# A best first search exact algorithm for the Multiple-choice Multidimensional Knapsack Problem

Abdelkader Sbihi

J Comb Optim (2007) 13:337–351
DOI 10.1007/s10878-006-9035-3

# A best first search exact algorithm for the Multiple-choice Multidimensional Knapsack Problem

**Abdelkader Sbihi**

Published online: 8 December 2006


**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 feasible solution as a starting lower bound, and (ii) at different levels of the search tree to determine an intermediate upper bound obtained by solving an auxiliary problem called $\text{MMKP}_{\text{aux}}$ and perform the strategy of fixing items during the exploration. The approach which we develop is of best-first search strategy. The method was able to optimally solve the MMKP. The performance of the exact algorithm is evaluated on a set of small and medium instances, some of them are extracted from the literature and others are randomly generated. 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 concerned with a more harder variant of the knapsack problem KP: the Multipe-choice Multidimensional Knapsack Problem namely MMKP. MMKP is an $\mathcal{NP}$-hard optimization problem (as a generalization of the single Knapsack Problem KP) which models many practical and real life problems. We cite the problem of quality adaptation and admission control for interactive multimedia systems (Chen et al., 1999), or service level agreement management in telecommunication networks problems (Watson, 2001). In the MMKP, we are given a set $\mathcal{N}$ of items divided into $n$ classes $J_i$, where each class $J_i$, $i = 1, \ldots, n$, has $r_i = |J_i|$

A. Sbihi (✉)
Lancaster University Management School, Optimisation Research Centre, Lancaster LA1 4YX, United Kingdom
e-mails: a.sbihi@lancaster.ac.uk; Abdelkader.Sbihi@univ-paris1.fr

items such that $\forall\ 1 \leq p, q \leq n$ and $p \neq q$, $J_p \cap J_q = \emptyset$ and $\cup_{p=1}^{n} J_p = \mathcal{N}$. 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:

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

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 class $J_i$ is picked.

This problem may be considered as a generalization of the well known Multiple-Choice Knapsack Problem namely MCKP by adding the multidimensionality in the capacity constraint (for more details, see Nauss (1978) and Pisinger (1995)).

The remaining of the paper is organized as follows. First (Section 2), we present a brief reference of some related works on the classical knapsack problem and its most known variants dealing with sequential exact and approximate algorithms. Second (Section 3), we present a method showing how to compute intermediate upper bounds at each developed node of the search tree for the MMKP. For this purpose we start by modifying (reducing) the original problem MMKP to an auxiliary problem called $\text{MMKP}_{\text{aux}}$. This transformation allowed to compute an upper bound UB for the auxiliary problem $\text{MMKP}_{\text{aux}}$. We establish the result for which an upper bound for $\text{MMKP}_{\text{aux}}$ also is an upper bound for the original problem MMKP. Third (Section 4), we prove the finitude of the approach and present the principle of the search tree exploration. We, then, present the main principles of the branch and bound algorithm `EMKP` and we prove its optimality. Fourth and last (Section 5) the performance of the proposed approach is tested on a set of problem instances extracted from the literature that have small sizes and which have been solved optimally and on others generated one.

## 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 resolution methods have been proposed (Martello et al., 1999; Balas and Zemel, 1980; Fayard and Plateau,

1982; Pisinger, 1997). The problem has been solved optimally and approximately by dynamic programming, or by different search tree procedures or other hybrid approaches.

Most exact algorithms for solving the knapsack problem (KP) variants are mainly based on (i) branch and bound search using depth-first search strategy (Balas and Zemel, 1980; Fayard and Plateau, 1982; Martello and Toth, 1988), (ii) dynamic programming techniques (see Pisinger, 1997), and (iii) hybrid algorithms combining dynamic programming and branch and bound procedures (Martello et al., 1999).

Research has been developed on some KP variants and different exact and approximate approaches have been tailored especially for these problems. One of the most famous is the Multi-Dimension Knapsack Problem (MDKP) which is one kind of KP where the constraints are multidimensional (Chu and Beasley, 1998; Shih, 1979). The Multiple-Choice Knapsack Problem (MCKP) is another variant of KP where the picking criterion of items is more restrictive (Nauss, 1978; Pisinger, 1995; Sinha and Zoltners, 1979). For the MCKP variant there are one or more disjoint classes of items. For the MDKP, Toyoda (1975) used the aggregate resources 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 (1998)). We can also cite the Multiple Knapsack Problem MKP as another type of knapsack problem which is a special case of the Generalized Assignment problem (GAP) (for more details see Shmoys and Tardos (1993)) and where the profit and the weight of an item can vary based on the specific knapsack that it is assigned to.

To our knowledge, very few papers dealing directly with the MMKP are available. Moser et al. (1997) have designed an approach based upon the concept of graceful degradation from the most valuable items based on Lagrange multipliers. Khan et al. (2002) have tailored an algorithm based on the aggregate resources already introduced by Toyoda (1975) for solving the MDKP. Finally, Hifi et al. (2004, 2005) proposed two different approximate approaches. The first approach is 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. The second approach is a reactive local search and where an explicit check for the repetition of configurations is added to the local search. The algorithm starts by an initial solution and improved by using a fast iterative procedure. Later, both deblocking and degrading procedures are introduced in order (i) to escape to local optima and, (ii) to introduce diversification in the search space. Finally, a memory list is applied in order to forbid the repetition of configurations.

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 which is a lower bound for the MMKP, and (ii) to determine a new upper bound for each level 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 $\text{MMKP}_{\text{aux}}$:

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

where $C = \sum_{k=1}^{m} C^k$ represents the sum of the $m$ capacity constraints and $w_{ij} = \sum_{k=1}^{m} w_{ij}^k$ represents the $m$ weigths sum of the $j$-th item belonging to the $i$-th class $J_i$. Also, let $N_f = |\mathcal{N}| - n = \sum_{i=1}^{n} r_i - n$ representing the total of the remaining items when fixing an item by class.
We also can remark that the above $\text{MMKP}_{\text{aux}}$ formulation also represents the MCKP problem.

Now, let UB be the expression which we compute with relation to the $\text{MMKP}_{\text{aux}}$ 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 vector this item noted $j_{\max}$ for each class $J_i$.
2. Let $R_{\max} = \sum_{i=1}^{n} w_{i j_{\max}}$ (resp. $V_{\max} = \sum_{i=1}^{n} v_{i j_{\max}}$) be the cumulated used weights (resp. the cumulated used profits) of all the fixed items noted $j_{\max}$ belonging to each class $J_i$, $i = 1, \ldots, n$. We can then distinguish two cases:

   - $R_{\max} > C$. In this case, the UB value defined above is computed for $\text{MMKP}_{\text{aux}}$ as follows:

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

   - $R_{\max} \leq C$. 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 single class $J = \{1, \ldots, j, \ldots, N_f\}$ with items indexed by $j = 1, \ldots, N_f$. Then, we sort the items in the classical decreasing order of the ratio $\frac{v_j}{w_j}$ for $j = 1, \ldots, N_f$. The problem generated by the set of these items represents a knapsack problem $\text{KP}(C - R_{\max})$ with a capacity constraint equals to $C - R_{\max}$. We, then, compute $\text{UB}_d$ as the relative Dantzig (1957) upper bound of $\text{KP}(C - R_{\max})$. This means that we fill the knapsack with the ordered items $j$ until its constraint is

violated.
Let, now $\ell$ such that $1 \leq \ell \leq N_f$ be a particular item which violates the capacity constraint $C - R_{\max}$ and defined in the following:

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

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

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

After computing the Dantzig upper bound $\mathrm{UB}_d$ for the knapsack problem with the remaining items, the UB value for the $\mathrm{MMKP}_{\mathrm{aux}}$ is computed, in this case, in the following:

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

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

**Proof:** We will show by contradiction the first case ($R_{\max} > C$) of this proposition. Indeed, 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.

$$\text{If} \quad \frac{a}{b} \geq \frac{c}{d} \quad \text{and} \quad \frac{x}{y} \geq \frac{z}{t} \quad \text{then} \quad \frac{a+x}{b+y} \geq \frac{c+z}{d+t} \tag{1}$$

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 $\mathrm{MMKP}_{\mathrm{aux}}$ with objective value $\overline{V} = \sum_{i=1}^{n} v_{ij_i}$ such that $V_{\max} \times (\frac{C}{R_{\max}}) < \overline{V}$. Thus $\frac{V_{\max}}{R_{\max}} < \frac{\overline{V}}{C}$. This implies $\frac{\overline{V}}{C} \leq \frac{\overline{V}}{\overline{R}}$, since $\overline{R} = \sum_{i=1}^{n} w_{ij_i} \leq C$ and $\frac{V_{\max}}{R_{\max}} < \frac{\overline{V}}{C} \leq \frac{\overline{V}}{\overline{R}}$. The last double inequality implies that $\frac{V_{\max}}{R_{\max}} < \frac{\overline{V}}{\overline{R}}$.

According to the relation (1) and to the decreasing order of the profits by weights ratios of the items:

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

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

For the second case ($R \leq C$), it is easy to see that the remaining items represent a single knapsack problem and, by the continuation, the calculated upper bound is the Dantzig (1957) upper bound. □

**Proposition 2.** *UB is an upper bound for the original problem MMKP.*
*The proof is naturally obvious.*

Besides, we applied the MRLS heuristic that we have proposed in Hifi et al. (2005) in order to determine an initial feasible solution that we consider as a starting lower bound (noted LB) for the problem. The reader may refer to Hifi et al. (2005) for more details about MRLS heuristic. Here we recall the main principle of the MRLS algorithm.

The MRLS approach can be summarized as follows: (i) starting with an initial solution for the MMKP, obtained by applying a fast constructive procedure, (ii) improving the current solution by running a complementary constructive procedure which applies a swapping criterion and, (iii) using the reactive strategy composed by a degrading procedure and finally, a tabu list is introduced in order to avoid some cycling during the search process (Hifi et al., 2005).

## 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 to enumerate the sorted solutions in the order $\hat{X}_p, \hat{X}_{p-1}, \ldots, \hat{X}_{\max}, \ldots, \hat{X}_1$ (of respective values taken in the order $Z_p \geq Z_{p-1} \geq \cdots \geq Z_{\max} \geq \cdots \geq Z_1$ with $Z_q = Z(\hat{X}_q)$, $q = 1, \ldots, p$) until finding the first feasible solution that we note by $\hat{X}_{\max}$ and of value $Z(\hat{X}_{\max})$.

### 4.1 The branching strategy

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

The classes are considered one by one, 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.:

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

If two items have the same profit value, we consider in first the one corresponding to the smallest $\sum_{k=1}^{m} \frac{w_{ij}^k}{C^k}$.

While developing the search tree and for a current node $n_{ij}$, we develop two nodes:

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

This is illustrated by the following outline: (Fig. 1)
The development of a node allows:

- on one hand, to generate a partial solution while developing the brother's node;

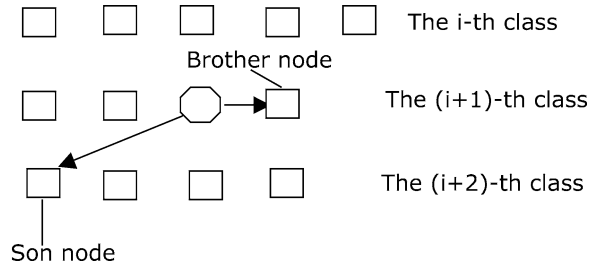


**Fig. 1** Development of the arborescence

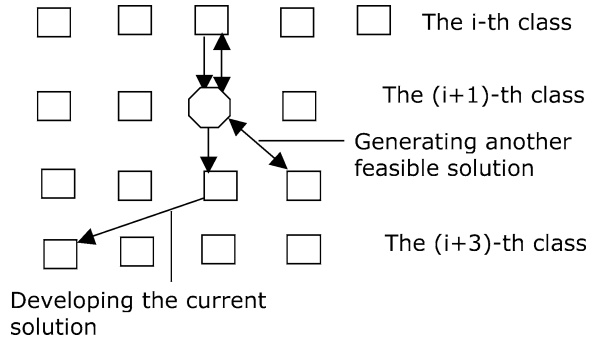


**Fig. 2** Developing a node outline

- on the other hand, to continue to build the partial running solution while developing the son's node.

This is illustrated in the Fig. 2.

The algorithm approach uses the *best-first search* strategy. The algorithm selects which node to develop regarding the value of the better solution (feasible or not) and which is 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 $Z_p = Z(\hat{X}_p)$, $p \geq 1$, be the values of the obtained solutions $\hat{X}_p$, $p \geq 1$, and which are feasible and/or not feasible for the MMKP. We ordered these values in the decreasing order such that:

$$Z(\hat{X}_1) \leq Z(\hat{X}_2) \leq \cdots \leq Z(\hat{X}_\ell) \leq \cdots \leq Z(\hat{X}_p).$$

**Lemma 1.** *Let $Z(\hat{X}_{\max})$, $1 \leq max \leq p$, 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 $\hat{X}_\ell$ are not feasible, for $1 \leq \ell \leq p$), then the set of feasible solutions for MMKP is an empty one.*

**Proof:** If $\hat{X}_{\max}$ is feasible, then $Z(\hat{X}_{\max})$ is value of an optimal solution since it is the biggest obtained value for a feasible solution. We will therefore have:

$$Z(\hat{X}_{\max}) \leq Z(\hat{X}_{\max+1}) \leq \cdots \leq Z(\hat{X}_p), \quad \text{for } 1 \leq \max \leq p.$$

With $\hat{X}_{\max+1}, \ldots, \hat{X}_p$, $p \geq 1$, are not feasible solutions. □

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

**Proof:**

1. To show that EMKP develops all the solutions, it suffices to show that for each node $n_{i,j}$, EMKP allows developing all its son's nodes.
   When the node $n_{i,j}$ is considered, we develop its son's node $n_{i+1,1}$. In the same way, when the node $n_{i+1,1}$ is treated, we develop, adding to its son's node, its brother's node $n_{i+1,2}$. In all the same, when the node $n_{i+1,2}$ is considered, we develop its brother's node $n_{i+1,3}$ and so on until developing 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 obtained first feasible solution is an optimal one for the MMKP, it is enough to show that the solutions which are generated by EMKP are obtained in the order $\hat{X}_p, \hat{X}_{p-1}, \ldots, \hat{X}_{\max}$, $1 \leq \max \leq p$ (with the respective values taken in the order $Z(\hat{X}_p) \geq Z(\hat{X}_{p-1}) \geq \cdots \geq Z(\hat{X}_{\max})$). This is true since the algorithm considers the nodes with regards to the best first strategey of exploration and regarding the better obtained solution with respect to this node.
   Thus by the Lemma 1, we can affirm that the obtained first feasible solution (if it exists) realizes the optimum for the MMKP. □

### 4.3 Reduction of the search space

In order to make the search for solutions more efficient and fast, some reductions of the space of search are necessary to reduce to the maximum the combinatorial explosion. These reductions consist with pruning the useless branches of the tree and which do not lead to the solutions which could be elite.

To perform the pruning in the tree, EMKP realizes these reductions thanks to the evaluation functions. In fact, for a given node $n_{ij}$, two types of tree pruning 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 removed (see Fig. 3).
   A son's node corresponding to an infeasible solution while the father's node corresponds to a feasible one might not be truncated (see Fig. 4).
   Just as a developed brother node and which corresponds to a nonrealizable solution may not be truncated either. In fact, the truncature of these nodes can non

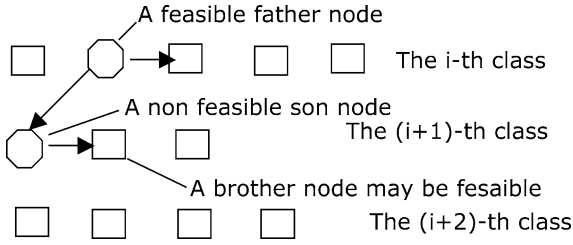


**Fig. 3** First outline of a fathoming node in the search tree

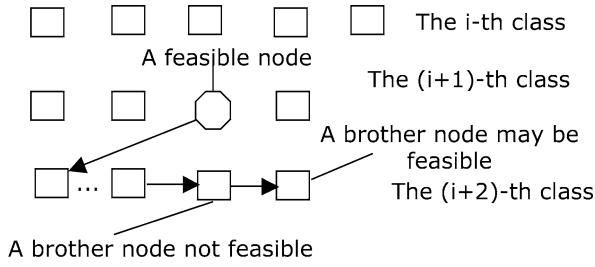


**Fig. 4** Second case of a node truncature

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

2. For this second case, we need to have as input entry the value of a lower bound LB for MMKP. We apply the MRLS heuristic (Hifi et al., 2005) and we consider the solution value as the starting lower bound LB for MMKP. Before developing a node $n_{ij}$ corresponding to a partial solution with a value $\hat{Z}$ (this means EMKP has up to now treated the $i$ first classes $J_1, \ldots, J_i$), we compute the upper bound UB described in the Section 3 for the MMKP subproblem constituted with the remaining $(n - i)$ non yet treated classes $J_p$, for $p = i + 1, \ldots, n$.

   If $UB + \hat{Z} < LB$, it is sure that any feasible solution which shall be developed from this node will have a strictly lower value to LB (feasible solution) and therefore it does not realize the optimum to the MMKP.

### 4.4 The EMKP algorithm

The EMKP algorithm starts by 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 MRLS heuristic solution value as a lower bound. All the class items are taken according to the decreasing order of their respective profits, $(v_{i1} \geq \cdots \geq v_{ij} \cdots \geq v_{ir_i}, \forall\, i \in \{1, \ldots, n\})$.

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

Here $UB(Node)$ denotes the upper bound calculated at the best current node by the $\text{MMKP}_{\text{Aux}}$ strategy developed in the Section 3. $X$ denotes the optimal solution and $Z = Z(X)$ the value of the optimal solution obtained by EMKP.

EMKP algorithm terminates as soon as one finds a feasible solution that is the optimal solution according to the Proposition 3.

```
Input: An MMKP instance I;
Ouput: An optimal solution for MMKP X;
Initialisation:
1.  X ⟵ MRLS(I); /* a starting heuristic solution */
2.  LB ⟵ V(X); /* MRLS solution value as a lower bound for the MMKP */
3.  Sort items class by class in the decreasing order of their profits ;
4.  Node = n11; /* a root node of the tree: item 1 of the class J1 */
5.  Z = v(Node) = v11; UB = UB(Node); /* objective value Z and upper bound UB */
6.  L = {Node};
Main steps:
7.  While (L ≠ ∅) do
8.  Node = Best_Node of L;
9.  Z = v(Node);
10. L ⟵ L \ {Node};
11. Node⟵ Son_Node;
12. If (Node ∈ Jn and X is feasible ) then /* Node belongs to the last class */
                                          /* and the solution is feasible */
        exit with this solution X;
13. EndIf
14. If (Node corresponds to a feasible solution) then
        develop a new Son_Node;
        UB = UB(Node)+Z; /* Z is the current solution value at Son_Node */
15.     If (UB ≥ LB) then
            Node =Son_Node;
            Z=Z+v(Node); /* v(Node) is the fixed item profit in Son_Node */
            Update(X);
            L ⟵ L ∪ {Node};
16.     Else (Node is not the last item of the current class)
            develop a Brother_Node;
            UB = UB(Node)+Z;
17.         If (UB ≥ LB) then
                Node = Brother_Node;
                Z=Z+v(Node);
                Update(X);
                L ⟵ L ∪ {Node};
18.         EndIf
19.     EndIf
20. EndIf
21. EndWhile
22. Exit with best feasible solution X of best value Z(X);
```

**Fig. 5** A branch and bound EMKP algorithm for the MMKP

## 5 Computational results

In this section, we detail the computational results carried out within EMKP in order to optimally solve the MMKP. First, we tailor EMKP on the instances extracted from the litertaure. We report the results obtained for this first set. Second, we randomly generate several problem instances of various sizes and densities.

On one hand, the proposed EMKP algorithm optimally solves all the instances of small size given by Khan et al. (2002). Indeed, EMKP necessitated a "shortly" execution time (three seconds to the maximum on a SUN UltraSparc10) to obtain the optimal solution. We present bellow the computational results and the

performance comparison between EMKP and the Khan et al. approach, here noted KLMA. We can remark that our approach is more efficient and able to reach exact solutions in small times of execution. Also, EMKP ouperforms Khan et al. algorithm. For the few available instances, hereafter the results given by both approaches.

On the other hand, EMKP was incapable to solve the other large instances (as, besides, the algorithm developed by Khan et al. (2002)). 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 set 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 several 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 in 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 set {10, 25, 50}, the number of items of the classes is set up in the discrete set {5, 10, 15, 20} and the number of constraints by instance is set up in the set {5, 7, 10}. The capacity vector $(C^k)_{(1\leq k\leq m)}$ of each instance constraint is put equals to:

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

The performance of the proposed algorithm tested on these instances is reported in the Tables 2–5.

For each table, we report:

- the number $n$ of classes (column 2);
- the number $r_i = r$ of items in each class $J_i$, $i = 1, \ldots, n$ (column 3);
- the number $m$ of constraints (column 4);

**Table 1** Performances comparison between EMKP and KLMA

| # Inst. | $n$ | $r$ | $m$ | $T_{\text{KLMA}}$ (sec) | $T_{\text{EMKP}}$ (sec) |
|---|---|---|---|---|---|
| I1 | 5 | 5 | 5 | 1.4 | 0.1 |
| I2 | 10 | 5 | 5 | 1.8 | 0.4 |
| I3 | 15 | 10 | 10 | 115.934 | 0.6 |
| I4 | 20 | 10 | 10 | 288.687 | 1.1 |
| I5 | 25 | 10 | 10 | 1073.236 | 2.1 |
| I6 | 30 | 10 | 10 | 5905.536 | 15.3 |
| I7 | 100 | 10 | 10 | 12983.339 | 37.6 |

**Table 2** Numerical results of the first group

| # Inst. | $n$ | $r$ | $m$ | LB | Opt | UB | $T$ (sec) |
|---|---|---|---|---|---|---|---|
| I1a1 | 10 | 5 | 5 | 1420 | 1482 | 1558 | 0.1 |
| I1a2 | 10 | 5 | 5 | 1392 | 1475 | 1513 | 0.1 |
| I1a3 | 10 | 5 | 5 | 1375 | 1436 | 1562 | 0.1 |
| I1a4 | 10 | 5 | 5 | 1387 | 1433 | 1520 | 0.1 |
| I1a5 | 10 | 5 | 5 | 1442 | 1548 | 1598 | 0.1 |
| I1b1 | 10 | 5 | 10 | 1559 | 1559 | 1602 | 0.1 |
| I1b2 | 10 | 5 | 10 | 1553 | 1573 | 1623 | 0.1 |
| I1b3 | 10 | 5 | 10 | 1461 | 1539 | 1569 | 0.1 |
| I1b4 | 10 | 5 | 10 | 1049 | 1609 | 1635 | 0.1 |
| I1b5 | 10 | 5 | 10 | 1371 | 1456 | 1502 | 0.1 |

**Table 3** Numerical results of the second group

| # Inst. | $n$ | $r$ | $m$ | LB | Opt | UB | $T$ (sec) |
|---|---|---|---|---|---|---|---|
| I2a1 | 10 | 10 | 5 | 1662 | 1662 | 1662 | 0.3 |
| I2a2 | 10 | 10 | 5 | 1649 | 1658 | 1679 | 0.8 |
| I2a3 | 10 | 10 | 5 | 1592 | 1600 | 1623 | 1.2 |
| I2a4 | 10 | 10 | 5 | 1426 | 1453 | 1493 | 0.9 |
| I2a5 | 10 | 10 | 5 | 1461 | 1534 | 1582 | 0.5 |
| I2b1 | 10 | 10 | 10 | 1665 | 1665 | 1682 | 0.4 |
| I2b2 | 10 | 10 | 10 | 1655 | 1672 | 1723 | 0.5 |
| I2b3 | 10 | 10 | 10 | 1629 | 1629 | 1723 | 0.2 |
| I2b4 | 10 | 10 | 10 | 1598 | 1614 | 1722 | 0.5 |
| I2b5 | 10 | 10 | 10 | 1591 | 1637 | 1682 | 0.4 |

**Table 4** Numerical results of the third group

| # Inst. | $n$ | $r$ | $m$ | LB | Opt | UB | $T$ (sec) |
|---|---|---|---|---|---|---|---|
| I3a1 | 25 | 10 | 5 | 4089 | 4137 | 4163 | 0.6 |
| I3a2 | 25 | 10 | 5 | 4201 | 4245 | 4286 | 0.8 |
| I3a3 | 25 | 10 | 5 | 4007 | 4043 | 4076 | 0.8 |
| I3a4 | 25 | 10 | 5 | 4010 | 4045 | 4052 | 0.8 |
| I3a5 | 25 | 10 | 5 | 4092 | 4123 | 4163 | 0.9 |
| I3b1 | 25 | 10 | 10 | 4160 | 4177 | 4232 | 1.3 |
| I3b2 | 25 | 10 | 10 | 4272 | 4278 | 4352 | 1.9 |
| I3b3 | 25 | 10 | 10 | 3877 | 3982 | 4019 | 2.2 |
| I3b4 | 25 | 10 | 10 | 4072 | 4105 | 4136 | 1.9 |
| I3b5 | 25 | 10 | 10 | 4011 | 4071 | 4116 | 0.8 |

- the lower bound LB (feasible solution) obtained while applying the MRLS heuristic proposed by Hifi et al. (2004) (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.

**Table 5** Numerical results of the fourth group

| # Inst. | $n$ | $r$ | $m$ | LB | Opt | UB | $T$ (sec) |
|---|---|---|---|---|---|---|---|
| I4a1 | 50 | 20 | 5 | 8514 | 8542 | 8569 | 7 |
| I4a2 | 50 | 20 | 5 | 8561 | 8564 | 8589 | 6.5 |
| I4a3 | 50 | 20 | 5 | 8619 | 8635 | 8676 | 8 |
| I4a4 | 50 | 20 | 5 | 8443 | 8459 | 8489 | 7.3 |
| I4a5 | 50 | 20 | 5 | 8432 | 8456 | 8482 | 15 |
| I4b1 | 50 | 20 | 5 | 8575 | 8590 | 8623 | 12 |
| I4b2 | 50 | 20 | 7 | 8561 | 8569 | 8587 | 9 |
| I4b3 | 50 | 20 | 7 | 8545 | 8551 | 8586 | 10 |
| I4b4 | 50 | 20 | 7 | 8526 | 8542 | 8583 | 11 |
| I4b5 | 50 | 20 | 7 | 8438 | 8473 | 8529 | 13 |
| I4c1 | 50 | 15 | 10 | 8289 | 8340 | 8369 | 22 |
| I4c2 | 50 | 15 | 10 | 8303 | 8345 | 8372 | 20 |
| I4c3 | 50 | 15 | 10 | 8472 | 8511 | 8568 | 35 |
| I4c4 | 50 | 15 | 10 | 8529 | 8582 | 8613 | 28 |
| I4c5 | 50 | 15 | 10 | 8323 | 8324 | 8392 | 18 |

## 5.2 Performance of the exact algorithm

Previously, we were interested in the behavior of the algorithms 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.

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 s (the maximal execution time realized by the algorithm on the instances of these groups). From Fig. 2, 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 Fig. 3, 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.

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 Fig. 4 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 Fig. 5 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 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. (2004). 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 $\text{MMKP}_{\text{aux}}$. 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.

**Acknowledgments** The author thanks the two anonymous referees for their helpful comments and suggestions.

## References

Balas E, Zemel E (1980) An algorithm for large zero-one knapsack problem. Oper Res 28:1130–1154
Chen G, Khan S, Li KF, Manning E (1999) Building an adaptive multimedia system using the utility model. In: Proceedings of international workshop on parallel and distributed realtime systems, San Juan, Puerto Rico
Chu P, Beasley JE (1998) A genetic algorithm for the multidimensional knapsack problem. J Heur 4:63–86
Dantzig GB (1957) Discrete variable extremum problems. Oper Res 5:266–277
Fayard D, Plateau G (1982) An algorithm for the solution of the 0-1 knapsack problem. Computing 28:269–287
Hifi M, Michrafy M, Sbihi A (2004) Heuristic algorithms for the multiple-choice multi-dimensional knapsack problem. J Operat Res Soc 55(12):1323–1332
Hifi M, Michrafy M, Sbihi A (2005) A reactive local search-based algorithm for the multiple-choice multidimensional knapasck problem. Computational Optimization and Applications, in press
Khan S, Li KF, Manning EG, Akbar MDM (2002) Solving the knapsack problem for adaptive multimedia systems. Studia Informatica, an International Journal, Special Issue on Cutting, Packing and Knapsacking problems 2/1:154–174
Khan S (1998) Quality adaptation in a multi-session adaptive multimedia system: model and architecture. PhD Thesis, Department of Electronical and Computer Engineering, University of Victoria
Luss H (1992) Minmax resource allocation problems: optimization and parametric analysis. Europ J Oper Res 60:76–86
Martello S, Toth P (1988) A newal gorithm for the 0-1 knapsack problem. Manag Sci 34:633–644

Martello S, Pisinger D, Toth P (1999) Dynamic programming and strong bounds for the 0-1 knapsack problem. Manag Sci 45:414–424

Moser M, Jokanović DP, Shiratori N (1997) An algorithm for the multidimesional multiple-choice knapsack problem. *IEECE Trans Fundament Electron* 80(3):582–589

Nauss MR (1978) The 0-1 knapsack problem with multiple-choice constraints. Europ J Oper Res 2:125–131

Pang JS, Yu CS (1989) A min-max resource allocation problem with substitutions. Europ J Oper Res 41:218–223

Pisinger D (1997) A minimal algorithm for the 0-1 knapsack problem. Oper Res 45:758–767

Pisinger D (1995) A minimal algorithm for the multiple-choice knapsack problem. Europ J Oper Res 83:394–410

Shih W (1979) A branch and bound method for the multiconstraint zero-one knapsack problem. J Oper Res Soc 30:69–378

Sinha A, Zoltners AA (1979) The multiple-choice knapsack problem. Oper Res 27:503–515

Sbihi A (2003) Hybrid methods in combinatorial optimisation: exact algorithms and heuristics. PhD Thesis, Cermsem UMR-CNRS 8095 Laboratory. University of Paris 1 Panthéon-Sorbonne

Shmoys DB, Tardos E (1993) An approximation algorithm for the generalized assignment problem. Math Prog 62:461–474

Toyoda Y (1975) A simplified algorithm for obtaining approximate solution to zero-one programming problems. Manag Sci 21:1417–1427

Watson RK (2001) Packet networks and optimal admission and upgrade of service level agreements: applying the utility model. MSc Thesis, Department of Electronical and Computer Engineering, University of Victoria