A "reduce and solve" approach for the multiple-choice multidimensional knapsack problem

Yuning Chen and Jin-Kao Hao *

LERIA, Université d'Angers
2 Boulevard Lavoisier, 49045 Angers Cedex 01, France
European Journal of Operational Research 239(2): 313-322, 2014

Abstract

The multiple-choice multidimensional knapsack problem (MMKP) is a well-known NP-hard combinatorial optimization problem with a number of important applications. In this paper, we present a "reduce and solve" heuristic approach which combines problem reduction techniques with an Integer Linear Programming (ILP) solver (CPLEX). The key ingredient of the proposed approach is a set of group fixing and variable fixing rules. These fixing rules rely mainly on information from the linear relaxation of the given problem and aim to generate reduced critical subproblem to be solved by the ILP solver. Additional strategies are used to explore the space of the reduced problems. Extensive experimental studies over two sets of 37 MMKP benchmark instances in the literature show that our approach competes favorably with the most recent state-of-the-art algorithms. In particular, for the set of 27 conventional benchmarks, the proposed approach finds an improved best lower bound for 11 instances and as a by-product improves all the previous best upper bounds. For the 10 additional instances with irregular structures, the method improves 7 best known results.

Keywords: Knapsack; Fixing heuristics; Linear relaxation; Hybridization.

1 Introduction

The multiple-choice multidimensional knapsack problem (MMKP) can be informally described as follows. We are given a set of items that are divided into

* Corresponding author.
Email addresses: yuning@info.univ-angers.fr (Yuning Chen), hao@info.univ-angers.fr (Jin-Kao Hao).

several groups and different types of limited resources. Each item requires a certain amount of each resource and generates a profit. The purpose of the MMKP is to select *exactly* one item from each group such that the total profit of the selected items is maximized while the consumption of each resource does not exceed the given limit (knapsack constraints).

Formally, given $G = \{G_1, G_2, ..., G_n\}$ the set of n disjoint groups (i.e., $G_i \cap G_j = \emptyset$ for each i, j, $1 \leq i \neq j \leq n$). Let $I = \{1, 2, ..., n\}$ be the group index set, $g_i = |G_i|$ the number of items of group $G_i \in G$, m the number of resource types, b^k the capacity of resource k ($1 \leq k \leq m$), $p_{ij} \geq 0$ the profit of the j^{th} item of G_i, w_{ij}^k the consumption for resource k of the j^{th} item of G_i. Additionally, let x_{ij} be the decision variable such that $x_{ij} = 1$ if the j^{th} item of group G_i is selected; $x_{ij} = 0$ otherwise. Then the MMKP can be stated as follows:

$$max \sum_{i \in I} \sum_{j \in \{1,...,g_i\}} p_{ij} x_{ij} \tag{1}$$

subject to:

$$\sum_{i \in I} \sum_{j \in \{1...g_i\}} w_{ij}^k x_{ij} \leq b^k, k \in \{1, ..., m\} \tag{2}$$

$$\sum_{j \in \{1...g_i\}} x_{ij} = 1, i \in I \tag{3}$$

$$x_{ij} \in \{0, 1\}, i \in I, j \in \{1, ..., g_i\} \tag{4}$$

The MMKP is tightly related to the conventional multidimensional knapsack problem (MKP) [2, 16, 19, 23, 24, 29] since the MMKP can be reduced to the MKP by restricting each group to a single item and dropping constraint (3). Like the MKP, the MMKP is known to be NP-hard. In addition to its theoretical importance, the MMKP is notable for its capacity of modeling a number of practical applications such as logistics [1], resource allocation [5], capital budgeting [18], and telecommunications [28].

Compared with the conventional MKP, the MMKP is somewhat less studied until recently. Yet, given both its theoretical and practical relevance, the MMKP is receiving increasing attention in recent years and a number of effective solution approaches have been proposed in the literature. For instance, exact methods based on the branch and bound framework were reported in [6,15,21]. These algorithms have the advantage of guaranteeing the optimality of the solution found. Unfortunately, due to the very high computational complexity of the MMKP, exact approaches apply only to instances of limited sizes (i.e., n=100 and m=10 for the instances we used). To handle larger

instances, several heuristic approaches were developed to seek sub-optimal solutions (corresponding to lower bounds) in an acceptable computing time.

For instance, in [10], the authors introduced a reactive local search algorithm and showed much better results than those reported in [17], which was the first paper dealing directly with the MMKP. Later, the authors of [4] proposed an efficient hybrid heuristic combining local branching with column generation techniques, which improved the lower bounds for many benchmark instances. In [9] an iterative relaxation-based heuristic was applied to solve the MMKP, where a series of small sub-problems are generated by exploiting information obtained from a series of relaxations. In [3], the authors employed a similar but more general approach called semi-continuous relaxation heuristic approach where variables are forced to take values close to 0 or 1. More recently, again based on the iterative relaxation-based heuristic framework, the authors of [20] explored a new strategy, consisting of a family of new cuts and a reformulation procedure used at each iteration to improve the performance of the heuristic and to define the reduced problem. This method reported most of the current best known results over the set of conventional MMKP benchmark instances, which will be used as one of our references for performance assessment and comparison. In [22], the authors proposed an original parameterized compositional pareto-algebraic heuristic (CPH) which explores incremental problem solving and parallelization. They reported interesting results on the well-known MMKP benchmark instances and introduced a set of new instances that we will use in our work. Finally, there are several other recent and interesting studies based on general approaches like ant colony optimization combined with local search [14], strategic oscillation exploring surrogate constraint information [13], Lagrangian neighborhood search [11] and tabu search [12].

In this paper, we present a "reduce and solve" heuristic approach that jointly makes use of problem reduction techniques and the state-of-the-art CPLEX ILP solver. The basic idea of the proposed approach is to employ some dedicated heuristics to fix a number of groups and variables in order to obtain a reduced critical subproblem which is then solved by the ILP solver. The key issue is how to choose the groups and variables to fix. For this purpose, we first define general fixing rules based on information from linear relaxation. To better explore the space of the reduced problems and achieve improved results (lower bounds), we additionally introduce specific strategies to enlarge progressively the reduced subproblems which are to be solved by CPLEX. Notice that our group and variable fixing techniques are in connection with the notion of strongly determined and consistent variables [7,8]. Similar strategies for temporary or definitive variable fixing are explored in other contexts like, for instance, 0-1 mixed integer programming and binary quadratic programming [26, 27, 29].

To assess the merit and limit of the proposed approach, we carry out extensive computational experiments based on two sets of benchmark instances from the literature. These experiments show that the proposed approach competes favorably with the state-of-the-art methods and is able to discover 11 improved lower bounds and in passing to improve all the current best upper bounds reported in the literature for the set of 27 conventional benchmark instances. Moreover the proposed approach improves 7 best known results for the 10 additional benchmarks with irregular structures.

The paper is organized as follows. In Section 2, we present in detail the proposed approach. We begin with the introduction of the group and variable fixing rules and then introduce two solution procedures for the exploration of different reduced problems. Section 3 is dedicated to an extensive computational assessment in comparison with the state-of-the-art approaches. We also show an analysis of the effect of the group and variable fixing techniques in Sections 4. Conclusions are given in the last section.

2 A "reduce and solve" approach for the MMKP

2.1 General approach

The "reduce and solve" approach proposed in this paper can be summarized as a three-step method.

(1) Group fixing: This step aims to identify some variables which are highly likely to be part of the optimal solution and fixes them to the value of 1. Given the constraint (3), once a group has a variable assigned the value of 1, the remaining variables of the group must be assigned the value of 0. We remove then the group (the group is said fixed) from the initial problem P, leading to a first reduced problem P'. Let q be the number of fixed groups.
(2) Variable fixing: For each of the $n - q$ remaining groups of the problem P', we identify some variables that are unlikely to be part of the optimal solution, fix these variables to 0 and remove them from problem P', leading to a further reduced problem P".
(3) ILP solving: We run CLPEX to solve P".

Given this general procedure, it is clear that the success of this approach depends on the methods used for group fixing (step 1) and variable fixing (step 2). We will explain in Sections 2.3 and 2.4 the heuristic fixing rules based on linear relaxation of the problem. However, whatever the method we use, it is possible that some variables are fixed to a wrong value. To mitigate

this risk, we introduce additional strategies to decrease gradually the number of fixed variables. By doing so, we explore different and increasingly larger reduced problems which provides a means to achieve improved solutions. These strategies are presented in Sections 2.5.2 and 2.5.3.

2.2 Basic definitions and notations

The following notations and definitions will be used in the presentation of the proposed approach.

- Let $G = \{G_1, G_2, ..., G_n\}$ be the given MMKP problem P with its index set $I = \{1, 2, ..., n\}$ and let x^* be an optimal solution of problem P.
- $LP(P)$ and $\bar{x}$ denote respectively the linear relaxation of P and an optimal solution of $LP(P)$.
- $\underline{v}(P)$ and $\bar{v}(P)$ denote respectively a lower bound and an upper bound of P.
- $(P|c)$, $LP(P|c)$ and $\bar{x}^c$ denotes respectively the problem P with exactly one additional constraint c, the linear relaxation of $(P|c)$ and an optimal solution of $LP(P|c)$.
- Integer group: Given the LP-relaxation optimal solution $\bar{x}$ of $LP(P)$, a group G_i of G is called *integer group* in $\bar{x}$ if $\exists j_i \in \{1, 2, ..., g_i\} : \bar{x}_{ij_i} = 1$.
- Fractional group: Given the LP-relaxation optimal solution $\bar{x}$ of $LP(P)$, a group G_i of G is called *fractional group* if $\exists J \subset \{1, 2, ..., g_i\}, |J| > 1 : \sum_{j \in J} \bar{x}_{ij} = 1, \forall j \in J : 0 < \bar{x}_{ij} < 1$. Let n_f be the number of fractional groups in $\bar{x}$.
- For $x \in [0, 1]^n$, we define $I^1(x) = \{i \in I : G_i$ is an integer group, $x_{ij_i} = 1\}$ as the index set of integer groups.
- For $x \in [0, 2]^n$, we define $I^2(x) = \{i \in I, \exists j_i \in \{1, 2, ..., g_i\} : x_{ij_i} = 2\}$, $I^{2*}(x) = I \setminus I^2(x)$, $F(x) = \{(i, j) : i \in I^{2*}(x), 0 < x_{ij} < 2\}$.
- r_{ij} is the reduced cost of x_{ij} from the LP-relaxation of P.

2.3 Group fixing

For an ILP problem, it is interesting to consider the relationship between its (integer) optimum x^* and its LP-relaxation optimum $\bar{x}$. When we studied several typical small MMKP instances, we observed that the binary optimum and the LP-relaxation optimum share some selected items (i.e., the corresponding variables receive the value of 1 in both the binary optimum and the LP optimum). It is thus expected that some groups could be fixed according to the LP-relaxation optimum $\bar{x}$.

In order to reduce the risk of fixing wrong groups, we try to identify the

set of "strongly determined" groups. For this purpose, we use a double LP-relaxation strategy. We first solve the LP-relaxation of the given problem P to identify a set of integer groups. Then we add an effective cut c to the original problem P and solve again the relaxed linear program $LP(P|c)$. The purpose of this second LP-relaxation with the cut c is to identify some unpromising (or unstable) integer groups in the first LP optimum $\bar{x}$ such that their variable assignments change in the new optimum $\bar{x}^c$. Those integer groups in $\bar{x}$ that "survive" in $\bar{x}^c$ are considered to be stable and are likely to be part of the true optimum x^*. We then fix these groups by assigning their variables to their LP optimal values. Once these groups are fixed, they are removed from the problem and will not be further considered.

Formally, our group fixing rule for the given problem P is summarized as follows.

(1) Solve its LP-relaxation $LP(P)$ and let $\bar{x}$ be its optimal solution. Let $I^1(\bar{x})$ be the index set of the integer groups in $\bar{x}$.
(2) Define the following cut c_k with respect to $\bar{x}$ (where k is a parameter) and let $(P|c_k)$ be the new problem with the added cut c_k.

$$\sum_{i \in I^1(\bar{x})} [(1 - x_{ij_i}) + \sum_{j \in \{1,\ldots,g_i\}, j \neq j_i} x_{ij}] \geq 2 * k \tag{5}$$

Solve the LP-relaxation $LP(P|c_k)$ and let $\bar{x}^{c_k}$ be the optimal solution.
(3) Define $\tilde{x} = \bar{x} + \bar{x}^{c_k}$, i.e., the value in each position of $\tilde{x}$ is obtained by adding the values of the corresponding position of vector $\bar{x}$ and $\bar{x}^{c_k}$. According to the definition of $I^2(x)$, it holds: $\forall i \in I^2(\tilde{x}) : \bar{x}_{ij_i} = 1 \wedge \bar{x}^{c_k}_{ij_i} = 1$.
(4) For each group G_i such that $i \in I^2(\tilde{x})$, fix the variables of G_i to their $\bar{x}$ values. As such we fix the groups of $I^2(\tilde{x})$ and put aside all their variables.

One notices that given the cut defined in Formula (5), the integer groups in $\bar{x}^{c_k}$ is a subset of the integer groups in $\bar{x}$. Intuitively, cut c_k forces at least k integer groups in the first LP optimum $\bar{x}$ to change their status in the second LP optimum $\bar{x}^{c_k}$. Such a changed group either becomes a fractional group or remains an integer group where the value of 1 is however assigned to a different variable. By varying k, we can control the number of groups to be fixed and as a consequence the size of the reduced subproblems. A large (resp. small) k fixes a small (resp. large) number of groups. With $k = n - n_f$ (n_f being the number of fractional groups in $\bar{x}$), all the integer groups in $\bar{x}$ change their status in $\bar{x}^{c_k}$ and consequently, no group will be fixed. Thus, $n - n_f$ can be considered as the maximum value for k. In Section 2.5, we will discuss how one can (partially) enumerate k in a potential interval, or set it to a promising value.

2.4 Variable fixing in unfixed groups

After the group fixing step, though $|I^2(\tilde{x})|$ groups of variables are fixed and removed from the problem, we still have $|I^{2*}(\tilde{x})|$ groups of variables whose values are not decided. The associated subproblem might be still too large to be efficiently solved by the ILP solver. To further reduce the search space, we need to further fix some variables in the $|I^{2*}(\tilde{x})|$ unfixed groups. Notice that contrary to group fixing where exactly one specific variable of each fixed groups is assigned the value of 1, the variable fixing step tries to identify, for each unfixed group, some variables that are to be assigned the value of 0 (i.e., the corresponding items will not be part of the optimal solution x^*). For this purpose, we will make use of another type of information from LP-relaxation, i.e., the reduced costs.

According to the LP theory, variables with small absolute reduced costs have more potential to be changed without affecting too much the objective value. It is then interesting to fix the variables with a large absolute reduced cost to their LP optimal values and leave the variables with a small absolute reduced cost to form the reduced problem to be solved by the ILP solver. In order to determine the number of the variables to fix (and thus the size of the reduced problem), we use an overall "threshold" of the absolute reduced costs.

Precisely, the "threshold" is given by the highest absolute reduced cost among all the variables that receive different values in the previous double LP-relaxation optima:

$RC_{max}(\tilde{x}) = max_{(i,j)\in F(\tilde{x})}\{|r_{ij}|\}$

where r_{ij} is the reduced cost of variable x_{ij} in the first linear relaxation and $F(\tilde{x})$ is the index set of variables whose values are greater than 0 and less than 2 in $\tilde{x}$.

Obviously, $RC_{max}(\tilde{x})$ is related to the results of the two linear programs $LP(P)$ and $LP(P|c_k)$, and more precisely it is related to the parameter k (see cut (5), Section 2.3). Generally, a small k gives a smaller threshold $RC_{max}(\tilde{x})$ which allows to fix more variables, leading thus to a smaller reduced problem. The reverse is also true.

Given $RC_{max}(\tilde{x})$, our variable fixing step can be summarized as follows. In each group, variables with an absolute reduced cost greater than $RC_{max}(\tilde{x})$ are set to the value of the LP-relaxation optimal solution. The remaining variables form the reduced problem.

Formally, let $S = \{(i,j) : i \in I, j \in \{1,...,g_i\}\}$ be the index set of all variables in the original problem P, $S_{free}(\tilde{x}) = \{(i,j) : i \in I^{2*}(\tilde{x}), j \in \{1,...,g_i\} : |r_{ij}| \leq$

$RC_{max}(\tilde{x})\}$ be the index set of the unfixed variables in the unfixed groups, the reduced problem associated to $\bar{x}$, $\tilde{x}$, S and $S_{free}(\tilde{x})$ is defined as follows:

$$P(\bar{x}, \tilde{x}, S, S_{free}(\tilde{x})) = \{P|(i,j) \in (S \backslash S_{free}(\tilde{x})) : x_{ij} = \bar{x}_{ij}\}$$

When k is small enough, the reduced problem can be solved to optimality within a very short time by means of the ILP solver. In practice, depending on the size of problem instances, the reduced problems are either solved to optimality or stopped due to a given CPU time limit.

2.5 Exploring the space of reduced problems: the PEGF and PERC algorithms

As group fixing and variable fixing are two relatively separated procedures, we propose two different ways of combining these two procedures to devise the MMKP algorithms. The first algorithm (named PEGF) partially enumerates the parameter k for group fixing and for each k uses $RC_{max}(\tilde{x})$ for variable fixing and constructing the reduced problem. For the second algorithm (named PERC), we set k to a promising value (identified in an empirical manner) and partially enumerate the absolute reduced cost threshold by increasing $RC_{max}(\tilde{x})$ progressively. Before presenting our PEGF and PERC algorithms in Sections 2.5.2 and 2.5.3, we first recall some known results which are used in our algorithms.

2.5.1 Known results

The Mixed Integer Programming (MIP) relaxation has been used in [29] to strengthen the bound of the LP relaxation, which was previously explored by the authors in [3] to design their algorithms for solving the MMKP. To introduce this technique, we consider a more general case of the MMKP. Let p and b be two vectors of coefficients, W be a matrix of coefficients, N be the set of binary variables. The MIP relaxation of problem P which is related to a subset J of the set N of items is expressed as follows:

$$MIP(P,J) \begin{cases} max\ px \\ s.t. \\ Wx \leq b \\ x_j \in \{0,1\}\ j \in J \\ x_j \in [0,1] \quad j \in N \setminus J \end{cases} \tag{6}$$

The following proposition shows the MIP relaxation of a problem provides stronger bounds than the linear relaxation (the proof can be found in [29]). This proposition is used in both PEGF and PERC to obtain an upper bound for the problem P.

Proposition 1. Let $v(P)$ be the optimal value of P, J and $J^{'}$ be two subsets of N with $J^{'} \subseteq J \subseteq N$, yields:

$$v(MIP(P,N)) = v(P) \leq v(MIP(P,J)) \leq v(MIP(P,J^{'})) \leq v(MIP(P,\emptyset)) = v(LP(P)) \quad (7)$$

Another result is the so-called "reduced cost constraint" which was introduced for the MKP in [25]. We observe that it also holds for the MMKP since the MMKP is different from the MKP only by some additional choice constraints. This reduced cost constraint was used to solve the MMKP in [3] as a variable fixing technique, and was also used in [20] as a cut which is progressively added to the original problem to further strengthen the model and to improve the quality of the subsequent upper bounds. Given the $\underline{v}(P)$ value of a feasible solution for problem P and the optimal solution value $\bar{v}(P)$ of its continuous relaxation, each current solution better than $\underline{v}(P)$ needs to satisfy the following relation:

$$\bar{v}(P) + \sum_{j \in N^-} r_j x_j - \sum_{j \in N^+} r_j (1 - x_j) + \sum_{i \in M} u_i s_i \geq \underline{v}(P) \quad (8)$$

where s_i are the slack variables, and r_j and u_j are the reduced costs associated to the nonbasic variables. N^- (resp. N^+) represents the set of indexes associated to nonbasic variables with a value equal to their lower bound(resp. upper bound). Reduced costs are positive for items in N^+ and negative in N^-. M is the index set of slack variables. Since $u_i \leq 0$ and $s_i \geq 0$, we can remove $\sum_{i \in M} u_i s_i$ and rewrite the "reduced cost constraint" as follows:

$$\sum_{j \in N^+} r_j (1 - x_j) + \sum_{j \in N^-} |r_j| x_j \leq \bar{v}(P) - \underline{v}(P) \quad (9)$$

This constraint implies that only the nonbasic variables with an absolute reduced cost smaller than or equal to $\bar{v}(P) - \underline{v}(P)$ can change their values with respect to the LP-relaxation optimal value if we are looking for a higher lower bound. Given the fixing rules of our proposed approach, this constraint ensures that including those variables in the unfixed groups whose absolute reduced costs are higher than the difference of an upper bound and a lower bound to the reduced problem gives no benefit of improving the lower bound. Our PERC algorithm thus uses the reduced cost constraint as one of its stopping criteria.

2.5.2 The PEGF algorithm

Algorithm 1 shows the basic steps of the PEGF algorithm. In the preprocessing step, we solve the LP-relaxation of the original problem P, generating an optimal solution $\bar{x}$ and the reduced costs of all the variables. At each iteration, the algorithm obtains $\bar{x}^{c_k}$ by solving the LP-relaxation of the current problem noted $(P|c_k)$. $\tilde{x}$ can then be computed by summing up $\bar{x}$ and $\bar{x}^{c_k}$.

Algorithm 1 Pseudo-code of the PEGF algorithm for the MMKP

1: **Input**:
   P: an instance of the $MMKP$;
   k_{start}: first value of k;
   T_{max}: limit on total computing time;
   t_{max}: time limit for solving the reduced problem;
   Δ: step length to increase k;
2: **Output**: the best lower bound $\underline{v}(P)$ found so far;
3: Solve $LP(P)$; keep an optimal solution $\bar{x}$ and the reduced cost r_{ij}, $\forall(i,j) \in S$;
4: Let TI be the temporary index set of fixed groups, t_{cur} be the current elapsed CPU time;
5: $\underline{v}(P) = -\infty$, $\bar{v}(P) = p^T\bar{x}$, $n_f = n - |I^1(\bar{x})|$, $k = k_{start}$, $TI = \emptyset$;
6: **while** $(\lfloor \bar{v}(P) - \underline{v}(P) \rfloor \geq 1) \wedge (k \leq n - n_f) \wedge (t_{cur} < T_{max})$ **do**
7: Solve $LP(P|c_k)$; keep an optimal solution $\bar{x}^{c_k}$;
8: $\tilde{x} = \bar{x} + \bar{x}^{c_k}$;
9: **if** $I^2(\tilde{x}) != TI$ **then**
10: $TI = I^2(\tilde{x})$;
11: **else**
12: skip to 23;
13: **end if**
14: Solve $MIP(P, S_{free}(\tilde{x}))$; keep an optimal solution $\bar{x}_{mip}$;
15: **if** $p^T\bar{x}_{mip} < \bar{v}(P)$ **then**
16: $\bar{v}(P) = p^T\bar{x}_{mip}$;
17: **end if**
18: Compute $RC_{max}(\tilde{x})$ and construct reduced problem $P(\bar{x}, \tilde{x}, S, S_{free}(\tilde{x}))$;
19: Run CPLEX with $min\{t_{max}, T_{max} - t_{cur}\}$ to solve the reduced problem $P(\bar{x}, \tilde{x}, S, S_{free}(\tilde{x}))$; keep a best solution x^0;
20: **if** $p^T x^0 > \underline{v}(P)$ **then**
21: $\underline{v}(P) = p^T x^0$;
22: **end if**
23: $k \leftarrow k + \Delta$;
24: **end while**

To avoid redundant computations, at each iteration, we check whether the current set of fixed groups is the same as that of the last iteration; and if this is the case, we increase k by a step length Δ and move to the next iteration. $\bar{v}(P)$ is updated each time a better upper bound is obtained by solving $MIP(P, S_{free}(\tilde{x}))$. By reference to $\tilde{x}$, the absolute reduced cost threshold $RC_{max}(\tilde{x})$ can be decided by comparing the values of all the "shakable" variables. After that, the reduced problem $P(\bar{x}, \tilde{x}, S, S_{free}(\tilde{x}))$ is constructed and

solved with time limit $min\{t_{max}, T_{max} - t_{cur}\}$ by the ILP solver. The so far best lower bound $\underline{v}(P)$ is updated when a new best solution is encountered.

The process stops when one of the following three stopping criteria is met: the upper bound equals the lower bound, or k exceeds the maximum value $n - n_f$, or the current execution time reaches the total time limit.

Notice that the first value of k, namely k_{start}, could be set to 0 from a general algorithmic viewpoint. However in practice, when an overall time limit is imposed, k_{start} is not suggested to be too small since too many groups would be fixed wrongly and it is a waste of computational resources to solve the related reduced problems. According to our experimental experience, we set $k_{start} = 13$ for the instances we used, which can be regarded as a good start point for our PEGF algorithm.

2.5.3 The PERC algorithm

Another way of exploring different reduced problems is to fix k to a specific value k_0 and then vary the absolute reduced cost threshold. Our PERC algorithm is based on this idea (see Algorithm 2).

To specify k_0, we devise the following empirical formula which is based on some characteristics of the given instance under consideration.

$$k_0 = min\{n - n_f, k_{start} + \lceil lg 1.2^n + 0.5m \rceil\} \tag{10}$$

To enumerate different thresholds for the absolute reduced costs, we rely again on information from LP-relaxation. Precisely, we solve first $LP(P)$ and $LP(P|c_{k_0})$ in the preprocessing phase to get the $RC_{max}(\tilde{x})$ value (see Section 2.4). This value is then used as the first absolute reduced cost threshold rd. At each iteration of the PERC algorithm, the upper bound and the lower bound are updated respectively by solving the $MIP(P, S_{free}(\tilde{x}))$ and the reduced problem $P(\bar{x}, \tilde{x}, S, S_{free}(\tilde{x}))$ (see Section 2.5.1). rd is then increased by a small value δ to expand the reduced problem to be solved in the next iteration of the algorithm.

The algorithm stops either when rd becomes larger than or equal to the gap between the upper bound and the lower bound, or the execution time reaches the total time limit. The rationale of the first stopping criterion is that according to the reduced cost constraint (see Section 2.5.1), including nonbasic variables whose absolute reduced costs are higher than the gap to the reduced problem cannot further improve the lower bound. Note that variables with an absolute reduced cost larger than $RC_{max}(\tilde{x})$ are all nonbasic variables, because basic variables having fractional values in the LP-relaxation optimal solution

are those variables whose absolute reduced costs are smaller than $RC_{max}(\tilde{x})$ and are already included in the reduced problem.

Algorithm 2 Pseudo-code of the PERC algorithm for the MMKP

Input:
  P: an instance of the $MMKP$;
  k_0: chosen value for k;
  T_{max}: limit on total computing time;
  t_{max}: time limit for solving the reduced problem;
  δ: step length to increase reduced cost threshold rd;
Output: the best lower bound $\underline{v}(P)$ found so far;
Solve $LP(P)$; keep an optimal solution $\bar{x}$ and the reduced cost r_{ij}, $\forall (i,j) \in S$;
Solve $LP(P|c_{k_0})$, keep an optimal solution $\bar{x}^{c_{k_0}}$;
$\tilde{x} = \bar{x} + \bar{x}^{c_{k_0}}$;
Compute $RC_{max}(\tilde{x})$;
$rd = RC_{max}(\tilde{x})$,$\underline{v}(P) = -\infty$, $\bar{v}(P) = p^T\bar{x}$; let t_{cur} be the current elapsed CPU time;
while $(rd < \bar{v}(P) - \underline{v}(P)) \wedge (t_{cur} < T_{max})$ **do**
  Solve $MIP(P, S_{free}(\tilde{x}))$; keep an optimal solution $\bar{x}_{mip}$;
  if $p^T\bar{x}_{mip} < \bar{v}(P)$ **then**
    $\bar{v}(P) = p\bar{x}_{mip}$;
  end if
  Construct reduced problem $P(\bar{x}, \tilde{x}, S, S_{free}(\tilde{x}))$;
  Run CPLEX with $min\{t_{max}, T_{max} - t_{cur}\}$ to solve the reduced problem $P(\bar{x}, \tilde{x}, S, S_{free}(\tilde{x}))$; keep a best solution x^0;
  if $p^T x^0 > \underline{v}(P)$ **then**
    $\underline{v}(P) = p^T x^0$;
  end if
  $rd \leftarrow rd + \delta$;
end while

3 Computational Experiments

3.1 Benchmark instances

To evaluate the efficiency of the proposed heuristics, we carry out extensive experiments on two sets of 37 benchmark instances. The first set of 27 instances [1] (7 instances named I07-I13 and 20 instances named INST01-INST20) is very popular in the literature and frequently used to test recent MMKP algorithms like [3, 4, 10, 20, 22]. The results on the instances of this set are reported in Sections 3.3 and 3.4. The second set (10 instances named INST21-INST30) is

[1] available at: http://www.laria.u-picardie.fr/hifi/OR-Benchmark/

introduced very recently in [22]. The results on the second set of instances are provided in Sections 3.5.

- Instances I07-I13 belong to the instance set generated and used in [15] (We do not consider the first 6 instances (I01-I06) because they can be easily solved by state-of-art MMKP exact algorithms and CPLEX). INST01-INST20 which are introduced in [10] according to the procedure proposed in [15], are large sized and more difficult. Thus in total, we consider 27 instances for our first experiments. These instances have a number of groups from 50 to 400, a number of items from 10 to 30 per group and 10 constraints. Except for the two smallest ones (INST01 and INST02), optimal solutions are still unknown for the remaining 25 instances. The best known results of the literature listed in different tables of this section are reported in [20] (algorithm MACH), [22] (algorithm CPH) and [4] (algorithm CH). Notice that in [20], the authors present results of three algorithm variants named MACH1, MACH2 and MACH3. And for MACH3, results corresponding to three different parameter settings are also provided. Here for each instance, we select the best results from all these MACH variants as the reference results. Also in [22], results obtained by the serial version and parallel version(executed on 10 processors) of the proposed algorithm were provided and we select those results obtained by the serial version as reference for our comparative study.
- Instances INST21-INST30 are new benchmarks introduced in [22]. Compared to the previous instances, these new instances have irregular structures such that the number of items per group varies and an item does not necessarily has resource consumption in all the resource dimensions. These instances have a number of groups from 100 to 500, a maximum number of items from 10 to 20 per group and a number of constraints from 10 to 40. The optimal solutions are unknown and the best known results are reported in [22] with the Compositional Pareto-algebraic Heuristic (CPH). We use these results as our reference for our experimental assessment.

3.2 Experiment settings

Our PEGF and PERC algorithms are coded in C++ [2] and compiled using GNU G++ on a PC running Ubuntu Linux V12.04 with an Intel Xeon E5440 processor (2.83GHz and 2G RAM). We use CPLEX 12.4 to solve both the LP-relaxation of the problem and the reduced problems.

For the set of 27 conventional benchmark instances (I07-I13, INST01-INST20), we report our results respectively with a total time limit of 500 seconds and 1

[2] The source code of our algorithms and the best solution certificates are available at http://www.info.univ-angers.fr/pub/hao/mmkp.html.

hour. These stopping conditions were previously used by the state-of-the-art algorithms to report their computational results [3,4,9,20]. With the time limit of 500 seconds which is considered to be relatively short in the literature, the time limit for a reduced problem (t_{max}) is set to be 75 seconds for PEGF and 500 seconds for PERC. With the time limit of 1 hour, the CPU time allowed to solve a reduced problem is set to 400 seconds for PEGF and 900 seconds for PERC.

For the set of 10 new benchmark instances (INST21-INST30), we adopt the time limit used in the reference paper [22], i.e., a total of 1200 seconds and 1 hour respectively. For the case of 1200 seconds, the time limit for a reduced problem (t_{max}) is set to be 200 seconds for PEGF and 600 seconds for PERC. With the time limit up to 1 hour, the CPU time allowed to solve a reduced problem is set to 400 seconds for PEGF and 900 seconds for PERC.

In all the cases, the step length to increase k (Δ) in PEGF is set to 3, and the step length to increase the reduced cost threshold (δ) in PERC is set to 1.

We mention that it is not a straight-forward task to make a full comparison between our results and those reported in the literature due to the differences in computing hardware, the version of CPLEX, programming language, etc. However, we hope the experimental studies shown in this section provide some useful indications about the performance of the proposed algorithms relative to the current best MMKP procedures.

3.3 Comparative results on 27 classical instances with a limit of 500 seconds

In this section, we report our results obtained by the PEGF and PERC algorithms with the time limit of 500 seconds, and compare them with the best known results over the set of the 27 classical benchmark instances. We also show a comparative study with respect to the best state-of-art algorithms.

Table 1 shows the results of our two algorithms together with the best known results in the literature. In Table 1, the best known lower bounds (column $\text{BKLB}_{\leq 1200s}$) are given by several MACH variants [20] (with a time limit of more than 500 seconds), CPH [22] (with a time limit of 1200 seconds) and CH [4] (with a time limit of 1200 seconds), while the best known upper bounds (column BKUB) are only provided by MACH [20]. Column $\underline{v}(P)$ and $\bar{v}(P)$ respectively indicate the lower bound and upper bound obtained by the corresponding algorithms, where the best known value is highlighted in bold while a '*' symbol indicates an improved best result. *cpu* gives the time when the algorithm encounters the best solution for the first time.

From Table 1, we observe that our PEGF and PERC algorithms attain respec-

Table 1
Computational results of the PEGF and PERC on the set of 27 classical MMKP benchmark instances with a time limit of 500 seconds. A value highlighted in bold denotes a best known result and a starred value indicates an improved lower bound.

Instance	$BKLB_{\leq 1200s}$	BKUB	PEGF			PERC		
			$\underline{v}(P)$	$\bar{v}(P)$	*cpu*	$\underline{v}(P)$	$\bar{v}(P)$	*cpu*
I07	24592_{CPH}	24605.2	24588	24604.5	214	24590	24605.1	75
I08	36894_{CH}	36902.9	**36894**	36900.6	397	36892	36901.1	74
I09	49182_{MACH}	49192.8	49181	49190.6	393	**49185***	49191.0	466
I10	61480_{MACH}	61485.4	61473	61483.5	144	61474	61483.6	145
I11	73789_{MACH}	73797.1	73787	73795.6	253	73787	73795.9	205
I12	86094_{MACH}	86099.8	86090	86098.8	206	86091	86098.8	479
I13	98440_{MACH}	98447.9	98436	98446.7	212	98439	98446.7	403
INST01	10738_{MACH}	10748.7	**10738**	10744.3	97	**10738**	10747.3	196
INST02	13598_{MACH}	13619.3	**13598**	13614.0	38	**13598**	13616.5	147
INST03	10955_{CPH}	10977.7	10947	10975.4	7	10943	10977.0	101
INST04	14456_{MACH}	14474.0	14447	14472.2	121	**14456**	14473.5	215
INST05	17061_{HMW}	17075.2	17057	17072.7	227	17055	17074.0	127
INST06	16838_{MACH}	16853.5	16832	16850.7	76	16833	16852.6	369
INST07	16442_{MACH}	16456.2	16440	16455.0	241	**16444***	16456.7	474
INST08	17510_{MACH}	17530.8	17507	17528.4	105	17503	17529.5	20
INST09	17761_{CH}	17777.3	17760	17776.1	124	17760	17776.7	394
INST10	19316_{MACH}	19334.7	19314	19333.1	275	19312	19333.7	365
INST11	19441_{MACH}	19459.9	19437	19457.9	114	**19449***	19459.5	330
INST12	21738_{MACH}	21754.9	**21738**	21753.4	4	**21738**	21754.5	175
INST13	21577_{MACH}	21591.3	**21577**	21590.1	198	21575	21591.0	15
INST14	32875_{MACH}	32886.4	32873	32884.5	427	32873	32884.9	209
INST15	39161_{MACH}	39173.5	**39161**	39172.8	311	**39161**	39172.9	146
INST16	43366_{MACH}	43378.6	**43367***	43376.4	98	43362	43377.1	157
INST17	54363_{MACH}	54371.3	54360	54370.2	128	54360	54370.3	50
INST18	60467_{MACH}	60477.9	**60467**	60476.1	219	60466	60476.1	22
INST19	64932_{MACH}	64943.0	**64932**	64941.1	150	64931	64941.1	352
INST20	75618_{MACH}	75626.4	75613	75625.4	317	75614	75625.4	51

tively 9 and 8 best known results for 9 out of the 27 instances among which 4 corresponds to improved best lower bounds (1 from PEGF and 3 from PERC). The best solutions provided by both PEGF and PERC are slightly different according to the instances, showing some complementarity of the two algorithms. The two algorithms attain together 14 best known results. For those instances where they fail to reach the best known results, the gaps between our results and the best known ones are very small. The largest gap is 0.01% for I10 (calculated by (OurResult-BestKnown)/BestKnown×100) .

Moreover, the values reported in columns $\bar{v}(P)$ show that our algorithms generate strong upper bounds. Compared to the best known upper bounds (column BKUB) provided by MACH, PEGF produces better results for all the instances. The same performance is observed for PERC with only one exception (i.e. INST07). Recall that the upper bounds are obtained by solving the MIP-relaxation of the original problem P (see equation (6)), where a subset of variables is forced to be binary. In our two algorithms, the variables forced to be binary are those in the reduced problem and they show to be "critical" ones as confirmed by both the strong upper bounds and the favorable lower bounds. In addition, column *cpu* shows that our algorithms visit most of their best solutions in less than 400 seconds, far before reaching the given time limit.

In this experiment, PEGF realizes on average 9 iterations across the instance set. The reduced problems can be solved to optimality within 37 seconds for all the instances when k is small enough (i.e., $k = k_{start}$). As k increases, the reduced problems become larger and thus harder to solve. For PERC, given the value of k calculated by Equation (10), the induced reduced problems cannot be optimally solved for most of the instances even if we impose a time limit of 500 seconds for each of them. So we observed only 1 iteration for all the instances except INST02 where there are two iterations. The encouraging results presented in Table 1 show that these reduced problems are promising and additionally confirm the merit of our fixing rules. Indeed, a decrease of t_{max} (the time limit for a reduced problem) allows the method to explore more different reduced problems and introduces some diversities on the final results. However, the peak performance of PERC is achieved by adopting the parameter settings introduced in Section 3.2.

To further evaluate the performance of our algorithms, we show a comparison with CPLEX, and 5 recent algorithms that achieve state-of-art performances:

- A Hybrid heuristic approach [20]. The best version MACH3 of the 5 algorithm variants is used for comparison (column MACH3). The reported results were performed on a Dell computer 2.4GHz and CPLEX 11.2 with an overall CPU limit > 500 seconds [3].
- A compositional pareto-algebraic heuristic method [22]. The best results obtained by the serial version of their approach (i.e., CPH+OptPP) are listed (column CPH). The tests were performed with an overall CPU limit of 1200 seconds on a PC with an Intel processor (2.8Ghz and 12G RAM).
- A column generation method [4]. We list its best reported results (column CH). The tests were performed on an UltraSparc10 2.5GHz with an overall CPU limit of 1200 seconds.

[3] The paper [20] indicates a limit of 500 seconds, however one observes that Table 2 of [20] includes computing times greater than 500 for several cases.

Table 2
Comparative results of PEGF and PERC (with a time limit of 500s) with CPLEX and 5 state-of-art approaches (MACH3 [20] (2013), CPH [22] (2013), CHMW [3] (2012), CH [4] (2009), HMW [9] (2009)) on the set of 27 classical MMKP benchmark instances. The comparison focus on solution quality. A value in bold indicates a best known lower bound or an improved lower bound. The row "♯Bests" indicates the number of best lower bounds (values in bold) obtained by the corresponding algorithm, "Sum." denotes the sum of the best solution values over the instance set.

Instance	$BKLB_{\leq 1200s}$	PEGF	PERC	CPLEX	MACH3	CPH	CH	HMW	CHMW
I07	24592_{CPH}	24588	24590	24584	24586	**24592**	24587	24586	24585
I08	36894_{CH}	**36894**	36892	36873	36888	36885	**36894**	36883	36885
I09	49182_{MACH}	49181	**49185**	49162	49180	49179	49179	49172	49172
I10	61480_{MACH}	61473	61474	61474	**61480**	61464	61464	61465	61460
I11	73789_{MACH}	73787	73787	73765	73783	73780	73780	73770	73778
I12	86094_{MACH}	86090	86091	86077	**86094**	86081	86080	86077	86077
I13	98440_{MACH}	98436	98439	98425	98438	98433	98433	98428	98429
INST01	10738_{MACH}	**10738**	**10738**	10709	10724	10727	**10738**	10714	10732
INST02	13598_{MACH}	**13598**	**13598**	**13598**	**13598**	**13598**	**13598**	13597	**13598**
INST03	10955_{CPH}	10947	10943	10935	10938	**10955**	10944	10943	10943
INST04	14456_{MACH}	14447	**14456**	14439	**14456**	14452	14442	14442	14445
INST05	17061_{HMW}	17057	17055	17043	17053	17055	17055	**17061**	17055
INST06	16838_{MACH}	16832	16833	16813	16832	16827	16823	16827	16823
INST07	16442_{MACH}	16440	**16444**	16419	16442	16440	16440	16434	16440
INST08	17510_{MACH}	17507	17503	17492	17508	17507	**17510**	17503	17505
INST09	17761_{CH}	17760	17760	17753	17760	17757	**17761**	17751	17753
INST10	19316_{MACH}	19314	19312	19299	19311	19314	19316	19311	19306
INST11	19441_{MACH}	19437	**19449**	19417	19437	19441	19441	19430	19434
INST12	21738_{MACH}	**21738**	**21738**	21723	**21738**	**21738**	21732	**21738**	**21738**
INST13	21577_{MACH}	**21577**	21575	21571	**21577**	**21577**	**21577**	21574	21574
INST14	32875_{MACH}	32873	32873	32866	32872	32872	32872	32869	32869
INST15	39161_{MACH}	**39161**	**39161**	39158	**39161**	39160	39160	39160	39160
INST16	43366_{MACH}	**43367**	43362	43359	43366	43363	43362	43363	43363
INST17	54363_{MACH}	54360	54360	54353	**54363**	54360	54360	54356	54352
INST18	60467_{MACH}	**60467**	60466	60461	**60467**	60465	60460	60462	60463
INST19	64932_{MACH}	**64932**	64931	64924	64931	64931	64925	64925	64924
INST20	75618_{MACH}	75613	75614	75605	75614	75613	75612	75609	75609
♯Bests		9	8	1	9	5	6	2	2
Sum.	1018684	1018614	1018618	1018297	1018597	1018566	1018548	1018437	1018461

- An iterative relaxation-based heuristic approach [9]. The best version of the 3 algorithm variants is compared in this paper (column HMW). The evaluations were performed with a CPU limit of 300 seconds on a Pentium IV 3.4GHz, but the version of CPLEX is not indicated.

- An iterative semi-continuous relaxation heuristic approach [3]. We select the best version among the 4 algorithm variants for comparison (column CHMW). The tests were performed on a Pentium IV 3.4GHz, using CPLEX 11.2 with a CPU limit of 400 seconds.

Table 2 summarizes the results of our PEGF and PERC algorithms along with those reported by the 5 reference algorithms and those obtained by CPLEX with a time limit of 500 seconds. The last two rows (♯Bests and Sum.) indicate respectively the number of best known results and the total objective value over the 27 instances reached by an algorithm. From Table 2, we can observe that our two algorithms attain respectively 9 (PEGF) and 8 (PERC) best known results (♯Bests) and their total objective values are higher (better) than those of the reference algorithms (Sum.) and CPLEX.

Compared to MACH3 which is one of the current best performing algorithms for the MMKP, our algorithms remain very competitive even if the improvement of our algorithms over MACH3 is relatively small. Notice that the overall qualities of the best known solutions of the benchmark instances are already very high (see Table 1, the gaps between the upper bounds and the lower bounds for all the instances are small). As such, even a small improvement could be difficult to reach and thus can be considered to be valuable. It should be noted that MACH3 was executed with a time limit of more than 500 seconds on a machine which is slightly faster than our computer according to the Standard Performance Evaluation Corporation (www.spec.org), though our CPLEX version is more recent.

This experiment demonstrates that our approach outperforms CPLEX and most of the reference algorithms and competes well with the best performing MACH3 algorithm.

3.4 Comparative results on 27 classical instances with a limit of 1 hour

In this section, we investigate the behaviors of our algorithms by extending the total time limit to 1 hour and comparing our algorithms with both CPLEX 12.4 and the results reported in [3] (see Table 3).

In [3], the authors proposed 4 different variants of a heuristic approach namely ILPH, IMIPH, IIRH and ISCRH, and they reported the results obtained by their approaches with a time limit of 1 hour. Table 3 lists for each instance the best lower bound visited by our PEGF and PERC algorithms, the value of the best feasible solution of CPLEX 12.4 obtained within 1 hour of computing time, and the best value of ILPH, IMIPH, IIRH and ISCRH from [3]. A value in bold means that our algorithms reach the best known result reported in the literature and a starred value represents a new best lower bound (a record

Table 3
Comparative results of PEGF and PERC (with a time limit of 1h) with CPLEX (with a time limit of 1h) and four variant algorithms proposed in [3] on the set of 27 classical MMKP benchmarks. The row "♯Bests" indicates the number of best lower bounds (values in bold) obtained by the corresponding algorithm, "♯RLB" represents the number of improved lower bounds (lower bounds starred) which can be obtained by the corresponding algorithm, "Sum." denotes the sum of the best solution values over the instance set.

Instance	$BKLB_{\leq 1200s}$	PEGF	PERC	CPLEX	ILPH	IMIPH	IIRH	ISCRH
I07	24592_{CPH}	24590	24592	**24595***	24591	24587	24592	24584
I08	36894_{CH}	**36894**	36892	36886	36888	36889	36888	36882
I09	49182_{MACH}	**49185***	**49185***	49173	49174	49183	49179	49173
I10	61480_{MACH}	61477	61478	61472	61469	61471	61466	61465
I11	73789_{MACH}	**73791***	73789	73783	73784	73779	73779	73780
I12	86094_{MACH}	**86095***	86091	86088	86091	86083	86091	86081
I13	98440_{MACH}	98441	98441	98435	98435	**98445***	98433	98424
INST01	10738_{MACH}	**10738**	**10738**	**10738**$^\circ$	**10738**	10728	10728	10717
INST02	13598_{MACH}	**13598**	**13598**	**13598**$^\circ$	**13598**	**13598**	**13598**	**13598**
INST03	10955_{CPH}	10947	10947	10944	10949	10943	10949	10943
INST04	14456_{MACH}	14447	**14456**	14442	14442	14445	14446	14445
INST05	17061_{HMW}	17057	**17061***	17053	17058	17055	17057	17053
INST06	16838_{MACH}	**16840***	**16840***	16832	16835	16832	16832	16823
INST07	16442_{MACH}	**16444***	**16444***	16440	16440	16440	16440	16429
INST08	17510_{MACH}	17508	**17514***	17510	17510	17511	17510	17505
INST09	17761_{CH}	**17763***	**17763***	17753	17760	17760	17753	17757
INST10	19316_{MACH}	19314	19316	19316	**19320***	**19320***	19314	19316
INST11	19441_{MACH}	**19449***	**19449***	19441	19446	19446	19446	19437
INST12	21738_{MACH}	21738	**21741***	21732	21733	21738	21738	21729
INST13	21577_{MACH}	21578	21578	21576	21577	**21580***	21577	21573
INST14	32875_{MACH}	**32875**	**32875**	32872	32873	32872	32872	32870
INST15	39161_{MACH}	39161	39162	**39163***	39161	39160	39162	39156
INST16	43366_{MACH}	**43367***	43366	43365	43366	43363	43363	43362
INST17	54363_{MACH}	**54363**	**54363**	54360	54361	54360	54358	54356
INST18	60467_{MACH}	**60467**	**60467**	60465	**60467**	**60467**	60465	60461
INST19	64932_{MACH}	**64932**	64931	64929	64930	**64932**	64929	64929
INST20	75618_{MACH}	75616	75614	75615	75613	75611	75613	75610
♯Bests		15	14	4	4	6	1	1
♯RLB		8	8	2	1	3	0	0
Sum.	1018684	1018675	1018691	1018576	1018609	1018598	1018578	1018458

lower bound) for the given instance. In the column "CPLEX", the values of two instances (INST01 and INST02) are marked with ∘ which means they are proved to be optimal by CPLEX. In the last three rows of Table 3, we indicate for each algorithm the number of best lower bounds obtained, the number of improved lower bounds (lower bounds starred) which can be obtained by the algorithm (♯RLB) and the sum of the best solution values over the instance set (Sum.).

From Table 3, we observe that our algorithms visit the best known solution for 19 out of the 27 instances. Among these best solutions, 11 were never reported by previous approaches. PEGF and PERC respectively obtain 15 and 14 best known lower bounds and their average solution qualities are both very high, which shows both algorithms are very effective under this stopping condition. Compared to CPLEX, our algorithms obtain 24 better lower bounds, 1 equal result and 2 worse results. Compared to the 4 variants proposed in [3], our algorithms compete favorably by holding more record lower bounds (11 vs 3). Finally, when we examine the total objective value attained by each algorithm over the whole set of the 27 instances, we observe that the proposed algorithms dominate both CPLEX and the four reference algorithms.

3.5 Comparative results on 10 recent instances with irregular structures

The last experiment is dedicated to the set of 10 large irregular benchmark instances introduced in [22]. Like [22], we adopt a total time limit of 1200 seconds and 1 hour, respectively. We also provide the results obtained by CPLEX for our comparative study.

In [22], the results of two variants of their proposed CPH approach were reported, namely CPH+OptPP with the default resource usage aggregation and with the scarcity-aware resource aggregation. We use the best solution values of their two algorithm variants as our reference. Table 4 shows our results (PEGF and PERC) as well as those of CPH and CPLEX under the two time conditions. For each instance, a solution value is highlighted in bold if it is the best among those results obtained by the four listed algorithms. In the last two rows of Table 4, we indicate for each algorithm the number of best lower bounds obtained (♯Bests) and the sum of the best solution values over the instance set (Sum.).

From Table 4, we observe that our algorithms compete very favorably with CPH and CPLEX on these instances. Our algorithms discovers improved best known solutions for 7 out of 10 instances. For the time limit of 1200 seconds, each of our two algorithms attains 5 best solutions against 2 for CPH and 1 for CPLEX. Both PEGF and PERC give better results than CPH and CPLEX in

Table 4
Comparative results on 10 large irregular instances from [22]. Our results are compared with those of CPH from [22] and CPLEX with a time limit of 1200 seconds and 3600 seconds (1 hour). A value highlighted in bold indicates the best solution value among the four compared results.

Instance	T_{max}=1200s				T_{max}=3600s			
	PEGF	PERC	CPH	CPLEX	PEGF	PERC	CPH	CPLEX
INST21	**44280**	**44280**	44262	44260	**44280**	**44280**	44270	44260
INST22	41966	41936	**41976**	41928	41966	41952	**41976**	41960
INST23	42522	**42584**	42502	42512	42522	**42584**	42562	42512
INST24	41820	41852	**41918**	41864	41876	41860	**41918**	41878
INST25	**44156**	**44156**	44138	44135	44156	**44159**	44156	44135
INST26	**44865**	44861	44832	44848	**44879**	44861	44869	44878
INST27	**87630**	**87630**	87616	87594	**87630**	**87630**	87616	87616
INST28	134638	134638	134634	**134648**	134642	134642	134634	**134648**
INST29	179222	**179224**	179186	179206	**179228**	179224	179206	179208
INST30	**214216**	214210	214198	214192	214216	**214230**	214198	214192
♯Bests	5	5	2	1	4	5	2	1
Sum.	875315	875371	875262	875187	875395	875422	875405	875287

8 out of 10 cases, while there are only two cases (INST22 and INST24) where CPH performs better and one case where CPLEX performs better (INST28). Regarding the total solution value, our two algorithms outperform both CPH and CPLEX. A similar observation can be made for the case of 1 hour execution. One exception is that the total solution value of PEGF is slightly worse than that of CPH though PEGF obtains better results for more cases than CPH does (6 vs 3). Moreover, the results show that our two algorithms are able to achieve further improved best lower bounds for 4 instances when more time is available. This experiment confirms that our proposed algorithms perform well on these irregular instances.

4 Discussion

In this section, we provide an analysis about the impact of the group and variable fixing techniques on the initial model. For this purpose, we focus on the experiment reported in Table 3 (with a time limit of 1 hour on the set of 27 classical instances). Specifically, for each of our two algorithms and each problem instance, we record the reduced problem that has led to the best solution value and calculate the values (as a percentage) of three indicators: the number of fixed groups ($\%_{FG}$), the number of fixed variables ($\%_{FV}$) and the size of the reduced subproblem ($\%_{RP}$).

Precisely, the number of fixed groups as a percentage over the total number of groups is given by $\%_{FG} = 100 \cdot |I^2(\tilde{x})|/n$, the number of fixed variables as a percentage over the total number of variables in the unfixed groups is defined by $\%_{FV} = 100 \cdot |S_{fixed}(\tilde{x})|/(|S_{fixed}(\tilde{x})| + |S_{free}(\tilde{x})|)$ where $S_{fixed}(\tilde{x}) = \{(i,j)|i \in I^{2*}(\tilde{x}), j \in \{1,...,g_i\}, |r_{ij}| > RC_{max}(\tilde{x})\}$, and the size of the reduced problem as a percentage over the size of the original problem is defined as $\%_{RP} = 100 \cdot |S_{free}(\tilde{x})|/|S|$.

Moreover, to show a general picture and simplify the presentation, we divide the whole set of 27 instances into three classes according to the value of groups n: $n \in [50, 90]$, $n \in [100, 200]$ and $n \in [250, 400]$. Table 5 displays for each instance class, the number of the instances in the class (#) and the aggregated statistics of the above three indicators ($\%_{FG}$, $\%_{FV}$, $\%_{RP}$) in terms of their minimum ($min.$), maximum ($max.$) and average values ($avg.$).

From Table 5, we observe that the percentage of the fixed groups increases generally as n increases for both algorithms. The average value of $\%_{FG}$ is less than 50% for the instance class with n smaller than or equal to 90. Then it jumps dramatically to 70% for the second larger instance class. This value even grows up to more than 80% when n is greater than or equal to 250. A similar trend is also observed for the fixed variables, the average value of $\%_{FV}$ shows a relatively steady rise across the three instance classes with the smallest value around 70% and the largest value around 84%. In contrast, the size of the reduced problem shows generally a negative correlation with the change of n for the two proposed algorithms. The average value of $\%_{RP}$ is around 15% for the first instance class. A sharp drop of this value is seen for the second instance class where the average percentage is slightly different for PEGF (4.93%) and PERC (6.36%). For the third large instance class, the average $\%_{RP}$ decreases to around 3%.

Given the above observations, we can conclude that, with the fixing rules proposed in this paper, the number of fixed groups and fixed variables increases as the size of problem enlarges, leading to a small and promising reduced problem whose size increases very slowly. Typically, there are less than 200 variables left in the reduced models for these instances where the size of the original problems can have up to 7000 variables (INST20).

5 Conclusions

The multiple-choice multidimensional knapsack problem (MMKP) is a highly useful model in practice and represents nevertheless a real challenge from a computational point of view. We have presented a "reduce and solve" approach for the MMKP which jointly uses problem reduction techniques and the state-

Table 5
Statistical data on the fixing process

n	#	*stat*	PEGF			PERC		
			$\%_{FG}$	$\%_{FV}$	$\%_{RP}$	$\%_{FG}$	$\%_{FV}$	$\%_{RP}$
$[50, 90]$	11	*min.*	30.00	66.00	9.75	30.00	58.12	10.44
		max.	64.44	76.36	22.00	56.67	75.90	26.80
		avg.	48.64	72.41	14.37	48.40	69.58	15.96
$[100, 200]$	8	*min.*	60.00	72.05	1.43	61.00	64.87	2.04
		max.	81.50	92.25	10.90	77.00	92.20	13.70
		avg.	70.07	83.80	4.93	69.38	80.11	6.36
$[250, 400]$	8	*min.*	80.67	77.91	1.28	80.00	72.81	1.67
		max.	87.43	92.00	4.10	84.75	91.67	5.17
		avg.	84.54	84.19	2.52	81.72	83.17	3.05

of-the-art ILP technology. On the one hand, the proposed approach employs a two-stage fixing method (group fixing and variable fixing) based on LP-relaxation to generate reduced problems which are solved by the ILP solver. To better explore the space of the reduced problems, the proposed approach makes uses of two dedicated strategies to progressively expand the reduced problems, leading to two algorithm variants (PEGF and PERC).

Using a set of 27 well-known classical MMKP benchmark instances and a set of 10 recent benchmarks with irregular structures from the literature, we have assessed the performance of the proposed approach and compared our results with some current best performing methods. The computational experiments have shown the proposed approach competes favorably with the state-of-the-art reference algorithms and dominates the well-known commercial CPLEX ILP solver. In particular, the proposed approach discovers 11 improved lower bounds and as a by-product provides 27 improved upper bounds for the set of classical MMKP instances. And for the set of 10 irregular benchmarks, the proposed approach is also able to discover 7 improved lower bounds. This study confirms the merit of a hybrid approach combining heuristics and a state of the art ILP solver.

Acknowledgment

We are grateful to the anonymous referees for their insightful comments and suggestions which helped us to improve the paper. The work is partially supported by the RaDaPop (2009-2013) and LigeRo projects (2009-2013) from the Region of Pays de la Loire (France). Support for Yuning Chen from the China Scholarship Council is also acknowledged.

References

[1] Basnet C., Wilson J. Heuristics for determing the number of warehouses for storing noncompatible products. International Transactions in Operational Research 2005; 12(5):527-538.

[2] Boussier S., Vasquez M, Vimont Y, Hanafi S, Michelon P. A multi-level search strategy for the 0-1 multidimensional knapsack problem. Discrete Applied Mathematics 2010; 158(2):97-109.

[3] Crévits I., Hanafi S., Mansi R., Wilbaut C. Iterative semi-continuous relaxation heuristics for the multiple-choice multidimensional knapsack problem. Computers and Operations Research 2012; 39(1): 32-41.

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

[5] Gavish B., Pirkul H. Allocation of databases and processors in a distributed data processing. In: Akola J, editor. Management of distributed data processing. North-Holland: Amsterdam;1982. P.215-231.

[6] Ghasemi T., Razzazi M. Development of core to solve the multidimensional multiple-choice knapsack problem. Computers and Industrial Engineering 2011; 60(2):349-360.

[7] Glover F. Heuristics for integer programming using surrogate constraints. Decision Sciences 1977; 8(1):156-166.

[8] Glover F. Adaptive memory projection methods for integer programming. In: Rego C, Alidaee B, editors. Metaheuristic optimization via memory and evolution. Kluwer Academic Publishers, 2005, p.425-440.

[9] Hanafi S., Mansi R., Wilbaut C. Iterative relaxation-based heuristics for the multiple-choice multidimensional knapsack problem. Volume 5818 of Lecture Notes in Computer Science 2009; P.73-83. Springer Verlag.

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

[11] Hifi M.,Wu L. An Equivalent Model for Exactly Solving the Multiple-choice Multidimensional Knapsack Problem. International Journal of Combinatorial Optimization Problems and Informatics 2012; 3(3):43-58.

[12] Hiremath CS, Hill RR. First-level tabu search approach for solving the multiple-choice multidimensional knapsack problem. International Journal of Metaheuristics 2013; 2(2):174-199.

[13] Htiouech S, Bouamama S. OSC: solving the multidimensional multi-choice knapsack problem with tight strategic Oscillation using Surrogate Constraints. International Journal of Computer Applications 2013; 73:1-22.

[14] Iqbal S, Faizul Bari Md, Sohel Rahman M. Solving the multi-dimensional multi-choice Knapsack Problem with the help of ants. Lecture Notes in Computer Science 2010; 6234: 312-323.

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

[16] Kellerer H, Pferschy U, Pisinger D. Knapsack problems. Springer, 2004.

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

[18] Pisinger D. Budgeting with bounded multiple-choice constraints. European Journal of Operational Research 2001; 129:471-480.

[19] Puchinger J., Raidl G., Pferschy U. The multidimensional knapsack problem: structure and algorithms. INFORMS Journal on Computing 2010; 22(2):250-265.

[20] Mansi R., Alves C, Valerio de Carvalho J.M., Hanafi S. A hybrid heuristic for the multiple choice multidimensional knapsack problem. Engineering Optimization 2013; 45(8):983-1004.

[21] Sbihi A. A best first search exact algorithm for the Multiple-choice Multidimensional Knapsack Problem. Journal of Combinatorial Optimization 2007; 13(4):337-351.

[22] Shojaei H, Basten T, Geilen M, Davoodi A. A fast and scalable multidimensional multiple-choice knapsack heuristic. ACM Transactions on Design Automation of Electronic Systems 2013; 18(4) 32 pages, DOI=10.1145/2541012.2541014.

[23] Vasquez M., Hao J.K. An hybrid approach for the 0-1 multidimensional knapsack problem. Proc. 17th International Joint Conference of Artificial Intelligence (IJCAI-01), Morgan Kaufmann, San Francisco, P.328-333.

[24] Vasquez M., Vimont Y. Improved results on the 0-1 multidimensional knapsack problem. European Journal of Operational Research 2005; 165(1):70-81.

[25] Vimont Y, Boussier S, Vasquez M. Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem. Journal of Combinatorial Optimization 2008; 15:165-178.

[26] Wang Y., Lu Z., Glover F., Hao J.K. Effective variable fixing and scoring strategies for Binary Quadratic Programming. Lecture Notes in Computer Science 2011; 6622:72-83.

[27] Wang Y., Lu Z., Glover F., Hao J.K. Backbone guided tabu search for solving the UBQP problem. Journal of Heuristics 2013; 19(4):679-695.

[28] Watson RK. Packet networks and optimal admission and upgrade of service level agreements: applying the utility model. M.A.Sc. Thesis, Department of ECE, University of Victoria, Canada, 2001.

[29] Wilbaut C, Hanafi S. New convergent heuristics for 0-1 mixed integer programming. European Journal of Operational Research 2009; 195(1):62-74.