An iterative pseudo-gap enumeration approach for the Multidimensional Multiple-choice Knapsack Problem

Gao, Chao; Lu, Guanzhou; Yao, Xin; Li, Jinlong

DOI:
10.1016/j.ejor.2016.11.042

An Iterative Pseudo-gap Enumeration Approach for the Multidimensional Multiple-choice Knapsack Problem

Chao Gao, Guanzhou Lu, Xin Yao, Jinlong Li

PII: S0377-2217(16)30967-5
DOI: 10.1016/j.ejor.2016.11.042
Reference: EOR 14121

To appear in: *European Journal of Operational Research*

Received date: 24 May 2016
Revised date: 21 November 2016
Accepted date: 23 November 2016

Please cite this article as: Chao Gao, Guanzhou Lu, Xin Yao, Jinlong Li, An Iterative Pseudo-gap Enumeration Approach for the Multidimensional Multiple-choice Knapsack Problem, *European Journal of Operational Research* (2016), doi: 10.1016/j.ejor.2016.11.042

Highlights

- A new method is proposed for the Multidimensional Multiple-choice Knapsack Problem.
- Pseudo-cuts are derived by a hypothesized pseudo-gap.
- A way to enumerate the pseudo-gap is introduced.
- This new approach was evaluated on 37 classic benchmark instances.
- It discovered 10 new lower bounds, outperforming one state-of-the-art.

An Iterative Pseudo-gap Enumeration Approach for the Multidimensional Multiple-choice Knapsack Problem

Chao Gao[a,c], Guanzhou Lu[a], Xin Yao[a,b], Jinlong Li[a,*]

[a] *USTC-Birmingham Joint Research Institution in Intelligent Computation and Its Applications (UBRI), School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230026, China*

[b] *The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K.*

[c] *Department of Computing Science, University of Alberta, Edmonton, Canada.*

Abstract

The Multidimensional Multiple-choice Knapsack Problem (MMKP) is an important NP-hard combinatorial optimization problem with many applications. We propose a new iterative *pseudo-gap* enumeration approach to solving MMKPs. The core of our algorithm is a family of additional cuts derived from the reduced costs constraint of the nonbasic variables by reference to a *pseudo-gap*. We then introduce a strategy to enumerate the *pseudo-gap* values. Joint with CPLEX, we evaluate our approach on two sets of benchmark instances and compare our results with the best solutions reported by other heuristics in the literature. It discovers 10 new better lower bounds on 37 well-known benchmark instances with a time limit of 1 hour for each instance. We further give direct comparison between our algorithm and one state-of-the-art "reduce and solve" approach on the same machine with the same CPLEX, experimental results show that our algorithm is very competitive, outperforming "reduce and solve" on 18 cases out of 37.

*Corresponding author

Email addresses: cgao3@ualberta.ca (Chao Gao), lrlgz@ustc.edu.cn (Guanzhou Lu), x.yao@cs.bham.ac.uk (Xin Yao), jlli@ustc.edu.cn (Jinlong Li)

Keywords: Integer programming; Heuristics; Multidimensional Multiple-choice Knapsack; Reduced cost constraint

1. Introduction

The Multidimensional Multiple-Choice Knapsack Problem (MMKP) is one of the hardest variants of the Knapsack Problem [1]. It has many real-world applications, such as logistics [2], running time resource management [3], global routing of wiring in circuits [4], web service composition [5] and capital budgeting [6], the strike force asset allocation problem [7], etc.

Suppose there is a set of items N, which is divided into n disjoint subsets, where each item has an m dimensional cost and a profit value, the MMKP asks to select exactly one item from each subset such that the summed cost on each dimension will not exceed the given bound, while maximizing the summed profit. In the literature, the requirement of selecting exactly one item from each subset is commonly named as the *choice-constraint*, the subset of items is referred to as *group*.

More formally, let $\mathbf{x}$ be a zero-one vector where $x_j = 1$ means item with index j is selected, p_j and vector $\mathbf{v_j} = (v_j^1, v_j^2, \ldots, v_j^m)$ are respectively the profit value and cost vector associated with j. The resource bound is given by vector $\mathbf{b} = (b^1, b^2, .., b^m)$, and N_i is the set of items in *group* i. We can formulate the MMKP as a $0-1$ Integer Linear Programming (ILP) problem:

$$\max \sum_{j \in N} p_j x_j \tag{1}$$

$$\text{subject to } \sum_{j \in N} v_j^k x_j \leq b^k, \quad k = 1, ..., m, \tag{2}$$

$$\sum_{j \in N_i} x_j = 1, \quad i = 1, ..., n, \ \cup_{i=1..n} N_i = N, \tag{3}$$

$$x_j \in \{0, 1\}, j = 1, ..., |N|. \tag{4}$$

In this paper, we propose a new approach, namely Iterative Pseudo-Gap Enumeration, for solving MMKPs. Our algorithm starts by obtaining an upper bound from solving the Linear Programming (LP) relaxation, and then by reference to a *pseudo-gap* and a reduced cost constraint, we propose to derive a new family of pseudo cuts that constrain variables from different groups. Finally, we introduce a simple strategy to enumerate the *pseudo-gap* iteratively. Joint with CPLEX to solve the strengthened problem at each iteration, we test our approach on 37 instances from the literature. It updates 10 new lower bounds, given a run-time of 1 hour for each instance, outperforms the state-of-the-art approach in the literature when running on the same machine.

The rest of our paper is organized as follows. In Section 2, we review the related work. In Section 3, we explain our approach in detail. We then present our experimental studies in Section 4. Section 5 concludes this paper.

2. Related work

A number of algorithms have been proposed for tackling MMKP. Exact methods based on Branch and Bound [8, 9, 10] are able to guarantee the

obtained solution to be optimal after the algorithm terminates, however systematic search without heuristics usually requires intractable computation time to obtain high quality solutions for large-scale instances.

It is believed that the first heuristic results were due to Moser et al. [11], who proposed a heuristic algorithm based on Lagrangian Relaxation that starts from building an infeasible solution, then repeatedly permutes to reduce the infeasibility. Their method was improved by Akbar et al. [12]. Khan et al. [13] proposed a heuristic based on the aggressive resource usage, and they claimed that their heuristic performs better than Moser's. However, a guided local search and a reactive local search heuristic both proposed by Hifi et al. [1, 14] were able to outperform Khan and Moser's heuristics. Then, a column generation method proposed by Cherfi and Hifi [15] obtained better results on the benchmark instances used by the previous heuristics. Cherfi and Hifi in [16] later proposed a hybrid algorithm that combines local branching with column generation and a truncated branch-and-bound. Cherfi and Hifi's hybrid algorithm outperformed all previous approaches substantially.

In fact, due to the different real-world application requirements, the approaches for tackling MMKPs can be grouped into two categories. The first ones are fast heuristics that focus on finding feasible solutions at a small computation cost, particularly to meet the requirement of real-time applications. The methods proposed by Ykman-Couvreur et al. [3], Htiouech et al. [17], Parra-Hernandez et al. [18], Xia et al. [19], and Shojaei et al. [20] belong to this route. The second ones pay more efforts on high quality solutions. The iterative relaxation based heuristic introduced by Hanafi et al. [21], a family of iterative semi-continuous relaxation heuristics named ILPH, IMIPH, IIRH and ISCRH proposed by Crévits et al. [22], and another hybrid heuristic by Mansi et al. [23] that consists of a family of cuts to define a reduced problem

and a reformulation procedure are all of this sort.

The most recent approach "reduce and solve" [24] adopts both group fixing and variable fixing to obtain reduced problems, and then solves the reduced problems by the Integer Linear Programming (ILP) solver CPLEX. Based on different enumerating methods, two variants namely PEGF and PERC are actually defined. The "reduce and solve" approach found most of the current best known results on the set of 27 standard benchmark instances and 10 new irregular structure instances introduced by a fully parameterized CPH heuristic based on pareto algebra [25]. The comparison between the "reduce and solve" approaches and CPH [24] over these 37 instances demonstrates that the two variants PEGF and PERC of "reduce and solve" are overall better than CPH.

It is worth noting that recent high solution quality aimed approaches [21, 22, 23, 24] share the similar idea to reduce the problem by proposed pseudo cuts, and then the reduced problem is solved by an ILP solver, namely CPLEX. The key difference of these approaches is their proposed pseudo cuts and how they iteratively adjust their pseudo cuts.

In this paper, we present a new Iterative Pseudo Gap Enumeration (IPGE) approach for the MMKP. We introduce the concept of *pseudo-gap* which serves as a hypothesized gap between the upper bound and lower bound of the original problem. Based on the *pseudo-gap*, we show that a new family of cuts could be derived by the reduced cost constraints [26, 27, 28]. After applying these cuts, the strengthened problem is solved by the ILP solver CPLEX. We further introduce a strategy to enumerate the *pseudo-gap*, thereby realizing an iterative method that converges to an optimal solution after the *pseudo-gap* becomes valid.

To evaluate the effectiveness of IPGE, we conduct experimental studies on

the 37 benchmark instances [24], among which 27 are with regular structures and 10 are with irregular structures, where regular or irregular structure indicates whether all groups of an instance have exactly the same number of items or not. The comparative experiments show that our algorithm competes favorably with the state-of-the-art "reduce and solve" approach. In particular, given a run time of 1 hour for each instance, IPGE is able to report 6 new better lower bounds on the 27 regular structure instances, and 4 on the 10 irregular structure instances, even though the best lower bounds from the literature have been regarded as very high.

3. An Iterative Pseudo Gap Enumeration Approach to the MMKP

IPGE is essentially a two-step iterative procedure. In the first step, a family of pseudo cuts/constraints is derived from the reduced cost constraints with regarding to a *pseudo-gap*. Then the original problem with these pseudo cuts is solved by calling an ILP solver in the second step. In this section, we first show how to generate the pseudo cuts given there is a *pseudo-gap* at hand, after that we present how the *pseudo-gap* is initially defined and adjusted iteratively, finally we give our complete algorithm.

3.1. Definitions

For the convenience of understanding, we introduce some definitions that are consistently used in this paper.

- P is the given MMKP problem instance.
- $\mathbf{x}^*$ denotes an optimal solution to P.
- $LP(P)$ is the Linear Programming Relaxation of P.
- $\bar{\mathbf{x}}$ is the optimal solution to $LP(P)$.

- r_j denotes the reduced cost corresponding to $\bar{x}_j$, i.e., variable with index j.
- $\underline{v}(P)$ and $\bar{v}(P)$ are a lower bound and an upper bound for problem P, respectively.
- Strengthened problem $(P|C)$ denotes the problem instance P after applying a set of constrains (cuts) in C.

With regard to $\bar{\mathbf{x}}$, we further define the following sets:

- $J^0(\bar{\mathbf{x}}) = \{j | j \in N, \bar{x}_j \text{ is nonbasic and } \bar{x}_j = 0\}$ denotes the index set of nonbasic variables with value 0.
- $J^1(\bar{\mathbf{x}}) = \{j | j \in N, \bar{x}_j \text{ is nonbasic and } \bar{x}_j = 1\}$ denotes the index set of nonbasic variables with value 1.
- $J(\bar{\mathbf{x}}) = J^0(\bar{\mathbf{x}}) \cup J^1(\bar{\mathbf{x}})$ includes the indices of all nonbasic variables in $\bar{\mathbf{x}}$.

3.2. The Reduced Cost Constraint

The reduced cost constraint is an important concept in the Multidimensional Knapsack problem (MKP) [27, 28], originally given by Schinzinger and Saunders [26]. It is also applicable to MMKP since MMKP is extended from MKP with an additional choice-constraint.

For both MMKP and MKP, suppose we have known a feasible solution with objective value $\underline{v}(P)$, then to find solutions with better lower bounds, the following constraint needs to be satisfied [27]:

$$\sum_{j \in J^0(\bar{x})} |r_j| x_j + \sum_{j \in J^1(\bar{x})} |r_j|(1 - x_j) \leq \bar{v}(P) - \underline{v}(P) \tag{5}$$

Following Constraint (5), some fixing rules are defined by Proposition 1.

Proposition 1. *$\forall j \in J(\bar{x})$, if $|r_j| > \bar{v}(P) - \underline{v}(P)$, then $x_j^* = \bar{x}_j$.*

The proof is trivial, since for any nonbasic variable x_j with $|r_j| > \bar{v}(P) - \underline{v}(P)$, if we let $x_j = 1 - \bar{x}_j$, Constraint (5) would certainly be violated. Therefore, if we want better lower bounds, Proposition 1 indicates that only those nonbasic variables with an absolute reduced cost not greater than $\bar{v}(P) - \underline{v}(P)$ in $\bar{\mathbf{x}}$ should allow to change their values. Indeed, Proposition 1 has been intensively used in [21, 22, 23, 24] to reduce the problem size by fixing the nonbasic variables with absolute reduced costs greater than the gap between the known upper bound and lower bound.

3.3. Derived Pseudo Cuts

An upper bound $\bar{v}(P)$ could be obtained by solving $LP(P)$. Assume that we have also a *pseudo-gap* named as ζ, we show that a new family pseudo cuts could be further derived following Constraint (5).

We partition $J^0(\bar{\mathbf{x}})$ into $n_0 + 2$ ($n_0 \geq 0$) disjoint subsets, i.e., $J^0(\bar{\mathbf{x}}) = J_0^0(\bar{\mathbf{x}}) \cup J_1^0(\bar{\mathbf{x}}) \cup ... \cup J_{n_0}^0(\bar{\mathbf{x}}) \cup J_{n_0+1}^0(\bar{\mathbf{x}})$. For any $0 \leq k \leq n_0$, subset $J_k^0(\bar{\mathbf{x}})$ has the property that 1) the summed absolute reduced costs of any k variables from $J_k^0(\bar{\mathbf{x}})$ does not exceed ζ, and 2) the sum of any $k+1$ variables' absolute reduced costs is greater than ζ. The special case is $k = 0$, in which situation we say condition 1) is satisfied automatically. $J_{n_0+1}^0(\bar{\mathbf{x}})$ is the residual subset that $J_{n_0+1}^0(\bar{\mathbf{x}}) = J^0(\bar{\mathbf{x}}) - \{J_0^0(\bar{\mathbf{x}}) \cup J_1^0(\bar{\mathbf{x}}) \cup ... \cup J_{n_0}^0(\bar{\mathbf{x}})\}$.

More formally, let $H_l(A)$ be a function that returns all *l-element* subsets of a given set A, where $l \leq |A|$. For any $0 \leq k \leq n_0$, the property of $J_k^0(\bar{\mathbf{x}})$ could be expressed by Formula (6).

$$\forall \mathbf{S} \in H_k(J_k^0(\bar{\mathbf{x}})), \sum_{j \in \mathbf{S}} |r_j| \leq \zeta \text{ and } \forall \mathbf{S} \in H_{k+1}(J_k^0(\bar{\mathbf{x}})), \sum_{j \in \mathbf{S}} |r_j| > \zeta \qquad (6)$$

This property of $J_k^0(\bar{\mathbf{x}})$ implies that at most k variables could be flipped from 0 to 1, since any $k+1$ variables' summed absolute reduced costs is greater than ζ, violating Constraint (5).

So, it is obvious that a set of cuts could be derived from Formula (6):

$$\sum_{j \in J_k^0(\bar{\mathbf{x}})} x_j \leq k, \qquad k = 0, 1, \ldots, n_0 \tag{cuts-0}$$

Analogously, we partition $J^1(\bar{\mathbf{x}})$ into $n_1 + 2$ ($n_1 \geq 0$) disjoint subsets, i.e., $J^1(\bar{\mathbf{x}}) = J_0^1(\bar{\mathbf{x}}) \cup J_1^1(\bar{\mathbf{x}}) \cup \ldots \cup J_{n_1}^1(\bar{\mathbf{x}}) \cup J_{n_1+1}^1(\bar{\mathbf{x}})$, such that

$$\forall \mathbf{S} \in H_k(J_k^1(\bar{\mathbf{x}})), \sum_{j \in \mathbf{S}} |r_j| \leq \zeta \text{ and } \forall \mathbf{S} \in H_{k+1}(J_k^1(\bar{\mathbf{x}})), \sum_{j \in \mathbf{S}} |r_j| > \zeta \tag{7}$$

Formula (7) and Constraint (5) indicate that for any $J_k^1(\bar{\mathbf{x}})$, at most k variables could be flipped from 1 to 0. Therefore, we obtain another set of cuts as follows:

$$\sum_{j \in J_k^1(\bar{\mathbf{x}})} x_j \geq |J_k^1(\bar{\mathbf{x}})| - k, \qquad k = 0, 1, \ldots, n_1 \tag{cuts-1}$$

It's quite different from the choice-constraint, the derived cuts (cuts-0) and (cuts-1) try to restrict those variables from different groups, so we call them Cross Group Cuts (CGCs). The fixing rule defined by Proposition 1 is just the special case of our CGCs (with $n_0 = n_1 = 0$). If $n_0 = n_1 = 0$, both $J^0(\bar{\mathbf{x}})$ and $J^1(\bar{\mathbf{x}})$ are divided into two parts, and variables indicated by $J_0^0(\bar{\mathbf{x}})$ and $J_0^1(\bar{\mathbf{x}})$ are fixed as 0 and 1, respectively. That is, our CGCs are a generalization of Proposition 1, producing more fixing rules implied by Constraint (5).

3.4. Implementation of the CGCs

It is still a question that, for a given ζ, how to efficiently generate all the CGCs. We present the procedure to computing CGCs in Algorithm 1. Note that before calling Algorithm 1, we suppose that the reduced costs for all variables have been computed by solving the $LP(P)$, and we sort the variables in $J^0(\bar{\mathbf{x}})$ and $J^1(\bar{\mathbf{x}})$ in non-increasing order according to their absolute reduce costs.

Algorithm 1: Generating the CGCs

Input: ζ: a pseudo gap value; l: the type of the CGCs, $l = 0, 1$; $J^l(\bar{\mathbf{x}})$: the nonbaisc variables with value equal to l, sorted in non-increasing order according to the absolute value of their reduced costs, *s.t.* $|r_{j_1}| \geq |r_{j_2}| \geq |\ldots \geq |r_{j_{|J^l(\bar{\mathbf{x}})|}}|$;

Output: C_l: the derived cuts-l. Any cut in C_l is represented by the indices of variables occurred in the cut and the bound value.

```
k ← 0;                                   /* k:the subscript of J_k^l(x̄) */
start ← 0;      /* start:the start position of J_k^l(x̄) in J^l(x̄) */
C_l ← ∅;
while start < |J^l(x̄)| − 1 − k do
    for s ← start to |J^l(x̄)| − 1 − k do
        if Σ_{i=s}^{s+k} |r_{j_i}| < ζ then
            break;
        end
    end
    end ← s + k;                  /* set the end position of J_k^l(x̄) */
    if l = 0 then
        C_l ← C_l ∪ (Σ_{i=start}^{end−1} x_{j_i} ≤ k);
    end
    if l = 1 then
        C_l ← C_l ∪ (Σ_{i=start}^{end−1} x_{j_i} ≥ (end − start) − k);
    end
    k ← k + 1;
    start ← end;
end
return C_l;
```

As we can see from Algorithm 1, the most time-consuming part to computing CGCs is the for-loop, which is used to find some continuous subsequences of the decreasing reduced cost sequence of $|r_{j_1}|, |r_{j_2}|, \ldots, |r_{j_{|J^l(\bar{\mathbf{x}})|}}|$. If a subsequence found is denoted by $J^l_k(\bar{\mathbf{x}})$, then the sum of the first k reduced costs in this subsequence, which is certainly one of the biggest sum of k elements in this subsequence, is smaller than ζ, and the sum of the last $k+1$ reduced costs in this subsequence, which is surely one of the smallest sum of $k+1$ elements in this subsequence, is greater than ζ. Algorithm 1 only conducts one reduced cost sequence scan, and then *start* and *end* mark all the subsequences successively. Hence, the time complexity of Algorithm 1 would be concluded as $O(|J^l(\bar{\mathbf{x}})|)$, which is very efficient.

3.5. Iteration Cut

We introduce yet another type of cut in our algorithm, which is used to avoid revisiting the searched space in the successive iterations. So this cut is used from the second iteration and henceforth.

At first, we explain some notations used in the following presentation. Let ζ_t and $(\text{cuts-0})_t$, $(\text{cuts-1})_t$ be the pseudo-gap and the derived CGCs in the t-th iteration, respectively, where $t = 1, 2, \ldots$. The set of feasible solutions defined by the strengthened ILP problem $(P|(\text{cuts-0})_t, (\text{cuts-1})_t)$ is denoted by F^t.

Since the greater ζ is, the looser our CGCs will be. If we set ζ_{t+1} greater than ζ_t, we will have $F^t \subseteq F^{t+1}$, which means a large amount of solutions explored in the t-th iteration do not need to be searched once again, so we want to reduce as many solutions searched as possible in the next iteration. A new constraint, entitled (cut-3), is deduced as below.

For $x_j = 0$ or 1, let n_0^t and n_1^t respectively be the n_0 and n_1 in iteration

t, we have

$$0 \leq \sum_{j \in J_k^0(\bar{\mathbf{x}})} x_j, \ k = 0, \ldots, n_0^t \tag{8}$$

and

$$\sum_{j \in J_k^1(\bar{\mathbf{x}})} x_j \leq |J_k^1(\bar{\mathbf{x}})_t|, \ k = 0, \ldots n_1^t \tag{9}$$

Combine the results of summing Inequalities (8) and (9) with different k values, a new inequality (10) is obtained

$$\sum_{i=0}^{n_1^t} \sum_{j \in J_i^1(\bar{\mathbf{x}})} x_j - \sum_{i=0}^{n_0^t} \sum_{j \in J_i^0(\bar{\mathbf{x}})} x_j \leq \sum_{i=0}^{n_1^t} |J_i^1(\bar{\mathbf{x}})| \tag{10}$$

So, if any constraint in (cuts-0) or (cuts-1) is violated, the left hand side of inequality (10) will be reduced at least by 1. Therefore, a new Constraint (cut-3) would be used to cut the visited solutions which makes the equality in Inequality (10) hold.

$$\sum_{i=0}^{n_1^t} \sum_{j \in J_i^1(\bar{\mathbf{x}})_t} x_j - \sum_{i=0}^{n_0^t} \sum_{j \in J_i^0(\bar{\mathbf{x}})_t} x_j \leq \sum_{i=0}^{n_1^t} |J_i^1(\bar{\mathbf{x}})_t| - 1 \tag{cut-3}$$

It should be noticed that Constraint (cut-3) is very loose in essence, however, since the computation of this cut does not cause computation overhead, it is still better with than without.

3.6. Iteratively Enumerating Pseudo Gap

So far, the only issue unsolved in our IPGE approach is how to set the pseudo-gap ζ and adjust its value iteratively. Ideally, the pseudo-gap ζ should

be equal to the smallest valid gap value, which will result in tight bounds while the optimal solutions of problem P still remain in $(P|C)$. However, we do not know the value of the smallest valid gap, so iteratively enumerating the pseudo gap values is suggested. The sequence of these pseudo gap values that will be used in our algorithm is called the strategy for setting ζ and is denoted by $\mathcal{S}$.

We observe that in MMKPs, the number of nonbasic variables indicated by $J^1(\bar{\mathbf{x}})$ is usually relatively small, but they tend to be critical, because if one of them is fixed to 1, the rest in the same group should be set to 0 due to the choice-constraint. Therefore, in our IPGE approach, the strategy $\mathcal{S}$ is defined as a sorted subsequence of all reduced costs of variables in $J^1(\bar{\mathbf{x}})$. More precisely, Let $\mathcal{S} \triangleq (\zeta_0, \zeta_1, \ldots)$ be an increasing sequence, where $\zeta_t \in \{|r_{j_{|J^1(\bar{\mathbf{x}})|}}|, \ldots, |r_{j_2}|, |r_{j_1}|\}$, subject to $\zeta_{t+1} - \zeta_t > \Delta,\ t \geq 0$, and Δ is a small positive constant used to avoid generating cuts based on very similar ζ values. The size of the strategy $\mathcal{S}$ is bounded by the gap between the known upper bound $\bar{v}(P)$ and the obtained best lower bound $\underline{v}(P)$ due to $\zeta_t \leq \bar{v}(P) - \underline{v}(P)$. To keep our algorithm as simple as possible, ζ_0 is set to the smallest absolute reduced costs of the variables in $J^1(\bar{\mathbf{x}})$. Δ is a tuning parameter which is set to 0.1.

The details of IPGE are illustrated in Algorithm 2. The LP-relaxation of the original MMKP is solved by the LP solver CPLEX at first, and an optimal solution is recorded as $\bar{\mathbf{x}}$, the reduced costs of all variables are stored in the vector $(r_1, r_2, \ldots, r_{|N|})$, and $J^0(\bar{\mathbf{x}})$, $J^1(\bar{\mathbf{x}})$ are calculated. The upper bound $\bar{v}(P)$ is set to the optimal objective value of the LP-relaxation of P, while $\underline{v}(P)$ is initialized with -1, which will be updated to the best solutions found iteratively in Algorithm 2. Once we have $(r_1, r_2, \ldots, r_{|N|})$ and $J^1(\bar{\mathbf{x}})$, we can also initialize $\mathcal{S}$, the strategy of setting the pseudo gap values.

Algorithm 2: Iterative Pseudo Gap Enumeration for MMKP (IPGE)

Input: P: a given MMKP instance; Δ: the minimum difference of ζ values; $TLimit$: time limit to stopping algorithm;

Output: $\mathbf{x}'$: the best solution found, or NIL if no feasible solution found.

Solve $LP(P)$ by CPLEX, obtain $\bar{\mathbf{x}}$ and $r_i, i = 1, 2, \ldots, |N|$;
Compute $J^0(\bar{\mathbf{x}}), J^1(\bar{\mathbf{x}})$;
$\bar{v}(P) \leftarrow \mathbf{p} \cdot \bar{\mathbf{x}}$; `/* p:the profit value vector given in` P `*/`
$\underline{v}(P) \leftarrow -1$;
$\mathbf{x}' \leftarrow$ NIL;
Sort $J^1(\bar{\mathbf{x}}), J^0(\bar{\mathbf{x}})$ respectively in decreasing order according to $|r_j|$;
Generate $\mathcal{S} = \{\zeta_0, \zeta_1, \ldots\}$ based on sorted $J^1(\bar{\mathbf{x}})$;
$t \leftarrow 0$; `/*` t`:iteration count */`
while *TLimit not reached and* $t < |\mathcal{S}|$ **do**
  generate $(\text{cuts-0})_t, (\text{cuts-1})_t$ by Algorithm 1 with ζ_t;
  if $t == 0$ **then**
    $C \leftarrow (\text{cuts-0})_t \cup (\text{cuts-1})_t$;
  else
    $C \leftarrow (\text{cuts-0})_t \cup (\text{cuts-1})_t \cup (\text{cut-3})_0 \cup (\text{cut-3})_1 \cup \ldots \cup (\text{cut-3})_{t-1}$;
  end
  calculate the remaining time limit $T_{remains}$;
  $\mathbf{x^t} \leftarrow$ solve the ILP model $(P|C)$ by CPLEX with $T_{remains}$;
  `/*` x^t`:the best solution found in the` t`-th iteration. */`
  if $\mathbf{p} \cdot \mathbf{x^t} > \underline{v}(P)$ **then**
    $\underline{v}(P) \leftarrow \mathbf{p} \cdot \mathbf{x^t}$;
    update $\mathcal{S}$ by replacing $\{\zeta_k, \zeta_{k+1}, \ldots | \zeta_k > \bar{v}(P) - \underline{v}(P)\}$ with $\zeta_k = \bar{v}(P) - \underline{v}(P)$;
    $\mathbf{x}' \leftarrow \mathbf{x^t}$;
  end
  $t \leftarrow t + 1$;
end
return $\mathbf{x}'$;

In the t-th iteration of Algorithm 2, the CGCs, $(cuts-0)_t$ and $(cuts-1)_t$, are generated by Algorithm 1 with a giving pseudo gap ζ_t, and the iteration cuts, $(cut-3)_k, k=0,1,\ldots,t-1$, are generated in the previous iterations. We note that $(cut-3)$ are used from the second iteration, so in the $t=0$ iteration, no $(cut-3)$ are applied. The union of these cuts, denoted by C, are used to define a strengthened ILP model $(P|C)$, which is solved by the ILP solver CPLEX with the remaining time budget.

As shown in Algorithm 2, another stopping criterion is when all ζ values in $\mathcal{S}$ are visited, which indicates that the returned solution from Algorithm 2 is an optimal solution of P.

In implementation, both the $LP(P)$ and the strengthened IP models are solved by the academic solver CPLEX. In each iteration, we call CPLEX by setting the best found integer solutions as its starting point so as to speed up the computation.

4. Computational Results

4.1. Problem Instances

In order to evaluate our IPGE approach, we execute it on the benchmark instances that have been widely studied in the literature. We use two different sets of benchmark instances. The first set contains 27 regular structure instances with identical group sizes [16, 21, 22, 23, 24, 25] . The second set has 10 irregular structure instances with varying group sizes. These two sets of problem instances are described in Tables 1 and 2, respectively.

Table 1: Details of the 27 regular structure instances. n indicates the number of groups, $|N_i|$ indicates the size of group $i, i = 1 \dots n$, and m is the resource dimension. $\bar{v}(P)$ is the best upper bound found in the literature.

Instance	n	$\|N_i\|$	m	$\bar{v}(P)$	Instance	n	$\|N_i\|$	m	$\bar{v}(P)$
I07	100	10	10	24604.1	INST08	80	10	10	17528.4
I08	150	10	10	36900.6	INST09	80	10	10	17776.1
I09	200	10	10	49190.6	INST10	90	10	10	19333.1
I10	250	10	10	61483.5	INST11	90	10	10	19457.9
I11	300	10	10	73795.6	INST12	100	10	10	21753.4
I12	350	10	10	86098.8	INST13	100	30	10	21590.1
I13	400	10	10	98446.7	INST14	150	30	10	32884.5
INST01	50	10	10	10738	INST15	180	30	10	39172.8
INST02	50	10	10	13598	INST16	200	30	10	43376.4
INST03	60	10	10	10975.4	INST17	250	30	10	54370.2
INST04	70	10	10	14472.2	INST18	280	20	10	60476.1
INST05	75	10	10	17072.7	INST19	300	20	10	64941.1
INST06	75	10	10	16850.7	INST20	350	20	10	75625.4
INST07	80	10	10	16455.0					

+ For INST01 and INST02, the $\bar{v}(P)$ is proved to be the optimal objective value.
+ The first 6 (I01 to I06) instances are ignored in our experiments, because the sizes of these problems are relatively small and usually used to evaluate fast heuristics, and their optima can be easily obtained by an ILP solver such as CPLEX.

Table 2: Details of the 10 irregular structure instances [25]. n indicates the number of groups, $|N_i|_{max}$ indicates the maximum size of groups, and m is the resource dimension. $\bar{v}(P)$ is the best known upper bound found in the literature.

Instance	n	$\|N_i\|_{max}$	m	$\bar{v}(P)$	Instance	n	$\|N_i\|_{max}$	m	$\bar{v}(P)$
INST21	100	10	10	44315	INST26	100	20	20	45011
INST22	100	10	20	42076	INST27	200	10	10	87650
INST23	100	10	30	42763	INST28	300	10	10	134672
INST24	100	10	40	42252	INST29	400	10	10	179245
INST25	100	20	10	44201	INST30	500	10	10	214257

The regular structure instances are available from the website [1]. The optima of instances included in Table 1 are unknown in the literature except for INST01 and INST02. A common feature of these instances in Table 1 is that their dimensionalities are all 10. The number of items in a group ranges

[1] `http://www.info.univ-angers.fr/pub/hao/mmkp.html`

from 10 to 30, number of groups ranges from 50 to 400, and hence the total number of variables scales from 500 to 7000. The best upper bounds are also included in Tables 1 and 2, and most of them were found by PEGF and PERC which are two variants from the "reduce and solve" approach [24].

Table 2 contains the 10 irregular structure instances that are recently introduced by CPH [25], on which a few results were reported in the literature. These instances in Table 2 are available from the website [2], and the optima of them are all unknown. The included upper bounds were reported in [25] by solving the LP relaxation problem. Note that we do not consider those 7 instances (RTI07 to RTI13) [25] used by CPH for the reason that their problem sizes are very small and were only suggested to evaluate real-time heuristics, which is not the subject of this paper.

4.2. Experimental Results

Our IPGE is programmed in C++, compiled using Visual Studio 2010 with CPLEX12.5. Our experiments are carried out on an Intel(R) Core(TM) i3-3220 3.3GHz CPU machine with 4GB RAM, running Windows 7 32Bit system. The same CPLEX12.5 is used to solve the LP relaxation as well as for solving the strengthened problem after applying the *pseudo-cuts* at each iteration. Setting the time limit as 1 hour for each instance, we first compare our computational results with the best solutions reported by other approaches in the literature, then, we give direct comparison between our algorithm and the recent "reduce and solve" approach.

4.2.1. Comparison of Best Solutions Found by Different Approaches

To show the solution qualities we have obtained by running IPGE, we compare our results with the *best* solutions reported by the other well-

[2] `http://www.es.ele.tue.nl/pareto/mmkp`

performing algorithms in the literature. The comparisons on the 27 regular structured instances and 10 irregular instances are shown in Tables 3 and 4, respectively. Since those results except IPGE are all from the literature, we list their detailed running configurations in below:

- CH: A hybrid heuristic proposed by Cherfi and Hifi [16], results were obtained on an UltraSparc10 2.5 GHz CPU with 1200 seconds as the time limit.

- ILPH-IMIPH-IIRH-ISCRH: The iterative semi-continuous approach in which four variants are proposed in [22],i.e., ILPH, IMIPH, IIRH and ISCRH. We combine the best results of each variant here. The results were obtained on a Pentium IV 3.4 Ghz CPU with CPLEX11.2, time limit 3600 seconds each instance.

- MACH1-2-3: Three variants are introduced in [23], they are MACH1, MACH2 and MACH3. We collect the best solutions of each variant which were obtained on a Dell 2.4GHz CPU Machine with CPLEX11.2, time limit roughly 500 seconds per instance.

- PEGF-PERC: The "reduce and solve" approach by Chen and Hao [24], running on an Intel Xeon 2.83GHz E5440 CPU with 2GB RAM and CPLEX12.4. Two variants PEGF and PERC were presented, and best results were obtained by setting the time limit as 3600 seconds.

- CPH: The fully parameterized heuristic [25] was running on Intel 2.8GHz CPU with 12G RAM and CPLEX version 9. When testing on the 27 regular instances, best results were found by parallel version pCPH with 10 processors, time limit was set to 1200 seconds. On the 10 irregular instances, time limit was set to 3600 seconds, best results were obtained

by two variants with *default aggregation* or *scare-aw. aggregation*, by CPH+OptPP.

Table 3: The comparison of best solutions on the 27 regular structured instances found by different algorithms, i.e., CH [16], MACH1-2-3: three MACH variants [23], pCPH: a fully parameterized heuristic with parallel implementation [25], PEGF-PERC: two variants of "reduce and solve" approach [24], ILPH-IMIPH-IIRH-ISCRH: four variants of the iterative semi-continuous approach [22] and our IPGE.

Inst.	BKLB	CH	ILPH-IMIPH-IIRH-ISCRH	MACH1-2-3	PEGF-PERC	pCPH	IPGE
I07	24595	24587	24592	24590	24592	24592	**24595**
I08	36896	36894	36889	36888	36894	36886	36893
I09	49185	49179	49183	49182	**49185**	**49185**	**49187***
I10	61480	61464	61471	**61480**	61478	61465	61479
I11	73791	73780	73784	73789	**73791**	73782	**73791**
I12	86095	86080	86091	86094	**86095**	86084	86094
I13	98445	98433	**98445**	98440	98441	98437	98443
INST01	10738	**10738**	**10738**	**10738**	**10738**	10733	**10738**
INST02	13598	13498	**13598**	**13598**	**13598**	**13598**	**13598**
INST03	10955	10944	10949	10949	10947	**10955**	10952
INST04	14456	14442	14446	**14456**	**14456**	14452	**14456**
INST05	17061	17055	17058	17057	**17061**	17059	**17061**
INST06	16840	16823	16835	16835	**16840**	16830	**16843***
INST07	16444	16440	16440	16442	**16444**	16440	16442
INST08	17514	17510	17511	17508	**17514**	17509	**17521***
INST09	17763	17761	17760	17760	**17763**	17754	**17763**
INST10	19320	19316	**19320**	19316	19316	19316	**19320**
INST11	19449	19441	19446	19441	**19449**	19441	19446
INST12	21741	21732	21738	21738	**21741**	21738	**21742***
INST13	21580	21577	**21580**	21577	21578	21577	**21580**
INST14	32875	32872	32873	32872	**32875**	32872	32873
INST15	39163	39160	39162	39161	39162	39161	**39163**
INST16	43367	43362	43366	43366	**43367**	43363	**43367**
INST17	54363	54360	54361	**54363**	**54363**	54360	**54363**
INST18	60467	60460	**60467**	**60467**	**60467**	60464	**60469***
INST19	64932	64925	**64932**	**64932**	**64932**	**64932**	**64933***
INST20	75618	75612	75613	**75618**	75616	75615	75616
#BKLB		1	7	8	18	4	18
Sum	1018731	1018445	1018648	1018657	1018703	1018600	1018728

[+] All results except those of IPGE are from the literature. BKLB is the best known lower bound. We note that the BKLBs of I07(24595), INST15(39163) were reported by Chen and Hao using CPLEX12.4 [24], the BKLB of I08(36896) was reported by Mansi et al [23] using CPLEX11.2, other BKLBs were found by algorithms in the table excluding IPGE.

[+] For those approaches consisting of two or more variants, we combine the best results of each variant together.

Table 3 summarizes the best lower bounds visited by the recent algorithms CH, ILPH-IMIPH-IIRH-ISCRH, MACH1-2-3, PERC-PEGF and pCPH, along with our computational results by IPGE. The BKLB stands for best known lower bound, boldface indicates the solution value attains or surpasses the

BKLB. We notice that IPGE has improved 6 lower bounds on the 27 regular instances in the literature, as starred.

In terms of solution quality, from Table 3, we can see IPGE has visited or updated 18 BKLBs, and its summed solution is superior to the rest five methods. Following IPGE, the two variants PEGF-PERC attained 18 BKLBs and the summed solution value is larger than the other 4 methods. The combined results of three MACH variants (column MACH1-2-3) are better than the four variants ILPH-IMIPH-IIRH-ISCRH, even though the time limit of MACH was smaller.

Table 4 contains the comparison of CPH, PEGF-PERC and IPGE. Among those 10 instances, PEGF-PERC attained 7 BKLBs, CPH attained 2, while CPLEX12.4 visited 1 [24]. It is easy to notice that the new computational results of IPGE contain 5 new BKLBs, and its summed value is higher than that of PEGF-PERC, although the best result combination of PEGF and PERC has two more better lower bounds than IPGE.

In summary, IPGE has discovered 11 new BKLBs on these 37 instances from the literature. By the comparisons above, we may further conclude that IPGE has obtained very competitive results. Considering that the results in Tables 3 and 4 of PEGF-PERC are the best of two different variants, it is tempting to say that IPGE performs slightly better than PEGF or PERC. However, due to the discrepancies of running configurations, it is far from fair to conclude which algorithm is better or worse by just comparing their best reported solutions. To have more accurate assessment of IPGE as well as give better comparative results, we will run PEGF and PERC on the same computer as IPGE and compare their performance directly in the next section.

Table 4: The comparison of best solutions on the 10 irregular structured instances found by CPH+OptPP: [25], PEGF-PERC: two variants of "reduce and solve" approach [24], and our IPGE.

Instance	BKLB	PEGF-PERC	CPH+OptPP (default or scar.-aw. aggr.)	IPGE
INST21	44280	**44280**	44270	**44284***
INST22	41976	41966	**41976**	41964
INST23	42584	**42584**	42562	42536
INST24	41918	41876	**41918**	**41998***
INST25	44159	**44159**	44156	44156
INST26	44879	**44879**	44869	44869
INST27	87630	**87630**	87616	**87634***
INST28	134648	134642	134634	**134654***
INST29	179228	**179228**	179206	179222
INST30	214230	**214230**	214198	**214242***
#BKLB		7	2	5
SUM	875532	875474	875405	875559

+ All results except those of IPGE are from the literature. BKLB is the best known lower bound, the BKLB of INST28(134648) was reported by Chen and Hao using CPLEX12.4 [24].

4.2.2. Direct Comparison with PEGF and PERC

The PEGF and PERC are two variants of the "reduce and solve" approach proposed by Chen and Hao [24]. In the previous section, we have seen the superior performance of PEGF-PERC and IPGE in terms of solution quality. However, it is still a question whether IPGE would outperform PEGF and PERC when running those algorithms in the same computer with the same CPLEX installed?

To answer such a question precisely, we download and compile the source code from the authors' website [3], compile it with the same compiler and CPLEX as IPGE. We then run PEGF and PERC on the same Intel i3-3220 CPU machine with the same time limit (1 hour each instance). We show the detailed comparative results in in Tables 5 and 6. The solutions by pure CPLEX12.5 on the same machine are also included as a reference of the solution quality of PEGF, PERC and IPGE.

[3] `http://www.info.univ-angers.fr/pub/hao/mmkp.html`

Tables 5 and 6 contain the comparative results of PEGF, PERC, IPGE and CPLEX12.5 running the same computer with the same time limit. For each instance, we use boldface to indicate whose solution is the best among the four. The starred case indicates that solution is better than the best known lower bound (BKLB) from the literature.

In Table 5, IPGE has 12 (#Best) solutions better than PEGF, PERC and CPLEX12.5, while PEGF and PERC each has 4, CPLEX12.5 has 1. In the last row of Table 5, we summarize the total of the reported values, where the value of IPGE is the highest. Both CPLEX12.5 and PEGF have updated 1 BKLB, i.e., I12 and INST04 respectively, while IPGE improved 6 (I09, INST06, INST08, INST12, INST18 and INST19).

Table 6 contains the computational results of IPGE, PEGF, PERC and CPLEX12.5 on the 10 irregular instances. It is easy to see that IPGE has 5 better lower bounds than the other, while PEGF and PERC each has 3 and 2 respectively. CPLEX12.5 failed to visit even one best solution. If comparing with PEGF and PERC separately, we can see that IPEG has 6 and 7 better results respectively. The summed values in the last row shows that the value of PEGF is much better than CPLEX12.5, but surprisingly, the value of PERC is much worse than CPLEX12.5. The summed value of IPGE still dominates all the rest, although the combined best results from PEGF and PERC are very close.

In summary, on the 37 tested instances, IPGE has 18 higher lower bounds than PEGF and PERC, while PEGF combined PERC has 11 better cases than IPGE in total. The summed results also indicate that IPGE has better performance than PEGF and PERC. Therefore, this direct comparison certificates that IPGE is a very competitive MMKP approach in finding high quality solutions. The authors of "reduce and solve" claim that the two

approaches PEGF and PERC have complementary property [24]. Our experiments demonstrate that our IPGE could be a further complement, since IPGE outperformed PEGF and PERC on 18 out of 37 instances.

Table 5: Direct comparison of IPGE, PEGF, PERC and CPLEX on the 27 standard regular instances.

Instance	IPGE	"reduce and solve"		CPLEX12.5
		PEGF	PERC	
I07	**24595**	24593	24592	24592
I08	36893	**36895**	**36895**	36880
I09	**49187***	49185	49185	49173
I10	**61479**	61478	61477	61474
I11	73791	73791	73790	73779
I12	86094	86093	86092	**86096***
I13	98443	**98444**	98438	98435
INST01	10738	10738	10738	10738
INST02	13598	13598	13598	13598
INST03	**10952**	10945	10947	10946
INST04	14456	**14457***	14456	14456
INST05	**17061**	17057	17057	17050
INST06	**16843***	16837	16838	16825
INST07	16442	**16444**	**16444**	16440
INST08	**17521***	17512	17515	17508
INST09	17763	17762	**17764**	17757
INST10	**19320**	19314	19316	19312
INST11	19446	19446	**19449**	19436
INST12	**21742***	21739	21739	21729
INST13	21580	21580	21578	21577
INST14	32873	32873	32873	32871
INST15	39163	39163	39162	39160
INST16	43367	43367	43367	43367
INST17	**54363**	54361	54360	54360
INST18	**60469***	60467	60467	60464
INST19	**64933***	64932	64930	64930
INST20	75616	75616	75614	75610
#Best	12	4	4	1
SUM	1018728	1018687	1018681	1018563

+ All approaches are running on the same Intel i3 3220 CPU machine with 4GB RAM.
+ Time limit is 3600 seconds for each instance.
+ The starred case indicates the newly improved BKLB. The boldface indicates the result is better than all other approaches in the table.

4.2.3. Convergence of IPGE

IPGE adopts a simple strategy that enlarges the *pseudo gap* each iteration, this makes IPGE converges to an optimal solution when ζ becomes a valid gap larger than the gap of an upper bound and lower bound. To better demonstrate the convergence property as well as investigate the performance

Table 6: Direct comparison of IPGE, PEGF, PERC and CPLEX on the 10 irregular instances.

Instance	IPGE	"reduce and solve"		CPLEX12.5
		PEGF	PERC	
INST21	**44284***	44280	44280	44272
INST22	41964	**41970**	41948	41934
INST23	42536	42560	**42464**	42520
INST24	**41998***	41792	41796	41904
INST25	44156	**44159**	44156	44144
INST26	**44869**	44865	44844	44844
INST27	87634*	87630	**87636***	87616
INST28	**134654***	134640	134652	134626
INST29	179222	**179230***	179218	179214
INST30	**214242***	214220	214214	214214
#Best	5	3	2	0
SUM	875559	875346	875208	875288

+ All approaches are running on the same Intel i3 3220 CPU machine with 4GB RAM.
+ TIme limit is 3600 seconds for each instance.
+ The starred case indicates the newly improved BKLB. The boldface indicates the result is better than all other approaches in the table. IPGE's result on INST27 is starred but not with boldface, since 87634 is better than the BKLB (87630) in the literature, however, our running of PERC has yielded an even better solution (87636).

of IPGE in detail, we illustrate the computation process of IPGE on several representative instances, i.e., I07, INST01, INST02, INST21 and INST23.

The detailed computational results of IPGE on INST01, INST02, I07, INST21 and INST23 are illustrated in Figures 1, 2, 3, 4 and 5. The optima of INST01 and INST02 are known in the literature, and it is observed that on both instances, IPGE terminates much earlier than the time limit of 3600 seconds, for the reason that the ζ has become valid, and hence the optimal solution of the strengthened problem is also optimal to the original problem. Note that for INST02, IPGE fails to yield a feasible solution on iteration 0 and 1, this is because that the very small ζ makes the derived pseudo cuts such strict that the *choice constraint* becomes unsatisfiable, hence the LB remains as -1.

Figure 3 demonstrates how IPGE performs on I07. It is easy to see that at iteration 9, within 300 seconds, the solution visited by IPGE is already 24594,

which is very high, compared to the best solutions reported in the literature. Similar result is observed on INST21 too — within 1200 seconds, at iteration 13, the solution of IPGE has surpassed the BKLB (44280), showing the efficiency of IPGE in discovering high quality solutions.

However, it is also observed that the solution of IPGE on INST23 is significant worse than those of PEGF and PERC, which may imply a potential drawback of IPGE at certain circumstance. For a better understanding of such a circumstance, we also present the detailed computation of IPGE on INST23 in Figure 5. One obvious character of Figure 5 is that there is a plateau from iteration 8 to 12 where IPGE's search over 900 seconds is futile. Recalling that the strengthened problem is primarily defined by the pseudo gap threshold ζ, by comparing the ζ value at different iteration, we may conclude that a minimal increase $\Delta = 0.1$ might be insufficient to include new promising regions to the next iteration for this instance. It seems a more intelligent *enumerate strategy* that taking into account the distribution of the values in $J^1(\bar{\mathbf{x}})$ might be able to help alleviate this problem, and further improve the efficiency of IPGE. We further note that for instances I07, INST21 and INST23, the solution statuses for the corresponding strengthened problems are all "optimal" except the last iteration. In the final iteration, IPGE terminates because it has reached the indicated time limit, which is a normal stopping condition for IPGE when solving large scale instances.

5. Conclusions and Future Work

We have presented a new approach for solving MMKPs namely IPGE. The key idea of IPGE is to use a family of pseudo-cuts to further strengthen the problem. Since the effectiveness of the generated cuts is directly related to the choices of the *pseudo-gap*, we then introduced a simple strategy that

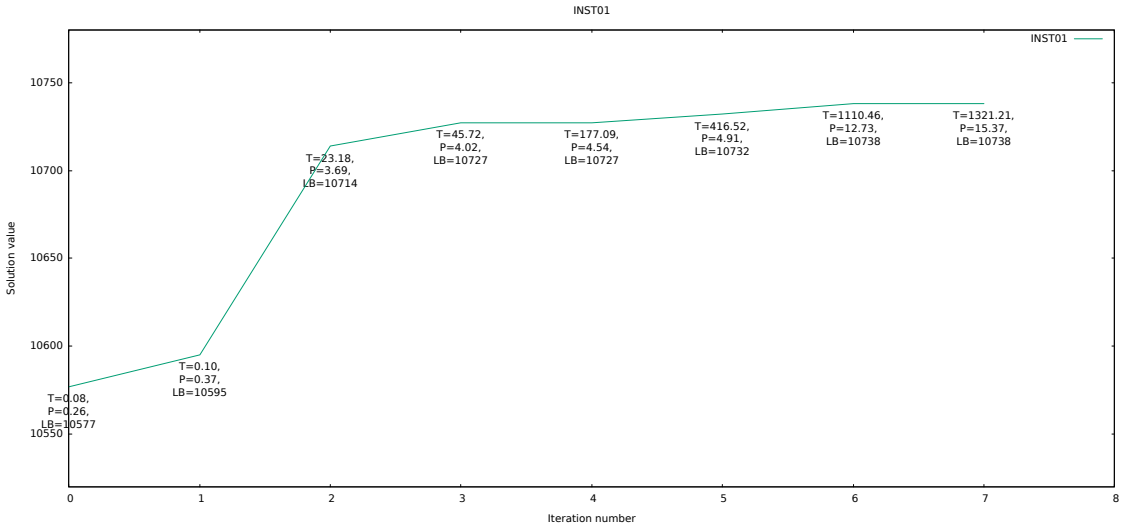

Figure 1: Detailed computational results of IPGE on INST01. T, P and LB in the figure represent respectively the cpu time, pseudo-gap value ζ as well as the discovered lower bound, at the corresponding iteration.

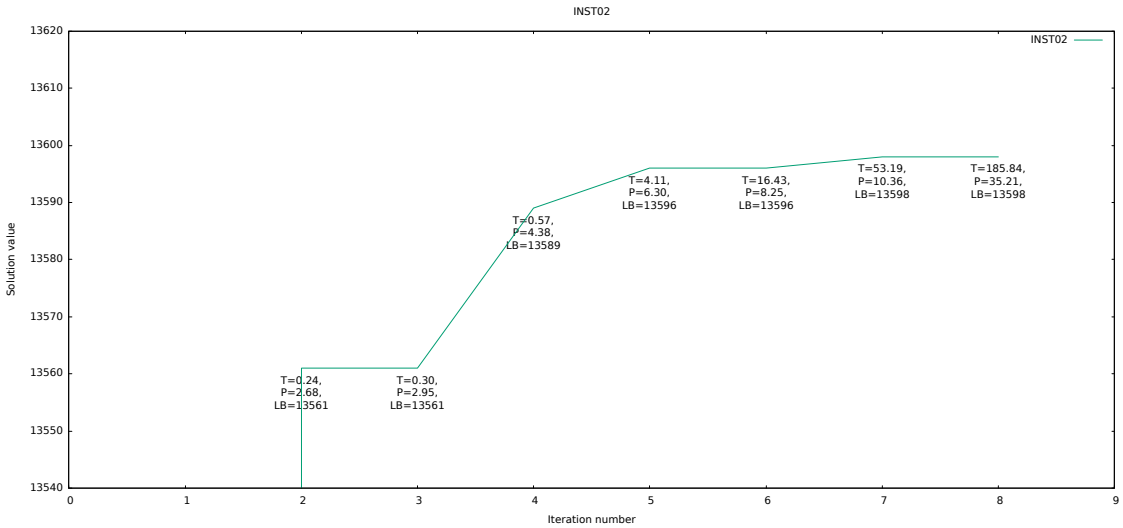

Figure 2: Detailed computational results of IPGE on INST02. T, P and LB in the figure represent respectively the cpu time, pseudo-gap value ζ as well as the discovered lower bound, at the corresponding iteration.

enumerates the *pseudo-gap* values for the next value in $J^1(\bar{\mathbf{x}})$ that has a minimum Δ increase, where Δ is set to 0.1 throughout our experiments.

Our approach has been evaluated on 37 benchmark instances with two

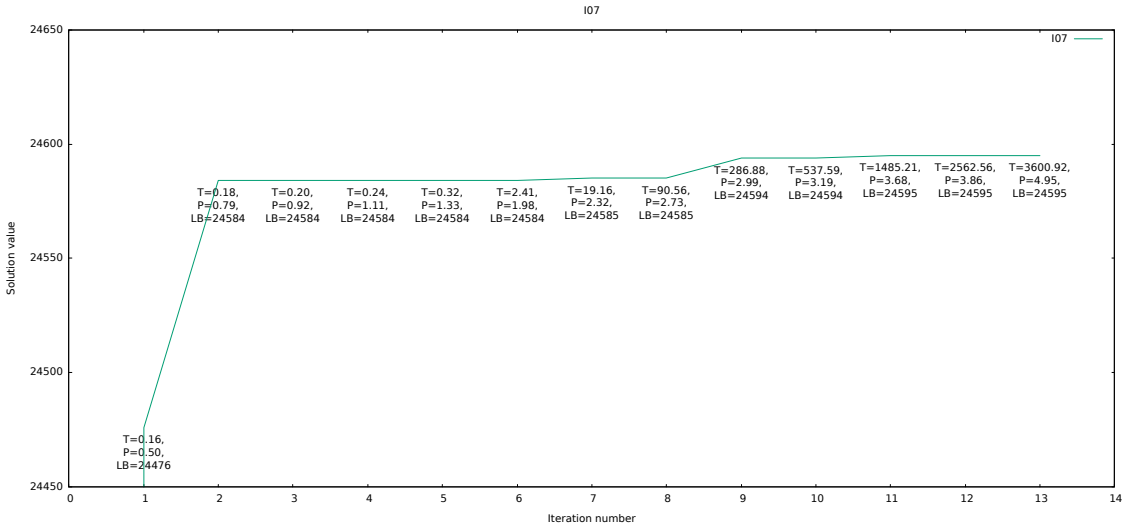

Figure 3: Detailed computational results of IPGE on I07. T, P and LB in the figure represent respectively the cpu time, pseudo-gap value ζ as well as the discovered lower bound, at the corresponding iteration.

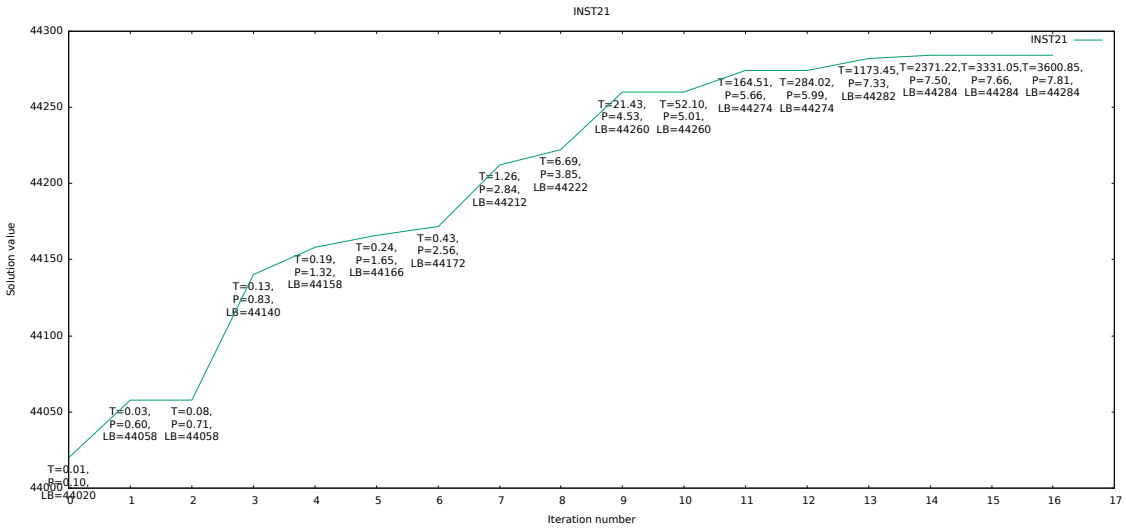

Figure 4: Detailed computational results of IPGE on INST21. T, P and LB in the figure represent respectively the cpu time, pseudo-gap value ζ as well as the discovered lower bound, at the corresponding iteration.

different characteristics. The experimental results indicate that IPGE could outperform the state-of-the-art "reduce and solve" approach on 18 cases. Moreover, IPGE improves 10 new lower bounds, even though the qualities of

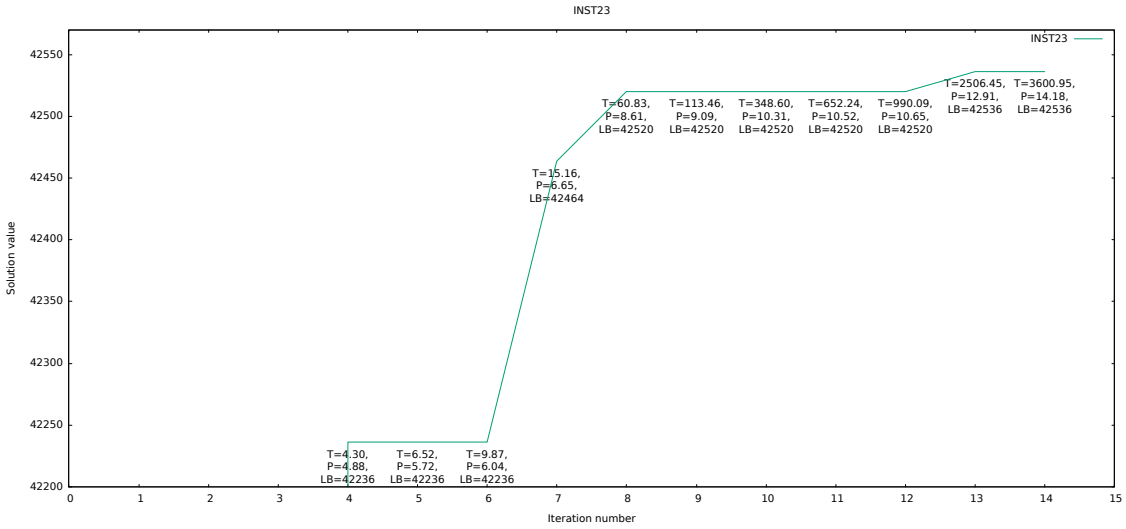

Figure 5: Detailed computational results of IPGE on INST23. T, P and LB in the figure represent respectively the cpu time, pseudo-gap value ζ as well as the discovered lower bound, at the corresponding iteration.

the best known solutions from the literature have been regarded as very high. Our result confirms that a proper integration of effective heuristics with a state-of-the-art ILP solver is a promising approach to solving MMKPs.

In spite of the good performance of IPGE, future work are needed in the following directions. First, we have not yet investigate other *enumeration strategies* other than the simple one proposed, more intelligent enumeration strategies are worth of trying, which may help enhance IPGE's performance on instances like INST23. Second, our iteration cut is in essence loose, perhaps better heuristic cuts or exact cuts exist, and if so, should be identified and added so as to further eliminate the regions visited in the previous iterations. Third, our experimental results have shown that the empirical performances of IPGE, PEGF and PERC seem to be complementary, however, it is still not very clear what kind of pseudo-cuts are most effective on which circumstances, the relation between the effectiveness of heuristic cuts and problem instance features remains as another future research task ahead.

Acknowledgment

We thank Yuning Chen and Jin-Kao Hao for answering questions about their solvers PEGF and PERC. We are grateful to the anonymous reviewers for their constructive comments, which have been very helpful for us in improving the quality and presentation of this paper. This research work was supported by the National Natural Science Foundation of China under Grants 61573328 and Grants 61329302. Xin Yao was supported by a Royal Society Wolfson Research Merit Award.

References

[1] M. Hifi, A. Sbihi, Heuristic algorithms for the multiple-choice multidimensional knapsack problem, Journal of the Operational Research Society 55 (12) (2004) 1323–1332(10).

[2] C. Basnet, J. Wilson, Heuristics for determining the number of warehouses for storing noncompatible products, International Transactions in Operational Research 12 (5) (2005) 527–538.

[3] C. Ykman-Couvreur, V. Nollet, F. Catthoor, H. Corporaal, Fast multi-dimension multi-choice knapsack heuristic for mp-soc run-time management, in: In Proc. SoC06, 2006, pp. 1 – 4.

[4] H. Shojaei, T. H. Wu, A. Davoodi, T. Basten, A pareto-algebraic framework for signal power optimization in global routing, in: Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on, 2010, pp. 407 – 412.

[5] T. Yu, Y. Zhang, K.-J. Lin, Efficient algorithms for web services selection with end-to-end qos constraints, ACM Trans. Web 1 (1) (2007) 1–26.

[6] D. Pisinger, Budgeting with bounded multiple-choice constraints, European Journal of Operational Research 129 (3) (2001) 471–480.

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

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

[9] M. S. Khan, Quality adaptation in a multisession multimedia system: Model, algorithms and architecture, Canada: Department of Electronical and Computer Engineering, University of Victoria. Thesis (PhD).

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

[11] M. Moser, D. P. Jokanovic, N. Shiratori, An algorithm for the multidimensional multiple-choice knapsack problem, Ieice Transactions on Fundamentals of Electronics Communications & Computer Sciences 80 (3) (1997) 582–589.

[12] M. M. Akbar, E. G. Manning, G. C. Shoja, S. Khan, Heuristic solutions for the multiple-choice multi-dimension knapsack problem, in: Proceedings of the International Conference on Computational Science-Part II, ICCS '01, Springer-Verlag, London, UK, UK, 2001, pp. 659–668.

[13] S. Khan, K. F. Li, E. G. Manning, M. M. Akbar, Solving the knapsack problem for adaptive multimedia systems., Studia Informatica An International Journal Special Issue on Cutting Packing & Knapsacking Problems 2 (4) (2002) 157–178.

[14] M. Hifi, M. Michrafy, A. Sbihi, A reactive local search-based algorithm for the multiple-choice multi-dimensional knapsack problem, Computational Optimization & Applications 33 (2-3) (2006) 271–285.

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

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

[17] S. Htiouech, S. Bouamama, R. Attia, Using surrogate information to solve the multidimensional multi-choice knapsack problem, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 2102–2107.

[18] R. Parra-Hernandez, N. J. Dimopoulos, A new heuristic for solving the multichoice multidimensional knapsack problem, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 35 (5) (2005) 708–717.

[19] Y. Xia, C. Gao, J. Li, A stochastic local search heuristic for the multidimensional multiple-choice knapsack problem, in: Bio-Inspired Computing-Theories and Applications, Springer, 2015, pp. 513–522.

[20] H. Shojaei, A. H. Ghamarian, T. Basten, M. Geilen, A parameterized compositional multi-dimensional multiple-choice knapsack heuristic for cmp run-time management, in: Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE, 2009, pp. 917–922.

[21] S. Hanafi, R. Mansi, C. Wilbaut, Iterative relaxation-based heuristics for the multiple-choice multidimensional knapsack problem, in: Hybrid Metaheuristics, Springer, 2009, pp. 73–83.

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

[23] R. Mansi, C. Alves, J. Valério de Carvalho, S. Hanafi, A hybrid heuristic for the multiple choice multidimensional knapsack problem, Engineering Optimization 45 (8) (2013) 983–1004.

[24] Y. Chen, J.-K. Hao, A reduce and solve approach for the multiple-choice multidimensional knapsack problem, European Journal of Operational Research 239 (2) (2014) 313 – 322.

[25] H. Shojaei, T. Basten, M. Geilen, A. Davoodi, A fast and scalable multidimensional multiple-choice knapsack heuristic, ACM Trans. Des. Autom. Electron. Syst. 18 (4) (2013) 51:1–51:32.

[26] R. M. Saunders, R. Schinzinger, A shrinking boundary algorithm for discrete system models, IEEE Trans. Systems Science and Cybernetics 6 (2) (1970) 133–140.

[27] Y. Vimont, S. Boussier, M. Vasquez, Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem, Journal of Combinatorial Optimization 15 (2) (2008) 165–178.

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