A hybrid heuristic for the multiple choice multidimensional knapsack problem

Raïd Mansi[†], Cláudio Alves[†,*], J. M. Valério de Carvalho[†], Saïd Hanafi[‡]

† Centro de Investigação Algoritmi da Universidade do Minho,

Escola de Engenharia, Universidade do Minho, 4710-057 Braga, Portugal

{raid.mansi,claudio,vc}@dps.uminho.pt

‡ Université Lille Nord de France, UVHC, LAMIH, CNRS, FRE 3304,

59313 Valenciennes, France

said.hanafi@univ-valenciennes.fr

April 21, 2011

Abstract

In this paper, we describe a new solution approach for the multiple choice multidimensional knapsack problem. The problem is a variant of the multidimensional knapsack problem where items are divided into classes, and exactly one item per class has to be chosen. Both problems are NP-hard. However, the multiple choice multidimensional knapsack problem appears to be more difficult to solve in part because of its choice constraints.

Many real applications lead to very large scale multiple choice multidimensional knapsack problems that can hardly be addressed using exact algorithms. In this paper, we explore a new hybrid heuristic that embeds several new procedures. Our heuristic is based on the resolution of linear programming relaxations of the problem and reduced problems that are obtained by fixing some variables of the problem. The solutions of these problems are used to update the global lower and upper bound for the optimal solution value. In this paper, we explore a new strategy for defining the reduced problems. We propose a new family of cuts and a reformulation procedure that is used at each iteration to improve the performance of the heuristic. We report on an extensive set of computational experiments for benchmark instances from the literature and for a large set of hard instances generated randomly. Our results show that our approach outperforms other state-of-the-art methods described so far, providing the best known solution for a significant number of benchmark instances.

1 Introduction

The multiple choice multidimensional knapsack problem (MMKP) is a combinatorial optimization problem with many applications on different fields such as the telecommunications, logistics and the financial sector [8, 32, 35, 11, 25, 26, 5]. The problem belongs to the family of knapsack problems [27, 22], and more specifically, it is a variant of the multidimensional knapsack problem (0-1 integer programming problem) which is characterized by the presence of more than a single resource (knapsack) constraint. In the MMKP, the items are divided into classes, and exactly one item per class must be chosen. Given a

[*]Corresponding author. Tel.: +351 253 604765; fax: +351 253604741.

Address: Dept. Produção e Sistemas, Universidade do Minho, 4710-057 Braga, Portugal

E-mail address: claudio@dps.uminho.pt (C. Alves).

profit for each item, the aim of the MMKP is to find the subset of items that maximizes the total profit and such that all the knapsack constraints are satisfied.

The MMKP is a NP-hard problem as it generalizes the standard knapsack problem. Unlike some other variants of the knapsack problem, the MMKP is very difficult to solve in practice. This is partly due to its choice constraints. Furthermore, even finding a feasible solution for the problem is NP-hard. As a consequence, it is not expectable that an exact method can provide optimal solutions in reasonable time for real-life applications. This fact has motivated the efforts of many researchers in developing fast and effective heuristics for this problem. Despite this renewed interest and the practical relevance of the problem, the number of approaches described in the literature for the MMKP remains small.

In this paper, we propose and analyze a new hybrid heuristic for the MMKP. The heuristic is based on the iterative computation of lower and upper bounds for the optimal solution value of the problem. These bounds are obtained by solving linear programming (LP) relaxations and reduced versions of the problem obtained by fixing the value of some variables. The global lower and upper bound and the information provided by the reduced problems is used to generate cutting planes and to fix some variables of the problem to their optimal value. To further improve the performance of our approach, we developed an original reformulation procedure that results in a reduction of the number of classes in the problem. To evaluate the performance of the heuristic, an extensive set of computational experiments was conducted on benchmark instances from the literature and on randomly generated instances. The results show the efficiency of our approach.

The paper is organized as follows. In Section 2, we review the main contributions described in the literature for the MMKP. In Section 3, we define formally all the elements of the MMKP. The different components of our algorithm are described in Section 4. The details related to the implementation of our approaches are given in Section 5. The results of our computational experiments are reported in Section 6, and some conclusions are drawn in Section 7.

2 Literature review

The first approaches described in the literature focused on a special case of the MMKP called the multiple choice knapsack problem (MCKP). In this problem, only a single knapsack constraint must be satisfied. Different exact methods are described in the literature for the MCKP. Sinha and Zoltners [37] proposed a branch-and-bound algorithm with a branching scheme guided by the solutions of the linear programming relaxations at each node. The computational experiments conducted by the authors on random instances show that the computing time increases faster with the number of classes than with the number of variables in the classes. Armstrong *et al.* [3] improved this algorithm and reduced the required memory space and computing time for the largest instances.

Dyer *et al.* [10] proposed a hybrid algorithm combining dynamic programming and branch-and-bound to solve the MCKP. The authors resort to Lagrangian duality to derive bounds at the branching nodes, and to apply a reduction procedure. The computational results reported in [10] show the potential of hybridization when compared to pure branch-and-bound algorithms.

Pisinger [31] described a polynomial partitioning algorithm to find an optimal solution for the LP relaxation. He also discussed the integration of this method into a dynamic programming algorithm that ensures the enumeration of a minimal number of classes. In its dynamic programming algorithm, classes are added to the problem core as needed. With this approach, the author improved the results obtained with other algorithms for large instances with more than 100000 variables.

To the best of our knowledge, the first results on the resolution of the MMKP are due to Moser *et al.* [28]. The authors developed a heuristic based on Lagrangian relaxation that starts from an infeasible solution, and permutes repeatedly the items to reduce this infeasibility. Their algorithm was improved later by Akbar *et al.* [1].

Khan *et al.* [24] proposed a heuristic based on the aggregation of the knapsack constraints as suggested by Toyoda in [38] for the multidimensional knapsack problem. They improved their approach by using a procedure for exchanging items. The heuristic was compared with a branch-and-bound algorithm and the method of Moser *et al.* [28]. The computational results provided in [24] show that the heuristic outperforms the other algorithms both in terms of the computing times and the quality of the solutions.

Parra-Hernandez and Dimopoulos [29] adapted the algorithm of Pirkul [30] for the multidimensional knapsack problem to solve the MMKP. Initially, they relax the choice constraints transforming the MMKP into a multidimensional knapsack problem with generalized upper bounds where at most one item can be selected from each class. An initial solution that may not be feasible is then obtained and improved using local search. Their algorithm finds better solutions than those obtained by Akbar *et al.* [1], but at the expense of a significant increase in the computing time.

In [17], Hifi *et al.* proposed two constructive heuristics and a guided local search method to solve the MMKP. Their approaches led to better results than those obtained by Moser *et al.* [28] and Khan *et al.* [24]. These results were later improved by the same authors in [18] using a reactive local search algorithm.

Akbar *et al.* [2] described a heuristic for the MMKP that relies on the generation of convex hulls and on the aggregation of the multidimensional knapsack constraints into a single one using a penalty vector. The authors obtained good results especially for the uncorrelated instances when compared to the algorithms of Moser *et al.* [28] and Akbar *et al.* [1]. In [19], Hiremath and Hill describe two greedy heuristics for the MMKP and provide an empirical study on specific test instances to show the performance of their algorithms.

In [7], Cherfi and Hifi described a branch-and-bound algorithm for the MMKP that uses a variant of the column generation algorithm and a rounding heuristic to solve some of the problems at the branching nodes. Different branching schemes were explored. Their approaches were compared with CPLEX [20] and the heuristic described in [18] using a set of benchmark instances. For 21 of 33 instances, the value of the best known lower bound was improved.

In [15], Hanafi *et al.* applied three iterative relaxation-based heuristics to solve the MMKP following the ideas introduced in [16]. The authors explored two different relaxations: LP relaxations and mixed integer programming relaxations where only part of the integrality constraints are relaxed. For the latter, they considered two strategies for choosing the subset of variables whose integrality constraints should be relaxed. Lower bounds are computed by solving a reduced problem obtained by fixing some of the variables to their value in the optimal linear solution. Their heuristics converge theoretically to an optimal solution. The authors compared their approach with the algorithms described by Cherfi and Hifi in [7]. They improved the lower bounds for 9 of 33 instances.

Using a similar approach as the one proposed in [15], Crévits *et al.* [9] explored a heuristic based on a new (semi-continuous) relaxation that consists in removing the integrality constraints and forcing the variables to be close to 0 or 1. This relaxation is more general than the LP and mixed integer programming relaxations used in [15]. The authors improved their algorithm by integrating a simple descent local search procedure that tries at each iteration to restore the feasibility of the solutions. This local search procedure exploits the special structure of the MMKP, and it is based on swapping objects between two classes. The authors resort also to fixation techniques based on the dual information obtained at each iteration after solving the associated relaxation. The results obtained with this method outperforms the results achieved in [15].

More recently, Cherfi and Hifi [6] described three new heuristics for the MMKP based respectively on a local branching algorithm, on a hybrid algorithm combining local branching with column generation, and on a truncated branch-and-bound algorithm that embeds the previous hybrid method. From the computational results reported in [6], it appears that the branch-and-bound algorithm dominates the other two methods. In fact, they obtained state-of-the-art results with this approach for the MMKP.

Some metaheuristics were also proposed recently for the MMKP [34, 21]. In [34], the authors describe an ant colony optimization algorithm for the problem. Their algorithm combines an efficient constructive procedure with an operator used to repair the infeasible solutions, and it further integrates dual information provided by a lagrangian relaxation of the problem. Iqbal *et al.* [21] developed also an ant colony optimization approach for the MMKP. The authors improved the convergence of their approach by using a local search routine in their algorithm. The authors claim that their method is able to find near optimal solutions in a short computing time.

Some exact algorithms were also described in the literature to solve the MMKP [36, 33, 14]. In [36], the author described a branch-and-bound algorithm that starts by computing a feasible solution using the heuristic proposed in [17]. The branching scheme of this algorithm consists in fixing one item in the solution, and in exploring the search tree using a best-first strategy. Upper bounds at the nodes of the branching tree are obtained by reducing the problem to a MCKP. The algorithm solved successfully small and medium sized instances with up to 50 classes, 15 items per class (750 variables) and 10 knapsack constraints. Razzazi and Ghasemi [33] proposed a different branch-and-bound method in which the nodes are explored using a depth-first strategy, and upper bounds are obtained by solving a surrogate relaxation of the problem. The computational results reported in [33] show that this algorithm outperforms the approach of Sbihi [36] for instances with up to 1000 items and 5 knapsack constraints. In [14], Ghasemi and Razzazi describe an exact algorithm for the MMKP based on an approximate core. The authors report on promising results for large uncorrelated instances and for correlated instances with up to 5 constraints.

In this paper, we describe a new hybrid heuristic for the MMKP that embeds several new procedures. Our approach is based on the resolution of LP relaxations and reduced problems obtained from the latter by fixing the values of some of the variables. We propose an alternative way of defining the reduced problems, and we describe a new set of cutting planes that can be derived from this original definition. These cuts are applied together with other families of cuts to strengthen the original formulation of the problem. To further improve the performance of our algorithm, we developed a reformulation procedure based on the combination of classes that aims at reducing the number of classes. Our algorithm was tested on a set of benchmark instances used in the literature. The results show that our approach outperforms the other state-of-the-art methods proposed so far, improving the value of the best known lower bounds for a significant number of instances. Additionally, we conducted a set of experiments on a large set of randomly generated instances. The results obtained for these instances confirm the quality of our approach.

3 The multiple choice multi-dimensional knapsack problem

The MMKP is defined by a set G of items divided into n disjoint groups (or classes) $G = G_1 \cup G_2 \cup \ldots \cup G_n$, with $G_i \cap G_{i'} = \emptyset$, for $i \neq i'$ and $i, i' \in \{1, \ldots, n\}$. Each class G_i has $n_i = |G_i|$ items. Each item $j \in G_i$, $i = 1, \ldots, n$ has a nonnegative profit c_{ij}, and a vector of coefficients $a_{ij} = (a_{ij}^1, a_{ij}^2, \ldots, a_{ij}^m)$ whose values are associated respectively to each of the m knapsack constraints of the problem. The capacities of the knapsacks are denoted by $b = (b^1, b^2, \ldots, b^m)$. The MMKP can be formulated as an integer programming problem as follows:

$$\max \quad \sum_{i=1}^{n} \sum_{j \in G_i} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \sum_{j \in G_i} a_{ij}^k x_{ij} \leq b^k, \; k = 1, \ldots, m, \tag{2}$$

$$\sum_{j \in G_i} x_{ij} = 1, \; i = 1, \ldots, n, \tag{3}$$

$$x_{ij} \in \{0, 1\}, \; i = 1, \ldots, n, \; j \in G_i. \tag{4}$$

The problem consists in choosing exactly one item per class so as to maximize the total profit without exceeding the capacities of the knapsacks. The binary variables x_{ij} take the value 1 if the j^{th} item in class i is selected, and 0 otherwise. Constraints (2) represent the knapsack constraints. Constraints (3) model the choice constraints, and ensure that one and only one item per class is selected. The maximization of the total profit is modeled by the linear objective function (1).

The LP relaxation of the MMKP is obtained from (1)-(4) by relaxing the integrality constraints (4) as follows:

$$\max \quad \sum_{i=1}^{n} \sum_{j \in G_i} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \sum_{j \in G_i} a_{ij}^k x_{ij} \leq b^k, \; k = 1, \ldots, m,$$

$$\sum_{j \in G_i} x_{ij} = 1, \; i = 1, \ldots, n,$$

$$x_{ij} \in [0, 1], \; i = 1, \ldots, n, \; j \in G_i.$$

Properties based on the simple dominance introduced by Armstrong *et al.* [3] can be used to reduce the size of the MMKP. For instance, an item j dominates an item l in a class i if the following conditions are satisfied: $c_{ij} \geq c_{il}$ and $a_{ij}^k \leq a_{il}^k$, for $k \in \{1, \ldots, m\}$. In that case, x_{il} can be set to 0.

4 Exploring a new hybrid heuristic

In this section, we describe the different components of our algorithm, but first we introduce some notations that will be used in the paper. Let P be an optimization problem (maximization) with a set N of binary variables, and let x^* and $\overline{x}$ be the optimal solutions of P and $LP(P)$ (the LP relaxation of P), respectively. In the paper, we will refer to basic and non-basic variables by reference to a LP basis that produces a given solution for $LP(P)$, *i.e.* by reference to a basic solution for $LP(P)$ with at most m' non-zero values (assuming that $LP(P)$ has m' constraints). Whenever we will refer to basic and non-basic variables, we will clearly mention to which basic solution we are referring to.

We denote by $v(P)$ the optimal value of P, and by $(P|C)$ the problem P with a set C of additional constraints. Let $\underline{v}(P)$ (resp. $\overline{v}(P)$) be a lower bound (resp. upper bound) for $v(P)$. By reference to the LP basis that produces $\overline{x}$, we define the following sets:

$$J^0(\overline{x}) = \{j \in N, x_j \text{ is non-basic and } \overline{x}_j = 0\}, \; J^1(\overline{x}) = \{j \in N, x_j \text{ is non-basic and } \overline{x}_j = 1\},$$
$$J(\overline{x}) = J^0(\overline{x}) \cup J^1(\overline{x}), \text{ and } J^*(\overline{x}) = \{j \in N, x_j \text{ is basic}\}.$$

The following problem associated to the optimal solution $\overline{x}$ of $LP(P)$ and to the index set $J(\overline{x})$ of non-basic variables is called the *reduced problem*:

$$P(\overline{x}, J(\overline{x})) = (P|x_j = \overline{x}_j, j \in J(\overline{x})). \tag{5}$$

4.1 General outline

The heuristic proposed in this paper for the MMKP is based on the iterative refinement of the upper and lower bounds for the optimal value of the MMKP. The bounds are improved by applying a method based on the general framework proposed by Wilbaut and Hanafi in [40] and enriched with new procedures. This framework consists in solving iteratively a relaxation of the current problem to update the value of the best upper bound, and in using the solutions of this relaxation to define a reduced problem which is solved in turn to improve the value of the best lower bound. The reduced problems are obtained from the current problem by fixing the values of some of its variables. In this paper, we explore a new strategy for fixing the values of the variables and defining the reduced problems. We propose also a new family of cuts that we use to further strengthen the model and to improve the quality of the subsequent upper bounds. Finally, we describe a reformulation procedure that is applied to improve the performance of the algorithm.

The outline of our heuristic is given in Figure 1. For the sake of clarity, we divided the main components of our heuristic into three blocks. Each block will be described in detail in a specific subsection.

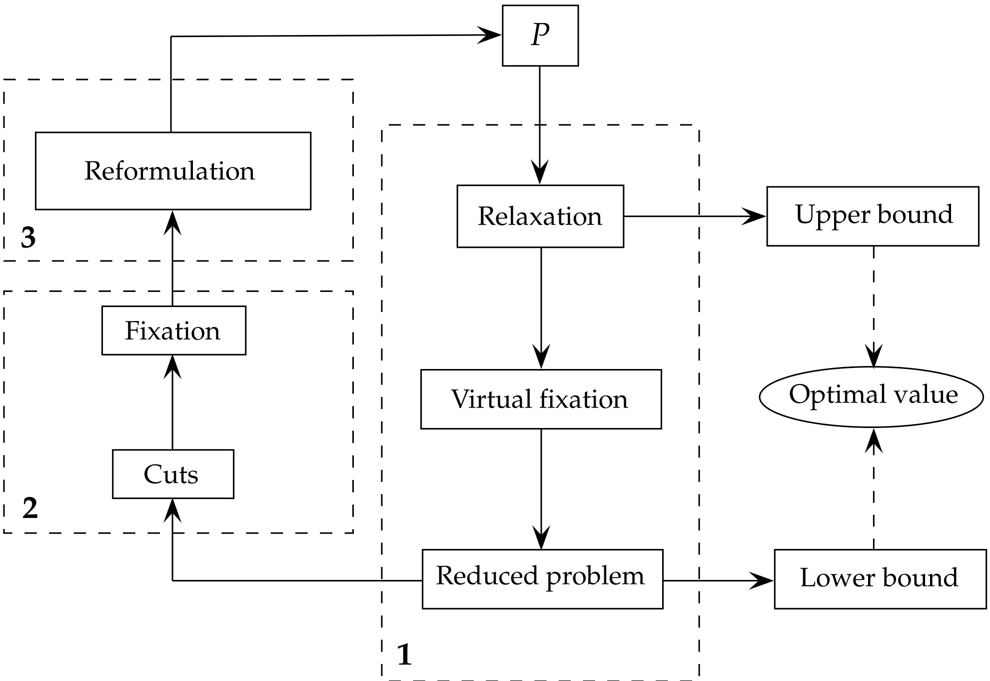

Figure 1: Outline of the heuristic

In our case, P denotes the MMKP which is represented initially by (1)-(4). At each iteration of the algorithm, we may fix the values of some of the variables of P to their optimal values by using the lower and upper bounds for the value of the optimal solution that are available at that moment. Additionally, cuts may be applied and followed by a reformulation of the problem as depicted in Figure 1. At the beginning of each iteration, the problem P that results from these operations performed in the previous iteration is called the current problem.

The upper bounds are computed by solving the LP relaxation $LP(P)$ of the current problem P. If the optimal solution of $LP(P)$ is better (smaller) than the current upper bound, then this bound is updated accordingly. In the next step, the solution $\overline{x}$ of $LP(P)$ is used to define a reduced problem. In our heuristic, we explore a new reduced problem denoted as $P(\overline{x}, J(\overline{x}, t))$, with $J(\overline{x}, t)$ being a subset of $J(\overline{x})$ ($J(\overline{x}, t) \subset J(\overline{x})$) associated with the solution $\overline{x}$ of $LP(P)$ and a parameter t. The parameter t identifies a subset of non-basic variables that will not be fixed in our reduced problem. In Figure 1, this process is denoted as "virtual fixation". It will be described in detail in Section 4.2. To the best of our knowledge, such a strategy was never used before to define the reduced problems. If the reduced problem is feasible, then its optimal solution is used to update the value of the best lower bound. These steps of the heuristic correspond the first block identified in Figure 1.

To avoid generating the same solution in a subsequent iteration, we enforce a so-called pseudo-cut in the current problem P to cut the solution of the reduced problem, and all the other dominated solutions. The formal definition of this pseudo-cut (that we will denote by $(cut-1)$) will be given in Section 4.3. Furthermore, we use the current lower and upper bound for the value of the optimal solution of the problem to apply valid cuts and to fix some its variables to their optimal values. The cuts are used to strengthen the model and to improve the quality of the upper bounds provided by the relaxations. Fixing the values of some variables allows us to reduce the size of the problem, and thus to accelerate the resolution of the relaxations and the reduced problems. The cuts and the fixing rules used in our heuristic will be discussed in Section 4.3. This part of our heuristic corresponds to the second block in Figure 1.

Finally, to improve the performance of our heuristic, we apply a reformulation procedure that consists in combining the classes two by two by enumerating all the possible combinations of variables related to these classes. This procedure leads to an equivalent reformulation of the original problem but with a smaller number of different classes. The coefficients of the variables in the knapsack constraints increase, and hence, fixing the values of the corresponding variables becomes easier. The procedure is applied only if a given percentage of the variables have been fixed. This procedure corresponds to the third block depicted in Figure 1. The details of the procedure are given in Section 4.4.

The process repeats until a stopping criterion is satisfied. In practice, the algorithm can be stopped after a given time limit has been reached, or when the gap between the upper and lower bound is less than a given tolerance.

4.2 Relaxation, virtual fixation procedure and reduced problem

At each iteration of the heuristic, we start by solving the LP relaxation $LP(P)$ of the current problem P to update the upper bound for the optimal value $v(P)$. The solution $\overline{x}$ of $LP(P)$ is then used to define a reduced problem that will be solved in turn to update the lower bound for $v(P)$. The reduced problem is obtained by fixing some of the variables of P to their values in $\overline{x}$. In this paper, we explore a new reduced problem that is different from the standard definition proposed in [40].

The standard definition of a reduced problem proposed in the general framework of Wilbaut and Hanafi [40] corresponds to (5). In this case, the variables of P that are fixed are the variables that are non-basic in the LP basis that produces $\overline{x}$. Let $\widehat{c}_j$ denote the reduced cost of the variable x_j in the LP basis related to $\overline{x}$. The following proposition holds ([4, 13, 12, 39]).

Proposition 1. *If the variable x_j is a non-basic variable in the LP basis that produces $\overline{x}$, and $|\widehat{c}_j| > c\overline{x} - \underline{v}(P)$, then at optimality we have $x_j^* = \overline{x}_j$.*

As a consequence, when we define the reduced problem using (5), we are assuming implicitly that $\underline{v}(P) = c\overline{x}$, since we are fixing all the non-basic variables such that $|\widehat{c}_j| > 0 = c\overline{x} - \underline{v}(P) = c\overline{x} - c\overline{x}$. From this point forward, we will use the term *virtual lower bound* to refer to this (assumed) lower bound, and we will denote it by $\overline{LB}$. Similarly, we will refer to this procedure as a *virtual fixation* procedure.

The following proposition shows the link between the optimal solution of a reduced problem and the corresponding virtual lower bound $\overline{LB}$.

Proposition 2. *If the reduced problem associated to the virtual lower bound $\overline{LB}$ is infeasible, or if the value of its optimal solution is strictly less than $\overline{LB}$, then $\overline{LB}$ is not a valid lower bound for P.*

Proof. The proof by negation follows trivially from Proposition 1. If $\overline{LB}$ is a valid lower bound for P, then the optimal solution of the obtained reduced problem after fixing the variables according to Proposition 1 exists, and its value is greater than or equal to $\overline{LB}$. If this solution does not exist, or if its value is smaller than $\overline{LB}$, this is contradictory with the fact that $\overline{LB}$ is a valid lower bound. $\square$

Obviously, if $\overline{LB}$ is not a valid lower bound, then it is an upper bound for the problem.

In our heuristic, the value of the virtual lower bound is chosen from the interval $[\underline{v}(P), c\overline{x}]$. For this purpose, we re-index first the non-basic variables associated to $\overline{x}$ such that $|\widehat{c}_j| \leq |\widehat{c}_{j+1}|$, *i.e.* such that the variables are sorted in increasing order of their corresponding reduced cost. Furthermore, let $J(\overline{x}, t) = J(\overline{x}) \setminus \{1, ..., t\}$ be the index set of the non-basic variables in $\overline{x}$ whose reduced cost is greater than $\widehat{c}_t$. The new fixing rule that we propose to define the reduced problem stands as follows:

if the variable x_j is a non-basic variable in the LP basis that produces $\overline{x}$ and if $j > t$, then we set $x_j = \overline{x}_j$ in our reduced problem.

The formal definition of our reduced problem follows:

$$P(\overline{x}, J(\overline{x}, t)) = (P | x_j = \overline{x}_j : |\widehat{c}_j| > |\widehat{c}_t|)\,. \tag{6}$$

This definition is equivalent to the following:

$$P(\overline{x}, J(\overline{x}, t)) = (P | x_j = \overline{x}_j : |\widehat{c}_j| > |\widehat{c}_{t+1}| - \epsilon)\,. \tag{7}$$

for a suitable value of ϵ such that $|\widehat{c}_{t+1}| - |\widehat{c}_t| \geq \epsilon > 0$. Because the alternative definition (7) of our reduced problem allows us to derive stronger cuts in a subsequent step of our heuristic (see ($cut-3$) described in Section 4.3), we will use it instead of (6).

To obtain our reduced problem, we fix the variables x_j of P that are non-basic in the LP basis that produces $\overline{x}$, and whose reduced cost $|\widehat{c}_j|$ is greater than $|\widehat{c}_{t+1}| - \epsilon$. From Proposition 1, that means that we are assuming $c\overline{x} - \underline{v}(P) = |\widehat{c}_{t+1}| - \epsilon$, and hence, that the virtual lower bound $\overline{LB}$ is equal to $c\overline{x} - (|\widehat{c}_{t+1}| - \epsilon)$. From Proposition 2, if $P(\overline{x}, J(\overline{x}, t))$ is infeasible for a given value t, or if the value of its optimal solution is less than $(c\overline{x} - (|\widehat{c}_{t+1}| - \epsilon))$, then this value is an upper bound for P. In the next section, we will show how this result can be used to derive new cutting planes ($cut-3$).

4.3 Cuts and fixing rule

In this section, we describe the cuts and fixing rules applied at each iteration of our heuristic after solving the reduced problem, and once the global lower and upper bounds have been updated. The addition of these cuts and the application of these fixing rules correspond to the second block depicted in Figure 1.

Three families of cuts are applied in our heuristic: a pseudo-cut to avoid generating the solution of the last reduced problem in subsequent iterations ($cut-1$), a cut based on the reduced costs of the solution $\overline{x}$ of $LP(P)$ ($cut-2$), and a new cut derived from the virtual fixation procedure described in the previous section ($cut-3$).

Let $J^0(\overline{x}, t) = \{j \in J(\overline{x}, t) : \overline{x}_j = 0\}$ and $J^1(\overline{x}, t) = \{j \in J(\overline{x}, t) : \overline{x}_j = 1\}$, such that $J(\overline{x}, t) = J^0(\overline{x}, t) \cup J^1(\overline{x}, t)$. The definition of the pseudo-cut for our reduced problem $P(\overline{x}, J(\overline{x}, t))$ follows from the pseudo-cut used in [16] for the standard reduced problem $P(\overline{x}, J(\overline{x}))$, which relies on the following proposition.

Proposition 3. *Let y be an optimal solution of the reduced problem $P\left(\overline{x}, J\left(\overline{x}\right)\right)$. An optimal solution of P is either the solution y, or an optimal solution of the problem:*

$$\left(P \,\middle|\, \left\{ \sum_{j \in J^1(\overline{x})} x_j - \sum_{j \in J^0(\overline{x})} x_j \leq \left|J^1\left(\overline{x}\right)\right| - 1 \right\}\right). \tag{8}$$

Proof. The proof can be found in [16]. □

In our case, the pseudo-cut is adapted from (8) and states as follows:

$$\sum_{j \in J^1(\overline{x},t)} x_j - \sum_{j \in J^0(\overline{x},t)} x_j \leq \left|J^1\left(\overline{x}, t\right)\right| - 1. \qquad (cut-1)$$

We omit the proof for the validity of this cut since it follows directly from the proof given in [16] for the standard reduced problem.

The second cut that we enforce in the current problem is based on the reduced costs of the variables in the solution $\overline{x}$ of $LP(P)$. This cut will be denoted by $(cut-2)$. It is defined as follows [39]:

$$\sum_{j \in J^0(\overline{x})} |\widehat{c}_j| x_j + \sum_{j \in J^1(\overline{x})} |\widehat{c}_j| \left(1 - x_j\right) \leq c\overline{x} - \underline{v}(P). \qquad (cut-2)$$

To show the validity of $(cut-2)$, we rewrite the LP relaxation $LP(P)$ in its standard form in the following way:

$$max\left\{cx : Ax + s = b, x \in [0,1]^n,\ s \geq 0\right\},$$

with s being the vector of slack variables. In the optimal basis related to $\overline{x}$, this LP relaxation can be written as follows:

$$\begin{aligned} \max \quad & c\overline{x} + \sum_{j \in J^0(\overline{x})} \widehat{c}_j x_j - \sum_{j \in J^1(\overline{x})} \widehat{c}_j \left(1 - x_j\right) + \widehat{d}s \\ \text{s.t.} \quad & \widehat{A}x + \widehat{t}s = \widehat{b}, \\ & x \in [0,1]^n,\ s \geq 0, \end{aligned}$$

with $(\widehat{c}, \widehat{d})$ being the vectors of reduced costs associated to the variables (x, s) of the optimal basis. By definition, for a given lower bound $\underline{v}(P)$, we have

$$c\overline{x} + \sum_{j \in J^0(\overline{x})} \widehat{c}_j x_j - \sum_{j \in J^1(\overline{x})} \widehat{c}_j \left(1 - x_j\right) + \widehat{d}s \geq \underline{v}(P),$$

which is equivalent to

$$\sum_{j \in J^0(\overline{x})} \widehat{c}_j x_j - \sum_{j \in J^1(\overline{x})} \widehat{c}_j \left(1 - x_j\right) \geq -c\overline{x} + \underline{v}(P) - \widehat{d}s,$$

and to

$$-\sum_{j \in J^0(\overline{x})} \widehat{c}_j x_j + \sum_{j \in J^1(\overline{x})} \widehat{c}_j \left(1 - x_j\right) \leq c\overline{x} - \underline{v}(P) + \widehat{d}s.$$

The slack variables that are basic have a reduced cost equal to 0, while the non-basic slack variables are equal to 0. Hence, the quantity $\widehat{d}s$ is equal to 0, and it can be removed from the above inequalities. Moreover, since $\widehat{c}_j \leq 0$ for $j \in J^0(\overline{x})$, and $\widehat{c}_j \geq 0$ for $j \in J^1(\overline{x})$, the inequality $(cut-2)$ follows. Note that the fixing rule described in Proposition 1 follows also from the definition of $(cut-2)$.

Our third cut follows from the virtual fixation procedure described in Section 4.2. In this section, we showed that our fixation procedure is equivalent to assuming a virtual lower bound $\overline{LB}$ equal to $c\overline{x} - (|\widehat{c}_{t+1}| - \epsilon)$. As mentioned also in this section, and according to Proposition 2, if the corresponding reduced problem is infeasible, or if its optimal value is less than $\overline{LB}$, then $\overline{LB}$ is in fact an upper bound for the optimal solution of the problem. In this case, we have:

$$cx < c\overline{x} - (|\widehat{c}_{t+1}| - \epsilon). \tag{9}$$

This constraint is not much useful essentially because of the degeneracy that it may induce. Indeed, both the hyperplanes of the objective function and the constraint (9) are parallel. The cutting planes that we propose here are similar to (9), but they are based on a different objective function which is derived from the original one. Our objective is to avoid the degeneracy issues related to (9).

At optimality, the reduced costs $\widehat{c}_j$ of the variables in $\overline{x}$ are as follows:

- $\widehat{c}_j = 0$, if $j \in J^*(\overline{x})$,
- $\widehat{c}_j \geq 0$, if $j \in J^1(\overline{x})$,
- $\widehat{c}_j \leq 0$, if $j \in J^0(\overline{x})$.

Let $J^*(\overline{x}, t) = N \setminus J(\overline{x}, t)$ (with N being the set of variables of P). Furthermore, let r be a vector such that

$$r_j = \begin{cases} c_j - \alpha_j, & \text{if } j \in J^0(\overline{x}, t), \\ c_j + \alpha_j, & \text{if } j \in J^1(\overline{x}, t), \\ c_j, & \text{if } j \in J^*(\overline{x}, t). \end{cases}$$

with α_j, $j \in J(\overline{x}, t)$, being positive scalars. Consider the LP relaxation of the problem P where the function cx is replaced by the function rx. We will denote the resulting problem by P'. From the definition of r, it follows that the reduced cost vector $\widehat{r}$ of the new problem is given by

$$\widehat{r}_j = \begin{cases} \widehat{c}_j - \alpha_j & \text{if } j \in J^0(\overline{x}, t), \\ \widehat{c}_j + \alpha_j, & \text{if } j \in J^1(\overline{x}, t), \\ \widehat{c}_j, & \text{if } j \in J^*(\overline{x}, t). \end{cases}$$

The solution $\overline{x}$ of $LP(P)$ remains optimal for the LP relaxation of the problem P'. Indeed, the objective function of P' is obtained by increasing the coefficients in the original objective function of P for non-basic variables that are equal to 1 in $\overline{x}$, and by decreasing the coefficients of non-basic variables that are equal to 0 in $\overline{x}$, while the coefficients of the remaining variables are not changed. Furthermore, $\overline{x}$ is feasible for P' and the associated reduced costs $\widehat{r}_j$, $j \in N$, satisfy the optimality conditions.

The next proposition introduces our third cut, which is derived from the objective function rx.

Proposition 4. *Let α_j, $j \in J(\overline{x}, t)$, be integer values. If the reduced problem $P'(\overline{x}, J(\overline{x}, t))$ is infeasible, or $v(P'(\overline{x}, J(\overline{x}, t))) < r\overline{x} - (|\widehat{r}_{t+1}| - \epsilon)$, then*

$$cx - \sum_{j \in J^0(\overline{x}, t)} \alpha_j x_j + \sum_{j \in J^1(\overline{x}, t)} \alpha_j (x_j - 1) \leq \lfloor c\overline{x} - (|\widehat{c}_{t+1}| - \epsilon) \rfloor - \alpha_{t+1}. \tag{cut − 3}$$

is a valid cutting plane for P.

Proof. If the reduced problem $P'(\overline{x}, J(\overline{x}, t))$ is infeasible, or if $v(P'(\overline{x}, J(\overline{x}, t)))$ is strictly less than $r\overline{x} - (|\widehat{r}_{t+1}| - \epsilon)$, then the following inequality is valid for the problem P', and hence it is also valid for P, since the set of feasible solutions is the same:

$$rx < r\overline{x} - (|\widehat{r}_{t+1}| - \epsilon). \tag{10}$$

From the definition of r, we can write the inequality (10) as

$$\sum_{j\in J^0(\overline{x},t)}(c_j-\alpha_j)x_j+\sum_{j\in J^1(\overline{x},t)}(c_j+\alpha_j)x_j+\sum_{j\in J^*(\overline{x},t)}c_jx_j<r\overline{x}-(|\widehat{r}_{t+1}|-\epsilon), \tag{11}$$

and

$$r\overline{x}=\sum_{j\in J^1(\overline{x},t)}(c_j+\alpha_j)\overline{x}_j+\sum_{j\in J^*(\overline{x},t)}c_j\overline{x}_j=\sum_{j\in J^1(\overline{x},t)}\alpha_j+c\overline{x}. \tag{12}$$

By replacing the value of $r\overline{x}$ from (12) in (11), and using the following equation

$$cx=\sum_{j\in J^0(\overline{x},t)}c_jx_j+\sum_{j\in J^1(\overline{x},t)}c_jx_j+\sum_{j\in J^*(\overline{x},t)}c_jx_j,$$

the inequality (11) can be rewritten as

$$cx-\sum_{j\in J^0(\overline{x},t)}\alpha_jx_j+\sum_{j\in J^1(\overline{x},t)}\alpha_jx_j<\sum_{j\in J^1(\overline{x},t)}\alpha_j+c\overline{x}-|\widehat{r}_{t+1}|+\epsilon. \tag{13}$$

By definition, we have $\widehat{r}_j=\widehat{c}_j-\alpha_j$, for $j\in J^0(\overline{x},t)$ and $\widehat{r}_j=\widehat{c}_j+\alpha_j$, for $j\in J^1(\overline{x},t)$. Since $\widehat{c}_j\leq 0$ for $j\in J^0(\overline{x})$, and $\widehat{c}_j\geq 0$ for $j\in J^1(\overline{x})$, and the scalars $\alpha_j\geq 0$ for $j\in J(\overline{x},t)$, we have $|\widehat{r}_j|=|\widehat{c}_j|+\alpha_j$ for $j\in J(\overline{x},t)$. Therefore, $|\widehat{r}_{t+1}|=|\widehat{c}_{t+1}|+\alpha_{t+1}$, and the inequality (13) becomes

$$cx-\sum_{j\in J^0(\overline{x},t)}\alpha_jx_j+\sum_{j\in J^1(\overline{x},t)}\alpha_j(x_j-1)<c\overline{x}-|\widehat{c}_{t+1}|-\alpha_{t+1}+\epsilon. \tag{14}$$

Since the coefficients α_j are integers, from the inequality (14), we conclude that the following inequality

$$cx-\sum_{j\in J^0(\overline{x},t)}\alpha_jx_j+\sum_{j\in J^1(\overline{x},t)}\alpha_j(x_j-1)\leq\lfloor c\overline{x}-(|\widehat{c}_{t+1}|-\epsilon)\rfloor-\alpha_{t+1}$$

is valid for the problem P. This completes the proof. □

In our case, we set α_j, $j\in J(\overline{x},t)$, equal to $\lceil(max_{i\in N}|\widehat{c}_i|)-|\widehat{c}_j|\rceil$. The consequence is that the variables with a small reduced cost will have a large coefficient in $(cut-3)$.

Based on the solution $\overline{x}$ and the updated upper and lower bounds, some of the variables may be fixed to their values in the optimal solution. In our heuristic, we applied the fixing rule presented in Proposition 1. Note that fixing one variable of the MMKP to 1 implies fixing to 0 all the other variables associated with the same class.

4.4 Reformulation procedure

After applying the cuts and the fixing rule described in the previous section, we proceed with a reformulation of the problem. The objective of this reformulation is to reduce the number of classes of the problem, and to increase the size of the coefficients associated with the variables in the knapsack constraints. The procedure consists in combining the classes of the problem two by two by enumerating all the possible combinations of variables of these two classes. The resulting reformulated problem is equivalent to the original one. The following example illustrates the application of this reformulation procedure.

Example 1. *Suppose that x_{i1}, x_{i2} and x_{i3} are the variables of a class i, and x_{j1}, x_{j2} and x_{j3} the variables of a class j. These two classes can be combined into a single class using the binary variables $x_{(i1,j1)}$, $x_{(i1,j2)}$, $x_{(i1,j3)}$, $x_{(i2,j1)}$, $x_{(i2,j2)}$, $x_{(i2,j3)}$, $x_{(i3,j1)}$, $x_{(i3,j2)}$ and $x_{(i3,j3)}$. The binary variable $x_{ih,jl}$ represents the selection of h^{th} object of class i and the l^{th} object of class j, i.e. $x_{ih,jl}=1\Leftrightarrow x_{ih}=1\wedge x_{jl}=1$.*

Note that if a variable has been previously fixed to the value 0, this variable is not combined with the variables of the other class. For example, if x_{i3} is set to 0, then the variables $x_{(i3,j1)}$, $x_{(i3,j2)}$ and $x_{(i3,j3)}$ will not be generated in our reformulated problem. Indeed, by definition, if $x_{i3} = 0$, we have $x_{(i3,j1)} = x_{(i3,j2)} = x_{(i3,j3)} = 0$.

For each pair of variables x_{ih} and x_{jl} of two classes i and j, respectively, such that neither x_{ih} nor x_{jl} have been fixed to the value 0, we introduce a binary variable $x_{ih,jl}$ in our reformulated problem (*i.e.* classes i and j are replaced by a single class). Furthermore, our reformulation procedure comprises the following steps:

- the choice constraints associated to classes i and j are replaced by a single choice constraint with the new variables (in Example 1, that will lead to the following choice constraint: $x_{(i1,j1)} + x_{(i1,j2)} + x_{(i1,j3)} + x_{(i2,j1)} + x_{(i2,j2)} + x_{(i2,j3)} + x_{(i3,j1)} + x_{(i3,j2)} + x_{(i3,j3)} = 1$);
- the coefficients $a^k_{ih,jl}$ of the new variables $x_{ih,jl}$ in the k^{th} knapsack constraint are made equal to $a^k_{ih} + a^k_{jl}$, for $k = 1, \ldots, m$;
- the coefficients $c_{ih,jl}$ for the new variables $x_{ih,jl}$ are set to $c_{ih} + c_{jl}$ in the objective function.

As referred to above, this procedure reduces the number of classes in the problem, but it also increases the value of the coefficients for the new variables. As a consequence, it helps in fixing more variables, and hence, in reducing even more the number of classes.

Since the application of this procedure at each iteration can increase significantly the number of variables of the problem, we apply it only if a given percentage of variables have been fixed. Furthermore, the procedure is applied only while the total number of generated variables remains smaller than a given limit. In our computational experiments, we explored different strategies for combining the classes. For instance, we applied the reformulation procedure by combining the classes with the smallest number of variables, by combining the classes with the smallest and the largest number of variables, and by combining the classes with the largest number of variables. The results are reported in Section 6.

5 Implementation details

Before discussing the results of our computational experiments, we describe first the details of the implementation of our heuristic. These details are illustrated in Algorithm 1.

As mentioned above, our heuristic starts by solving the LP relaxation $LP(P)$ of the current problem P. Its optimal solution is denoted by $\overline{x}$. From this solution, we define the reduced problem $P(\overline{x}, J(\overline{x}, t))$ associated with the parameter t (the t^{th} smallest reduced cost of the non-basic variables after a reordering has been applied). The solution of the reduced problem is denoted by $\underline{x}$ in Algorithm 1. The LP relaxations and the reduced problems are solved with a commercial solver. In our computational experiments, we used CPLEX 11.2 [20]. Note that although the reduced problems remain instances of the MMKP, they are much smaller than the original instance, and, in practice, they are solved efficiently with CPLEX as illustrated in the computational results reported in the following section.

The solutions of the LP relaxation and reduced problem allow us to update respectively the upper and lower bounds for the problem (and hence, the optimality gap). The cuts described in Section 4.3 are then applied to the current problem, and followed by the fixing procedure related to Proposition 1. Finally, the reformulation procedure is invoked if and only if a given percentage (min_{fix}) of variables have been fixed since the last reformulation, and with a maximum number of allowed variables (max_G) for the reformulated problem. In Algorithm 1, and for illustration purposes, we considered that the classes that are combined are those with the smallest number of variables. However, as alluded in Section 4.4,

Algorithm 1: Hybrid heuristic for the MMKP

Input:

P: instance of the MMKP;

t: parameter used to define $P(\overline{x}, J(\overline{x}, t))$;

T: limit on the total computing time;

δ: limit value for the optimality gap used as a stopping criterion;

M: large positive value;

max_G: max. number of variables that can be generated through the reformulation procedure;

min_{fix}: min. percentage of variables that must have been fixed to apply the reformulation procedure;

Output:

$\underline{v}(P)$: global lower bound for the problem;

$\underline{v}(P) := -M$; $\overline{v}(P) := M$; $gap := 2M$; $f' := 0$;

while $(CPU_{tot} < T)$ ***and*** $(gap \geq \delta)$ **do**

  Let $\overline{x}$ be the optimal solution of the LP relaxation $LP(P)$ of the problem P;

  if $(c\overline{x} < \overline{v}(P))$ **then**

    Update the global upper bound $\overline{v}(P)$ to $c\overline{x}$;

  end

  Let $\underline{x}$ be the optimal solution of the reduced problem $P(\overline{x}, J(\overline{x}, t))$ associated with t;

  if $(c\underline{x} > \underline{v}(P))$ **then**

    Update the global lower bound $\underline{v}(P)$ to cx^*;

  end

  $gap := \overline{v}(P) - \underline{v}(P)$;

  $P := (P|(cut-1), (cut-2), (cut-3))$;

  Apply the fixing rule based on Proposition 1 to the variables of P;

  Let f'' be the number of variables that have been fixed in the previous step;

  Let f be the number of variables in the current problem P;

  $f' := f' + f''$;

  $(i', j') := Argmin\{|G_i| + |G_j| : (i, j)\}$;

  if $(f'/f \geq min_{fix})$ **then**

    while $(|G| - |G_{i'}| - |G_{j'}| + (|G_{i'}| \times |G_{j'}|)) < max_G$ **do**

      `/* Note that, in the above expression, the values` $|G_i|$ `do not take into account the variables whose values may have been fixed to 0, since these variables are not combined with other variables as mentioned in Section 4.4 */`

      Combine the classes i' and j' of problem $P \longrightarrow P$;

      $(i', j') = Argmin\{|G_i| + |G_j| : (i, j)\}$;

    end

    $f' := 0$;

  end

end

other strategies may be followed. In our computational experiments, we report on results for alternative combination strategies.

Our heuristic stops after one of the two following criteria has been met. The first one indicates that a proven optimal solution was found, or no feasible solution exists (when $\underline{v}(P) = -M$), and it is defined by the two following conditions:

- the optimality gap is less than a small positive value δ (note that if $\delta = 1$, and since all the coefficients in the problem are integer, this criterion will only be satisfied when an optimal integer solution has been reached);

- the number of classes in the current problem is equal to 1 (in this case, we just have to choose the best variable in this class).

The second criterion is a limit on the total computing time. When the heuristic stops because of this limit, $\underline{v}(P)$ represents a heuristic solution value for the problem. In Algorithm 1, the total computing time spent by the heuristic is denoted by CPU_{tot}. The convergence of our heuristic can be ensured under the same conditions as those stated in ([40]). However, since the convergence can be long, that justifies the consideration of a time limit as a stopping criterion.

6 Computational experiments

In this section, we report on the computational experiments that we conducted to evaluate the performance of our heuristic. We used two sets of instances for this purpose: a set of benchmark instances from the literature and a set of hard instances generated randomly. Our heuristic was coded in C++, and our tests were run on a Dell computer with 2.4 GHz and 4 GB of RAM. As referred to in Section 5, we used CPLEX 11.2 to solve the LP relaxation of the problem and the reduced problem at each iteration of the heuristic.

We performed two sets of experiments. The first set was conducted on the benchmark instances. Our objective was both to compare our heuristic with other state-of-the-art methods using the same instances as those used by the authors of these methods, and to analyze the behavior of our heuristic for different sets of parameters and strategies. Our second set of experiments was conducted on 256 instances generated randomly using the procedure described in [23]. Our aim was to compare the performance of our algorithm with other state-of-the-art methods on a large set of instances with different characteristics, and to analyze the impact of these varying characteristics on the quality of the results provided by our approach.

In Table 1, we describe the characteristics of the benchmark instances. The set is composed by 27 instances. The main characteristics of these instances are the number of knapsack constraints (m), the number of classes (n), the number of items in each class (n_i), and the total number of variables of the problem (n'). The first 7 instances, denoted by $I07, \ldots, I13$, were generated and used by Khan *et al.* [23]. Note that the original set was composed by 13 instances $I01$ to $I13$. However, we did not consider the first 6 instances because these are easy instances that are typically solved by all the methods considered in this paper (including ours) and by CPLEX within a short computing time. The instances denoted by $INST01, \ldots, INST20$ are large and difficult instances. They were generated by Hifi *et al.* [18] according to the procedure described by Khan in [23].

In Table 1, we provide also the complete list of results obtained with the best version of the algorithms proposed in [6] (column CH), and the best algorithms described in [15] (column HMW) and [9] (column CHMW). We will refer to these methods respectively by CH, HMW and CHMW. These methods are state-of-the-art for the MMKP, and hence we will compare our heuristic to them. Additionally, we report on the best solutions obtained with CPLEX 11.2 within a time limit of one hour (column CPLEX). Column $\underline{v}(P)$ in Table 1 represents the value of the lower bound obtained by the corresponding algorithm. Column *cpu* indicates the total computing time that was needed to compute the corresponding lower bound. The best lower bound provided by CH, HMW, CHMW and CPLEX is reported in column MAX_{LIT}. From this point forward, the best bound obtained from CH, HMW, CHMW and CPLEX will be denoted by MAX_{LIT}. Furthermore, in Table 1, we highlighted in bold the bounds given by each one the methods and CPLEX whenever they are equal to the best bound MAX_{LIT} for the corresponding instance.

In Table 2, we compare the results obtained with our heuristic using three different strategies for combining the classes in the reformulation procedure. These strategies are related to the cardinalities of the classes that are combined in this reformulation. In the column MACH1, we give the results obtained when we combine the classes (i', j') with the smallest number of items ($(i', j') := Argmin\{|G_i| + |G_j| : (i, j)\}$). In column MACH2, we report on the results achieved when the classes (i', j') with the largest number of items are combined ($(i', j') := Argmax\{|G_i| + |G_j| : (i, j)\}$). Column MACH3 gives the results obtained when we combine the class i' with the smallest number of items with the class j' that has the largest number of items ($i' := Argmin\{|G_i| : i\}$, $j' := Argmax\{|G_j| : j\}$). From this point forward, we will refer to each one of these methods respectively by MACH1, MACH2 and MACH3.

The results given by MACH1, MACH2 and MACH3 are compared with the best bound MAX_{LIT} given by the methods CH, HMW, CHMW and CPLEX. Whenever one of our methods (MACH1, MACH2 or MACH3) improves the best bound MAX_{LIT}, the corresponding value is highlighted in bold. When one of our bounds equals the best bound MAX_{LIT}, the corresponding value appears in italic. Similarly, if the best bound MAX_{LIT} is the best among all the methods considered in this paper and CPLEX, but if at least one of our methods provides the same bound, then this value appears in italic. If a bound is the only best among all the methods, then its value is highlighted in bold and followed by an asterisk. For these tests, we set the parameter t equal to 7. The parameters max_G and min_{fix} were set equal to 100000 and 30%, respectively. The parameter δ is set to a small positive value strictly less than 1. Our experiments were conducted with a time limit of 500 seconds.

The quality of the results provided by the three strategies is quite similar. All the strategies improve the best lower bound for a significant number of instances. Here, it is important to note that this best bound MAX_{LIT} is chosen from a set of 3 different methods (CH, HMW and CHMW) and CPLEX. The best bound given by MACH1 is equal to MAX_{LIT} for 6 instances. It improves the value of MAX_{LIT} for 13 instances and it provides the only best bound for 5 instances. MACH2 improves the bound for 12 instances with 5 of these bounds being the only best among all the other methods. For 10 instances, the bound given by MACH2 is equal to MAX_{LIT}. MACH3 improves the bound of 11 instances, and provides the only best bound for 8 of these instances. For 5 instances, it provides a bound that is equal to MAX_{LIT}. If we compare the best bound MAX_{LIT} with the best bound provided by MACH1, MACH2 and MACH3, we conclude that our methods improved the lower bound for 16 of the 27 instances, and gave a bound equal to MAX_{LIT} for 6 instances. As referred to above, the quality of the results is not significantly affected by the strategy used to combine the classes in our reformulation procedure. However, because MACH3 gave the only best bound for a larger number of instances, we used the corresponding strategy for the next experiments.

In Table 3, we report on the computational experiments that we conducted to evaluate the impact of the parameter t on the quality of the results provided by our heuristic. We tested the heuristic for three different values of t ($t \in \{3, 7, 15\}$). The columns $\underline{v}(P)$ and $\overline{v}(P)$ represent respectively the lower and upper bound given by the corresponding method. We used the version MACH3 of the heuristic, and we kept all the other parameters unchanged. As happened with the strategy used to combine the classes in the reformulation procedure, the quality of the results do not change significantly for different values of t. For $t = 3$, our heuristic provides a bound better than MAX_{LIT} for 11 instances with 7 of these

Instance	m	n	n_i	n'	CPLEX	CH		HMW		CHMW		MAX_{LIT}
						$\underline{v}(P)$	*cpu*	$\underline{v}(P)$	*cpu*	$\underline{v}(P)$	*cpu*	
I07	10	10	100	1000	**24589**	24587 24587	< 600 <1200	24586	<300	24585	41	24589
I08	10	10	150	1500	**36896**	36894 36894	< 600 <1200	36883	<300	36885	291	36896
I09	10	10	200	2000	49169	**49179** 49179	< 600 <1200	49172	<300	49172	193	49179
I10	10	10	250	2500	61462	61464 61464	< 600 <1200	**61465**	<300	61460	137	61465
I11	10	10	300	3000	73776	73777 **73783**	< 600 <1200	73770	<300	73778	133	73783
I12	10	10	350	3500	**86087**	86080 86080	< 600 <1200	86077	<300	86077	325	86087
I13	10	10	400	4000	98431	98433 **98438**	< 600 <1200	98428	<300	98429	313	98438
INST01	10	50	10	500	**10738**	**10738** 10738	< 600 <1200	10714	<300	10732	318	10738
INST02	10	50	10	500	**13598**	**13598** 13598	<600 <1200	13597	<300	**13598**	17	13598
INST03	10	60	10	600	10939	**10944** 10944	< 600 <1200	10943	<300	10943	233	10944
INST04	10	70	10	700	14442	14442 14442	< 600 <1200	14442	<300	**14445**	11	14445
INST05	10	75	10	750	17053	17053 17053	< 600 <1200	**17061**	<300	17055	396	17061
INST06	10	75	10	750	16826	**16827** 16827	< 600 <1200	16823	<300	16823	55	16827
INST07	10	80	10	800	**16440**	**16440** 16440	< 600 <1200	16434	<300	**16440**	196	16440
INST08	10	80	10	800	17502	**17510** 17510	< 600 <1200	17503	<300	17505	95	17510
INST09	10	80	10	800	17751	**17761** 17761	< 600 <1200	17751	<300	17753	271	17761
INST10	10	90	10	900	19304	**19316** 19316	< 600 <1200	19311	<300	19306	365	19316
INST11	10	90	10	900	19433	**19441** 19441	< 600 <1200	19430	<300	19434	339	19441
INST12	10	100	10	1000	21720	21732 21732	< 600 <1200	**21738**	<300	**21738**	43	21738
INST13	10	100	30	3000	21573	**21577** 21577	< 600 <1200	21574	<300	21574	147	21577
INST14	10	150	30	4500	32869	**32874** 32874	< 600 <1200	32869	<300	32869	41	32874
INST15	10	180	30	5400	**39160**	**39160** 39160	< 600 <1200	**39160**	<300	**39160**	233	39160
INST16	10	200	30	6000	**43363**	43362 43362	< 600 <1200	**43363**	<300	**43363**	83	43363
INST17	10	250	30	7500	**54360**	54352 **54360**	< 600 <1200	54356	<300	54352	89	54360
INST18	10	280	20	6500	**60465**	60460 60460	< 600 <1200	60462	<300	60463	116	60465
INST19	10	300	20	6000	**64928**	64925 64925	< 600 <1200	64925	<300	64924	32	64928
INST20	10	350	20	7000	**75617**	75612 75612	< 600 <1200	75609	<300	75609	253	75617

Table 1: Benchmark instances and results from the literature

bounds being the only best among all the cases reported in Table 3. For 7 instances, the bound is equal to MAX_{LIT}. MACH3 with $t = 7$ improves MAX_{LIT} for 11 instances with 9 of these bounds being the only best. It gives a bound equal to MAX_{LIT} for 5 instances. MACH3 with $t = 15$ gives a bound equal to MAX_{LIT} in 4 cases, and it improves it for 10 instances (5 of these bounds are also the only best among all these approaches). Note that the results are slightly worse for $t = 15$. This can be explained by the fact that the corresponding reduced problems are larger, and hence, CPLEX finds more difficulties in solving them. If we compare MAX_{LIT} with the best bounds given by MACH3 with $t = 3$, $t = 7$ and $t = 15$, the number of improved bounds is now equal to 15, while the number of instances for which our approaches return a bound equal to MAX_{LIT} is 5. Additionally, we report in Table 3 the values of the best upper bounds $\overline{v}(P)$ given by MACH3 for each value of t. Increasing the value of t has an impact on the strength of the pseudo-cut $(cut - 1)$ and the cut $(cut - 3)$. As a consequence, the value of the best upper bound decreases naturally for increasing values of t as it can be observed in Table 3.

The last set of experiments was performed on 256 hard instances generated randomly with the procedure described in [23] and with different values for the number m of knapsack constraints, the number n of different classes and the number n_i of items per class, namely $m \in \{10, 15, 20, 25\}$, $n \in \{100, 250, 500, 700\}$ and $n_i \in \{10, 15, 20, 25\}$. For each combination of these parameters, we generated 4 different instances. These experiments were conducted with a time limit of 600 seconds.

In the Appendix A, we provide the exhaustive list of computational results obtained for these instances with CPLEX, HMW, CHMW and MACH3 with $t = 7$ and the same parameters as those used in the previous experiments. In the Tables 5-8, the results are grouped according to the number m of knapsack constraints. For these experiments, we used a time limit of 600 seconds. For the sake of clarity, we give in Table 4 the number of times each one of these methods provide the best bound among all the methods and the number of times these bounds were the only best. In the row *global*, these values are given for the whole set of 256 instances. To evaluate the impact of each parameter, we give also the number of times that each method finds the best and the only best bound for instances with the same value of m, n and n_i. These results are reported also in Table 4. Hence, the other rows of this table correspond to the results obtained for subsets of 64 instances.

Our heuristic outperforms the other methods including CPLEX. It finds the best bound for 184 of the 256 instances, and for 176 of these instances, these bounds are also the only best among all the methods and CPLEX. The behavior of the heuristic remains approximately the same for increasing values of m, n and n_i as it can be observed in Table 4. Whatever the parameter that is considered, our heuristic reaches the best bound much more times than the other approaches including CPLEX, and, on average, in 96% of the cases these bounds are also the only best among all these methods.

Moreover, our heuristic always find a feasible solution for these instances, while the other methods fail in finding a feasible solution for some of them. In the Tables 5-8, when a method do not find a feasible solution within the time limit, a $-$ appears in the corresponding cell. This fact illustrates the hardness of these instances.

7 Conclusions

In this paper, we described a new hybrid heuristic for the MMKP. The problem is very relevant in practice because many real applications can be formulated as a multiple choice multidimensional knapsack problem. This problem is also very difficult to solve in part because of the choice constraints. Until now few approaches were proposed in the literature to solve it.

The approach described in this paper consists on a hybrid heuristic that refines iteratively the value of the global lower and upper bound for the optimal solution value of the problem. The lower bounds are computed by solving at each iteration a LP relaxation of the problem that is strengthened with

Instance	MAX_{LIT}	MACH1		MACH2		MACH3	
		$\underline{v}(P)$	*cpu*	$\underline{v}(P)$	*cpu*	$\underline{v}(P)$	*cpu*
I07	24589	24584	1	24585	101	24584	1
		24587	227	**24590***	344	24586	120
I08	**36896***	36886	248	36872	17	36885	408
		36887	443	36887	29	36888	479
I09	49179	49170	9	49173	72	49166	67
		49182*	102	**49180**	207	**49180**	134
I10	61465	**61467**	55	**61467**	31	**61472**	382
		61469	74	**61470**	406	**61480***	443
I11	73783	73775	31	**73788**	138	73781	373
		73785	139	**73789***	406	*73783*	564
I12	86087	*86087*	61	86086	149	**86088**	329
		86091	475	**86088**	218	**86094***	453
I13	98438	98431	24	98436	65	98436	66
		98440*	50	*98438*	130	*98438*	252
INST01	*10738*	10720	145	10714	12	10719	68
		10738	169	*10738*	106	10724	444
INST02	*13598*	13596	1	13595	20	13597	113
		13598	86	*13598*	70	*13598*	342
INST03	10944	10940	190	**10945**	84	10937	45
		10947	324	**10949***	158	10938	257
INST04	14445	**14447**	213	14426	15	14438	58
		14454	394	*14445*	26	**14456***	461
INST05	**17061***	17053	2	17044	141	17004	1
		17057	88	17053	263	17053	5
INST06	16827	**16830**	138	**16829**	142	**16828**	58
		16835*	486	**16832**	525	**16832**	244
INST07	16440	16412	12	16425	72	*16440*	24
		16440	33	*16440*	76	**16442***	554
INST08	**17510***	17501	75	17499	9	17501	25
		17502	512	17504	119	17508	377
INST09	**17761***	17748	69	17739	21	17756	279
		17752	221	17755	50	17760	318
INST10	*19316*	19314	97	19314	37	19309	82
		19316	157	*19316*	140	19311	147
INST11	*19441*	19419	6	19440	457	19434	458
		19432	41	*19441*	519	19437	528
INST12	*21738*	21728	114	21728	6	21733	129
		21738	277	*21738*	56	*21738*	248
INST13	*21577*	21575	129	21575	227	21575	255
		21577	454	*21577*	415	*21577*	544
INST14	**32874***	32869	31	32871	239	32870	26
		32871	100	32872	490	32872	509
INST15	39160	*39160*	49	39158	74	39158	75
		39161*	407	*39160*	266	**39161***	231
INST16	43363	**43365**	294	*43363*	240	*43363*	165
		43366*	409	**43364**	489	**43366***	227
INST17	54360	54357	35	54358	232	*54360*	423
		54361	106	**54361**	477	**54363***	490
INST18	60465	60462	32	60464	103	60464	61
		60466	128	**60466**	163	**60467***	171
INST19	64928	*64928*	11	**64929**	243	**64930**	161
		64931	136	**64932***	304	**64931**	386
INST20	75617	75612	93	75613	194	75613	102
		75614	179	**75618***	316	75614	171

Table 2: Comparison between alternative reformulation strategies

Instance	MAX_{LIT}	$t=3$		$t=7$		$t=15$	
		$\underline{v}(P)$	$\overline{v}(P)$	$\underline{v}(P)$	$\overline{v}(P)$	$\underline{v}(P)$	$\overline{v}(P)$
I07	*24589*	*24589*	24606,0	24586	24605,6	24587	24605,2
I08	**36896***	36889	36903,4	36888	36903,2	36888	36902,9
I09	49179	**49182***	49193,3	**49180**	49193,1	49175	49192,8
I10	61465	**61475**	61485,7	**61480***	61485,7	**61476**	61485,4
I11	73783	**73788***	73797,2	*73783*	73797,2	**73785**	73797,1
I12	86087	**86090**	86100,0	**86094***	86100,0	**86091**	86099,8
I13	98438	**98440***	98448,2	*98438*	98448,1	*98438*	98447,9
INST01	**10738***	10725	10750,2	10724	10749,2	10719	10748,7
INST02	*13598*	*13598*	13624,1	*13598*	13622,1	13597	13619,3
INST03	10944	**10949***	10980,2	10938	10979,9	10940	10977,7
INST04	14445	**14447**	14475,5	**14456***	14475,5	**14447**	14474,0
INST05	**17061***	17053	17076,5	17053	17076,0	17050	17075,2
INST06	16827	*16827*	16854,4	**16832**	16854,1	**16838***	16853,5
INST07	16440	*16440*	16457,9	**16442***	16457,7	**16442***	16456,2
INST08	*17510*	*17510*	17531,7	17508	17531,3	*17510*	17530,8
INST09	**17761***	17758	17778,4	17760	17778,1	17760	17777,3
INST10	**19316***	19309	19335,7	19311	19335,3	19314	19334,7
INST11	**19441***	19437	19461,2	19437	19460,8	19437	19459,9
INST12	*21738*	21737	21755,7	*21738*	21755,5	21737	21754,9
INST13	*21577*	*21577*	21592,4	*21577*	21592,1	*21577*	21591,3
INST14	32874	32871	32886,9	32872	32886,8	**32875***	32886,4
INST15	39160	**39161***	39174,1	**39161***	39174,0	39159	39173,5
INST16	43363	**43365**	43378,9	**43366***	43378,9	**43366***	43378,6
INST17	54360	*54360*	54371,8	**54363***	54371,7	*54360*	54371,3
INST18	60465	**60467***	60478,2	**60467***	60478,0	**60466**	60477,9
INST19	64928	**64931***	64943,4	**64931***	64943,4	**64931***	64943,0
INST20	**75617***	75614	75626,7	75614	75626,7	75615	75626,4

Table 3: Comparative results on benchmark instances for different values of t

different families of cuts. The upper bounds are obtained by solving a reduced problem induced by the optimal solution of the LP relaxation. In this paper, we explored a new strategy for defining the reduced problems. After solving the reduced problem, we apply different cuts to the current problem, and we fix some variables to their optimal value. Finally, to improve the performance of the heuristic, we apply a reformulation to the current problem.

To evaluate the performance of our heuristic, we performed an extensive set of computational experiments on benchmark instances from the literature and on hard instances generated randomly. The benchmark instances were also used to compare the quality of the results provided by different strategies and parameters of our heuristic. The results obtained for the two sets of instances show that our approach outperforms other state-of-the-art methods described in the literature and also CPLEX which is one the best commercial solvers that is currently available. The quality of the results remains approximately unchanged for different strategies and parameters of the heuristic.

		best				*only best*			
		CPLEX	**HMW**	**CHMW**	**MACH3**	**CPLEX**	**HMW**	**CHMW**	**MACH3**
global		15	22	45	184	13	18	41	176
m	10	6	9	6	49	4	6	4	45
	15	4	2	5	56	4	1	4	53
	20	4	7	10	43	4	7	10	43
	25	1	4	24	36	1	4	23	35
n	100	3	8	9	47	3	8	7	44
	250	4	1	12	47	4	1	12	47
	500	5	6	12	45	4	4	11	42
	700	3	7	12	45	2	5	11	43
n_i	10	4	5	10	47	4	4	9	45
	15	4	3	7	54	3	2	5	51
	20	3	11	13	39	2	9	13	38
	25	4	3	15	44	4	3	14	42

Table 4: Results for the hard instances generated randomly

Acknowledgements

We would like to thank the anonymous referees for their constructive comments, which led to a clearer presentation of the material.

This work was partially supported by the Portuguese Science and Technology Foundation through the postdoctoral grant SFRH/BPD/64766/2009 for Raïd Mansi, by the Algoritmi Research Center of the University of Minho for Cláudio Alves and José Valério de Carvalho, and by the International Campus on Safety and Intermodality in Transportation, the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the Ministry of Higher Education and Research, and the National Center for Scientific Research for Saïd Hanafi.

References

[1] M. Akbar, O. Ergun, and A. Punnen. Heuristic solutions for the multiple-choice multi-dimensional knapsack problem. In *International Conference on Computer Science, San Francisco, USA*, 2001.

[2] M. Akbar, M. Rahman, M. Kaykobad, E. Manning, and G. Shoja. Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Computers and Operations Research*, 33(5):1259–1273, 2006.

[3] R. Armstrong, D. Kung, P. Sinha, and A. Zoltners. A computational study of a multiple-choice knapsack algorithm. *ACM Transactions on Mathematical Software*, 9(2):184–198, 1983.

[4] E. Balas and C. Martin. Pivot and complement - a heuristic for zero-one programming. *Management Science*, 26(1):86–96, 1980.

[5] C. Basnet and J. Wilson. Heuristics for determining the number of warehouses for storing non-compatible products. *International Transactions in Operational Research*, 12(5):527–538, 2005.

[6] N. Cherfi and M. Hifi. Hybrid algorithms for the multiple-choice multi-dimensional knapsack problem. *International Journal of Operational Research*, 5(1):89–109, 2009.

[7] N. Cherfi and M. Hifi. A column generation method for the multiple-choice multi-dimensional knapsack problem. *Computational Optimization and Applications*, 46(1):51–73, 2010.

[8] C. Chocolaad. Solving geometric knapsack problems using tabu search. *AFIT/GOR/ENS/98M-05, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio*, 1998.

[9] I. Crévits, S. Hanafi, R. Mansi, and C. Wilbaut. Iterative semi-continuous relaxation heuristics for the multiple-choice multidimensional knapsack problem. *Computers and Operations Research (in press)*, 2011.

[10] M. Dyer, W. Riha, and J. Walker. A hybrid dynamic programming/branch-and-bound algorithm for the multiple-choice knapsack problem. *Journal of Computational and Applied Mathematics*, 58:43–54, 1995.

[11] Y. Feng. Resource allocation in ka-band satellite systems. *MSc Thesis, Institute for Systems Research, University of Maryland*, 2001.

[12] A. Freville and G. Plateau. Heuristics and reduction methods for multiple constraints 0-1 linear programming problems. *European Journal of Operational Research*, 24:206–215, 1986.

[13] B. Gavish and H. Pirkul. Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. *Mathematical Programming*, 31:78–105, 1985.

[14] T. Ghasemi and M. Razzazi. Development of core to solve the multidimensional multiple-choice knapsack problem. *Computers and Operations Research*, 60:349–360, 2010.

[15] S. Hanafi, R. Mansi, and C. Wilbaut. Iterative relaxation-based heuristics for the multiple-choice multidimensional knapsack problem. volume 5818 of *Lecture Notes in Computer Science*, pages 73–83. Springer Verlag, 2009.

[16] S. Hanafi and C. Wilbaut. Improved convergent heuristics for the 0-1 multidimensional knapsack problem. *Annals of Operations Research*, 183(1):125–142, 2011.

[17] M. Hifi, M. Michrafy, and A. Sbihi. Heuristic algorithms for the multiple-choice multidimensional knapsack problem. *Journal of the Operational Research Society*, 55(12):1323–1332, 2004.

[18] M. Hifi, M. Michrafy, and A. Sbihi. A reactive local search-based algorithm for the multiple-choice multi-dimensional knapsack problem. *Computational Optimization and Applications*, 33:271–285, 2006.

[19] C. Hiremath and R. Hill. New greedy heuristics for the multiple-choice multi-dimensional knapsack problem. *International Journal of Operational Research*, 2(4):495–512, 2007.

[20] Ilog. *Ilog CPLEX 10.0 Reference Manual.* 9, rue de Verdun, BP 85, F-92453, Gentilly, France, 2006.

[21] S. Iqbal, M. Bari, and M. Rahman. Solving the multi-dimensional multi-choice knapsack problem with the help of ants. In *Proceedings of the 7th international conference on Swarm intelligence (ANTS 2010)*, 2010.

[22] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[23] 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, 1998.

[24] S. Khan, K. Li, E. Manning, and M. Akbar. Solving the knapsack problem for adaptive multimedia systems. *Studia Informatica, Special Issue on Combinatorial Problems*, 2(2):157–178, 2002.

[25] A. Lardeux, G. Knippel, and G. Geffard. Efficient algorithms for solving the 2-layered network design problem. In *Proceedings of the International Network Optimization Conference, Evry, France*, pages 367–372, 2003.

[26] V. Li, G. Curry, and E. Boyd. Towards the real time solution of strike force asset allocation problems. *Computers and Operations Research*, 31(2):273–291, 2004.

[27] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations.* J. Wiley and Sons, New-York, 1990.

[28] M. Moser, D. Jokanovic, and N. Shiratori. An algorithm for the multidimensional multiple-choice knapsack problem. *IEICE Transactions in Fundamentals*, E80-A(3), 1997.

[29] R. Parra-Hernandez and N. Dimopoulos. A new heuristic for solving the multi-choice multidimensional knapsack problem. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, 35(5):708–717, 2002.

[30] H. Pirkul. Efficient algorithms for the capacitated concentrator location problem. *Computers and Operations Research*, 14(3):197–208, 1987.

[31] D. Pisinger. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83(2):394–410, 1995.

[32] G. Raidl. The multiple container packing problem: A genetic algorithm approach with weighted codings. *ACM SIGAPP Applied Computing Review, ACM Press*, 7(2):22–31, 1999.

[33] M. Razzazi and T. Ghasemi. An exact algorithm for the multiple-choice multidimensional knapsack based on the core. *Advances in Computer Science and Engineering, Communications in Computer and Information Science*, 6:275–282, 2008.

[34] Z. Ren and Z. Feng. An ant colony optimization approach to the multiple-choice multidimensional knapsack problem. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation (GECCO 2010).* Association for Computing Machinery, 2010.

[35] J. Romaine. Solving the multidimensional multiple knapsack problem with packing constraints using tabu search. *Rapport de recherche AFIT/GOR/ENS/99M-15, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio*, 1999.

[36] A. Sbihi. A best first search exact algorithm for the multiple-choice multidimensional knapsack problem. *Journal of Combinatorial Optimization*, 13(4):337–351, 2007.

[37] P. Sinha and A. Zoltners. The multiple-choice knapsack problem. *Operations Research*, 27(3):503–515, 1979.

[38] Y. Toyoda. A simplified algorithm for obtaining approximate solutions to zero-one programming problems. *Management Science*, 21(12):1417–1427, 1975.

[39] Y. Vimont, S. Boussier, and M. Vasquez. Reduced costs propagation in an efficient implicit enumeration for the 0-1 multidimensional knapsack problem. *Journal of Combinatorial Optimization*, 15(2):165–178, 2008.

[40] C. Wilbaut and S. Hanafi. New convergent heuristics for 0-1 mixed integer programming. *European Journal of Operational Research*, 195:62–74, 2009.

A Tables of computational results for the random instances

The complete list of computational results obtained for the randomly generated instances is reported in the Tables 5-8. The column *Instance* identifies the corresponding instance. The characteristics of each instance is given in the columns 2 to 5. The best lower bound provided by each method is given in the columns 6 to 9, and the best bound is reported in column MAX. In the columns 11 to 14, we give for each instance the difference between the bound provided by each method and CPLEX and the best bound given in the column MAX. The entries $-$ in these tables mean that no feasible solution was found by the corresponding method within the time limit of 600 seconds.

Instance	m	n	n_i	n'	CPLEX	HMW	CHMW	MACH3	MAX	CPLEX	HMW	CHMW	MACH3
1	10	100	10	1000	25647	25653	25653	25656	25656	-9	-3	-3	0
2	10	100	10	1000	29563	29588	29598	29598	29598	-35	-10	0	0
3	10	100	10	1000	29154	29164	29170	29168	29170	-16	-6	0	-2
4	10	100	10	1000	30148	30155	30168	30172	30172	-24	-17	-4	0
5	10	100	15	1500	32692	32703	32715	32715	32715	-23	-12	0	0
6	10	100	15	1500	33228	33231	33224	33234	33234	-6	-3	-10	0
7	10	100	15	1500	32197	32216	32221	32232	32232	-35	-16	-11	0
8	10	100	15	1500	29189	29202	29203	29221	29221	-32	-19	-18	0
9	10	100	20	2000	31251	31243	31266	31274	31274	-23	-31	-8	0
10	10	100	20	2000	27270	27274	27274	27290	27290	-20	-16	-16	0
11	10	100	20	2000	29250	29256	29263	29272	29272	-22	-16	-9	0
12	10	100	20	2000	30262	30256	30259	30270	30270	-8	-14	-11	0
13	10	100	25	2500	30854	30865	30865	30867	30867	-13	-2	-2	0
14	10	100	25	2500	40848	40839	40863	40869	40869	-21	-30	-6	0
15	10	100	25	2500	25349	25355	25354	25366	25366	-17	-11	-12	0
16	10	100	25	2500	31368	31367	31376	31389	31389	-21	-22	-13	0
17	10	250	10	2500	71651	71645	71650	71664	71664	-13	-19	-14	0
18	10	250	10	2500	72908	72909	72905	72924	72924	-16	-15	-19	0
19	10	250	10	2500	69178	69181	69195	69194	69195	-17	-14	0	-1
20	10	250	10	2500	60383	60381	60377	60393	60393	-10	-12	-16	0
21	10	250	15	3750	78134	78126	78134	78145	78145	-11	-19	-11	0
22	10	250	15	3750	81875	81874	81874	81893	81893	-18	-19	-19	0
23	10	250	15	3750	93128	93123	93146	93147	93147	-19	-24	-1	0
24	10	250	15	3750	76849	76842	76855	76864	76864	-15	-22	-9	0
25	10	250	20	5000	68228	68226	68220	68229	68229	-1	-3	-9	0
26	10	250	20	5000	73187	73187	73189	73195	73195	-8	-8	-6	0
27	10	250	20	5000	74446	74452	74436	74457	74457	-11	-5	-21	0
28	10	250	20	5000	50753	50756	50754	50766	50766	-13	-10	-12	0
29	10	250	25	6250	77217	77202	77219	77231	77231	-14	-29	-12	0
30	10	250	25	6250	67248	67258	67262	67261	67262	-14	-4	0	-1
31	10	250	25	6250	92189	92175	92195	92210	92210	-21	-35	-15	0
32	10	250	25	6250	65990	65987	65979	66003	66003	-13	-16	-24	0
33	10	500	10	5000	150941	150949	150935	150960	150960	-19	-11	-25	0
34	10	500	10	5000	180760	180750	180775	180786	180786	-26	-36	-11	0
35	10	500	10	5000	125884	125888	125898	125903	125903	-19	-15	-5	0
36	10	500	10	5000	150884	150871	150871	150875	150884	0	-13	-13	-9
37	10	500	15	7500	176298	176297	176285	176319	176319	-21	-22	-34	0
38	10	500	15	7500	146265	146265	146260	146265	146265	0	0	-5	0
39	10	500	15	7500	138815	138797	138799	138812	138815	0	-18	-16	-3
40	10	500	15	7500	131294	131291	131297	131311	131311	-17	-20	-14	0
41	10	500	20	10000	136460	136468	136460	136467	136468	-8	0	-8	-1
42	10	500	20	10000	159003	158990	159013	159012	159013	-10	-23	0	-1
43	10	500	20	10000	139052	139060	139045	139053	139060	-8	0	-15	-7
44	10	500	20	10000	136493	136500	136491	136482	136500	-7	0	-9	-18
45	10	500	25	12500	91981	91975	91985	91988	91988	-7	-13	-3	0
46	10	500	25	12500	149533	149520	149530	149538	149538	-5	-18	-8	0
47	10	500	25	12500	157014	157016	157024	157049	157049	-35	-33	-25	0
48	10	500	25	12500	159474	159459	159473	159495	159495	-21	-36	-22	0
49	10	700	10	7000	207721	207711	207720	207741	207741	-20	-30	-21	0
50	10	700	10	7000	186795	186781	186799	186802	186802	-7	-21	-3	0
51	10	700	10	7000	225246	225242	225251	225261	225261	-15	-19	-10	0
52	10	700	10	7000	183299	183307	183296	183307	183307	-8	0	-11	0
53	10	700	15	10500	183777	183777	183772	183791	183791	-14	-14	-19	0
54	10	700	15	10500	208352	208367	208353	208360	208367	-15	0	-14	-7
55	10	700	15	10500	180365	180363	180368	180374	180374	-9	-11	-6	0
56	10	700	15	10500	236443	236429	236438	236432	236443	0	-14	-5	-11
57	10	700	20	14000	201626	201624	201620	201609	201626	0	-2	-6	-17
58	10	700	20	14000	254092	254094	254088	254076	254094	-2	0	-6	-18
59	10	700	20	14000	215601	215606	215591	215599	215606	-5	0	-15	-7
60	10	700	20	14000	201642	201642	201635	201628	201642	0	0	-7	-14
61	10	700	25	17500	233793	233784	233792	233812	233812	-19	-28	-20	0
62	10	700	25	17500	184861	184869	184884	184892	184892	-31	-23	-8	0
63	10	700	25	17500	226774	226813	226821	226828	226828	-54	-15	-7	0
64	10	700	25	17500	240735	240736	240749	240777	240777	-42	-41	-28	0

Table 5: Comparative results for hard instances generated randomly (Part I)

Instance	m	n	n_i	n'	**CPLEX**	**HMW**	**CHMW**	**MACH3**	**MAX**	**CPLEX**	**HMW**	**CHMW**	**MACH3**
65	15	100	10	1000	42394	42367	42383	42417	42417	-23	-50	-34	0
66	15	100	10	1000	52358	52343	52337	52376	52376	-18	-33	-39	0
67	15	100	10	1000	43916	43912	43933	43959	43959	-43	-47	-26	0
68	15	100	10	1000	45837	45843	45860	45866	45866	-29	-23	-6	0
69	15	100	15	1500	43530	43558	43551	43593	43593	-63	-35	-42	0
70	15	100	15	1500	47011	46990	47011	47020	47020	-9	-30	-9	0
71	15	100	15	1500	42994	43019	43023	43044	43044	-50	-25	-21	0
72	15	100	15	1500	52433	52461	52438	52495	52495	-62	-34	-57	0
73	15	100	20	2000	53552	53550	53569	53603	53603	-51	-53	-34	0
74	15	100	20	2000	41560	41596	41600	41607	41607	-47	-11	-7	0
75	15	100	20	2000	47653	47666	47706	47687	47706	-53	-40	0	-19
76	15	100	20	2000	44524	44558	44558	44571	44571	-47	-13	-13	0
77	15	100	25	2500	51643	51646	51646	51699	51699	-56	-53	-53	0
78	15	100	25	2500	29685	29728	29703	29738	29738	-53	-10	-35	0
79	15	100	25	2500	45124	45150	45135	45174	45174	-50	-24	-39	0
80	15	100	25	2500	42707	42687	42694	42707	42707	0	-20	-13	0
81	15	250	10	2500	104987	105000	104982	105014	105014	-27	-14	-32	0
82	15	250	10	2500	126147	126093	126146	126144	126147	0	-54	-1	-3
83	15	250	10	2500	107400	107384	107404	107435	107435	-35	-51	-31	0
84	15	250	10	2500	99998	99951	99959	100006	100006	-8	-55	-47	0
85	15	250	15	3750	106528	106528	106551	106573	106573	-45	-45	-22	0
86	15	250	15	3750	115310	115317	115302	115340	115340	-30	-23	-38	0
87	15	250	15	3750	120296	120280	120291	120329	120329	-33	-49	-38	0
88	15	250	15	3750	104119	104090	104080	104132	104132	-13	-42	-52	0
89	15	250	20	5000	149167	149106	149177	149229	149229	-62	-123	-52	0
90	15	250	20	5000	100566	100583	100574	100616	100616	-50	-33	-42	0
91	15	250	20	5000	85582	85590	85600	85622	85622	-40	-32	-22	0
92	15	250	20	5000	122974	122961	122958	122989	122989	-15	-28	-31	0
93	15	250	25	6250	113149	113147	113186	113177	113186	-37	-39	0	-9
94	15	250	25	6250	115665	115694	115661	115691	115694	-29	0	-33	-3
95	15	250	25	6250	120657	120689	120664	120718	120718	-61	-29	-54	0
96	15	250	25	6250	124332	124323	124366	124396	124396	-64	-73	-30	0
97	15	500	10	5000	197493	197514	197503	197529	197529	-36	-15	-26	0
98	15	500	10	5000	227613	227621	227599	227643	227643	-30	-22	-44	0
99	15	500	10	5000	230048	230023	230023	230083	230083	-35	-60	-60	0
100	15	500	10	5000	259906	259886	259918	259961	259961	-55	-75	-43	0
101	15	500	15	7500	188279	188258	188256	188276	188279	0	-21	-23	-3
102	15	500	15	7500	215783	215772	215817	215835	215835	-52	-63	-18	0
103	15	500	15	7500	193086	193062	193072	193099	193099	-13	-37	-27	0
104	15	500	15	7500	200828	200819	200833	200869	200869	-41	-50	-36	0
105	15	500	20	10000	183727	183715	183759	183767	183767	-40	-52	-8	0
106	15	500	20	10000	236172	236195	236181	236195	236195	-23	0	-14	0
107	15	500	20	10000	261043	260988	261085	261077	261085	-42	-97	0	-8
108	15	500	20	10000	246071	246061	246069	246137	246137	-66	-76	-68	0
109	15	500	25	12500	218952	218890	218970	218981	218981	-29	-91	-11	0
110	15	500	25	12500	186416	186412	186433	186448	186448	-32	-36	-15	0
111	15	500	25	12500	168988	168914	168958	168970	168988	0	-74	-30	-18
112	15	500	25	12500	206485	206472	206491	206505	206505	-20	-33	-14	0
113	15	700	10	7000	290726	290682	290713	290741	290741	-15	-59	-28	0
114	15	700	10	7000	336198	336161	336177	336215	336215	-17	-54	-38	0
115	15	700	10	7000	273110	273121	273141	273157	273157	-47	-36	-16	0
116	15	700	10	7000	311667	311648	311673	311714	311714	-47	-66	-41	0
117	15	700	15	10500	309071	309088	309140	309152	309152	-81	-64	-12	0
118	15	700	15	10500	333527	333523	333551	333602	333602	-75	-79	-51	0
119	15	700	15	10500	361570	361550	361640	361640	361640	-70	-90	0	0
120	15	700	15	10500	333536	333493	333540	333558	333558	-22	-65	-18	0
121	15	700	20	14000	271095	271108	271172	271187	271187	-92	-79	-15	0
122	15	700	20	14000	379698	379727	379746	379808	379808	-110	-81	-62	0
123	15	700	20	14000	337557	337538	337611	337628	337628	-71	-90	-17	0
124	15	700	20	14000	278176	278167	278193	278189	278193	-17	-26	0	-4
125	15	700	25	17500	316994	317058	317061	317111	317111	-117	-53	-50	0
126	15	700	25	17500	338051	338024	338074	338081	338081	-30	-57	-7	0
127	15	700	25	17500	303082	303068	303088	303127	303127	-45	-59	-39	0
128	15	700	25	17500	337928	337951	338012	338047	338047	-119	-96	-35	0

Table 6: Comparative results for hard instances generated randomly (Part II)

Instance	m	n	n_i	n'	CPLEX	HMW	CHMW	MACH3	MAX	CPLEX	HMW	CHMW	MACH3
129	20	100	10	1000	44209	44209	44201	44233	44233	-24	-24	-32	0
130	20	100	10	1000	64592	64666	64601	64648	64666	-74	0	-65	-18
131	20	100	10	1000	63218	63132	63158	63207	63218	0	-86	-60	-11
132	20	100	10	1000	63142	63176	63185	63250	63250	-108	-74	-65	0
133	20	100	15	1500	62006	62097	62127	62119	62127	-121	-30	0	-8
134	20	100	15	1500	49742	49736	49741	49826	49826	-84	-90	-85	0
135	20	100	15	1500	56202	56250	56158	56212	56250	-48	0	-92	-38
136	20	100	15	1500	70568	70562	70559	70593	70593	-25	-31	-34	0
137	20	100	20	2000	60240	60205	60211	60268	60268	-28	-63	-57	0
138	20	100	20	2000	52813	52831	52795	52808	52831	-18	0	-36	-23
139	20	100	20	2000	67212	67282	67245	67304	67304	-92	-22	-59	0
140	20	100	20	2000	68251	68199	68255	68243	68255	-4	-56	0	-12
141	20	100	25	2500	68345	68402	68370	68392	68402	-57	0	-32	-10
142	20	100	25	2500	60331	60376	60322	60370	60376	-45	0	-54	-6
143	20	100	25	2500	71194	71244	71300	71309	71309	-115	-65	-9	0
144	20	100	25	2500	63294	63304	63336	63411	63411	-117	-107	-75	0
145	20	250	10	2500	154959	154959	154999	155028	155028	-69	-69	-29	0
146	20	250	10	2500	161468	161552	161632	161638	161638	-170	-86	-6	0
147	20	250	10	2500	151335	151200	151276	151317	151335	0	-135	-59	-18
148	20	250	10	2500	156542	156470	156512	156623	156623	-81	-153	-111	0
149	20	250	15	3750	140878	140825	140874	140976	140976	-98	-151	-102	0
150	20	250	15	3750	157171	157185	157219	157333	157333	-162	-148	-114	0
151	20	250	15	3750	148228	148293	148264	148338	148338	-110	-45	-74	0
152	20	250	15	3750	144486	144527	144542	144626	144626	-140	-99	-84	0
153	20	250	20	5000	131234	131264	131227	131299	131299	-65	-35	-72	0
154	20	250	20	5000	157459	157463	157509	157550	157550	-91	-87	-41	0
155	20	250	20	5000	168691	168639	168692	168739	168739	-48	-100	-47	0
156	20	250	20	5000	150077	149948	150053	150058	150077	0	-129	-24	-19
157	20	250	25	6250	139006	138982	138982	138971	139006	0	-24	-24	-35
158	20	250	25	6250	167590	167549	167608	167658	167658	-68	-109	-50	0
159	20	250	25	6250	139006	138968	139048	139056	139056	-50	-88	-8	0
160	20	250	25	6250	153960	153912	153981	153968	153981	-21	-69	0	-13
161	20	500	10	5000	285685	285690	285734	285777	285777	-92	-87	-43	0
162	20	500	10	5000	293248	293262	293287	293367	293367	-119	-105	-80	0
163	20	500	10	5000	278100	278128	278128	278149	278149	-49	-21	-21	0
164	20	500	10	5000	313179	313124	313201	313246	313246	-67	-122	-45	0
165	20	500	15	7500	312022	312115	312146	312190	312190	-168	-75	-44	0
166	20	500	15	7500	329461	329392	329447	329521	329521	-60	-129	-74	0
167	20	500	15	7500	297105	297037	297135	297199	297199	-94	-162	-64	0
168	20	500	15	7500	304604	304581	304656	304761	304761	-157	-180	-105	0
169	20	500	20	10000	252941	252985	253047	253085	253085	-144	-100	-38	0
170	20	500	20	10000	330203	330086	330208	330239	330239	-36	-153	-31	0
171	20	500	20	10000	267677	267757	267757	267763	267763	-86	-6	-6	0
172	20	500	20	10000	365215	365344	365355	365318	365355	-140	-11	0	-37
173	20	500	25	12500	330540	330543	330623	330642	330642	-102	-99	-19	0
174	20	500	25	12500	343182	343188	343323	343336	343336	-154	-148	-13	0
175	20	500	25	12500	335616	335563	335707	335681	335707	-91	-144	0	-26
176	20	500	25	12500	253229	253252	253288	253321	253321	-92	-69	-33	0
177	20	700	10	7000	446033	445986	446047	446129	446129	-96	-143	-82	0
178	20	700	10	7000	417822	418014	417897	417897	418014	-192	0	-117	-117
179	20	700	10	7000	445813	445823	445870	445917	445917	-104	-94	-47	0
180	20	700	10	7000	414029	414119	414250	414254	414254	-225	-135	-4	0
181	20	700	15	10500	367135	367114	367256	367276	367276	-141	-162	-20	0
182	20	700	15	10500	416127	416243	416289	416321	416321	-194	-78	-32	0
183	20	700	15	10500	514034	514170	514223	514301	514301	-267	-131	-78	0
184	20	700	15	10500	409044	408979	409138	409136	409138	-94	-159	0	-2
185	20	700	20	14000	465956	466054	466108	466179	466179	-223	-125	-71	0
186	20	700	20	14000	427553	427643	427713	427704	427713	-160	-70	0	-9
187	20	700	20	14000	420306	420494	420636	420642	420642	-336	-148	-6	0
188	20	700	20	14000	319086	319254	319223	319224	319254	-168	0	-31	-30
189	20	700	25	17500	438213	438464	438497	438490	438497	-284	-33	0	-7
190	20	700	25	17500	372238	372283	372399	372384	372399	-161	-116	0	-15
191	20	700	25	17500	406975	407167	407198	407249	407249	-274	-82	-51	0
192	20	700	25	17500	319726	319760	319851	319844	319851	-125	-91	0	-7

Table 7: Comparative results for hard instances generated randomly (Part III)

Instance	m	n	n_i	n'	CPLEX	HMW	CHMW	MACH3	MAX	CPLEX	HMW	CHMW	MACH3
193	25	100	10	1000	–	90052	90021	89961	90052	–	0	-31	-91
194	25	100	10	1000	73883	73893	74001	73989	74001	-118	-108	0	-12
195	25	100	10	1000	62173	62297	62391	62424	62424	-251	-127	-33	0
196	25	100	10	1000	70459	70703	70550	70691	70703	-244	0	-153	-12
197	25	100	15	1500	–	73972	73996	74025	74025	–	-53	-29	0
198	25	100	15	1500	68931	68925	68960	68992	68992	-61	-67	-32	0
199	25	100	15	1500	62330	62597	62676	62715	62715	-385	-118	-39	0
200	25	100	15	1500	75160	75144	75209	75331	75331	-171	-187	-122	0
201	25	100	20	2000	74014	74231	74291	74289	74291	-277	-60	0	-2
202	25	100	20	2000	70032	71238	71223	71225	71238	-1206	0	-15	-13
203	25	100	20	2000	81446	81455	81637	81750	81750	-304	-295	-113	0
204	25	100	20	2000	68221	68263	68319	68303	68319	-98	-56	0	-16
205	25	100	25	2500	82547	82977	82914	83082	83082	-535	-105	-168	0
206	25	100	25	2500	75123	75459	75535	75636	75636	-513	-177	-101	0
207	25	100	25	2500	77701	77898	77932	77983	77983	-282	-85	-51	0
208	25	100	25	2500	74515	74407	74444	74484	74515	0	-108	-71	-31
209	25	250	10	2500	–	158620	158804	158881	158881	–	-261	-77	0
210	25	250	10	2500	201012	202910	203002	202918	203002	-1990	-92	0	-84
211	25	250	10	2500	177375	177801	177903	177888	177903	-528	-102	0	-15
212	25	250	10	2500	210756	212672	212884	212832	212884	-2128	-212	0	-52
213	25	250	15	3750	181069	181205	181232	181259	181259	-190	-54	-27	0
214	25	250	15	3750	203124	204595	204611	204610	204611	-1487	-16	0	-1
215	25	250	15	3750	170253	171471	171529	171712	171712	-1459	-241	-183	0
216	25	250	15	3750	202152	202384	202445	202531	202531	-379	-147	-86	0
217	25	250	20	5000	182681	182997	182962	183048	183048	-367	-51	-86	0
218	25	250	20	5000	208419	210360	210441	210409	210441	-2022	-81	0	-32
219	25	250	20	5000	197646	199100	199285	199357	199357	-1711	-257	-72	0
220	25	250	20	5000	191156	192861	192987	193009	193009	-1853	-148	-22	0
221	25	250	25	6250	186969	187076	187099	187078	187099	-130	-23	0	-21
222	25	250	25	6250	190958	190983	191001	190999	191001	-43	-18	0	-2
223	25	250	25	6250	211040	211710	211916	211926	211926	-886	-216	-10	0
224	25	250	25	6250	188814	189492	189659	189635	189659	-845	-167	0	-24
225	25	500	10	5000	474294	476417	476498	476255	476498	-2204	-81	0	-243
226	25	500	10	5000	379188	379094	379219	379285	379285	-97	-191	-66	0
227	25	500	10	5000	349181	349108	349484	349546	349546	-365	-438	-62	0
228	25	500	10	5000	330694	330901	330997	330935	330997	-303	-96	0	-62
229	25	500	15	7500	468047	467925	468124	468261	468261	-214	-336	-137	0
230	25	500	15	7500	340568	340782	340815	340873	340873	-305	-91	-58	0
231	25	500	15	7500	384522	–	385608	385753	385753	-1231	–	-145	0
232	25	500	15	7500	402351	–	403270	403225	403270	-919	–	0	-45
233	25	500	20	10000	369021	371964	371842	371903	371964	-2943	0	-122	-61
234	25	500	20	10000	408873	408988	409057	409041	409057	-184	-69	0	-16
235	25	500	20	10000	338972	339194	339229	339230	339230	-258	-36	-1	0
236	25	500	20	10000	362530	364122	364399	364372	364399	-1869	-277	0	-27
237	25	500	25	12500	337273	337429	337550	337550	337550	-277	-121	0	0
238	25	500	25	12500	379660	379699	379819	379809	379819	-159	-120	0	-10
239	25	500	25	12500	368785	369899	369820	369926	369926	-1141	-27	-106	0
240	25	500	25	12500	352006	–	352389	352374	352389	-383	–	0	-15
241	25	700	10	7000	481548	481675	–	481819	481819	-271	-144	–	0
242	25	700	10	7000	570815	–	572398	572538	572538	-1723	–	-140	0
243	25	700	10	7000	506108	506083	506305	506263	506305	-197	-222	0	-42
244	25	700	10	7000	515755	–	517018	517051	517051	-1296	–	-33	0
245	25	700	15	10500	512698	515844	515992	516011	516011	-3313	-167	-19	0
246	25	700	15	10500	487539	487444	487816	487789	487816	-277	-372	0	-27
247	25	700	15	10500	444301	445601	445885	445889	445889	-1588	-288	-4	0
248	25	700	15	10500	598648	–	599698	599705	599705	-1057	–	-7	0
249	25	700	20	14000	533768	534695	534924	534879	534924	-1156	-229	0	-45
250	25	700	20	14000	560251	562482	–	562621	562621	-2370	-139	–	0
251	25	700	20	14000	524107	524149	–	524373	524373	-266	-224	–	0
252	25	700	20	14000	540631	–	–	542123	542123	-1492	–	–	0
253	25	700	25	17500	512971	–	514582	514590	514590	-1619	–	-8	0
254	25	700	25	17500	509267	–	511121	511112	511121	-1854	–	0	-9
255	25	700	25	17500	621424	–	–	622626	622626	-1202	–	–	0
256	25	700	25	17500	471164	–	472559	472504	472559	-1395	–	0	-55

Table 8: Comparative results for hard instances generated randomly (Part IV)