Yugoslav Journal of Operations Research
33 (2023), Number 1, 91–110
DOI: https://doi.org/10.2298/YJOR210915015C

OPTIMIZING OVER THE EFFICIENT SET OF THE BINARY BI-OBJECTIVE KNAPSACK PROBLEM

Djamal CHAABANE
Faculty of Mathematics, Department of Operations Research, Laboratory AMCD-RO, USTHB, Bab-Ezzouar, Algiers, Algeria
chaabane_dj@yahoo.fr

Nadia LACHEMI
Faculty of Mathematics, Department of Operations Research, Laboratory AMCD-RO, USTHB, Bab-Ezzouar, Algiers, Algeria
nadia0989@hotmail.fr

Received: September 2021 / Accepted: September 2022

Abstract: This paper deals with the problem of optimizing a linear function over the efficient set of a 0-1 bi-objective knapsack problem. Such a function represents the main criterion of the problem posed. The resolution process is based essentially on dynamic programming. The proposed method provides a subset of efficient solutions including one which optimizes the main criterion without having to enumerate all the efficient solutions of the problem. Numerical experiments are reported, different instances with large sizes of the associated efficient sets are considered to show the efficiency of our algorithm compared with an approach proposed in the literature.

Keywords: Multiple Objective Programming, Bi-objective Knapsack Problem, Dynamic Programming, Efficient Set, Optimal Solution.

MSC: 90C05, 90C27, 90C39.

1. INTRODUCTION

The multiple objective knapsack problem is a well-known combinatorial optimization problem which appears as sub problem in many real world applications like logistics, industry, economics, transport and many fields. In this work we consider the 0-1 bi-objective knapsack problem (denoted by BOKP). This problem has received a significant attention in the literature, and has been widely studied by many researchers, and several approaches have been proposed to solve it: exact algorithms which provide the set (or

a subset) of efficient solutions and approximate algorithms which produce a good approximation of the efficient set. One of the well-known methods in the literature is the two-phase method. It was introduced by Ulungu [1, 2] where in the first phase, the set of supported efficient solutions is found by solving a weighted sum of the objective functions. In the second phase, the unsupported efficient solutions are found by using either an exact or an approximate method. Captivo et al. [3], however, modeled the BOKP by a bi-objective shortest path problem, then used a labeling algorithm to solve it, in comparison to the two phases method [1, 2], better results were obtained. Bazgan et al. [4] developed three complementary dominance relations in a dynamic programming algorithm. These relations were applied to fathom states that can't lead to efficient solutions. The approach only returns efficient solutions in criteria space and requires powerful programming tools for large size instances. This method was improved in [5] by Figueira et al. who have proposed new fathoming techniques to speed up the resolution process. Recently Correia et al. [6] thought to improve the memory usage of a dynamic programming algorithm for computing the set of efficient solutions in both objective and decision spaces. Besides exact methods, some good approximations of the nondominated solutions of a BOKP were also presented. Erlebach et al. [7] developed a fully polynomial time approximation scheme (FPTAS) to computes a $(1 + \epsilon)$-approximation for every nondominated solution. In [8] Bazgan et al. proposed a new FPTAS using dominance relations on all objective values. Some authors have also observed that a pretreatment of the problem is very important before solving it. Jorge et al. in their paper [9] exposed several properties allowing to find some parts of the structure of all efficient solutions. Daoud and Chaabane [10] proposed a new way to reduce the dimension of the BOKP based on the item's efficient ratio. Better results where obtained compairing to those of Jorge et al. (see [9]).
The set of all efficient solutions could be very large and choosing among this set according to the decision maker preference would be very difficult, thus optimizing the preference (linear in the decision variables in our case) is imposed. This idea allows to calculate a few number of efficient solutions, one of which optimizes the added criterion.

The problem of optimizing a linear criterion over the efficient set of a multiple objective programming problem is a linear program with nonconvex constraints. The difficulty of this problem is due to the nonconvexity of the feasible set, therefore delicate methods are necessary. The problem has been studied widely in the literature in the case of continuous variables (see, e.g., [11, 12, 13, 14, 15]), algorithms, theoretical results and some examples were proposed, but not enough experimental study was provided. These approaches and others were reviewed in [16]. As for integer variables, in the last three decades, since 1992 (see [17]), it became one of the most important and interesting areas in multicriteria programming. Nguyen [17] was the first to try to optimize over the integer efficient set, where only an upper bound value for the main objective is proposed. Then Abbas and Chaabane [18] developed for the first time a decision space search algorithm. They used different types of cuts to ensure an improvement on the main objective value. A criteria space search was proposed by Jorge [19] which is inspired by the work presented in [20]. Jorge's approach defined a sequence of progressively more constrained single-objective integer problems that eliminates unacceptable points. Unlike the previous methods, this method was implemented and tested on different problem instances randomly generated. Chaabane and Pirlot [21] developed a cutting plane algorithm combining Sylva and Crema way of reducing the feasible set and exploring the admissible region to improve the optimal efficient solution. Another algorithm was proposed by Chaabane et al. [22]; it employs an augmented weighted Tchebychev norm, to reduce the admissible region by the successive addition of constraints. The algorithm was tested on a variety of randomly generated problems. Boland et al. [23] employed a decomposition and search procedure to revise Jorge's proposal [19] and confirmed their improvement by a comparative experimental study. Mahdi and Chaabane [24] and Drici et al. [25] considered the case when the main criterion is a fractional function, they used cutting plane and branch and bound techniques respectively. Although some existing algorithms developed for multiple objective integer programs are applicable to binary variables, to our knowledge no study has been specially developed for this case.

In this work an algorithm is developed to optimize a linear function over the efficient set of a BOKP without having to explicitly enumerate all the efficient solutions. Our proposal is based on dynamic programming instead of linear programming or cut techniques which are the most used in the above cited works. The algorithm is inspired by the dynamic programming algorithm presented in [4]. However, we use some of the dominance relations proposed in [4] in order to eliminate some partial solutions that can not provide efficient solutions. Another relation that we propose is used to discard some partial solutions that can not improve the main criterion, this relation allows to eliminate early more solutions either efficient or not. In order to obtain an optimal solution, we should keep the solutions in decision space in memory while the algorithm is running; this should require more memory space and a longer computation time with dynamic programming. Our algorithm employs an efficiency test program to ensure the efficiency of a solution and returns the corresponding efficient solution in decision space without storing the decision variables in memory. We assess the performance of the proposed algorithm on well-known publicly available instances of BOKP. We compare our results with the results given by Jorge's algorithm [19] in terms of the percentage of computed efficient solutions and cpu time.

In the following section some basic concepts, notations and definitions concerning the dynamic programming applied to our problem are introduced. The informal explanation the algorithm is described in section 3. Section 4 represents the technical description of the proposed algorithm illustrated by a didactic example. In section 5 a computational comparative study on different existing instances is provided. Finally, some conclusions and perspectives are given in the last section.

2. DEFINITIONS AND BASIC CONCEPTS

Let a knapsack of capacity W and a set of n items of weights w_j and profits p_j^1, p_j^2 on criteria Z_1 and Z_2 respectively, $j \in \{1, \ldots, n\}$. A BOKP consists of selecting a subset of items, which maximizes the overall profits on criteria Z_1 and Z_2, and putting them in the knapsack without exceeding its capacity.

The problem can be stated, mathematically, as follows:

$$(BOKP)\begin{cases} \max\ Z_i(x) &= \sum_{j=1}^{n} p_j^i x_j,\ i=1,2 \\ & \sum_{j=1}^{n} w_j x_j \le W, \\ & x_j \in \{0,1\}\ \forall j \in \{1,\ldots,n\} \end{cases} \tag{1}$$

where x_j is a decision variable which equals to 1 if the item j is included in the knapsack and 0 otherwise. For each item, we correspond three parameters: non-negative integer profit p_j^i for $i \in \{1,2\}$ and $j \in \{1,\ldots,n\}$, positive integer weight w_j for $j \in \{1,\ldots,n\}$. We assume a positive integer value of the knapsack capacity W. In order to avoid trivial solutions we suppose that

$$w_j \le W,\ \ \forall j = 1,\ldots,n, \text{ and} \sum_{j=1}^{n} w_j > W.$$

We denote by $p^{(i)}$ the i^{th} row vector $(p_j^i)_{j\in\{1,\ldots,n\},i\in\{1,2\}}$.

Let X be a non-empty finite set of all feasible solutions, $\forall x, \tilde{x} \in X$, x dominates $\tilde{x}$ $(x\underline{\Delta}\tilde{x})$ if and only if $Z_i(x) \ge Z_i(\tilde{x})$, $i = 1,2$, with at least one strict inequality. If $\nexists x \in X$ such that $x\underline{\Delta}\tilde{x}$ then $\tilde{x}$ is efficient. Its corresponding vector $Z(\tilde{x}) = (Z_1(\tilde{x})\ Z_2(\tilde{x}))'$ is said to be nondominated. The sets E and ND denote the set of all efficient solutions and nondominated vectors respectively. Let $\phi(x) = \sum_{j=1}^{n} d_j x_j$ a linear function called "main objective", where $d_j, j \in \{1,\ldots,n\}$ is supposed to be a positive integer.

The problem of optimizing a linear function over the efficient set of BOKP is given by :

$$(P_E)\begin{cases} max\,\phi(x) &= \sum_{j=1}^{n} d_j x_j \\ & x = (x_1\ \ldots\ x_n)' \in E \end{cases} \tag{2}$$

Let's consider the following problem:

$$(BOKP_k)\begin{cases} max\ \ Z_i(x) &= \sum_{j=1}^{k} p_j^i x_j, \quad i=1,2 \\ & \sum_{j=1}^{k} w_j x_j \le W, \\ & x_j \in \{0,1\} \quad \forall j \in \{1,\ldots,k\}. \end{cases} \tag{3}$$

The problem $(BOKP_k)$ is the BOKP problem induced by the k first items $(k = 1,\ldots,n)$. A state $s^k = (s_1^k, s_2^k, s_3^k, s_4^k)$ represents a feasible solution of the $(BOKP_k)$, where s_i^k is the value on criterion i, $(i = 1,2)$, s_3^k and s_4^k its associated weight and main objective value respectively. The set of all feasible solutions of $(BOKP_k)$, $k \in \{1,\ldots,n\}$ is defined by S^k:

$$S^k = S^{k-1} \cup \quad \left\{ \left(s_1^{k-1} + p_k^1, s_2^{k-1} + p_k^2, s_3^{k-1} + w_k, s_4^{k-1} + d_k\right) \mid \right.$$
$$\left. s_3^{k-1} + w_k \leq W,\ s^{k-1} \in S^{k-1} \right\}$$

The initial set of states, S^0, contains only the state $s^0 = (0,0,0,0)$ which corresponds to the empty knapsack. The final set of states S^n defines the set of all feasible solutions of $(BOKP)$. Efficiency and dominance on S^k are defined in the same manner as on X. Thus, for $s^k, \tilde{s}^k \in S^k$, $s^k \underline{\Delta} \tilde{s}^k$ if and only if $s_i^k \geq \tilde{s}_i^k, \forall i \in \{1,2\}$, with at least one strict inequality.

A state $s^n \in S^n$ is an **extension** of $s^k \in S^k$, $(k \leq n)$ if and only if

$$s_i^n = \qquad s_i^k + \sum_{j \in J} p_j^i, i = 1,2;\ s_3^n = s_3^k + \sum_{j \in J} w_j;\ s_4^n = s_4^k + \sum_{j \in J} d_j$$

$$\text{such that} \quad J = \{j \in \{k+1, \ldots, n\} \mid s_3^k + \sum_{j \in J} w_j \leq W\}$$

Let $s^k, \tilde{s}^k \in S^k$. The lexicographic relation $\underline{\Delta}_{lex}$ is defined on S^k by: $s^k \underline{\Delta}_{lex} \tilde{s}^k \Leftrightarrow s_t^k > \tilde{s}_t^k$ where $t = \min\{i \in \{1,2\} : s_i^k \neq \tilde{s}_i^k\}$ or $s_i^k = \tilde{s}_i^k, \forall i \in \{1,2\}$. The lexicographic relation $\geq_{lex}$ is defined on S^k by: $s^k \geq_{lex} \tilde{s}^k \Leftrightarrow s_3^k < \tilde{s}_3^k$ or ($s_3^k = \tilde{s}_3^k$ and $s^k \underline{\Delta}_{lex} \tilde{s}^k$).

In the following sections we denote by E_f the efficient solutions subset of $(BOKP)$, by ND_f, we mean the subset of nondominated solutions corresponding to E_f, ϕ_{inf} is a lower bound of the main objective ϕ, x_{opt} is an optimal solution of (P_E) and ϕ_{opt} the optimal value of the main objective ϕ.

3. GENERAL DESCRIPTION OF THE APPROACH

In this section, we describe the general principle of the method by highlighting each used procedure.

The method consists of m stages $(m \leq n)$. First, we start from a lower bound ϕ_{inf} of the function ϕ, then at each stage k, $k = 1, \ldots, m$ we generate and trim progressively a subset of states $C^k \subseteq S^k$ from C^{k-1}, with $C^0 = S^0$. We use some comparison relations to discard some states in C^k that can not lead to efficient solutions improving ϕ_{inf}. We calculate a subset of efficient solutions improving the main criterion value then update ϕ_{inf}.

In order to avoid generating useless states in C^k, we use the two relations D_r^k and $D_{\underline{\Delta}}^k$ proposed in [4]. Another relation that we call Z_b^k is used to omit states from C^k whose upper bound are dominated by some nondominated solutions already found. This relation is proposed in [4] to compare between specific extensions of a state and an upper bound of the extensions of another state. To discard states that can't conduce to an improvement of the function ϕ we propose a new relation called Φ_b^k.

3.1. Finding a lower bound of the main objective ϕ

To determine a lower bound of the main objective ϕ, we have to find at least one efficient solution of $(BOKP)$.
Two extreme efficient solutions x^1_{opt} and x^2_{opt}, however, are found lexicographically as follows:

1. The single objective problem (P_1) induced by the first objective Z_1 is solved,

$$(P_1) : \left\{ max \ \ Z_1(x) = p^{(1)}x, x \in X \right\},$$

let x^1 be an optimal solution of (P_1). In case it is unique then $x^1_{opt} = x^1$. Otherwise, we solve the problem

$$(P_2) : \left\{ max \ \ Z_2(x) = p^{(2)}x, x \in X, p^{(1)}x = p^{(1)}x^1 \right\},$$

among the optimal solutions found, we select the best one, x^1_{opt} at the main criterion ϕ.
2. In the same manner we obtain the second efficient solution x^2_{opt}.

Let ϕ_{inf} be a lower bound of the function ϕ, $\phi_{inf} = \max\{\phi(x^1_{opt}), \phi(x^2_{opt})\}$.

3.2. Efficiency test

As a generated state $s^* = (s^*_1, s^*_2, s^*_3, s^*_4) \in S^n$ associated to a feasible solution x^*, where $p^{(1)}x^* = s^*_1$ and $p^{(2)}x^* = s^*_2$, in the proposed procedure is feasible, not necessarily efficient, we suggest using Ecker and Kouada efficiency test (see [26]) to know the nature of the new solution.
Let $(BOKP) : \{''max'' \ (p^{(1)} \ p^{(2)})x, \ x \in X\}$ be the bi-objective problem defined in (1).
Where $p^{(1)}, p^{(2)}$ are the objective coefficients of $(1 \times n)$ array and X is the corresponding admissible region.

Theorem 1. *$x^* \in X$ is an efficient solution if and only if the optimal value of the objective function Θ is null in the following mixed integer linear programming problem:*

$$(P_\Psi(x^*)) \begin{cases} max \ \ \Theta = \sum_{i=1}^{2} \psi_i \\ p^{(1)}x = \psi_1 + p^{(1)}x^* \\ p^{(2)}x = \psi_2 + p^{(2)}x^* \\ x \in X; \ \ \psi_i \ \ \textit{are real non negative for } i = 1, 2. \end{cases} \qquad (4)$$

3.3. Comparison between states

The following relations are used to compare between states.

Definition 2 (Relation D_r^k [4]). *Let $s^k \in S^k$ and $\tilde{s}^{k-1} \in S^{k-1}, k = 1, \dots, n$. D_r^k is defined as:*

$$s^k D_r^k \tilde{s}^{k-1} \Leftrightarrow \begin{cases} s^k & = & (\tilde{s}_1^{k-1} + p_k^1, \tilde{s}_2^{k-1} + p_k^2, \tilde{s}_3^{k-1} + w_k, \tilde{s}_4^{k-1} + d_k) \\ & & \textit{and} \\ \tilde{s}_3^{k-1} & \leq & W - \sum_{j=k}^{n} w_j \end{cases} \tag{5}$$

The equation $\tilde{s}_3^{k-1} \leq W - \sum_{j=k}^{n} w_j$ means that all the remaining items (items $k, \dots, n$) can be added to the knapsack, so the only extension of s^k that can represent an efficient solution is the one that contains objects $k, \dots, n$, thus it is not necessary to calculate extensions without item k. This relation is used to determine states in C^{k-1} which can belong to C^k at the stage k. The states in $C^k, k = 1, \dots, n$ are sorted according to the decreasing order with respect to the relation $\geq_{lex}$ (see [4]). This order allows to determine easily j the first index of states in C^{k-1} which is not dominated with respect to the relation D_r^k, thus it is unnecessary to include states $s^{k-1(1)}, \dots, s^{k-1(j-1)}$ in C^k.

Definition 3 (Relation $D_{\underline{\Delta}}^k$ [4]). *Let two states $s^k, \tilde{s}^k \in S^k, k = 1, \dots, n$. $D_{\underline{\Delta}}^k$ is defined as:*

$$s^k D_{\underline{\Delta}}^k \tilde{s}^k \Leftrightarrow s^k \underline{\Delta} \tilde{s}^k \quad \textit{and} \quad s_3^k \leq \tilde{s}_3^k, \ k < n \tag{6}$$

Since $s_3^k \leq \tilde{s}_3^k$, then all objects that can be selected from $\tilde{s}^k$ can also be from s^k. Hence, if $s^k \underline{\Delta} \tilde{s}^k$ then the state s^k will provide solutions which are at least as good as those provide from $\tilde{s}^k$. The latter can be omitted for the rest.

Let $u = (u_1, u_2, u_\phi)$ be an upper bound of a feasible solution represented by a state s^k. Where u_1, u_2 are upper bounds of objective 1 and 2 respectively, u_ϕ an upper bound of ϕ. We use the upper bound described in [4, 27]:
Let $O_i, i = 1, 2$, the decreasing order of profits to weight ratios p_j^i / w_j, $j \in \{k+1, \dots, n\}$, $i = 1, 2$. t_i is the position of the first item in $\{k+1, \dots, n\}$ that cannot be added to s^k when the items are reordered according to order O_i. Let $\overline{W}(s^k) = W - s_3^k$ be the residual capacity associated to s^k. An upper bound u_i of the objective i is calculated as follows:

$$u_i = s_i^k + \sum_{j=k+1}^{t_i - 1} p_j^i + max \left\{ \lfloor \overline{W}(s^k) \frac{p_{t_i+1}^i}{w_{t_i+1}} \rfloor, \lfloor p_{t_i}^i - (w_{t_i} - \overline{W}(s^k)) \frac{p_{t_i-1}^i}{w_{t_i-1}} \rfloor \right\}. \tag{7}$$

Similarly, for the main objective ϕ:

$$u_\phi = s_4^k + \sum_{j=k+1}^{t_\phi - 1} d_j + max \left\{ \lfloor \overline{W}(s^k) \frac{d_{t_\phi+1}}{w_{t_\phi+1}} \rfloor, \lfloor d_{t_\phi} - (w_{t_\phi} - \overline{W}(s^k)) \frac{d_{t_\phi-1}}{w_{t_\phi-1}} \rfloor \right\}. \tag{8}$$

Where t_ϕ is the position of the first item in $\{k+1,\dots,n\}$ that cannot be added to s^k when the items are reordered according to order O_ϕ. O_ϕ is the decreasing order of profits to weight ratios d_j/w_j, $j \in \{k+1,\dots,n\}$.

In addition to the previous relations, we introduce a new relation (definition 4) in order to reduce efficiently the set C^k and eliminate early more useless states.

Definition 4 (Relation Z_b^k). *Let $s^k \in S^k, k = 1,\dots,n$ and $u = (u_1, u_2, u_\phi)$ its associated upper bound. If $\exists s = (s_1, s_2) \in ND_f$ such that $(s_1, s_2)\underline{\Delta}(u_1, u_2)$, thus s^k can't produce an efficient solution, Thus it can be eliminated.*

In the following proposition, we show how to exclude states that can not be optimal for the main problem (P_E) by comparing an upper bound of the main objective function of a current state to the present lower bound of the main objective.

Proposition 5 (Relation Φ_b^k). *Let $s^k \in S^k, k = 1,\dots,n$ and $u = (u_1, u_2, u_\phi)$ its associated upper bound. Let ϕ_{inf} be the lower bound of the objective ϕ updated at the stage k. If $\phi_{inf} \geq u_\phi$ then s^k can't lead to an improvement of the main objective value, thus s^k can be omitted (s^k is dominated with respect to Φ_b^k).*

Proof. Let ϕ_{inf} be the lower bound of the objective ϕ updated at the stage k. Consider that s^k is dominated with respect to Φ_b^k. This implies that there exists another state $\tilde{s}^k \in S^k$ leading to an efficient state s^*, $s^* = (s_1^*, s_2^*, s_3^*, s_4^*)$ with $s_4^* = \phi_{inf}$. Let u_ϕ be an upper bound of s^k on the main objective value, it means that $u_\phi \geq s_4^n$, for any extension $s^n = (s_1^n, s_2^n, s_3^n, s_4^n)$ of the state s^k. Since $\phi_{inf} \geq u_\phi$, thus s^k is dominated by $\tilde{s}^k$ and it can not provide an efficient state with better value of the main objective ϕ. $\square$

3.4. General step *k*

Let $C^{k-1} = \{s^{k-1(1)},\dots,s^{k-1|C^{k-1}|}\}$ the set of states kept in step $k-1$, a subset of states C^k is generated and reduced progressively (algorithm 1) as follows.
First we determine the states in C^{k-1} that can belong to C^k. To do this, the dominance relation D_k^r is applied. The order of states in C^{k-1} allows to define the index j such that states $s^{k-1(j+1)}, s^{k-1(j+2)},\dots,s^{k-1(|C^{k-1}|)}$ can be added in C^k. Then we start generating new states from C^{k-1} according to the dominance relation $D_{\underline{\Delta}}^k$.

Algorithm 1 Procedure $generate\&trim(C^{k-1})$

```
%$C^{k-1} = \{s^{k-1(1)}, \ldots, s^{k-1(i)}, \ldots, s^{k-1(|C^{k-1}|)}\}$ in the decreasing order of $\geq_{lex}$.
$C^k := \emptyset$, $M^k := \emptyset$, $i := 1, j := 1$
% $j$, is the index of the first state of $C^{k-1}$ that is not dominated according to $D^k_r$.
while $j \leq |C^{k-1}|$ and $s_3^{k-1(j)} + \sum_{l=k}^{n} w_l \leq W$
  $j := j + 1$
while $i \leq |C^{k-1}|$ and $s_3^{k-1(i)} + w_k \leq W$
  $s^k := (s_1^{k-1(i)} + p_k^1, s_2^{k-1(i)} + p_k^2, s_3^{k-1(i)} + w_k, s_4^{k-1(i)} + d_k)$
  while $j \leq |C^{k-1}|$ and $s^{k-1(j)} \geq_{lex} s^k$
    $Maintain\_NDS(s^{k-1(j)}, M^k, C^k)$
    $j := j + 1$
  $Maintain\_NDS(s^k, M^k, C^k)$
  $i := i + 1$
while $j \leq |C^{k-1}|$
  $Maintain\_NDS(s^{k-1(j)}, M^k, C^k)$
  $j := j + 1$
```

We maintain a subset $M^k \subseteq C^k$ which stores nondominated states of C^k. A new generated state s^k is inserted in C^k if and only if there exists no state m^k in the current M^k such that $m^k \underline{\Delta} s^k$. The subsets M^k and C^k are updated at each insertion of new state. M^k is sorted according to a decreasing order with respect to the relation $\underline{\Delta}_{lex}$ (algorithm 2).

Algorithm 2 Procedure $Maintain\_NDS(s^k, M^k, C^k)$

```
%$M^k = \{m^{k(1)}, \ldots, m^{k(|M^k|)}\}$ in decreasing order of $\underline{\Delta}_{lex}$.
$dom := false, \ell := 1$
while $\ell \leq |M^k|$ and $m_1^{k(\ell)} \geq s_1^k$
  $\ell := \ell + 1$
if $\ell > 1$ and $m_2^{k(\ell-1)} > s_2^k$
  $dom := true$
if $dom = false$
  $M^k := M^k \cup \{s^k\}$ in the $\ell^{th}$ position of $M^k$
  $C^k := C^k \cup \{s^k\}$
  if $\ell > 1$ and $m_1^{k(\ell-1)} = s_1^k$ and $m_2^{k(\ell-1)} < s_2^k$
    $M^k := M^k \setminus \{m^{k(\ell-1)}\}$
    $\ell := \ell - 1$
  $\ell = \ell + 1$
  while $\ell \leq |M^k|$ and $s_1^k \geq m_1^{k(\ell)}$
    $M^k := M^k \setminus \{m^{k(\ell)}\}$
    $\ell := \ell + 1$
```

Once the generation and trim of states is finished, we look for efficient states improving ϕ_{inf} as follows:
For each state of M^k, an extension is generated according to O_1. Only extensions improving the criterion ϕ and nondominated by any solution in ND_f are listed. These latter are tested for their efficiency then E_f, ND_f and ϕ_{opt} are updated. This procedure is repeated when generating an extension according to O_2 (algorithm 3).

Algorithm 3 Procedure $test\_nondominated(ND_f, E_f, \phi_{inf}, s^n)$

```
$dom := false$
**if** $s_4^n \leq \phi_{inf}$
  $dom := true$
**else**
  $\ell := 1$
  **while** $\ell \leq |ND_f|$ and $dom = false$
    **if** $s^\ell \underline{\Delta} s^n$
      $dom := true$
    $\ell := \ell + 1$
  **if** $dom = false$
    solve $(P_\Psi)$
    Let $x^*$ the optimal solution of $(P_\Psi)$
    **if** $\Theta = 0$
      $ND_f := ND_f \cup \{(s_1^n, s_2^n)\}$, $E_f := E_f \cup \{x^*\}$, $\phi_{inf} := s_4^n$
    **else**
      **if** $\phi(x^*) > \phi_{inf}$
        $ND_f := ND_f \cup \{(Z_1(x^*)\ Z_2(x^*))\}$, $E_f := E_f \cup \{x^*\}$, $\phi_{inf} := \phi(x^*)$
```

Finally, we apply relations Z_b^k and Φ_b^k as follow: For each state s^k of C^k, we calculate its associated upper bound on both two criteria $u = (u_1, u_2)$. If there exists a nondominated solution s in ND_f such that $(s_1, s_2) > (u_1, u_2)$ thus s^k is removed, else we compute u_ϕ its upper bound on main objective, the s^k is removed if $\phi_{inf} \geq u_\phi$ (algorithm 4).

Algorithm 4 Procedure $removing\_states(C^k, ND_f, \phi_{inf})$

```
$i := 1$
while $i \leq |C^k|$
  Calculate $(u_1, u_2)$ an upper bound of $(s_1^{k(i)}, s_2^{k(i)})$
  $j := 1, remove := false$
  while $j \leq |ND_f|$ and $not(remove)$
    if $(s_1^{(j)}, s_2^{(j)}) \underline{\Delta} (u_1, u_2)$
      $remove := true$, $C^k := C^k \setminus \{s^{k(i)}\}$
    else
      $j := j + 1$
  if $remove := false$
    calculate $u_\phi$ an upper bound of $s_4^{k(i)}$
    if $\phi_{inf} \geq u_\phi$
      $C^k := C^k \setminus \{s^{k(i)}\}$
  $i := i + 1$
```

The algorithm terminates at a step n when all the objects are considered or at a step $m \leq n$, when all states of the previous step are omitted (C^{m-1} is empty and we can not generate states).

3.5. Algorithm

Algorithm 5 Optimizing over the (BOKP) efficient set

```
Input:
$\downarrow p^1_{(1\times n)}, p^2_{(1\times n)}$: criteria vectors
$\downarrow w_{(1\times n)}$: vector of weights
$\downarrow W$: Knapsack capacity
$\downarrow d_{(1\times n)}$: main criterion vector
Output:
$\uparrow x_{opt}$: optimal solution of the problem $(P_E)$
$\uparrow \phi_{opt}$: optimal value of the main criterion $\phi$
Initialisation:
$C^0 := \{s^0\} = (0,0,0,0)$, $k := 1$
Calculate $\phi_{inf}$ a lower bound of $\phi$, let $x^1, x^2$ the extreme efficient solutions
$E_f := \{x^1, x^2\}$, $ND_f := \{Z(x^1), Z(x^2)\}$
while $k \leq n$ and $|C^{k-1}| > 0$
  $generate\&trim(C^{k-1})$
  if $k = n$
    for $j = 1$ to $|M^k|$
      $test\_nondominated(ND_f, E_f, \phi_{inf}, s^{k(j)})$
  else
    for $i = 1$ to 2
      reorder items $k+1, \ldots, n$ according to $O_i$
      for $j = 1$ to $|M^k|$
        $s^n := m^{k(j)}$
        for $\ell = k+1$ to $n$
          if $s^n_3 + w_j \leq W$
            $s^n := (s^n_1 + p^1_\ell, s^n_2 + p^2_\ell, s^n_3 + w_\ell, s^n_4 + d_\ell)$
        $test\_nondominated(ND_f, E_f, \phi_{inf}, s^n)$
    $removing\_states(C^k, ND_f, \phi_{inf})$
  $k = k + 1$
$\phi_{opt} := \phi_{inf}$
```

DIDACTIC EXAMPLE

Consider the following example

$$(BOKP)\left\{\begin{array}{l} \max\ Z_1(x) = 15x_1 + 16x_2 + 8x_3 + 12x_4 + 6x_5 + 14x_6 \\ \max\ Z_2(x) = 3x_1 + 10x_2 + 12x_3 + 6x_4 + 11x_5 + 7x_6 \\ \text{s.t.} \left\{\begin{array}{l} 8x_1 + 3x_2 + 9x_3 + 5x_4 + 6x_5 + 8x_6 \leq 19 \\ x_j \in \{0,1\}, j \in \{1,2,\ldots,6\}. \end{array}\right. \end{array}\right.$$

Let the main problem be

$$(P_E)\left\{\begin{array}{l} \max\ \ \phi(x) = 2x_1 + 5x_2 + 9x_3 + 6x_4 + 4x_5 + 7x_6 \\ \text{s.t. } x \in E. \end{array}\right.$$

- Initialization:
  - $C^0 := \{s^0\} = \{(0,0,0,0)\}$.
  - Finding a lower bound of the main objective ϕ:
    The two extreme efficient solutions are: $x^1 = (1\,1\,0\,0\,0\,1)'$, $x^2 = (0\,1\,1\,0\,1\,0)'$,
    ϕ_{inf}=max$\{\phi(x^1), \phi(x^2)\}$=max$\{14, 18\} = 18$, $x_{opt} = x^2 = (0\,1\,1\,0\,1\,0)'$,
    $Z(x^1) = (45, 20)$, $Z(x^2) = (30, 33)$, $E_f = \{(1\,1\,0\,0\,0\,1)', (0\,1\,1\,0\,1\,0)'\}$,
    $ND_f = \{(45, 20), (30, 33)\}$.

- Step 1.
  $k = 1, C^1 = \emptyset, M^1 = \emptyset$.
  1. Generate and trim states:
     First we test if states of C^0 can be included in C^1. $s^0 = (0, 0, 0, 0)$, $\sum_{j=1}^{6} w_j = 39 > W$, then s^0 can be included in C^1.
     We generate s^1 from s^0, $s^1 = (0, 0, 0, 0) + (15, 3, 8, 2) = (15, 3, 8, 2)$.
     $C^1 := \{(0, 0, 0, 0), (15, 3, 8, 2)\}$, $M^1 := \{(15, 3, 8, 2)\}$.
  2. Generate extensions: the order of items $2, 3, \ldots, 6$ according to O_1 is: $2, 4, 6, 5, 3$.
     Thus $s^n = (15, 3, 8, 2) + (16 + 12, 10 + 6, 3 + 5, 5 + 6) = (43, 19, 16, 13)$,
     $s^n_4 < \phi_{inf}$. s^n does not improve ϕ_{inf}.
     The order of items $2, 3, \ldots, 6$ according to O_2 is: $2, 5, 3, 4, 6$.
     $s^n = (15, 3, 8, 2) + (16 + 6, 10 + 11, 3 + 6, 5 + 4) = (37, 24, 17, 11)$.
     $s^n_4 < \phi_{inf}$.
  3. Removing states:
     - Calculate upper bounds of states of C^1:
       <u>State s^0:</u> $(u_1, u_2) = (49, 34)$, it is not dominated by any solution in ND_f, $u_\phi = 22 > \phi_{inf}$, thus s^0 can not be removed from C^1.
       <u>State s^1:</u> $(u_1, u_2) = (46, 30)$, it is not dominated by any solution in ND_f, $u_\phi = 15 < \phi_{inf}$, s^1 can be omitted from C^1,
       $C^1 := C^1 \setminus \{s^1\} = \{(0, 0, 0, 0)\}$.

- Step 2.
  $k = 2, C^2 = \emptyset, M^2 = \emptyset$.
  1. Generate and trim states:
     $C^1 = \{s^0\} = \{(0, 0, 0, 0)\}$, $\sum_{j=2}^{6} w_j = 31 > W$ then s^0 can be included in C^2.
     We generate s^2 from s^0, $s^2 = (0, 0, 0, 0) + (16, 10, 3, 5) = (16, 10, 3, 5)$.
     $C^2 := \{s^0, s^2\} = \{(0, 0, 0, 0), (16, 10, 3, 5)\}$, $M^k := \{s^2\} = \{(16, 10, 3, 5)\}$
  2. Generate extensions: the order of items $3, 4, \ldots, 6$ according to O_1 is: $4, 6, 5, 3$.
     Thus $s^n = (16, 10, 3, 5) + (12 + 14, 6 + 7, 5 + 8, 6 + 7) = (42, 23, 16, 18)$.
     s^n is not dominated by any solution in ND_f but
     $s^n_4 = 18 = \phi_{inf}$, it does not improve ϕ_{inf}, it is therefore not worth to test its

efficiency.
The order of items $3, 4, \ldots, 6$ according to O_2 is: $5, 3, 4, 6$.
$s^n = (16, 10, 3, 5) + (6 + 8, 11 + 12, 6 + 9, 4 + 9) = (30, 33, 18, 18)$.
$s^n_4 = 18 = \phi_{inf}$.

3. Removing states:

   - Calculate upper bounds of states of C^2:
     State s^0: $(u_1, u_2) = (32, 29)$, it is not dominated. $u_\phi = 19 > \phi_{inf}$, so s^0 can not be eliminated from C^2.
     State s^2: $(u_1, u_2) = (44, 34)$, it is not dominated by any solution in ND_f, $u_\phi = 22 > \phi_{inf}$, the state s^2 is maintained in C^2.

- Step 3.
  $k = 3$, $C^3 = \emptyset$, $M^3 = \emptyset$.

1. Generate and trim states:
   $C^2 = \{s^0, s^2\} = \{(0, 0, 0, 0), (16, 10, 3, 5)\}$. s^0 and s^2 can be included in C^3 since $s^0_3 + \sum_{j=3}^{6} w_j = 28 > W$ and $s^2_3 + \sum_{j=3}^{6} w_j = 31 > W$.
   We generate $s^{3(1)}$ from s^0, $s^{3(1)} = (0, 0, 0, 0) + (8, 12, 9, 9) = (8, 12, 9, 9)$.
   $C^3 := \{(0, 0, 0, 0), (8, 12, 9, 9)\}$, $M^k := \{(8, 12, 9, 9)\}$.
   We generate $s^{3(2)}$ from s^2, $s^{3(2)} = (16, 10, 3, 5)+(8, 12, 9, 9) = (24, 22, 12, 14)$.
   $C^3 := C^3 \cup \{s^2, s^{3(2)}\} = \{(0, 0, 0, 0), (16, 10, 3, 5), (8, 12, 9, 9), (24, 22, 12, 14)\}$.
   $M^3 := M^3 \setminus \{(s^{(1)}\} \cup \{s^{3(2)}\} = \{(24, 22, 12, 14)\}$.
2. Generate extensions:
   State $s^{3(2)}$: the order of items $4, 5, 6$ according to O_1 is: $4, 6, 5$.
   $s^n = (24, 22, 12, 14) + (12, 6, 5, 6) = (38, 28, 17, 20)$. $s^n_4 > \phi_{inf}$, then we test the efficiency of s^n by solving the problem:

$$(P_\Psi)\begin{cases} \max V = \psi_1 + \psi_2 \\ 15x_1 + 16x_2 + 8x_3 + 12x_4 + 6x_5 + 14x_6 = \psi_1 + 38 \\ 3x_1 + 10x_2 + 12x_3 + 6x_4 + 11x_5 + 7x_6 = \psi_2 + 28 \\ 8x_1 + 3x_2 + 9x_3 + 5x_4 + 6x_5 + 8x_6 \le 19 \\ x_j \in \{0, 1\},\ j = 1, 2, \ldots, 6;\ \ \psi_1, \psi_2 \text{ are real non negative.} \end{cases}$$

   Let x^* the optimal solution of (P_ψ), $V = 0$, so the state s^n is efficient, thus x^* is efficient, $x_{opt} := (0\,1\,1\,1\,0\,0)'$, $\phi_{inf} = s^n_4 = 20$.
   $E_f := E_f \cup \{x^*\} = \{(1\,1\,0\,0\,0\,1)', (0\,1\,1\,0\,1\,0)', (0\,1\,1\,1\,0\,0)'\}$, $ND_f := ND_f \cup \{(s^n_1, s^n_2)\} = \{(45, 20), (30, 33), (36, 28)\}$.
   The order of items $4, 5, 6$ according to O_2 is: $5, 4, 6$. Thus $s^n = (24, 22, 12, 14)+ (6, 11, 6, 4) = (30, 33, 18, 18)$, $s^n_4 < \phi_{inf}$.
3. Removing states:
   - Calculate upper bounds of states in C^3:
     State s^0: $(u_1, u_2) = (32, 24)$ which is dominated by $(36, 28)$, so $C^3 := C^3 \setminus \{s^0\}$.

State s^1: $(u_1, u_2) = (48, 34)$, it is not dominated so we calculate u_ϕ, $u_\phi = 22 > \phi_{inf}$, state s^1 is maintained in C^3.
State $s^{3(1)}$: $(u_1, u_2) = (26, 29)$, it is dominated by $(30, 33)$ thus it is eliminated.
$C^3 := C^3 \setminus \{s^{3(1)}\}$.
State $s^{3(2)}$: $(u_1, u_2) = (38, 33)$, it is not dominated, $u_\phi = 21 < \phi_{inf}$, state $s^{3(2)}$ is maintained in C^3.

All the steps are summarized in table 1.

Table 1: The obtained results at each step

Step	C^k	M^k	Removed states	x_{opt}	ϕ_{opt}
1	$\{(0,0,0,0),(15,3,8,2)\}$	$\{(15,3,8,2)\}$	$\{(15,3,8,2)\}$	$(0\,1\,1\,0\,1\,0)'$	18
2	$\{(0,0,0,0),(16,10,3,5)\}$	$\{(16,10,3,5)\}$	$\{\}$	$(0\,1\,1\,0\,1\,0)'$	18
3	$\{(0,0,0,0),(16,10,3,5),$ $(8,12,9,9),(24,22,12,14)\}$	$\{(24,22,12,14)\}$	{(0,0,0,0),(8,12,9,9)}	$(0\,1\,1\,1\,0\,0)'$	20
4	$\{(16,10,3,5),(28,16,8,11),$ $(24,22,12,14),(36,28,17,20)\}$	$\{(36,28,17,20)\}$	$\{(16,10,3,5),(24,22,12,14)\}$	$(0\,1\,1\,1\,0\,0)'$	20
5	$\{(28,16,8,11),(34,27,14,15),$ $(36,28,17,20)\}$	$\{(36,28,17,20)\}$	$\{(28,16,8,11),(34,27,14,15)\}$	$(0\,1\,1\,1\,0\,0)'$	20
6	$\{(36,28,17,20)\}$	$\{(36,28,17,20)\}$	$\{\}$	$(0\,1\,1\,1\,0\,0)'$	20

The algorithm terminates with: $x_{opt} = (0\,1\,1\,1\,0\,0)'$, $\phi_{opt} = 20$,
$E_f = \{(1\,1\,0\,0\,0\,1)', (0\,1\,1\,0\,1\,0)', (0\,1\,1\,1\,0\,0)'\}$, $ND_f = \{(45, 20), (30, 33), (36, 28)\}$.
The set of all efficient solutions of this BOKP problem is:
$E = \{(1\,1\,0\,0\,0\,1)', (0\,1\,0\,1\,0\,1)', (1\,1\,0\,0\,1\,0)', (0\,1\,0\,0\,1\,1)', (0\,1\,1\,1\,0\,0)', (0\,1\,1\,0\,1\,0)'\}$,
with $ND = \{(45, 20), (42, 23), (37, 24), (36, 28), (30, 33)\}$

4. EXPERIMENTAL STUDY

In this section we describe the experiments implemented to asses the performance of the algorithm described above. However, in order to do so, we compare it against the algorithm proposed by Jorge [19]. This algorithm was developed for optimizing a linear function over the efficient set of a multiobjective integer program which can be adapted to the problem (2). In this study we are mainly interested on the percentage of nondominated solutions computed before reaching an optimal solution, on the total number of nondominated solutions of an instance and cpu time spent by each algorithm. The following notations are adopted:

- $|ND|$: number of nondominated criterion vectors of an instance.
- $|ND_f|$: number of nondominated criterion vectors browsed before reaching an optimal solution.

- $time$: computational time.
- $\%ND_f$: percentage of nondominated criterion vectors browsed.
  $\%ND_f = \frac{|ND_f|}{|ND|} \times 100$.

4.1. Jorge's algorithm

For solving the problem (P_E) according to Jorge [19] we adapt the following steps:

- Step 0. Initialisation
  Let $\phi^l = -\infty$, $\phi^u = +\infty$ a lower and an upper bound of ϕ respectively, $ND_f = \emptyset$, $l = 1$ the number of iterations.
  Solve $R \equiv max\{\phi(x)/x \in X\}$, if R is infeasible $\Rightarrow$ STOP, (P_E) is infeasible. Otherwise, let x^l be an optimal solution of R.
- Step 1. If $x^l \in E \Rightarrow$ STOP. $ND_f \leftarrow ND_f \cup \{Z_1(x^l)\ Z_2(x^l)\}$, $x_{opt} = x^l$ is an optimal solution of (P_E). Otherwise, set $\phi^u = \phi(x^l)$ and go to Step 2.
- Step 2. Find $\hat{x}^l \in E$ such that $\hat{x}^l \underline{\Delta} x^l$ and let $\bar{x}^l$ an optimal solution of the problem: $T_l \equiv max\{\phi(x)/Z_i(x) = Z_i(\hat{x}^l), i = 1, 2,\ x \in X\}$. If $\phi(\bar{x}^l) > \phi^l$ set $\phi^l = \phi(\bar{x}^l)$ and $x_{opt} = \bar{x}^l$. $ND_f \leftarrow ND_f \cup \{Z_1(\bar{x}^l)\ Z_2(\bar{x}^l)\}$, if $\phi^u = \phi^l \Rightarrow$ STOP. x_{opt} is an optimal solution of (P_E).
- Step 3. Let $R_l \equiv max\{\phi(x)/x \in X - \bigcup_{h=1}^{l} L_h\}$, where $L_h = \{x \in \{0,1\}^n / Z_i(\bar{x}^h) \geq Z_i(x), i = 1, 2$ with at least one strict inequality$\}$.
  If R_l is infeasible $\Rightarrow$ STOP. x_{opt} is an optimal solution of (P_E). Otherwise, let x^{l+1} be an optimal solution of R_l. If $\phi(x^{l+1}) \leq \phi^l \Rightarrow$ STOP. x_{opt} is an optimal solution of (P_E). Otherwise, set $l = l + 1$ and go to step 1.

4.2. Computational results

The experiments were performed on an Intel(R) Core I5, 2.30 GHz with 4Go of RAM. Our algorithm as well as Jorge's algorithm have been implemented in MATLAB R2017a under 64-bit Windows 10. We used the Cplex 12.9 library (academic version) for solving integer programming problems. A set of instances named 1B with large number of non-dominated solutions collected in MOCOlib library
http://xgandibleux.free.fr/MOCOlib/MOKP.html is considered. The set 1B contains 40 instances divided on four classes $1B/A$, $1B/B$, $1B/C$ and $1B/D$.
Class 1B/A The weights and the profits are uniformly generated in $[1, 100]$.
Class 1B/B Created from 1B/A by replacing the second vector of profits by the first one in reverse order.
Class 1B/C The vector of profits is uniformly generated in $[1, 100]$ with plateaus of values of length $l \in [1, 0.1 \times n]$. The weights are generated independently following a uniform distribution in $[1, 100]$, it is an instance with repeated profits.
Class 1B/D Created from the previous class by replacing the second vector of profits by the first one in reverse order.

The main criterion ϕ is generated randomly according to a uniform distribution in $[1, 100]$ for each instance.

The results of the experimental study are summarised in the following table.

Table 2: Computational results of the algorithms on set 1B of instances

Class	Instance	$\|ND\|$	Jorge's algorithm			Proposed algorithm		
			$\|ND_f\|$	$time(s)$	$\%ND_f(\%)$	$\|ND_f\|$	$time(s)$	$\%ND_f(\%)$
A	$2kp50$	34	15	3.21	44.12	5	0.69	14.71
	$2kp100$	172	52	27.12	30.23	10	19.20	5.82
	$2kp150$	244	116	128.40	47.54	7	91.33	2.87
	$2kp200$	439	165	401.82	37.59	10	229.01	2.28
	$2kp250$	629	224	571.94	35.61	12	687.11	1.91
	$2kp300$	713	247	999.87	34.64	13	998.52	1.83
	$2kp350$	871	332	3100.78	38.12	11	2294.24	1.26
	$2kp400$	1000	421	4555.64	42.10	8	4280.01	0.80
	$2kp450$	1450	596	5984.29	41.10	14	5907.08	0.97
	$2kp500$	1451	630	9990.68	43.42	13	9368.45	0.90
B	$2kp50$	52	17	2.29	38.60	3	0.08	5.77
	$2kp100$	174	50	29.87	28.74	10	18.04	5.75
	$2kp150$	348	129	180.00	37.07	11	105.88	3.16
	$2kp200$	398	164	599.22	41.21	9	391.80	2.27
	$2kp250$	629	271	1111.58	43.08	14	761.01	2.23
	$2kp300$	617	233	1205.30	37.76	11	1047.98	1.79
	$2kp350$	955	414	1999.54	43.35	10	2398.01	1.05
	$2kp400$	1109	506	5094.68	45.63	12	4918.33	1.09
	$2kp450$	1626	759	11029.83	46.68	15	8446.18	0.93
	$2kp500$	1669	800	13036.48	47.93	20	9987.39	1.20
C	$2kp50$	66	20	2.98	30.30	3	0.67	4.55
	$2kp100$	64	23	17.55	35.94	8	11.81	12.50
	$2kp150$	166	67	91.81	40.36	7	60.18	4.22
	$2kp200$	328	134	403.47	40.85	11	248.68	3.36
	$2kp250$	528	247	800.39	46.78	16	542.98	3.03
	$2kp300$	400	246	960.11	61.50	12	733.39	3.00
	$2kp350$	845	375	2924.58	44.38	10	1299.30	1.19
	$2kp400$	946	436	3900.51	46.02	12	3125.97	1.27
	$2kp450$	655	276	3694.58	42.14	12	5210.00	1.83
	$2kp500$	522	349	6986.35	66.68	10	8889.16	1.92
D	$2kp50$	69	16	2.50	23.19	6	0.99	8.70
	$2kp100$	76	26	11.09	34.21	4	9.07	5.26
	$2kp150$	132	41	17.00	31.06	5	16.98	3.79
	$2kp200$	361	152	370.91	42.11	14	244.80	3.88
	$2kp250$	424	187	673.35	44.10	13	671.08	3.07
	$2kp300$	214	114	501.37	53.27	9	793.55	4.21
	$2kp350$	472	223	930.22	47.25	18	1005.28	3.82
	$2kp400$	1670	661	9205.70	39.58	21	8947.24	1.26
	$2kp450$	398	215	3651.25	54.02	7	4877.36	1.76
	$2kp500$	654	330	7000.27	50.46	9	9262.01	1.38

Table 3: Mean time on set 1B of instances

n	50	100	150	200	250	300	350	400	450	500
$Mean\ time(s)$	0.60	14.53	68.59	278.57	665.54	893.36	1749.20	5317.88	6110.15	9376.75

The results of Table 2 show that our algorithm generates only between 0.8% and 14.71% of nondominated solutions before reaching an efficient solution that optimizes the main criterion. Jorge's algorithm generates between 23.19% and 66.68%. For all instance classes, our algorithm computes fewer number of nondominated solutions, as shown in the figures 1, 2, 3 and 4. Considering cpu time, we can observe that, out of the 40 instances tested, Jorge's algorithm performs better than our method on only 8 of them. However, our approach is better on either 32 instances. We can also see that our algorithm consistently outperforms Jorge's one for large nondominated sets, as indicated by instances with up to 1000 nondominated solutions.

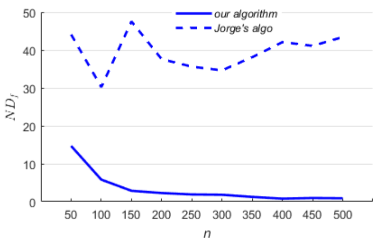

Figure 1: Percentage of nondominated solutions computed by the methods (Class A)

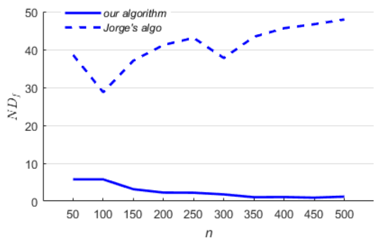

Figure 2: Percentage of nondominated solutions computed by the methods (Class B)

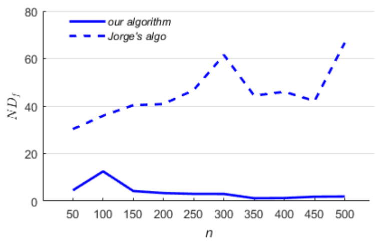

Figure 3: Percentage of nondominated solutions computed by the methods (Class C)

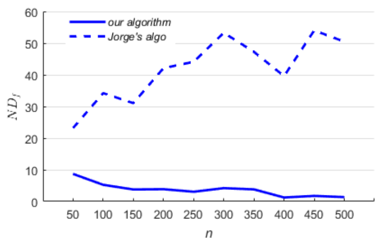

Figure 4: Percentage of nondominated solutions computed by the methods (Class D)

We analyse in figure 5, the computational time of our method, it increases with the dimension of the instance for the four classes. In addition, when the number of efficient solutions becomes much larger, the execution time grows significantly (instance 1B/D 2kp400).

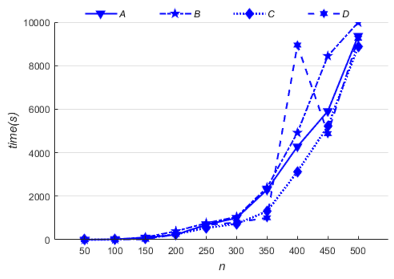

Figure 5: Computational time of our algorithm on set 1B

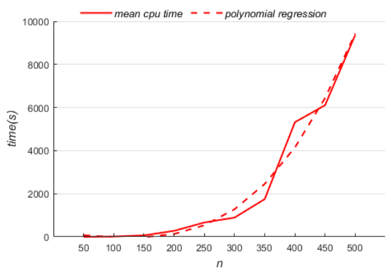

Figure 6: Complexity estimation of our algorithm

The proposed method's runtime complexity can be estimated as a polynomial function $\mathcal{C}(n) = \mathcal{P}(n)$ for the tested instances. We obtain $\mathcal{P}(n) = 0.0001n^3 - 0.0209n^2 - 0.6050n + 143.3234$ by performing a polynomial regression in MATLAB between the problem size $(n = 50, 100, \ldots, 500)$ and the mean cpu time of each size on the four classes (see table 3). The mean time of our method and the function $\mathcal{C}(n)$ are plotted in figure 6. Because they appear to be somewhat similar, we can conclude that the complexity of our method on the instances under consideration is polynomial.

5. CONCLUSION

In this work, we have developed a new exact algorithm specifically designed to optimize a linear function over the efficient set of a BOKP problem.

It is based on dynamic programming, the use of different relations to compare between states helps to remove early states that can not produce nondominated solutions improving the main criterion. An experimental study was carried out on several instances with large dimensions, and considerable cardinal of nondominated set. Reading the results of our experimentation shows that the proposed method outperforms in terms of cpu time 80 percent the algorithm of Jorge on the studied instances. Another advantage resides in the fact that our method can be easily extended to other multiobjective combinatorial optimization problems, particularly with quadratic objective functions.

Performing an item order before solving the problem and including other comparison relations to improve both quality and cpu time together are the main subject of our future research.

Funding. This research received no external funding.

REFERENCES

[1] E.L., Ulungu, *Optimisation combinatoire multicritre: Détermination de l'Ensemble des Solutions Efficaces et Méthodes Interactives*, these de doctorat, universitéde Mons-Hainaut, octobre 1993.

[2] M. Visée, J. Teghem, M. Pirlot, and EL. Ulungu, "Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem", *Journal of Global Optimization*, vol. 12, pp. 139-55, 1998.

[3] E. Captivo, J. Climaco, J. Figueira, E. Martins and J. L. Santos, "bicriteria 0-1 knapsack problems using a labeling algorithm", *Computers & Operational Research*, vol. 30, pp. 1865-1886, 2003.

[4] C. Bazgan, H. Hugot and D. Vanderpooten, "Solving efficiently the 0-1 multi-objective knapsack problem", *Computers & Operations Research*, vol. 36, pp. 260-279, 2009.

[5] J. Figueira, L. Paquete, M. Simões and D. Vanderpooten, "Algorithmic improvements on dynamic programming for the bi-objective {0,1} knapsack problem". *Comput Optim Appl*, vol. 56, pp. 97-111, 2013.

[6] P. Correia, L. Paquete and J. Figueira, "Compressed data structures for bi-objective {0,1}-knapsack problems", *Computers and Operations Research*, vol. 89, pp. 82-93, 2018.

[7] T. Erlebach, H. Kellerer and U. Pferschy, "Approximating multi-objective knapsack problems", *Management Sciences*, vol. 48, pp. 1603-1612, 2002.

[8] C. Bazgan, H. Hugot and D. Vanderpooten, "Implementing an efficient fptas for the 0-1 multi-objective knapsack problem", *European Journal of Operational Research*, vol. 198, pp. 47-56, 2009.

[9] J. Jorge, X. Gandibleux and M. Wiecek, *A priori reduction of the size of the binary multiobjective knapsack problem*, MOPGP'08 Multi-Objective Programming and Goal Programming, Portsmouth, UK, 2008.

[10] M. Daoud and D. Chaabane, “New reduction strategy in the biobjective knapsack problem”, *Intl. Trans. in Op. Res*, vol. 25, pp. 1739-1762, 2016.

[11] H.P. Bensen, “Optimization over the Efficient Set”, *J. Math. Anal. Appl*, vol. 98, pp. 562-580, 1984.

[12] H.P. Bensen, “A finite Non adjacent Extreme Point Search Algorithm over the Efficient Set”, *J. Opt. Theor. Appl*, vol. 73, pp. 47-64, 1992.

[13] H.P. Benson and S. Sayin, “Optimizing over the Efficient Set: Four Special Cases”, *J. Optim. Theor. Appl*, vol. 80, pp. 3-18, 1994.

[14] J.G. Ecker and J.H. Song, “Optimizing a Linear Function over an Efficient Set”, *J. Optim. Theor. Appl*, vol. 83, pp. 541-563, 1994.

[15] S. Sayin, “Optimizing over the Efficient Set using a Top-Down Search of Faces”, *Oper. Res*, vol. 48, pp. 65-72, 2000.

[16] Y. Yamamoto, “Optimization over the efficient set: Overview, dedicated to Professor Reiner Horst on his 60th birthday”, *Journal of Global Optimization*, vol. 22, pp. 285-317, 2002.

[17] N.C. Nguyen, *An algorithm for optimizing a linear function over the integer efficient set*, Konrad-Zuse-Zentrum fur Informations technik, Berlin, 1992.

[18] M. Abbas and D. Chaabane, “Optimizing a linear function over an integer efficient set”, *European Journal of Operational Research*, vol. 174, pp. 1140-1161, 2006.

[19] J.M. Jorge, “An algorithm for optimizing a linear function over an integer efficient set”, *European Journal of Operational Research*, vol. 195, pp. 98-103, 2009.

[20] J. Sylva and A. Crema,“A method for finding the set of non-dominated vectors for multiple objective integer linear programs”, *European Journal of Operational Research*, vol. 158, pp. 46-55, 2004.

[21] D. Chaabane and M. Pirlot, “A method for optimizing over the integer efficient set”, *Journal of Industrial and Management Optimization*, vol. 6, pp. 811-823, 2010.

[22] D. Chaabane, B. Brahmi and Z. Ramdani, “The augmented weighted Tchebychev norm for optimizing a linear function over an integer efficient set of a multicriteria linear program”, *International Transactions in Operational Research*, vol. 19, pp. 531-545, 2012.

[23] N. Boland, H. Charkhgard and M. Savelsbergh, “A new method for optimizing a linear function over the efficient set of a multiobjective integer program”, *European Journal of Operational Research*, vol. 260, pp. 904-919, 2017.

[24] S. Mahdi and D. Chaabane, “A linear fractional optimization over an integer efficient set”, *RAIRO Operations Research*, vol. 49, pp. 265-278, 2015.

[25] W. Drici, F.Z. Ouail and M. Moulai, “Optimizing a linear fractional function over the integer efficient set”, *Ann Oper Res*, vol. 267, pp. 135-151, 2017.

[26] J.G. Ecker and I.A. Kouada, “Finding Efficient Points for Multi-objective Linear Programs”, *Mathematical Programming*, vol. 8, pp. 375-377, 1975.

[27] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & sons, New York, 1990.