Optimizing over the efficient set of the binary bicriteria knapsack problem

Lachemi Nadia, Chaabane Djamal

Faculty of Mathematics, Department of Operations Research, Laboratory AMCD-RO
USTHB, Bab-Ezzouar, Algiers, Algeria
Nadia0989@hotmail.fr
chaabane_dj@yahoo.fr

Abstract: This paper deals with the problem of optimizing a linear criterion, that we call main criterion, over the efficient set of the 0/1 bicriteria knapsack problem. The resolution process is based essentially on dynamic programming. We obtain at the end of the algorithm a subset of efficient solutions including one which optimizes the main criterion without having to enumerate all the efficient solutions of the problem. A numerical experiment is reported. The percentage of the subset of efficient solutions obtained and the CPU time can proof the efficiency of the method.
Keywords: *Bicriteria knapsack problem, Optimizing, dynamic programming, efficient set, optimal solution.*

1 Introduction

The problem of optimizing a function over the efficient set of a multiple objective programming problem arises in a variety of applications. It has received the attention of many researchers and became one of the most important and interesting areas in multicriteria programming. The difficult of this problem is mainly due to the nonconvexity of the efficient solutions set. In the case of continuous variables, the problem has been intensively studied by many authors, see the overview [3]. When the decision variables are integers, there have been relatively few studies dealing with this case. For the first time Nguyen [7] made an attempt to optimize on the integer efficient set, then Abbas and al. [4], Chaabane and al [5], Jorge [6] and others have developed other approaches. Especially for the binary variables case, to our knowledge no study has been developed. In this work an algorithm is developed that optimizes an arbitrary linear function over the efficient set of a (BOKP) without explicitly having to enumerate all the efficient solutions. The proposed algorithm is based on dynamic programming instead of the linear programming and cut techniques which are the most used in the previous works. Consider the problem:

$$(BOKP)\begin{cases} max \quad Z_i(x) = \sum_{j=1}^{n} p_i^j x^j, \quad i = 1,2 \\ \sum_{j=1}^{n} w^j x^j \leq W, \quad j = \overline{1:n} \\ x_j \in \{0,1\}^n. \end{cases} \tag{1}$$

x^j is a decision variable, equals to 1 if the item j is in the knapsack and 0 otherwise.
w^j is the weight of object j.

p_i^j is the profit of object j on objective $i, i = 1, 2$.
W is the knapsack capacity.
All coefficients p_i^j, w^j and W are supposed to be positive integers, $j = \overline{1, n}, \quad i = 1, 2$.
To avoid trivial solutions, we suppose that: $w^j \leq W, \quad \forall j = 1, ..., n$, and $\sum_{j=1}^{n} w^j > W$.
The set of all feasible solutions is denoted by X.
Let $x, x' \in X$, x dominates x' if and only if $Z_i(x) \geq Z_i(x')$, $i = 1, 2$, with at least one strict inequality, we denote $(x \underline{\Delta} x')$.
Let x^* a feasible solution, it is said to be efficient for the $(BOKP)$ if and only if there is no other feasible solution which dominates it. Its corresponding vector $Z(x^*) = (Z_1(x^*), Z_2(x^*))$ is a nondominated vector.
The set of efficient solutions is denoted $E(BOKP)$, its corresponding set of nondominated vectors is denoted $ND(BOKP)$. The problem of optimization over the efficient set of the $(BOKP)$ is given by:

$$(P_E) \qquad max\{\phi(x) = \sum_{j=1}^{n} d^j x^j, j = \overline{1, n}; \quad x = (x^1, ..., x^n) \in E(BOKP)\}. \tag{2}$$

Where $\phi(x) = dx$ is a linear function and called "main objective".
In this work we suppose that: $d \in \mathbb{R}_+^n$.

2 Definitions and basic concepts

Let:

$$(BOKP_k) \begin{cases} max \quad Z_i(x) = \sum_{j=1}^{k} p_i^j x^j, \quad i = 1, 2 \\ \sum_{j=1}^{k} w^j x^j \leq W, \quad j = \overline{1:k} \\ x_j \in \{0, 1\}^k. \end{cases} \tag{3}$$

The (BOKP) induced by the k first items $(k = 1, ..., n)$.
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 at any stage k is defined by S^k, where:
$S^k = S^{k-1} \cup \{s_1^{k-1} + p_1^k, s_2^{k-1} + p_2^k, s_3^{k-1} + w^k, s_4^{k-1} + d^k / s_3^{k-1} + w^k \leq W, \ s^{k-1} \in S^{k-1}\}$.
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 stage S^n define the set of all feasible solutions of the (BOKP).
The notions of efficiency and dominance are also valid on the S^k since a state represents a feasible solution.
A state $s^n \in S^n$ is an **extension** of $s^k \in S^k (k \leq n)$ if and only if $s_i^n = s_i^k + \sum_{j \in J} p_i^j, i = 1, 2,$
$s_3^n = s_3^k + \sum_{j \in J} w^j$ and $s_4^n = s_4^k + \sum_{j \in J} d^j$, $J = \{j \in \{k+1, ..., n\}, \ s_3^k + \sum_{j \in J} w^j \leq W\}$.

- **Notations:** we denote by:
  $NDS_k(BOKP)$: the subset of non dominated solutions obtained at a stage k.
  ϕ_{inf}: a lower bound of the main objective ϕ.
  x_{opt}: optimal solution of the problem (P_E).
  ϕ_{opt}: optimal value of the main objective ϕ.

3 General description of our approach

The method consists on n stages. At each stage k, $k = 1, ..., n$, we generate and trim progressively a subset of states $C^k \subseteq S^k$ according to the previous subset of states C^{k-1}, with

$C^0 = S^0 = \{(0,0,0,0)\}$. We omit states that can't lead to efficient solution and among the remaining states we omit whose that can't improve the main objective value. At the end of a stage $k, k = 1, ..., n$, we get a reduced subset of states C^k, a subset of efficient solutions and an updated lower bound of the function ϕ. In order to compare between states we have conserved the two relations D_r^k and $D_{\underline{\Delta}}^k$ proposed by Bazgan and al. [1]. Another relation is proposed that we call Z_b^k, it is used to discard states whose upper bound are dominated by some efficient solutions. To discard states that can't conduce to an improvement of the function ϕ we propose the relation Φ_b^k.

3.1 Comparison between states

Let two states $s^k, \tilde{s}^k \in S^k$ at a stage $k, k = 1, ..., n$.
Relation D_r^k: This relatin is defined as follow:

$$s^k D_r^k \tilde{s}^k \Leftrightarrow \begin{cases} \tilde{s}^k \in S^{k-1}; \\ s^k = (\tilde{s}_1^k + p^1, \tilde{s}_2^k + p^2, \tilde{s}_3^k + w^k) \ \& \\ \tilde{s}_3^k \leq W - \sum_{j=k}^{j} w^j \end{cases} \tag{4}$$

This relation means that the only extension of s^k that can represents an efficient solution is the one that contains all the remanning objects, so it is not necessary to calculate other extensions, thus $\tilde{s}^k$ is omitted.
Relation $D_{\underline{\Delta}}^k$: The dominance notion defined on X is also valid on S^k, thus $s^k \underline{\Delta} \tilde{s}^k$ if and only if $s_i^k \geq \tilde{s}_i, i = 1, 2$. The relation $D_{\underline{\Delta}}^k$ is defined as:

$$s^k D_{\underline{\Delta}}^k \tilde{s}^k \Leftrightarrow s^k \underline{\Delta} \tilde{s}^k \ \ \& \ \ s_3^k \leq \tilde{s}_3^k, \ k < n \tag{5}$$

Let $u = (u_1, u_2, u_\phi)$ an upper bound of a feasible solution represented by a state s^k. Where u_1, u_2 is the upper bound on objective 1 and 2 respectively, u_ϕ the upper bound on objective ϕ. u is calculated as it was proposed by Martello and al.[2].
Relation Φ_b^k: Let ϕ_{inf} 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.
Relation Z_b^k: If $\exists s = (s_1, s_2) \in NDS_k(BOKP)$ such that $(s_1, s_2) \underline{\Delta} (u_1, u_2)$, thus s^k can't let to efficient solution, so it can be eliminated.

3.2 Finding a lower bound of the main objective ϕ

To determine a lower bound of the main objective ϕ, we must find at least one efficient solution of the $(BOKP)$, we propose to calculate the two extreme efficient solutions x^1 and x^2 according to the lexicographic method. $\phi_{inf} = max\{\phi(x^1), \phi(x^2)\}$.

3.3 Efficiency test

To test the efficiency of solutions, we apply the theorem 3.1 proposed by Benson in [8].

3.4 Algorithm

The general step k, $k = 1, ..., n-1$ of the algorithm can be summarised as follow:

1. Construct the set C^k by generating progressively states from C^{k-1} with respect to the relations D_r^k and $D_{\underline{\Delta}}^k$.

2. Set $M^k \subseteq C^k$ the subset of nondominated states according to the relation $\underline{\Delta}$.

3. Generate extensions for M^k states and maintain just P^k the subset of nondominated extensions which improve the main objective value. They represent potentially nondominated solutions of the $(BOKP)$.

4. Test the efficiency of the solutions represented by P^k states, update $NDS_k(BOKP)$, ϕ_{inf}.

5. For each state in C^k, calculate its upper bound on the main objective ϕ.

6. Eliminate from C^k states dominated with respect to the relation Φ_b^k, update C^k.

7. For each state in C^k, calculate its upper bound on objective 1 and 2.

8. Eliminate from C^k states dominated with respect to the relation Z_b^k, update C^k.

At the final step n we we apply steps 1 and 2. Let M^n the subset of nondominated states according to the relation $\underline{\Delta}$, only states improving the function ϕ are kept and tested for efficiency. Update $NDS_n(BOKP)$ which represents the subset of non dominated solutions of the $(BOKP)$, *xopt* the optimal solution of the problem (P_E) and ϕ_{opt} its corresponding value on objective ϕ.

4 Conclusion

In this paper, we presented an exact algorithm for optimizing a linear function on the efficient set of a $(BOKP)$ which is based essentially on dynamic programming. Our objective was to reach the optimal solution without having to enumerate all the efficient solutions of the problem. The experimental study shows that a very small percentage of efficient solutions is browsed. The computing time is admissible but it grows very fast with the number of efficient solutions of the problem. To improve our method, we think to built an efficient item order which helps to speed up the resolution process.

References

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

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

[3] Y. Yamamoto. *Optimization over the efficient set:* Overview, dedicated to Professor Reiner Horst on his 60th birthday. Journal of Global Optimization 22, 1-4, 285-317, 2002.

[4] M. Abbas, D. Chaabane. *Optimizing a linear function over an integer efficient set.* European Journal of Operational Research 174, 2, 1140-1161, 2006.

[5] D. Chaabane, M. Pirlot. *A method for optimizing over the integer efficient set.* Journal of Industrial and Management Optimization 6, 4, 811-823, 2010.

[6] J.M. Jorge. *An algorithm for optimizing a linear function over an integer efficient set.* European Journal of Operational Research 195, 1, 98-103, 2009.

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

[8] H.P. Benson, *Existence of efficient solutions for vector maximization problems.* Journal of Optimization Theory and Applications 26, 569-580, 1978.