 29*, 1444–1453.

Klamroth, K., Lacour, R., & Vanderpooten, D. (2015). On the representation of the search region in multi-objective optimization. *European Journal of Operational Research, 245*, 767–778.

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

Kubica, B. J., & Woźniak, A. (2008). Interval methods for computing the pareto-front of a multicriterial problem. In R. Wyrzykowski, J. Dongarra, K. Karczewski, & J. Wasniewski (Eds.), *Parallel processing and applied mathematics*. In *Lecture Notes in Computer Science: Vol. 4967* (pp. 1382–1391). Springer Berlin Heidelberg.

Marinescu, R. (2009). Exploiting problem decomposition in multi-objective constraint optimization. In I. Gent (Ed.), *Principles and practice of constraint programming*. In *Lecture Notes in Computer Science: Vol. 5732* (pp. 592–607). Springer Berlin Heidelberg.

Martello, S., & Toth, P. (1990). Knapsack problems: Algorithms and computer implementations. *Wiley-Interscience series in discrete mathematics and optimization*. J. Wiley & Sons.

Martin, B. (2014). *Rigorous algorithms for nonlinear biobjective optimization* Ph.D. thesis. Université de Nantes, France.

Martin, B., Goldsztejn, A., Granvilliers, L., & Jermann, C. (2014). On continuation methods for non-linear bi-objective optimization: towards a certified interval-based approach. *Journal of Global Optimization*, 1–14.

Mavrotas, G., & Diakoulaki, D. (1998). A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research, 107*(3), 530–541.

Mavrotas, G., & Diakoulaki, D. (2005). Multi-criteria branch and bound: A vector maximization algorithm for Mixed 0–1 Multiple Objective Linear Programming. *Applied Mathematics and Computation, 171*(1), 53–71.

Mezmaz, M.-S., Melab, N., & Talbi, E.-G. (2007). A grid-based parallel approach of the multi-objective branch and bound. In *Proceedings of the 15th euromicro international conference on parallel, distributed and network-based processing* (pp. 23–30).

Milano, M. (Ed.) (2004). Constraint and integer programming. *Operations Research/Computer Science Interfaces Series* (Vol. 27). Springer.

Özpeynirci, O., & Köksalan, M. (2010). An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs. *Management Science, 56*, 2302–2315.

Parragh, S. N., & Tricoire, F. (2014). Branch-and-bound for bi-objective optimization. http://www.optimization-online.org/DB_HTML/2014/07/4444.html.

Przybylski, A., Gandibleux, X., & Ehrgott, M. (2010a). A recursive algorithms for finding all nondominated extreme points in the outcome set of a multiobjective integer program. *INFORMS Journal on Computing, 22*, 371–386.

Przybylski, A., Gandibleux, X., & Ehrgott, M. (2010b). A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization, 7*, 149–165.

Ramos, R. M., Alonso, S., Sicilia, J., & González, C. (1998). The problem of the optimal biobjective spanning tree. *European Journal of Operational Research, 111*, 617–628.

Reiter, P., & Gutjahr, W. J. (2012). Exact hybrid algorithms for solving a bi-objective vehicle routing problem. *Central European Journal of Operations Research, 20*(1), 19–43.

Rollon, E., & Larrosa, J. (2009). Constraint optimization techniques for exact multi--objective optimization. In V. Barichard, M. Ehrgott, X. Gandibleux, & V. T'Kindt (Eds.), *Multiobjective programming and goal programming*. In *Lecture Notes in Economics and Mathematical Systems: Vol. 618* (pp. 89–98). Springer Berlin Heidelberg.

Rong, A., & Figueira, J. R. (2014). Dynamic programming algorithms for the bi-objective integer knapsack problem. *European Journal of Operational Research, 236*, 85–99.

Ruzika, S., & Wiecek, M. M. (2005). Approximation methods in multiobjective programming. *Journal of Optimization Theory and Applications, 126*, 473–501.

Scholz, D. (2010). The multicriteria big cube small cube method. *TOP, 18*(1), 286–302.

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

Stidsen, T. (2014). Personal communication.

Stidsen, T., & Andersen, K. A. (2015). Optimized parallel branch & cut algorithm for bi-objective TSP. *Proceedings of the conference of the 2nd workshop on recent advances in multi-objective Optimization, Nantes, France*.

Stidsen, T., Andersen, K. A., & Dammann, B. (2014). A branch and bound algorithm for a class of biobjective mixed integer programs. *Management Science, 60*(4), 1009–1032.

Talbi, E.-G., Mostaghim, S., Okabe, T., Ishibuchi, H., Rudolph, G., & Coello, C. A. C. (2008). Parallel approaches for multiobjective optimization. In J. Branke, K. Deb, K. Miettinen, & R. Słowiński (Eds.), *Multiobjective optimization*. In *Lecture Notes in Computer Science: Vol. 5252* (pp. 349–372). Springer Berlin Heidelberg.

Ulungu, E. L. B., & Teghem, J. (1997). Solving multi-objective knapsack problem by a branch-and-bound procedure. In J. Climaco (Ed.), *Multicriteria analysis* (pp. 269–278). Springer Verlag, Berlin.

Villarreal, B., & Karwan, M. H. (1981). Multicriteria integer programming: A (hybrid) dynamic programming recursive approach. *Mathematical Programming, 21*(1), 204–223.

Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., & Gandibleux, X. (2013). 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.

Visée, M., Teghem, J., Pirlot, M., & Ulungu, E. L. B. (1998). Two-phases method and branch and bound procedures to solve the bi-obective knapsack problem. *Journal of Global Optimization, 12*, 139–155.

Zhang, W. (2008). *Parallel multi-objective branch and bound*. Technical University of Denmark. Master's thesis