A New Hybrid Algorithm for the Multiple-Choice Multi-Dimensional Knapsack Problem

MAHMOUD ZENNAKI
Computer science department, faculty of mathematics and computer science
University of Science and Technology of Oran 'Mohamed Boudiaf' USTO MB
PO 1505 El M'naoeur, Bir el Djir, Oran, Algeria
mahmoud.zennaki@univ-usto.dz

Abstract: In this paper, we approximately solve the multiple-choice multi-dimensional knapsack problem. We propose a hybrid algorithm based on branch and bound method and Pareto-algebraic operations. The algorithm starts by an initial solution and then combines one-by-one groups of the problem instance to generate partial solutions in each iteration. Most of these partial solutions are discarded by Pareto dominance and bounding process leading at the end to optimality or near optimality in the case when only a subset of partial solutions is maintained at each step. Furthermore, a rounding procedure is introduced to improve the bounding process by generating high quality feasible solutions during algorithm execution. The performance of the proposed heuristic has been evaluated on several problem instances. Encouraging results have been obtained.

Key-Words: combinatorial optimization, heuristics, knapsacks, branch and bound.

1 Introduction

Recently, a more complex variant of knapsack problem, an NP-hard problem, has been studied by a large number of researchers. This variant called multi-dimensional multiple-choice knapsack problem (MMKP) is among the most challenging of the encountered optimization problems. The MMKP problem instances appear for example in chip multiprocessor run-time resource management, global routing of wiring in circuits [1] and other practical problems as the service level agreement and, the model of allocation resources [2].

The basic 0-1 knapsack problem considers n items, where each item has a profit value v and a resource cost given by the weight w. The objective is to put items in a knapsack so that the resource capacity of the knapsack is not exceeded and the summed value of packed items is maximal. Instead of n items, n groups of items may be considered where one item from each group must be selected, leading to the multiple-choice knapsack problem (MCKP). The multi-dimensional knapsack problem (MDKP) is another variant in which a multi-dimensional resource cost is considered for each item and each dimension has its own capacity. The multi-dimensional multiple-choice knapsack problem (MMKP) [3] combines the two aforementioned variants, and is the focus of this paper.

Formally, the MMKP can be stated as follows: given n classes J_i of items, where each class J_i, $i = 1, \dots, n$, has r_i items. Each item j, $j = 1, \dots, r_i$, of class J_i has the non-negative profit value v_{ij}, and requires resources given by the weight vector $W_{ij} = (w_{ij}^1, \dots, w_{ij}^m)$ where each weight component w_{ij}^k, $k = 1, \dots, m$ is also a nonnegative value. The amounts of available resources are given by a vector $C = (C^1, \dots, C^m)$. The aim of the MMKP is to pick exactly one item from each class in order to maximize the total profit value of the pick, subject to resource constraints. If we consider decision variables $x_{ij} = 1$ when item j of the i-th class is picked, 0 otherwise, the MMKP can be formulated in an Integer Linear Program (ILP) as follows:

$$(ILP)\begin{cases} \max Z = \sum_{i=1}^{n}\sum_{j=1}^{r_i} v_{ij}x_{ij} \\ s.t \sum_{i=1}^{n}\sum_{j=1}^{r_i} w_{ij}^k x_{ij} \le C^k, k \in \{1, \dots, m\} \\ \sum_{j=1}^{r_i} x_{ij} = 1, \ i \in \{1, \dots, n\} \\ x_{ij} \in \{0,1\}, \ i \in \{1, \dots, n\}, \ j \in \{1, \dots, r_i\} \end{cases}$$

We are talking about regular MMKP instances when $r_i = r \ \forall i = 1, \cdots n$ unlike irregular ones.

This paper presents BPH, for Branch and bound Pareto-algebraic Heuristic. BPH is a heuristic based on Branch and Bound (B&B) and uses the principle of Pareto algebra [4], [5], which is a framework for calculation of Pareto-optimal solution in multi-dimensional optimization problems. This combination of B&B and Pareto algebra concepts allow solving to near-optimality large MMKP instances in a reduced CPU time. At each step of the heuristic, partial solutions are computed by considering groups of items one at a time. Based on Pareto algebra, some dominated solutions are discarded, and then bound principle is applied by solving correspondent ILP relaxation. The bound process can discard more partial solutions, leading to a reduced number of active partial solutions during heuristic execution. When handling large instances, the practical run-time of BPH is determined by the number of partial solutions considered in each step. This provides a parameter to control the run-time of these computations and to trade-off run-time and quality of the final solution.

The remainder of the paper is organized as follows. In section 2, we present a brief reference of some sequential exact and approximate algorithms for MMKP. Section 3 introduces the BPH heuristic by means of an example; it further presents the relevant concepts of Pareto algebra. Then, in section 4, the proposed algorithm is presented in detail. In section 5, the performance of BPH is tested on a set of problem instances extracted from the literature including some very large instances. We conclude in the last section our paper with some interesting remarks.

2 Related works

As for other combinatorial optimization problems, two types of solution approaches for MMKP have been proposed in the literature. Exact solutions find an optimal solution for MMKP instances; heuristic solutions try to find a near-optimal solution, but require much less computation time than exact solutions.

Almost all successful exact methods are based on branch and bound algorithm. In [6], the author uses a branch and bound search tree to represent the solution space and linear programming to find bounds. The order in which the decision variables are considered has an important effect on the size of the search tree. Sbihi in [7] and [8] proposes a more powerful exact branch and bound algorithm for MMKP based on best-first search strategy. The approach fixes selected items during exploration, using linear programming to compute bounds during the search.

The literature also describes heuristic methods. The first heuristics developed in [3], [6], [9], [10], [11], [12] and [13] share the idea to project all resource dimensions of a candidate solution to a single aggregate resource, effectively reducing the multi-dimensional search space into a two-dimensional search space. Items are sorted with respect to a specific utility metric, which is unique for each approach. The approaches first find a feasible solution for an MMKP instance and then iterate over the sorted list of items to improve the candidate solution. Column generation approach has been also used in [14], explicitly targeting large-scale MMKP problems. The approach uses a rounding stage and then restricts the resource constraints and solves an exact instance of the restricted MMKP. The authors in [15] extend this approach and propose a hybrid algorithm that combines local branching and column generation techniques to generate higher-quality solutions. Cherfi provides a final extension of this algorithm, called BLHG, in his PhD thesis [16]. Recently, Crevits et al. [17] proposed a new iterative relaxation-based heuristic for MMKP. The approach generates upper bounds for the problem using relaxation and refinement. It generates lower bounds based on a restricted version of the problem. A new semi-continuous relaxation leads to high solution quality. In [18], the authors propose for the first time a compositional Pareto-algebraic MMKP heuristic. It is parameterized by the number of partial solutions considered in each compositional step. Pareto optimality is the central criterion to compare partial solutions and discard suboptimal ones. The authors achieve remarkable solution quality and computation times. This work has been extended in [1] improving both quality and speed and enabling a better tuning to the problem at hand by fully parameterizing the heuristic.

Motivated by the success of branch and bound algorithm in exact methods and Pareto-algebra in approximate methods, we propose a more powerful heuristic based on a combination of the two aforementioned approaches enhanced by a rounding procedure which can generate high quality feasible solutions during the search process.

3 Pareto-algebraic concepts

This section describes basic Pareto-algebraic notions [4], [5]. We give an example illustrating how MMKP can be solved using a Pareto algebra approach. We then define the basic concepts and operations for multi-dimensional optimization. Finally, we discuss algorithmic aspects for Pareto algebra.

3.1 An example

We illustrate in figure 1 the Pareto-algebraic solution to MMKP. A regular MMKP instance with three groups of same number of items is given. Each item has a value v and a two-dimensional weight w_1 and w_2. The amounts of available resources are given by C^1 and C^2.

First, groups of items are represented as sets of tuples. For example, group J_1 is represented by the set $\{(13,7,9); (18,5,11); (11,6,10)\}$, where the first element of each triple represents the value of the item, and the other two elements represent its two dimensional weight. Then, the sets of items are iteratively combined by means of the so-called product-sum operation $\otimes$ [1]. This operation takes all combinations of items from the two groups $(\text{card}(J_1 \otimes J_2) = \text{card}(J_1) \times \text{card}(J_2))$, summing for each combination their values and weights. An important observation is that elements of this operation can be seen as partial solutions to the MMKP instance and have the same representation as items. Thus, the final solution of MMKP can be obtained by combining groups one-by-one, $(J_1 \otimes J_2) \otimes J_3$.

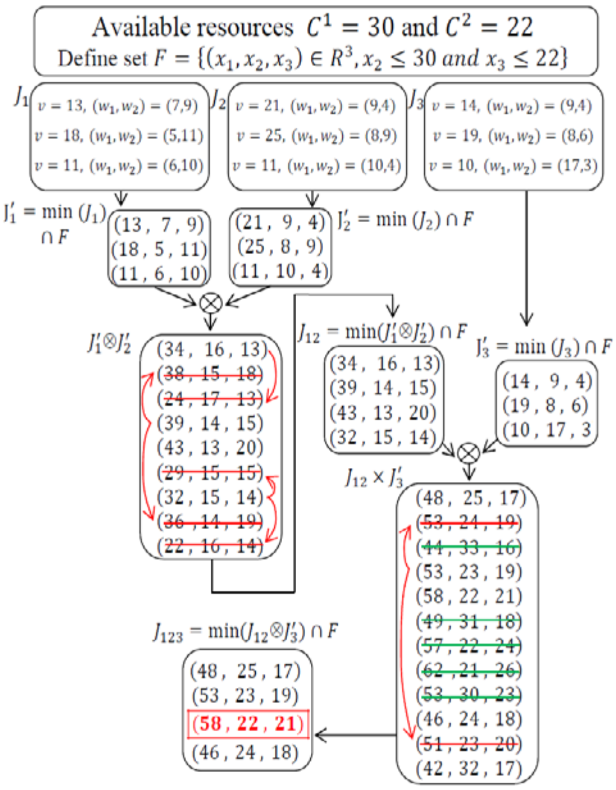

Fig. 1. Example of Pareto-algebraic solution

An important aspect of the solution process is that throughout the application of product-sum operation between groups of items, some of partial solutions in the resulting sets can be discarded. One reason to discard a partial solution is that it violates resource constraints. That is represented in figure 1 by the intersection of sets of tuples with the set of feasible configurations F. Another relevant reason to discard a solution candidate is that it is not better than some other candidate. For example, partial solution (39,14,15) from $J_1' \otimes J_2'$ dominates both (38,15,18) and (36,14,19), because it has an equal or higher value combined with same or lower weights compared to dominated configurations. Such dominated partial solution candidates are not of interest because they can never lead to the optimal solution in the end. Removing dominated elements from a set is referred to as *Pareto minimization*, denoted by the 'min' operation. The result of Pareto minimization is the set of so-called *Pareto points*, i.e., the points (partial solutions, in this case) that are not dominated by any of the other points in the set. This is where the name Pareto algebra comes from. It defines an algebra of operations on sets of Pareto points [4]. We can also see in Figure 1 that the min operation is applied earlier on groups of items to remove any dominated or infeasible items. It is denoted for group J_1 by $J_1' = \min(J_1) \cap F$.

Finally, a set of feasible non dominated solution candidates for the MMKP instance is obtained. In figure 1, the set J_{123} represents this final set which consists of four remaining candidates, the one with the highest value, (58,22,21), is the solution to the MMKP instance.

The essence of this method was used in [1] for the design of an efficient heuristic since exact solution cannot be practical. The number of partial solutions grows exponentially even with eliminating dominated and infeasible solutions. However, the amount of dominated and infeasible solutions remains very low to envisage any exact solution to the MMKP. Another way to discard partial solutions can be very profitable to the solution process. This is one of the key ideas used in this paper.

Before discussing algorithmic aspects of the Pareto-algebraic approach, we summarize and formalize the basic definitions of Pareto algebra.

3.2 Pareto algebra definitions

We summarize in this section some of the definitions given in [5]. Pareto algebra is based on partial order $\preccurlyeq$ that denotes preferred values of a set Q of quantities (quality metric or any other quantified aspect). If $\preccurlyeq$ is total, the quantity is basic. In figure 1, weights w_1 and w_2 are taken from intervals of non-negative real numbers that form two basic quantities with the usual total order $\leq$;

value v is taken from the basic quantity of real numbers with total order ≥ as the preference.

A configuration space S is the Cartesian product $Q_1 \times \cdots \times Q_n$ of a finite number of n quantities; a configuration $\bar{c} = (c_1, \cdots, c_n)$ is an element of such a space. The groups of items in an MMKP instance can thus be captured as configuration sets in Pareto algebra.

The already introduced product-sum operation $\otimes$ can be now defined as follows: $C_1 \otimes C_2 = \{\bar{c}_1 + \bar{c}_2 \,/\, \bar{c}_1 \in C_1;\ \bar{c}_2 \in C_2\}$ with + denoting the element-wise addition on configurations while dominance relation defines preference among configurations. If $\bar{c}_1, \bar{c}_2 \in S$, then $\bar{c}_1$ dominates $\bar{c}_2$ iff for every quantity Q_k, $\bar{c}_1(Q_k) \preccurlyeq \bar{c}_2(Q_k)$ expressing that $\bar{c}_1$ is in all aspects at least as good as $\bar{c}_2$ and thus $\bar{c}_2$ can never lead to optimal configuration. Dominance is reflexive, i.e., a configuration dominates itself. The irreflexive strict dominance relation is denoted $\prec$. A configuration is a *Pareto point* of a configuration set iff it is not strictly dominated by any other configuration. Configuration set C is *Pareto minimal* iff it contains only Pareto points. Figure 1 shows the Pareto minimal configuration set $J_{12} = \min(J_1' \otimes J_2') \cap F$. The crossed out configurations are dominated and hence not Pareto points.

A key objective of Pareto algebra is to support *compositional reasoning* with sets of Pareto points. This means that Pareto minimization may be performed at any intermediate step. For this to work, it should not be possible that configurations that are dominated in intermediate steps could potentially be optimal later on, because dominated configurations are discarded at intermediate steps.

3.3 Algorithmic aspects

Many algorithms have been developed to tackle the problem of Pareto minimization in Pareto algebra (see [19] for an overview). A very simple algorithm for Pareto minimization is the Simple Cull algorithm [20], [21], also known as a block-nested loop algorithm [22]. Simple Cull is a nested loop that essentially compares configurations in the set of N points to be minimized in a pairwise fashion, leading to an $O(N^2)$ worst-case complexity even if observed run-time complexity is often lower. Kung and Bentley [23], [24] developed a divide-and-conquer approach, that is still the algorithm with the best worst-case computational complexity $O(N(logN)^{d-1})$, where d is the dimensionality of these points. Due to its simplicity and practical efficiency, Simple Cull was used in [1] as a core of a Pareto-algebraic heuristic. Procedure *ProductSum-Min* gives the correspondent pseudo code in the form of a three-deep nested loop that takes two configuration sets (groups of items or sets of partial solutions) and a vector of (resources or weights) bounds as input. It outputs a Pareto-minimal result set of feasible, compound configurations, i.e., partial solutions to the MMKP instance that consider all the groups of items that contributed to the two input configuration sets. Line 1 initializes the result set. Lines 2-4 create, and iterate over, all compound configurations, realizing the product-sum operation. Line 5 then checks feasibility of newly created configurations, enforcing the resource constraints. Lines 6-10 check whether the new compound configuration dominates any compound configurations in the result set that so far were Pareto points, or whether the new configuration is itself dominated. After completion of the loops, line 11 returns the result.

```
Input: $C1$, $C2$ configuration sets, $b$ a vector of bounds
Output: result a minimized, compound configuration set,
with only feasible configurations
// Initialize the result to the empty set
1. result = $\emptyset$
// Iterate over all combinations of configurations
2. for all $\bar{c}_i \in C1$ do
3.    for all $\bar{c}_j \in C2$ do
        // Create a compound configuration
4.        $\bar{c} = \bar{c}_i + \bar{c}_j$
        // Continue only if the configuration is feasible
5.        if feasible($\bar{c}$,$b$) then
            // Minimize the result set
            // Maintain an attribute to check whether
            // $\bar{c}$ is dominated
6.          dominated = false
            // Iterate over all configurations so far
            // in the result set
7.            for all $\bar{c}' \in result$ do
                // Check whether the configuration $\bar{c}'$ of
                //the result set is dominated; if so, remove it
8               if $\bar{c} \prec \bar{c}'$ then result = result \ {$\bar{c}'$}
                // Check whether configuration $\bar{c}$ is itself
               //dominated; if so, stop minimization
9.             else if $\bar{c}' \prec \bar{c}$ then dominated = true; break
  // Add configuration $\bar{c}$ to the result set if not dominated
10.         if not dominated then result = result ∪ {$\bar{c}$}
11. return result
```

Fig. 2. Simple Cull-based procedure

Note that Procedure *ProductSum-Min* follows the structure of the two-deep nested loop of Simple Cull; in Simple Cull, the outer loop iterates over all points in the set to be optimized and the inner loop iterates over the result set being maintained. In *ProductSum-Min*, the outer loop of Simple Cull is turned into the nested loop of lines 2 and 3 to create all the configurations that need to be considered for the result. The size of this configuration set is $N = |C1 \times C2|$; the worst-case size of the result set is also N (when all compound configurations are Pareto points). The worst-case complexity of *ProductSum-Min* is therefore $O(N^2)$, i.e., quadratic in terms of the size of the set to be minimized, which is exactly the complexity of Simple Cull.

4 A branch and bound based heuristic

This section introduces our MMKP branch and bound-based heuristic. The proposed approach combines the classical branch and bound algorithm with the Pareto-algebraic heuristic described in [1]. This was motivated by the success of B&B method to solve some small to medium MMKP instances. We first explain the basic algorithm and its parameters. We then provide more detail about these parameters, namely the initial step, the best-first search strategy applied in branch and bound algorithm, and the rounding procedure used to improve solution quality during algorithm execution.

4.1 The basic algorithm

The algorithm is based on combining one-by-one groups of the MMKP instance using Pareto-algebra product operation as explained in figure 1. It is obvious that exact solution based on Pareto-algebra product cannot be considered for large instances. Explicit enumeration of all configurations lead to a solution space of cardinality $\prod_{i=1}^{n} r_i$ or in the case of regular instances r^n. For a medium regular MMKP instance with $n = 30$ and $r = 10$ we have a space of size 10^{30}! This is why discarding partial solutions is more than necessary.

The heuristic developed in [1] called CPH for Compositional Pareto-algebraic Heuristic consists on discarding dominated and infeasible configurations, and since dominance does not often appear, a parameter L that limits the number of partial solutions is considered in each step of the heuristic. It is clear that this parameter has a great influence on final solution quality. A small value of L deteriorates noticeably solution quality. Our basic idea is to add a more powerful criterion to discard partial solutions. This is achieved by combining B&B algorithm to CPH heuristic. The enumeration is done in a more intelligent way allowing greater values of parameter L, which lead to improve solution quality particularly when tackling large MMKP instances.

We consider Pareto-algebra product operation as a kind of branching, and then an upper bound is computed for each configuration. Based on this upper bound, a configuration can be discarded even if it is feasible or not dominated. The basic algorithm is detailed in figure 3.

Input: MMKP instance consisting of a vector of configuration sets S and a vector of resource capacity b, and L maximum number of partial solutions maintained at each iteration
Output: Z_{best} best solution found

```
// Keep only Pareto points in the configuration sets
1. for all $C_i \in S$ do min($C_i$)
//Generate an initial Solution $Z_{init}$
2. $Z_{init}$ = initialSol()
3. $Z_{best} = Z_{init}$
//Sort vector $S$ in order to consider groups with fewer items first
4. sort($S$)
// Initialize the set of partial solutions $C_{ps}$
5. $C_{ps} = S[1]$; $S = S - S[1]$
// Combining groups one-by-one
6. for all $C_i \in S$ do
// Combine partial solutions with configuration set $C_i$
// (Branching step) and discard from $C_{ps}$ any infeasible
// or dominated configuration
      $C_{ps}$ = ProductSum-Min($C_{ps}$, $C_i$, $b$)
      // Bounding step
      for all configuration $\bar{c}_j \in C_{ps}$ do
            // Compute a bound value
            $Z_{sup}$ = computeBound($\bar{c}_j$)
            if $Z_{sup} + value(\bar{c}_j) \leq Z_{best}$ then
                  // Discard partial solution due to 'Bound'
                  $C_{ps} = C_{ps} \setminus \{\bar{c}_j\}$
            else
                  $Z'$ = rcRound($Z_{sup}$)
                  // Update eventually $Z_{best}$
                  if $Z' + value(\bar{c}_j) > Z_{best}$ then
                        $Z_{best} = Z' + value(\bar{c}_j)$
      // Apply best-first search strategy
      $C_{ps}$ = selectBest($C_{ps}, L$)
7. $Z$ = bestConfiguration($C_{ps}$)
   if $Z > Z_{best}$ then $Z_{best} = Z$
8. return $Z_{best}$
```

Fig. 3. A B&B Pareto-algebraic heuristic

As a first step, the algorithm tries to find Pareto points in each configuration set. Dominated configurations cannot contribute to an optimal solution of the MMKP instance. However, in the publicly available benchmarks used in experimentations, there is practically no dominated items, but in some real case instances, they can be more. In lines 2 and 3, we generate a feasible initial solution which is set to the actually best solution.

The quality of this solution has an influence on the efficiency of B&B. Line 4 concern irregular instances when groups have different numbers of items, in this case groups are sorted on their cardinality r_i, in ascending order, because it is beneficial to consider groups with fewer items first. The fewer items the two configuration sets being combined have, the faster is the product operation and the more accurate is the result after reducing the set of partial solutions. Line 5 does the initialization for the Pareto-product operation. In each iteration, a configuration set C_{ps} of partial solutions is maintained. This set is initialized with the first configuration set in vector S, $S[1]$, which is subsequently removed from the vector of configurations.

The for loop in line 6 implements the combination one-by-one of configuration sets. This combination can be considered as a kind of branching in a classical B&B algorithm; it iterates over the list of remaining configuration sets. In each iteration, it performs four actions. First the dominated and infeasible configurations are discarded by the ProductSum-Min procedure. Second, a nested for-loop implements the bound part of the algorithm. For each configuration, a bound value is computed by solving the correspondent relaxation LP. Although it may seem time consuming, but bounding process can discard more configurations than dominance and, using efficient solver like Cplex [25] which can solve very large LP in a reduced CPU time, gives in experimentations reasonable running time even for large MMKP instances. With this bound denoted Z_{sup}, we can discard the current configuration if $Z_{sup} + value(\bar{c}_j)$ is less than or equal to Z_{best}; $value(\bar{c}_j)$ denotes the objective function value of partial solution $\bar{c}_j$. Otherwise, the configuration remain in C_{ps}. The third action consists of improving eventually the value of Z_{best}. Since it is unusual to get an integer solution after solving the corresponding LP with Cplex, we designed a rounding procedure based on reduced costs (more details are given in section 4.4). This technique can generate high quality feasible solutions leading to an improvement of Z_{best}. The gradual improvement of Z_{best} during the iterative process leads to discard a large number of partial solutions. Fourth, we apply the best-first search strategy by sorting the configuration on their bound value Z_{sup}, in descending order. Then, at most L configurations are selected in the new C_{ps}. The experimentations show that in some cases the number of discarded partial solutions is so important that the number of remaining configurations is less than L. In this case the solution is exact; otherwise the algorithm becomes a heuristic.

At line 7, C_{ps} contains feasible Pareto-optimal configurations that are all candidate solutions for the considered MMKP instance. Any maximal-value configuration among these candidates which is better than Z_{best}, is typically already a good or even optimal solution for the MMKP instance. Finally, the best solution found is returned in line 8.

4.2 A starting solution

The algorithm starts with an initial solution using greedy procedures. A *constructive procedure* (CP) and a *complementary* one (CCP), proposed in [10], are used to construct an initial solution by applying CP and improve it by applying CCP. The first procedure CP operates in a greedy way in order to produce a feasible solution without focalizing on the quality of the obtained solution. Procedure CCP is applied in order to improve the quality of the initial solution (See [10] for more details). In some cases, particularly when solving large MMKP instances, the procedure CP cannot always produce a feasible solution; we use as a substitute to CP, a simple descent procedure based on a swap neighborhood which leads in most cases to feasible solutions. One can also use Cplex MIP solver (for Mixed Integer Programming) in a reduced time to generate an acceptable feasible solution. In the worst case, when any feasible solution can be obtained we set Z_{best} to -1.

4.3 Best-first search strategy

The best-first search strategy keeps the most relevant configurations in the set of partial solutions C_{ps}. This set is sorted on the bound value Z_{sup}, in descending order. Of course, there is no guarantee that optimal solution will be obtained from this set, but when combined with the rounding procedure detailed in the next section, it leads to high quality feasible solutions, discarding thereby more partial solutions in the case of an improvement of Z_{best}.

4.4 Rounding procedure

Final configurations which correspond to feasible solutions to MMKP instance are obtained only at the end of the algorithm. This leads to a non efficient B&B algorithm; the best solution Z_{best} is updated only at the end of the iterative process. In some cases, solutions of LP are integer, but this happens rarely! To overcome this drawback, a rounding technique is introduced to generate feasible solutions during the search process, based on reduced or marginal costs denoted in simplex theory by $c_j - z_j$. First consider the following LP which corresponds to ILP-MMKP continuous relaxation:

$$(LP)\begin{cases} \max Z = \sum_{i=1}^{n}\sum_{j=1}^{r_i} v_{ij}x_{ij} \\ st \sum_{i=1}^{n}\sum_{j=1}^{r_i} w_{ij}^{k}x_{ij} \le C^k,\ \ k \in \{1,\dots,m\} \\ \sum_{j=1}^{r_i} x_{ij} = 1,\ \ i \in \{1,\dots,n\} \\ 0 \le x_{ij} \le 1,\ \ i \in \{1,\dots,n\}, j \in \{1,\dots,r_i\} \end{cases}$$

Note by $x^* = (x_{ij})$, the optimal non integer solution of LP-MMKP. As stated in linear programming theory, the vector x^*consists of basic variables x_{ij}^B and, free variables x_{ij}^F which are all equal to zero. Because x^* is optimal, all reduced costs $(c_j - z_j)$ of free variables are negative meaning that if a free variable is swapped with a basic variable, the objective function value will decrease. Since x^* is a non integer solution, we have:

$$\exists\, i^* \in \{1,\dots,n\}, \qquad \sum_{j=1}^{r_{i^*}} x_{i^*j}^B = 1$$
$$and \qquad (0 < x_{i^*j}^B < 1 \quad \forall j = 1,\dots,r_{i^*})$$

meaning that it exists at least one group with non integer basic variables. It's clear that rounding non integer basic variables $x_{i^*j}^B$ would very likely lead to an infeasible solution. For this, we suggest to set to zero all non integer basic variables, and then select a free variable $x_{i^*j^*}^F$ to be set to 1, and in order to generate high quality feasible solution, the selected free variable $x_{i^*j^*}^F$ is that which has the smallest reduced cost in order to minimize the loss of objective function value. Formally, decision variables for group i^* are modified as follows:

$$\begin{cases} x_{i^*j}^B = 0 & for\ j = 1,\dots r_{i^*} \\ x_{i^*j^*}^F = 1 & for\ j^* = \arg\min_j \{c_j - z_j\} \end{cases}$$

This process can be iterated until a feasible solution is generated or simply return the obtained solution even if it is infeasible. In any case, the generated solution by this procedure is of interest only if it is feasible and has value greater than the best solution Z_{best}.

5 Computational results

The purpose of this section is to experimentally investigate the various aspects of BPH on standard benchmarks. We evaluate the performance of BPH compared to the state-of-the-art best results. The obtained results are also compared to those obtained when running one hour Cplex Solver v12.2 on the same set of instances. Our algorithms were coded in C++ and all experiments were done on a PC with a 2.8 GHz Intel i5 CPU and 3GB of memory.

Table 1. Small to medium size test problem details

Regular instances					
#Inst	n	r_i	m	$\sum_{i=1}^{n} r_i$	Opt
I01	5	5	5	25	173
I02	10	5	5	50	364
I03	15	10	10	150	1602
I04	20	10	10	200	3597
I05	25	10	10	250	3905,7
I06	30	10	10	300	4799.3
Irregular instances					
#Inst	n	r_{max}	m	$\sum_{i=1}^{n} r_i$	Opt
RTI07	10	5	5	23	564
RTI08	20	10	10	109	6576
RTI09	30	10	10	158	7806.2
RTI10	30	20	10	235	7032
RTI11	30	20	20	208	6880
RTI12	40	10	10	241	11564
RTI13	50	10	10	295	10561

Table 2. Large size test problem details

Regular instances					
#Inst	n	r_i	m	$\sum_{i=1}^{n} r_i$	Upper b.
I07	100	10	10	1000	24607.95
I08	150	10	10	1500	36904.41
I09	200	10	10	2000	49193.87
I10	250	10	10	2500	61486.30
I11	300	10	10	3000	73797.74
I12	350	10	10	3500	86100.45
I13	400	10	10	4000	98448.64
Irregular instances					
		r_{ma}			
INST21	100	10	10	565	44315
INST22	100	10	20	538	42076
INST23	100	10	30	541	42763
INST24	100	10	40	584	42252
INST25	100	20	10	871	44201
INST26	100	20	20	842	45011
INST27	200	10	10	1076	87650
INST28	300	10	10	1643	134672
INST29	400	10	10	2223	179245
INST30	500	10	10	2704	214257

5.1 Problem details

The problems we considered are summarized in Tables 1 and 2. We tested a total of 30 instances corresponding to two groups: (i) regular instances with groups containing the same number of items, that is $r = r_i, \ \forall i = 1, \dots, n$ and (ii) irregular instances with heterogeneous groups; we denote by r_{max} the highest number of items in a group, that is $r_{max} = \max\{r_i, i = 1, \dots, n\}$. The instances I01 to I13 are the well studied regular benchmarks defined in [26], and the instances from RTI01 to INST30 are new instances described in [27].

5.2 Various aspects of BPH

In this section, various aspects of BPH are experimented, especially the impact of B&B algorithm and its ability to discard partial solutions more than Pareto-dominance. It is why we compare in Tables 3 and 4 for each instance, the number of solutions discarded by Pareto-dominance and bounding process. The running time is also carefully examined to show the impact of solving a large number of LP during the Pareto-product process. Recall that we use Cplex 12.2 as LP solver and we set parameter L to 100 to maintain 100 partial solutions during each iteration. The final aspect of BPH we experiment is the rounding procedure and its ability to generate feasible solutions which can improve the bounding process. It is why we present in Tables 3 and 4 the value of the solution found in the first iterations of BPH.

Table 3. Experimentation of discarding rate

#Inst	CONFIGURATION DISCARD		
	Pareto	B&B	%
I01	21	68	23/76
I02	28	791	2/98
I03	43	275	13/87
I04	110	117	48/52
I05	0	10	0/100
I06	0	2174	0/100
RTI07	2	18	10/90
RTI08	98	244	29/71
RTI09	7	1090	0.6/99.4
RTI10	57	201	22/78
RTI11	211	172	55/45
RTI12	144	312	31/69
RTI13	127	274	31/69
I07	6	122	4/96
I08	148	97	60/40
I09	112	138	45/55
I10	110	155	41/59
I11	131	176	43/57
I12	127	188	40/60
I13	150	162	48/52
INST21	80	298	21/79
INST22	266	2182	11/89
INST23	736	9025	8/92
INST24	0	797	0/100
INST25	241	181	57/43
INST26	0	1941	0/100
INST27	86	183	32/68
INST28	133	269	33/67
INST29	168	160	51/49
INST30	39	1637	2/98
AVERAGE			25/75

Table 4. First and final solution given by BPH

#Inst	Initial solution	First Sol.		Final solution		T(s)
		Value	%	Value	%	
I01	161	169	2.3	173	0	0,25
I02	341	355	2.4	364	0	1,15
I03	1511	1546	3.5	1601	0.1	1,98
I04	3397	3454	3.9	3557	1.1	3,5
I05	3591.59	3905.7	0	3905.7	0	0.31
I06	4567.9	4798.8	0.0	4798.8	0.0	5.68
RTI07	564	564	0	564	0	0.3
RTI08	-	6164	6.2	6536	0.6	2
RTI09	7461.2	7748.2	0.7	7806.2	0	1.8
RTI10	5519.4	6359.8	9.5	6979.4	0.7	4
RTI11	-	6075	11	6779.4	1.4	5
RTI12	-	-	-	11476	0.7	8
RTI13	-	9793	7.2	10420	1.3	10
I07	23753	24291	1.2	24584	0.0	85
I08	35485	36681	0.6	36773	0.3	213
I09	47685	48978	0.4	49060	0.2	425
I10	59492	59492	3.2	61391	0.1	739
I11	71378	73561	0.3	73645	0.2	1145
I12	83293	85794	0.3	85996	0.1	1713
I13	95141	98183	0.2	98368	0.0	2349
INST21	-	43636	1.5	44070	0.5	57
INST22	-	-	-	41734	0.8	587
INST23	-	-	-	42172	1.3	2051
INST24	-	41776	1.1	41776	1.1	460

Experiments show the limits of Pareto-dominance. In fact, table 3 shows that among all discarded solutions, only 25% of partial solutions have been discarded by Pareto-dominance. In addition, we include in this rate configurations which have been discarded due to infeasibility. In Contrast to Pareto-dominance, the bounding process led to the elimination of a large number of configurations. Furthermore, the running time remains reasonable despite solving a large number of LP. This is due to Cplex solver efficiency when dealing with LPs but also to the ability of bounding process to discard more configurations. The comparative study we present in the next section will also focus on running time. Moreover, the value of the solution obtained in the first iterations of BPH is presented to show the effectiveness of rounding procedure. In fact, Pareto-product based algorithms can find feasible solutions only in the final iteration, when the last group is combined with the set C_{ps} of partial configurations. Due to rounding procedure, high quality feasible solutions have been generated throughout algorithm execution. In most cases, value of the solution found in the first iterations is significantly better than the starting solution generated par CP and CCP algorithms, and in addition obtained in a negligible time. This is particularly useful in real-time applications when the running time is a crucial parameter. On the other hand, the progressive improvement of the best solution Z_{best} by rounding procedure has contributed to improve the bounding process.

5.3 Comparative study

In the final set of experiments, we compare BPH with the state-of-the-art heuristics both for regular and irregular instances. We focus only on medium to large instances. We present in table 5 the main results obtained on the classical large regular benchmarks I07 to I13 since the early Moser's Work on MMKP [3]. Therefore, we review results obtained in [9], [2], [16], [17], [1]. Medium to large irregular benchmarks results are summarized in table 6. These instances have been experimented for the first time in [1]. For both tables, we include results obtained by MIP solver Cplex 12.2 during one hour of running time and also results obtained by a classical genetic algorithm we applied on MMKP instances.

Table 5. Results for large regular benchmarks

#Inst	Moser 1997	Khan 2002	Hifi 2005	Cherfi 2009	Crevits 2012	Shoj.**20'** 2013	Cplex **(1h)**	G.A. Value	G.A. T(s)	BPH Value	BPH T(s)	Upper Bound
I07	23556	23912	24587	24587	24592	24592	24589	24185	440	**24584**	**85**	24607.95*
I08	35373	35979	36877	36894	36888	36886	36886	35826	553	**36773**	**213**	36904.41*
I09	47205	47901	49167	49179	49179	49185	49176	47874	718	**49060**	**425**	49193.87*
I10	58648	59818	61437	61464	61466	61465	61465	59680	577	61391	739	61486.30*
I11	70532	71760	73773	73783	73779	73782	73788	72004	1213	73645	1145	73797.74*
I12	82377	84141	86069	86080	86091	86084	86080	83815	1370	**85996**	1713	86100.45*
I13	94166	96003	98429	98438	98433	98437	98437	95576	1488	98368	2349	98448.64*

Table 6. Results for medium to large irregular benchmarks

#Inst	Shojaei 2013 Value	Shojaei 2013 T(s)	Cplex (1h)	G.A.	B&B Pareto-algebraic Heuristic First Sol.	B&B Pareto-algebraic Heuristic Final Sol.	B&B Pareto-algebraic Heuristic T(s)	Opt/Upper Bound*
RTI10	6096	0.27	Opt	7003	6359.8	**6979.4**	4	7032
RTI11	5449	0.45	Opt	-	**6075**	**6779.4**	5	6880
RTI12	10860	0.12	Opt	-	-	**11476**	8	11564
RTI13	8636	0.20	Opt	-	**9793**	**10420**	10	10561
INST21	44270	3600	44262	43172	43636	44070	57	44315*
INST22	41976	3600	41976	40302	-	41734	587	42076*
INST23	42562	3600	42550	-	-	42172	2051	42763*
INST24	41918	3600	41918	-	41776	41776	460	42252*
INST25	44156	3600	44146	40441	-	44001	71	44201*
INST26	44869	3600	44835	-	44701	44790	638	45011*
INST27	87616	3600	87600	84121	86752	87442	239	87650*
INST28	134634	3600	134610	127960	134114	134288	655	134672*
INST29	179206	3600	179202	172794	178660	178912	1346	179245*
INST30	214198	3600	214198	208424	**214178**	**214178**	4225	214257*

We can draw several conclusions from these results. First, tables 5 and 6 show that BPH results are competitive in terms of quality and running time especially those given in bold. Second, in columns reporting the pure Cplex results with a time budget of one hour, we may conclude that hybrid heuristics outperform pure Cplex when given equal time budgets. Third, the results we obtained with genetic algorithms are disappointing despite the use of several repair operators to deal with infeasible solutions generated by crossover operator. In fact, we are persuaded that in the case of MMKP, any purely heuristic approach is doomed to fail. Hybridization with exact methods is better suited. Finally, note that with BPH heuristic, we have only one parameter to adjust; the parameter L representing the number of partial solutions maintained at each iteration, while in all the state-of-the-art heuristics, there are several parameters to consider, and it is well known that when using approximate algorithms to solve optimization problems, different parameter settings lead to results of variable quality and the configuration of these parameters is a difficult task.

5 Conclusion

We have solved the multiple-choice multi-dimensional knapsack problem using a hybrid algorithm: Branch and bound Pareto-algebraic

Heuristic. The algorithm is mainly based on Pareto-product operation which combines one-by-one all groups of the MMKP instance at hand. Then, a large part of generated partial solutions is discarded either by Pareto-dominance or better by B&B method. A rounding procedure is used to generate high quality feasible solutions during BPH execution, improving the bounding process. Computational results show that BPH yields high quality solutions within a reasonable computing time, and can generate good solutions within a negligible run time due to rounding technique.

References:

[1] Shojaei H., Basten T., Geilen M., and Davoodi A., “A fast and scalable multi-dimensional multiple-choice Knapsack heuristic”, *in ACM Trans. on Design Automation of Electronic Systems (TODAES)*. Volume 18 Issue 4, 2013.

[2] Hifi M., Michrafy M., and Sbihi A, “A reactive local search-based algorithm for the multiple-choice multidimensional knapsack problem”. *Comput. Optim. Appl. 33,* 2-3, 271–285, 2006.

[3] Moser M., Jokanovic D.P., and Shiratori N.. “An algorithm for the multidimensional multiple-choice knapsack problem”. *IEICE Trans. Fund. Electron. Comm. Comput. Sci. 80,* 3, 582–589, 1997.

[4] Geilen M., Basten T., Theelen B.D., and Otten R., “An algebra of Pareto points”. In *Proc. ACSD*. IEEE,88–97, 2005

[5] Geilen M., Basten T., Theelen B.D., and Otten R., “An algebra of Pareto points”. *Fundamenta Informaticae 78,* 1, 35–74. 2007

[6] Khan S., “Quality adaptation in a multisession multimedia system: Model, algorithms and architecture”. Ph.D. thesis, Univ. of Victoria, Victoria, B.C., Canada, 1998.

[7] Sbihi A., “Hybrid methods in combinatorial optimization: Exact algorithms and heuristics”. Ph.D. thesis, Univ. of Paris I, France, 2003.

[8] Sbihi A., “A best first search exact algorithm for the multiple-choice multidimensional knapsack problem”. *J. Comb.Optim. 13,* 4, 337–351, 2007.

[9] Khan S., Li K.F., Manning E.G., and Akbar M.M., “Solving the knapsack problem for adaptive multimedia systems”. *Stud. Inform. Univ. 2,* 1, 157–178, 2002.

[10] Hifi M., Michrafy M., and Sbihi A., “Heuristic algorithms for the multiple-choice multidimensional knapsack problem”. *Operational Research Society 55,* 12, 1323–1332, 2004.

[11] Parra-Hernandez R., and Dimopoulos N.J., “A new heuristic for solving the multichoice multidimensional knapsack problem”. *IEEE Trans. on Systems, Man, and Cybernetics A 35,* 5, 708–717, 2005.

[12] Akbar M.M., Rahman M.S., Kaykobad M., Manning E.G., and Shoja G.C, “Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls”. *Comput. Oper. Res. 33,* 5, 1259–1273, 2006.

[13] Hiremath C.S., and Hill R.R., “New greedy heuristics for the multiple-choice multi-dimensional knapsack problem”. *Int. J. Operational Research 2,* 4, 495–512, 2007.

[14] Cherfi N., and Hifi M., “A column generation method for the multiple-choice multi-dimensional knapsack problem”. *Comput. Optim. Appl. 46*, 1, 51–73.

[15] Cherfi N., and Hifi M., “Hybrid algorithms for the multiple-choice multi-dimensional knapsack problem”. *Int. J.Operational Research 5,* 1, 89–109, 2009.

[16] Cherfi N., “Hybrid algorithms for knapsack problems”. Ph.D. thesis, University of Paris I, France, 2009.

[17] Crevits I., Hanafi S., Mansi R., and Wilbaut C., “Iterative semi-continuous relaxation heuristics for the multiple-choice multidimensional knapsack problem”. *Computers & Operations Research 39*, 32 – 41, 2012.

[18] Shojaei H., Ghamarian A., Basten T., Geilen M., Stuijk S., and Hoes R., “A parameterized compositional multi-dimensional multiple-choice knapsack heuristic for CMP run-time management”. In *Proc. DAC*. ACM, 917–922, 2009.

[19] Godfrey P., Shipley R., and Gryz J., “Maximal vector computation in large data sets”. In *Proc. 31st Int. Conf. on VLDB*. VLDB Endowment, 229–240, 2005.

[20] Preparata F., and Shamos M., “*Computational Geometry – An Introduction*”. Springer. 1985.

[21] Yukish M., “Algorithms to identify Pareto points in multi-dimensional data sets”. Ph.D. thesis, Pennsylvania State University, 2004.

[22] Borzsonyi S., Kossmann D., and Stocker K., “The skyline operator”. In *Proc. IEEE Conf. on Data Engineering*. IEEE, 421–430, 2001.

[23] Kung H., Luccio F., and Preparata F., “On finding the maxima of a set of vectors”. *Journal of the ACM 22*, 469– 476, 1975.

[24] Bentley J., “Multidimensional divide-and-conquer”. *Communications of the ACM 23*, 214 – 229, 1980.

[26] Cplex, "IBM ILOG Cplex". http://www-01.ibm.com/software/websphere/products/optimization/academic-initiative/, 2012.
[26] MMKP benchmarks, "MMKP benchmarks". ftp://cermsem.univparis1.fr/pub/CERMSEM/hifi/MMKP/MMKP.html, 2010.
[27] MMKP benchmarks, "MMKP benchmarks". http://www.es.ele.tue.nl/pareto/mmkp/, 2012.