An Exact Algorithm for the Multiple-choice Multidimensional Knapsack Problem

Mhand Hifi, Slim Sadfi, Abdelkader Sbihi

UNIVERSITÉ PARIS 1 – PANTHÉON - SORBONNE –

CERMSEM

UMR CNRS 8095

Cahiers de la MSE

An Exact Algorithm for the Multiple-choice Multimensional Knapsack Problem

Mhand HIFI

Slim SADFI

Abdelkader SBIHI

2004.24

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Maison des Sciences Économiques, 106-112 boulevard de L'Hôpital, 75647 Paris Cedex 13

ISSN : 1624-0340

An Exact Algorithm for the Multiple-choice Multidimensional Knapsack Problem

Mhand HIFI*† Slim SADFI*† Abdelkader SBIHI†

Email: {Mhand.Hifi,Abdelkader.Sbihi}@univ-paris1.fr; se.sad@voila.fr

Abstract. In this paper, we propose an optimal algorithm for the Multiple-choice Multidimensional Knapsack Problem MMKP. The main principle of the approach is twofold: (*i*) to generate an initial solution, and (*ii*) at different levels of the tree search to determine a new upper bound used with a best-first search strategy. The developed method was able to optimally solve the MMKP. The performance of the exact algorithm is evaluated on a set of small and medium instances. This algorithm is parallelizable and it is one of its important feature.

Keywords. *Combinatorial optimization, Branch and bound, Sequential algorithm, Knapsack.*

1 Introduction

In this article, we are concerning with a variant of the knapsack problem, the Multipe-choice Multidimensional Knapsack Problem MMKP. MMKP is an NP-Hard problem which models many practical and real life problems. We cite the problem of quality adaptation and admission control for interactive multimedia systems ([2]), or service level agreement management in telecommunication networks problems ([20]). In the MMKP, we are given n classes J_i of items, where each class J_i, $i = 1, \ldots, n$, has r_i items. Each item j, $j = 1, \ldots, r_i$, of class J_i has the nonnegative profit value v_{ij}, and requires resources given by the weight vector $W_{ij} = (w_{ij}^1, w_{ij}^2, \ldots, w_{ij}^m)$ and where each weight component w_{ij}^k, $k = 1, \ldots, m$ also is a nonnegative value. The amounts of available resources are given by a vector $C = (C^1, C^2, \ldots, C^m)$. The aim of the MMKP is to pick exactly one item from each class in order to maximize the total profit value of the pick, subject to the resource constraints. Formally, the MMKP can be stated as follows:

*LaRIA, 5 rue du Moulin Neuf, 80000 Amiens, France.
†CERMSEM UMR-CNRS 8095, MSE, 106-112, Bd de l'Hôpital, 75013 Paris, France.

$$(MMKP)\begin{cases} \text{maximize} & Z(x) = \sum_{i=1}^{n}\sum_{j=1}^{r_i} v_{ij}x_{ij} & \\ \text{subject to} & \sum_{i=1}^{n}\sum_{j=1}^{r_i} w_{ij}^k x_{ij} \leq C^k, & k \in \{1,\ldots,m\} \\ & \sum_{j=1}^{r_i} x_{ij} = 1, & i \in \{1,\ldots,n\} \\ & x_{ij} \in \{0,1\}, & i \in \{1,\ldots,n\},\ j \in \{1,\ldots,r_i\} \end{cases}$$

The variable x_{ij} is either equal to 0, implying item j of the i-th class J_i is not picked, or equal to 1 implying item j of the i-th J_i class is picked.

This problem may be considerd as a generalization of the well known Multiple-Choice Knapsack Problem MCKP (for more details, see Nauss [13] and Pisinger [16]).

The paper is organized as follows. First, we present a brief reference of some related works on the classical knapsack problem and its variants dealing with sequential exact and approximate algorithms. Second, we modify the original problem MMKP to an auxiliary problem MMKP$_{aux}$. This transformation permits to us to develop the computing of an upper bound for the auxiliary problem. We establish the result for which an upper bound for MMKP$_{aux}$ also is an upper bound for the original problem MMKP. Third, we prove the finitude of the approach and show the principle of search tree exploration. We, then, present the main principles of the branch and bound algorithm EMKP and we prove its exactness. Fourth and finally, different sets of instances of different sizes and densities are generated where we run EMKP to optimally solve them.

2 Literature survey

There exist several approaches for solving the knapsack problem KP and its variants. For the (un)bounded single constraint KP, a large variety of solution methods have been proposed (see Martello *et al.* [11], Balas and Zemel [1], Fayard and Plateau [6] and Pisinger [15]). The problem has been solved optimally and approximately by dynamic programming, or by the search tree procedures or other hybrid approaches.

Another problem, namely the max-min allocation problem, has been studied (see Luss [10] and, Pang and Yu [14]). Different exact and approximate approaches have been tailored especially for this problem. The Multi-Dimension Knapsack Problem (MDKP) is one kind of KP where the constraints are multidimensional (see Chu and Beasley [3]). The Multiple-Choice Knapsack Problem (MCKP) is another variant of KP where the picking

criterion of items is more restrictive (see Nauss [13]). For MDKP, Toyoda [19] used the aggregate resource consumption. The solution of the MDKP needs iterative picking of items until the resource constraint is violated. Other approaches have been used with great success, achieved via the application of local search techniques and metaheuristics to MDKP. Among these approaches, we can cite the tabu search, genetic algorithms, simulated annealing and hybrid algorithms (for more details the reader can refer to Chu and Beasley [3]).

To our knowledge, very few papers dealing directly with the MMKP are available. Moser *et al.* [12] have designed an approach based upon the concept of graceful degradation from the most valuable items based on Lagrange multipliers. Khan *et al.* [8] have tailored an algorithm based on the aggregate resources already introduced by Toyoda [19] for solving the MDKP. Finally, Hifi *et al.* [7] proposed a guided local search-based heuristic in which the trajectories of the solutions were oriented by increasing the cost function with a penalty term; it penalizes bad features of previously visited solutions.

In this paper, we present a branch and bound algorithm to optimally solve the MMKP problem. The main principle of the approach is twofold: (i) on one hand to generate an initial solution, and (ii) to determine a new upper bound used at different levels of the exploration with a best-first search strategy.

3 An upper bound for the Multiple-choice Multidimensional Knapsack Problem

First, we consider the following modified problem of MMKP which we call an auxiliary problem to the MMKP and which we note MMKP_{aux}.

$$(MMKP_{aux}) \begin{cases} \text{maximize} & Z(x) = \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{r_i} v_{ij}x_{ij} & \\ \text{subject to} & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{r_i} w_{ij}x_{ij} \leq C & \\ & \displaystyle\sum_{j=1}^{r_i} x_{ij} = 1, & i \in \{1,\ldots,n\} \\ & x_{ij} \in \{0,1\}, & i \in \{1,\ldots,n\},\ j \in \{1,\ldots,r_i\}. \end{cases}$$

Where $C = \displaystyle\sum_{k=1}^{m} C^k$ represents the sum of the m capacity constraints and $w_{ij} = \displaystyle\sum_{k=1}^{m} w_{ij}^k$ represents the m weigths sum of the j-th item of the i-th class J_i. Also, let $NR = \displaystyle\sum_{i=1}^{n} r_i - n$.

We also can remark that the above MMKP$_{aux}$ formulation also represents the MCKP problem.

Let now UB be the expression that we compute with relation to the MMKP$_{aux}$ as follows :

1. For each class J_i, $i = 1, \ldots, n$, we select the item $j \in J_i$, $j = 1, \ldots, r_i$, which realizes the most valuable (the biggest) ratio of the profit by weight $\frac{v_{ij}}{w_{ij}}$. Next, we consider in the solution this item noted j_{max} for each class J_i.

2. Let $R_{max} = \sum_{i=1}^{n} w_{ij_{max}}$ (resp. $V_{max} = \sum_{i=1}^{n} v_{ij_{max}}$) be the cumulated used weights (resp. the cumulated used profits) of all the j_{max} fixed items of each class J_i, $i = 1, \ldots, n$. We then can distinguish two cases :

   - $R_{max} > C$. In this case, the upper bound value for MMKP$_{aux}$ is given by :

   $$UB = \sum_{i=1}^{n} v_{ij_{max}} \times (\frac{C}{\sum_{i=1}^{n} w_{ij_{max}}}).$$

   - $R_{max} < C$. In this case, we consider all the remaining items as belonging to only one class (we merge all the classes J_i, $i = 1, \ldots, n$, into one class J). We obtain a class $J = \{1, \ldots, j, \ldots, NR\}$ with items indexed by $j = 1, \ldots, NR$. Then, we sort the items in the classical decreasing order of the ratio $\frac{v_j}{w_j}$ for $j = 1, \ldots, NR$. The problem generated by the set of these items represents a knapsack problem KP$_{C-R_{max}}$ with a capacity constraint of $C - R_{max}$. We, then compute UB$_d$ as the relative Dantzig [5] upper bound of KP$_{C-R_{max}}$. This means that we fill the knapsack with the ordered items j until its constraint is violated.
   Let, now ℓ such that $1 \leq \ell \leq NR$ be a particular item which violates the capacity constraint $C - R_{max}$ and defined in the following :

   $$\ell = min\{j : \sum_{j=1}^{\ell-1} w_j \leq C - R_{max} < \sum_{j=1}^{\ell} w_j\}.$$

   This item ℓ is called the critical item of the knapsack KP$_{C-R_{max}}$.
   With relation to the critical element ℓ, the Dantzig upper bound UB$_d$ of KP$_{C-R_{max}}$ is computed as follows :

   $$UB_d = \sum_{j=1}^{\ell-1} v_j + \Big(\frac{(C - R_{max}) - \sum_{j=1}^{\ell-1} w_j}{w_\ell}\Big) \times v_\ell.$$

After computing the Dantzig upper bound UB_d for the knapsack problem with the remaining items, the MMKP_{aux} upper bound UB is, in this case, given by :

$$UB = V_{max} + UB_d.$$

Proposition 1 *UB is an upper bound for the auxiliary problem $MMKP_{aux}$.*

Proof 1 *We will show by the absurd the first case ($R_{max} > C$) of this proposition. For to do, we may recall the following property :*
Let (a, c, x, z) be some nonnegative numbers and (b, d, y, t) be some strictely nonnegative numbers.

$$If \ \frac{a}{b} \geq \frac{c}{d} \ and \ \frac{x}{y} \geq \frac{z}{t} \ (1)$$

$$then \ \frac{a+x}{b+y} \geq \frac{c+z}{d+t} \qquad (2).$$

One supposes that there exists a solution $\overline{X} = (\overline{x}_{1j_1}, \ldots, \overline{x}_{ij_i}, \ldots, \overline{x}_{nj_n})$ for the $MMKP_{aux}$ with objective value $\overline{V} = \sum_{i=1}^{n} v_{ij_i}$ such that :

$$V_{max} \times (\frac{C}{R_{max}}) < \overline{V}.$$

$$Then \ \frac{V_{max}}{R_{max}} < \frac{\overline{V}}{C}.$$

$$Now \quad \frac{\overline{V}}{C} \leq \frac{\overline{V}}{\overline{R}}, \ since \ \overline{R} = \sum_{i=1}^{n} w_{ij_i} \leq C.$$

$$So \quad \frac{V_{max}}{R_{max}} < \frac{\overline{V}}{C} \leq \frac{\overline{V}}{\overline{R}}.$$

$$The \ last \ double \ inequality \ implies \ that \quad \frac{V_{max}}{R_{max}} < \frac{\overline{V}}{\overline{R}}.$$

According to the relations $(1) - (2)$ and the decreasing order of the ratios of the profits by weights of the items :

$$\frac{\sum_{i=1}^{n} v_{ij_{max}}}{\sum_{i=1}^{n} w_{ij_{max}}} \geq \frac{\sum_{i=1}^{n} v_{ij_i}}{\sum_{i=1}^{n} w_{ij_i}} \ since \ \frac{v_{ij_{max}}}{w_{ij_{max}}} \geq \frac{v_{ij_i}}{w_{ij_i}} \ \forall i = 1, \ldots, n.$$

Consequently, having $\frac{\overline{V}}{\overline{R}}$ is the biggest ratio and by the same $UB \leq \overline{V}$ is absurd.

For the second case ($R < C$), it is easy to see that the remaining items represent a single knapsack problem and, by the continuation, the calculated upper bound is the boundary of Dantzig [5].

Proposition 2 *UB is an upper bound for the original problem MMKP.*

Proof 2 *Let $\overline{x}$ be an optimal solution for the MMKP, i.e:*

$$\begin{cases} & Z_{MMKP}(\overline{x}) = \sum_{i=1}^{n}\sum_{j=1}^{r_i} v_{ij}\overline{x}_{ij} & \\ \text{such that} & \sum_{i=1}^{n}\sum_{j=1}^{r_i} w_{ij}^k\overline{x}_{ij} \leq C^k, & k = 1,\ldots,m \\ & \sum_{j=1}^{r_i}\overline{x}_{ij} = 1, & i = 1,\ldots,n \\ & \overline{x}_{ij} \in \{0,1\}. & \end{cases}$$

$$\text{Since } \sum_{i=1}^{n}\sum_{j=1}^{r_i} w_{ij}^k\overline{x}_{ij} \leq C^k,\ \forall k = 1,\ldots,m, \text{ then } \sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{r_i} w_{ij}^k\overline{x}_{ij} \leq \sum_{k=1}^{m} C^k.$$

The last inequality implies that:

$$\sum_{i=1}^{n}\sum_{j=1}^{r_i} w_{ij}\overline{x}_{ij} \leq C.$$

From which, $\overline{x}$ is a feasible solution for the modified problem $MMKP_{aux}$.
Now $\forall x,\ Z_{MMKP_{aux}}(x) \leq UB$, consequently and in particular for $\overline{x}$, we have the result :

$$Z_{MMKP_{aux}}(\overline{x}) = Z_{MMKP}(\overline{x}) \leq UB.$$

Besides, we applied the heuristic Der_Algo proposed in Hifi *et al.* [7] in order to determine an initial lower bound (noted LB) for the problem.

4 A branch and bound algorithm

In this section let us propose a branch and bound algorithm to solve the MMKP problem. The principle of the algorithm is to develop a search tree allowing enumerating the sorted solutions in the order $\hat{X}_p, \hat{X}_{p-1}, \ldots, \hat{X}_{max}$ (of respective values taken in the order $Z_p(\hat{X}_p) \geq Z_{p-1}(\hat{X}_{p-1}) \geq \ldots \geq Z_{max}(\hat{X}_{max})$) until to find the first feasible solution.

4.1 Development of the search tree

Each developed node in the search tree, corresponds to a fixed item in the solution. A branch of the search tree corresponds to a solution. We note by n_{ij} the developed node corresponding to the item j of the class J_i.

The classes are considered a to an, and for each class J_i, $i = 1, \ldots, n$, one sorts the items j, $j = 1, \ldots, r_i$, according to the decreasing order of their respective profits, i.e :

$$v_{i1} > v_{i2} > \ldots > v_{ij} > \ldots > v_{ir_i}.$$

During the development of the search tree and for a node n_{ij}, we develop two nodes :

- the *brother's* node $n_{i,j+1}$ that corresponds, if it exists to the item that comes next in the same classe J_i ;
- the *son's* node $n_{i+1,1}$ that corresponds to the first item of the class J_{i+1} if the $(i+1)$-th class which comes next, exists;

This is illustrated by the following outline :

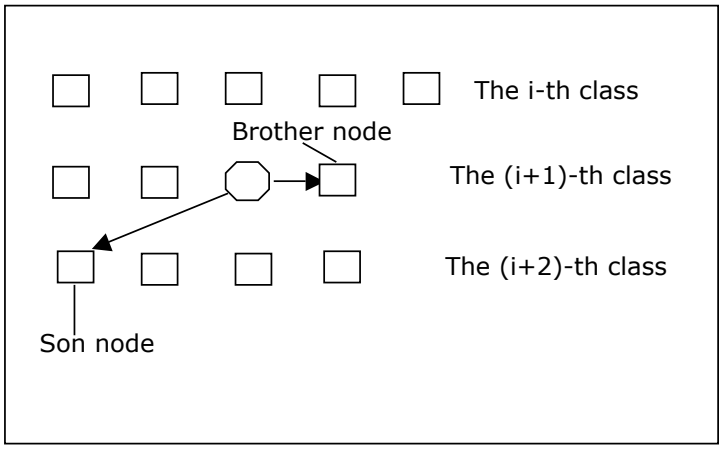

Figure 1: Development of the arborescence.

The development of a node allows:

- on one hand, to generate a partial solution while developing the brother's node;
- on the other hand, to continue the development of the partial running solution while developing the son's node.

This is illustrated by the following outline :

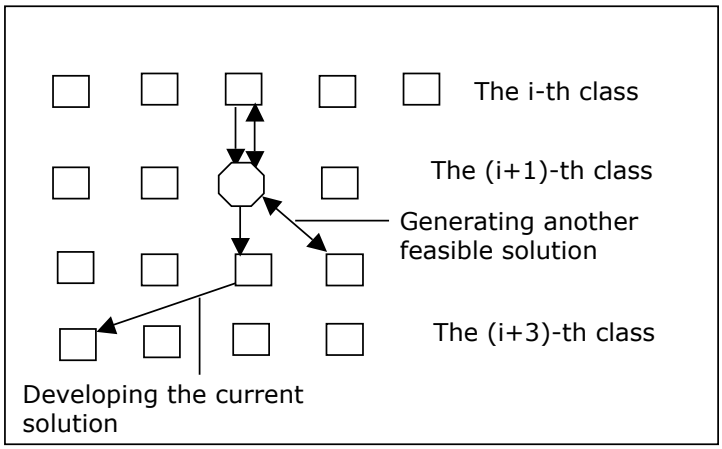

Figure 2: Developing a node outline.

The approach of the algorithm uses the strategy by *the best first*. The choice of the node to develop is done regarding the value of the better solution (feasible or not) that will be able to be obtained from this node. The value of this better solution is easy to compute. In fact, for a given node $n_{i,j}$, the solution with the bigger value from this node is the solution obtained while adding the items $(i+1,1),(i+2,1),\ldots,(n,1)$ (the first item 1 of each class J_i), since all the every class items were considered according to the decreasing order of their respective profits.

4.2 Principle of the algorithm

Let $Z_p(\hat{X}_p)$ be all the values of the solutions $\hat{X}_p$ which are feasible and not feasible for the MMKP such that:

$$Z_1(\hat{X}_1) \leq Z_2(\hat{X}_2) \leq \ldots \leq Z_l(\hat{X}_l) \leq \ldots \leq Z_p(\hat{X}_p).$$

Lemma 1 *Let $Z_{max}(\hat{X}_{max})$ be the biggest value such that $\hat{X}_{max}$ is feasible, then $\hat{X}_{max}$ is an optimal solution for the MMKP.*
If such a solution does not exist (all the solutions $\hat{X}_l$ are not feasible, for $1 \leq l \leq p$), then the set of feasible solutions for MMKP is empty.

Proof 3 *If $\hat{X}_{max}$ is feasible, then $Z_{max}(\hat{X}_{max})$ is value of an optimal solution since it is the biggest obtained value for a feasible solution.*

We will therefore have:

$$Z_{max}(\hat{X}_{max}) \leq Z_{max+1}(\hat{X}_{max+1}) \leq \ldots \leq Z_l(\hat{X}_l) \leq \ldots \leq Z_p(\hat{X}_p).$$

With $\hat{X}_{max+1}, \ldots, \hat{X}_l, \ldots, \hat{X}_p$ *are non feasible solutions.*

Proposition 3 *The search tree EMKP algorithm develops all the solutions. Adding to this, the first obtained feasible solution is optimum for the MMKP.*

Proof 4 *1. To show that EMKP develops all the solutions, it suffices to show that for all nodes* $n_{i,j}$*, EMKP allows developing all its son's nodes.*

When the node $n_{i,j}$ *is considered, we develop his son's node* $n_{i+1,1}$*. In the same manner, when the node* $n_{i+1,1}$ *is treated, we develop, what's more of his son's node, his brother's node* $n_{i+1,2}$*. In all the same, when the node* $n_{i+1,2}$ *is considered, we develop his brother's node* $n_{i+1,3}$ *and so on until to develop the last node* $n_{i+1,r_{i+1}}$*. EMKP therefore allows to develop all the son nodes* $n_{i+1,1}, n_{i+1,2}, \ldots, n_{i+1,r_{i+1}}$ *of the current node* $n_{i,j}$*.*

2. To show that the first obtained feasible solution is optimum for the MMKP, it sufficed to show that the solutions generated by EMKP are obtained in the order $\hat{X}_p, \hat{X}_{p-1}, \ldots, \hat{X}_{max}$ *(of respective values taken in the order* $Z_p(\hat{X}_p) \geq Z_{p-1}(\hat{X}_{p-1}) \geq \ldots \geq Z_{max}(\hat{X}_{max})$*). This is true since the algorithm considers the nodes in the best first strategey of exploration regarding the better obtained solution with respect to this node.*

4.3 Fathoming the nodes in the search tree

The troncature of the branches in the search tree is done thanks to the evaluation functions. From a given node n_{ij}, two types of troncatures can be performed :

1. A son's node developed from a father's node such that the latter corresponds to a non feasible solution is fathomed (see Figure 3). It is necessary to notice that a son's node that corresponds to a non feasible solution while the father's node corresponds to a feasible solution might not be truncated (cf. Figure 4). Just like a developed brother's node and that corresponds to a non feasible solution has as well not to be truncated.

   In fact, the troncature of these nodes can reduce the search space of the feasible solutions since the brother's node of these nodes can correspond to a feasible solution.

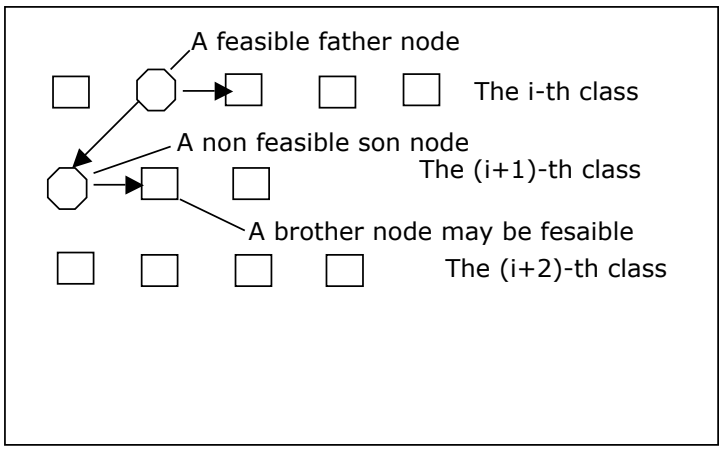

Figure 3: First outline of a node fathoming of the search tree .

2. For this second case, we need to have as input entry the value of a lower bound LB. We consider the lower bound LB presented in Hifi *et al.* [7]. Before developing a node n_{ij} corresponding to a partial solution of value $\hat{Z}$, we compute the upper bound UB described in the section 3 for the MMKP subproblem constituted with the remaining $(n-i)$ non treated classes J_p, for $p = i+1, \ldots, n$.
   If $UB + \hat{Z} < LB$, we are sure that any feasible solution developed from this node will have a strictly inferior value to LB (feasible solution) and therefore it is not optimum.

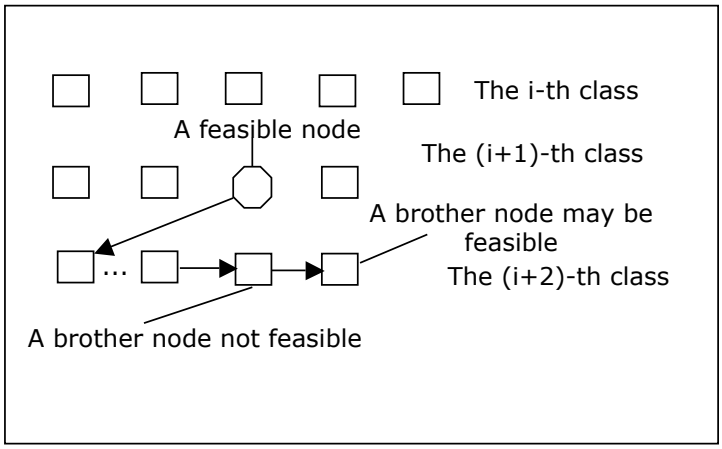

Figure 4: Second case of a node troncature.

4.4 The algorithm EMKP

The EMKP algorithm starts with developing the first node $n_{1,1}$ that corresponds to the first item 1 of the first class J_1. It takes as input entry the value of the lower bound LB

developed in Hifi *et al.* [7]. All the class items are taken according to the decreasing order of their respective profits.
EMKP algorithm terminates as soon as one finds a feasible solution that is the optimal solution according to the proposition 3.

The main steps of the EMKP branch and bound algorithm for the MMKP are described in the figure 5.

```
Entrée: An MMKP instance;
Sortie: An optimal solution for MMKP;
Initialisation:
   1. LB; /* a lower bound for MMKP */
   2. Sort the every class items in the decreasing order of the profits;
   3. L = {n11}; /* a root node of the arborescence */
Main step:
   4. While (L ≠ ∅) do
     4.1. node = best node of L;
     4.2. L ⟵ L \ {node};
     4.3. If node corresponds to a feasible solution, then develop a new son node;
     4.4. If (son node belongs to the last class and the solution is feasible then
             exit with solution;
     4.5. If (UB(son node) > LB) then
             L ⟵ L ∪ {son node};
     4.6. If (node is not the last item of a class) then
             Develop a brother node and put it in L;
   5. EndWhile
```

Figure 5: A branch and bound EMKP algorithm for the MMKP.

5 Computational results

On one hand, the proposed EMKP algorithm optimally solves all the instances of small size given by Khan *and al.* [8]. Of more, EMKP necessitated a "shortly" execution time (three seconds to the maximum on a SUN UltraSparc10) to obtain the optimal solution.

On the other hand, EMKP was incapable to solve the other large instances (as, besides, the algorithm developed by Khan *and al.* [8]). This is principally because of the RAM memory of the used machine.

5.1 Instances generation

For that, we have tested the EMKP algorithm on another tset of instances of small and medium size. These instances were randomly generated, divided up on four groups. Each of the groups represents the instances having the same number of classes. For these

different groups, the number of items as well as the number of constraints are different. Besides, we generate each group instance as follows : (i) to fix the number of classes, (ii) to fix the number of items of each class, (iii) to fix the number of constraints and (iv) to randomly generate the profits in the interval $[0, 150]$ and weights in the interval $[0, 50]$.

The number of classes is set up in the discrete interval $\{10, 25, 50\}$, the number of items of classes is set up in the discrete interval $\{5, 10, 15, 20\}$ and the number of constraints by instance is set up in the interval $\{5, 7, 10\}$. The capacity (C^k) of each instance constraint is put equals to :

$$C^k = (1/2) \times (\sum_{i=1}^{n} \min_{1 \leq j \leq r_i} \left\{ w_{ij}^k \right\} + \sum_{i=1}^{n} \max_{1 \leq j \leq r_i} \left\{ w_{ij}^k \right\}), \ k = 1, \ldots, m.$$

The performance of the proposed algorithm tested on these instances is outlined in the tables 1, 2, 3, and 4.

For each figure, we report :

- the number n of classes (column 2), ;
- the number r_i of items in each class J_i, $i = 1, \ldots, n$ (column 3);
- the number m of constraints (column 4);
- the lower bound LB (feasible solution) obtained while applying the heuristic proposed by Hifi *et al.* [7] (column 5);
- the upper bound UB proposed in the proposition 2 (column 6);
- the optimal solution Z produced by the EMKP algorithm (column 7);
- the execution time (noted T and measured in secondes : column 8) that EMKP necessitates to obtain the optimal solution.

5.2 Performance of the exact algorithm

Previously, we were interested in the behavior of algorithm while varying the number of items by class and the number of constraints (the two first groups). In a second time, we increased the number of classes. Finally, we considered a group while increasing the number of classes, while varying the number of items by class and while doing another variation on the number of constraints. The latter case is represented by the fourth group for which one wished to see the limits (on the used machine) of the proposed algorithm.

Inst.	n: # Classes	r_i: # Items by class	m: # Constraints	LB	Opt	T in seconds
I1a1	10	5	5	1420	1482	0.1
I1a2	10	5	5	1392	1475	0.1
I1a3	10	5	5	1375	1436	0.1
I1a4	10	5	5	1387	1433	0.1
I1a5	10	5	5	1442	1548	0.1
I1b1	10	5	10	1559	1559	0.1
I1b2	10	5	10	1553	1573	0.1
I1b3	10	5	10	1461	1539	0.1
I1b4	10	5	10	1049	1609	0.1
I1b5	10	5	10	1371	1456	0.1

Table 1: Numerical results of the first group

Inst.	n: # Classes	r_i: # Items by class	m: # Constraints	LB	Opt	T in seconds
I2a1	10	10	5	1662	1662	0.3
I2a2	10	10	5	1649	1658	0.8
I2a3	10	10	5	1592	1600	1.2
I2a4	10	10	5	1426	1453	0.9
I2a5	10	10	5	1461	1534	0.5
I2b1	10	10	10	1665	1665	0.4
I2b2	10	10	10	1655	1672	0.5
I2b3	10	10	10	1629	1629	0.2
I2b4	10	10	10	1598	1614	0.5
I2b5	10	10	10	1591	1637	0.4

Table 2: Numérical results of the second group

Inst.	n: # Classes	r_i: # Items by class	m: # Constraints	LB	Opt	T in seconds
I3a1	25	10	5	4089	4137	0.6
I3a2	25	10	5	4201	4245	0.8
I3a3	25	10	5	4007	4043	0.8
I3a4	25	10	5	4010	4045	0.8
I3a5	25	10	5	4092	4123	0.9
I3b1	25	10	10	4160	4177	1.3
I3b2	25	10	10	4272	4278	1.9
I3b3	25	10	10	3877	3982	2.2
I3b4	25	10	10	4072	4105	1.9
I3b5	25	10	10	4011	4071	0.8

Table 3: Numerical results of the third group

The two first groups are composed of ten instances each. For each of the groups, the half of the instances possesses five constraints and the other half possesses ten constraints. The number of classes of the group of these instances is set up to ten and we apply a variation on the number of items for each class.

We considered the instances of the two first groups as being instances of small size. For the set of these instances, EMKP converged towards the optimal solution while consuming an execution time less than 1.2 seconds (the maximal execution time realized by the

algorithm on the instances of these groups). From Figure 1, we can notice that the variation on the number of constraints is not signifcative for these instances, since we can consider them as instances of small size (the number of variables does not significantly increase). After increasing the number of items by class, we notice that, from Figure 2, the execution time increased for the instances of the second group. In this case, one can notice that the execution time is a growing function of the number of items.

Inst.	n: # Classes	r_i: # Items by class	m: # Constraints	LB	Opt	T in seconds
I4a1	50	20	5	8514	8542	7
I4a2	50	20	5	8561	8564	6.5
I4a3	50	20	5	8619	8635	8
I4a4	50	20	5	8443	8459	7.3
I4a5	50	20	5	8432	8456	15
I4b1	50	20	5	8575	8590	12
I4b2	50	20	7	8561	8569	9
I4b3	50	20	7	8545	8551	10
I4b4	50	20	7	8526	8542	11
I4b5	50	20	7	8438	8473	13
I4c1	50	15	10	8289	8340	22
I4c2	50	15	10	8303	8345	20
I4c3	50	15	10	8472	8511	35
I4c4	50	15	10	8529	8582	28
I4c5	50	15	10	8323	8324	18

Table 4: Numerical results of the fourth group

For the second case, we considered the third group composed from ten instances where the number of classes by instance were increased. This increase allows also to increase the number of variables by instance. In this case, we can note that from Figure 3 the execution time practically doubled.

Finally, we considered the fourth group (composed from fifteen instances) in which the number of classes was set up to 50 while we applied variations on the number of items by class and on the number of constraints by instance. Let us note that this group possesses instances having 1000 items divided up on 50 classes of 20 items and subject to 7 constraints. From Figure 4 one can notice that the execution time increases with the number of constraints.

Of more, we note than despite the decrease of the number of items by class on the five last instances, the execution time remains more important because of the number of constraints on these instances.

6 Conclusion

In this paper, we proposed an exact algorithm for the problem of the multiple-choice multidimensional knapsack problem. The algorithm applied a branch and bound procedure using the best-first strategy search. Previously, we used a lower bound as a starting solution (feasible solution) computed by the application of the heuristic developed by Hifi *et al.* [7]. In a second time, we carried out the reduction of the initial problem in the form of a multiple-choice knapsack problem MCKP and we called MMKP_{aux}. The latter problem allowed us to calculate an upper bound UB for the original problem as well as intermediary upper bounds for the proposed method. Besides, the combination between the initial lower bound and the intermediary upper bounds allowed to fathom many branches of the search tree. The experimental study showed that the proposed method was able to solve instances of small and medium sizes of which the number of variables being able to include is up to 1000 items, divided up on 50 classes with 20 items and subject up to 7 constraints.

References

[1] E. Balas and E. Zemel, An algorithm for large zero-one knapsack problem, *Operations Research*, vol. 28, pp.1130-1154, 1980.

[2] G. Chen, S Khan, KF Li and E Manning, Building an adaptive multimedia system using the utility model. In *Proceedings of International Workshop on Parallel and Distributed Realtime Systems*, San Juan, Puerto Rico, 1999.

[3] P. Chu and J.E. Beasley, A genetic algorithm for the multidimensional knapsack problem, Journal of Heuristics, vol. 4, pp. 63-86, 1998.

[4] JE Beasley, PC Chu, A genetic algorithm for the set covering problem. *Europ J Opl Res*, vol. 94, 392-404, 1996.

[5] GB Dantzig, Discrete variable extremum problems, *Operations Research*, vol. 5, 266-277, 1957.

[6] D. Fayard and G. Plateau, An algorithm for the solution of the 0-1 knapsack problem, *Computing,* vol. 28, pp. 269-287, 1982.

[7] M. Hifi, M. Michrafy and A. Sbihi, Heuristic algorithms for the multiple-choice multidimensional knapsack problem, working paper, *CERMSEM, Université Paris 1*, (Presented in seminary Modélisation et Résolution Algorithmique), 2002.

[8] S. Khan, K. F. Li, E. G. Manning and MD. M. Akbar. Solving the knapsack problem for adaptive multimedia systems, *Studia Informatica, an International Journal*, Special Issue on Cutting, Packing and Knapsacking problems, vol. 2/1, pp. 154-174, 2002.

[9] S. Khan, Quality adaptation in a multi-session adaptive multimedia system: model and architecture, PhD Thesis, Department of Electronical and Computer Engineering, University of Victoria, May 1998.

[10] H. Luss, Minmax resource allocation problems: optimization and parametric analysis, European Journal of Operational Research, vol.60, pp. 76-86, 1992.

[11] S. Martello, D. Pisinger and P. Toth, *Dynamic programming and strong bounds for the 0-1 knapsack problem*, Management Science, vol. 45, pp. 414-424, 1999.

[12] M. Moser, D. P. Jokanović and N. Shiratori. An algorithm for the multidimesional multiple-choice knapsack problem, *IEECE Transactions on Fundamentals of Electronics,* vol. 80, No 3., pp.582-589,1997.

[13] MR Nauss, The 0-1 knapsack problem with multiple-choice constraints, *European Journal of Opeartional Research* **2:** 125-131, 1978.

[14] J.S Pang and C.S Yu, A min-max resource allocation problem with substitutions, European Journal of Operational Research, vol.41, pp.218-223, 1989.

[15] D. Pisinger, A minimal algorithm for the 0-1 knapsack problem, *Operations Research*, vol. 45, pp. 758-767, 1997.

[16] D Pisinger, A minimal algorithm for the Multiple-choice Knapsack Problem, *European Journal of Operational Research*, vol. 83, 394-410, 1995.

[17] J Richardson, M Palmer, G Liepins, M Hilliard, Some guidelines for genetic algorithms with penalty functions. In: Schaffer J (eds). *Proceedings of the Third International Conference on Genetic Algorithms* pp 191-197. Maurgan, 1989. Kaufmann.

[18] A Sbihi, Hybrid methods in combinatorial optimisation: exact algorithms and heuristics. PhD Thesis, CERMSEM Laboratory, University of Paris 1 Panthéon-Sorbonne, December 2003.

[19] Y. Toyoda, A simplified algorithm for obtaining approximate solution to zero-one programming problems, *Management Science*, vol. 21, pp. 1417-1427, 1975.

[20] Watson RK (2001). *Packet Networks and optimal admission and upgrade of service level agreements: applying the utility model.* M.Sc thesis, Dept. of ECE, University of Victoria.