A Hybrid Algorithm for Solving the Multiple Knapsack Problem with Setup

Samah Boukhari, Isma Dahmani, Mhand Hifi

A Hybrid Algorithm for Solving the Multiple Knapsack Problem with Setup

Samah Boukhari[1], Isma Dahmani[2], Mhand Hifi[3]

[1] LaROMaD, USTHB, BP 32 El Alia, 16111 Alger, Algérie
[2] AMCD-RO, USTHB, BP 32, El Alia, 16111 Alger, Algerie
{boukhari.samah.ro, dahmani.isma}@gmail.com
[3] EPROAD, UPJV, 7 rue du Moulin Neuf, 80000 Amiens, France
hifi@u-picardie.fr

Mots-clés : *Heuristic, knapsack, optimization.*

1 Introduction

In real-time production, often the manufacturer prefers to act according to the orders received from customers before starting to produce and then to supply. Such a case may be encountered in several companies, where complex products should be provided, specialized machines should be established, electronic circuits must be achieved, etc. On the one hand, the real-time production remains the preferable option for any company, because it may products according to the current demand. On the other hand, it is often necessary that the deadline fixed by the customer remains compatible enough with the necessary production time. Some customers prefer to pay a high price for a reduced production time, while others prefer to wait for a reduced price (for more details, the reader can refer to Yanchun [6]). Therefore,

— one can assume that orders are distributed in classes such that each class contains orders,
— the production capacity related to the factory at period T is fixed,
— each order is characterized by its cost and its setup time, and
— each family is characterized by both a cost and a setup time.

Hence, such a problem can be modeled as the Multiple Knapsack Problem with Setup (namely MKPS).

The MKPS is a combinatorial optimization problem, where its goal is to maximize the profit of the selected items by adding a negative cost related to the classes containing the aforementioned selected items. The MKPS can be viewed as a variant of the well-known Knapsack Problem (noted KP) and an extension of the Knapsack Problem with Setup (noted KPS, cf. Khemakhem and Chebil [3]). KPS is represented by a set of items divided into a set of classes, where each class is characterized by both fixed cost and fixed capacity while an item can be selected if the class containing the activated item.

The MKPS is a more complex version of KPS, where almost of one knapsack, there are T knapsacks and in this case, the setup cost is related to its corresponding knapsack constraint. As described above, these types of problems can be encountered in many real-world industrial and financial applications, such as order acceptance and production scheduling, aviation security system, resource allocation and transportation. (for more details, the reader can refer to Boukhari *et al.* [1], Chebil and Khemakhem [2], Khemakhem and Chebil [3], McLay [5] and Lahyani *et al.* [4]).

2 Problem definition

An instance of MKPS is characterized by a set of T knapsacks of different capacities $r_t,\ t \in \{1,\ldots,T\}$, and a set of m disjoint families (classes) of items. Each family $i,\ i \in \{1,\ldots,m\}$ consists of n_i items and is characterized by a knapsack-dependent integer setup cost f_{it} and an integer parameter s_i representing its capacity consumption. Each item $j,\ j \in \{1,...,n_i\}$ of a family i is associated with a knapsack-dependent profit p_{ijt} and a capacity consumption w_{ij}. An item can be selected only if the corresponding class is activated and a class can only be setup in one knapsack. Furthermore, activating a class induces a knapsack-dependent setup cost that should be considered both in the objective function and related constraints. The goal of the problem is to maximize selected items from different disjoint classes without violating the capacity constraints.

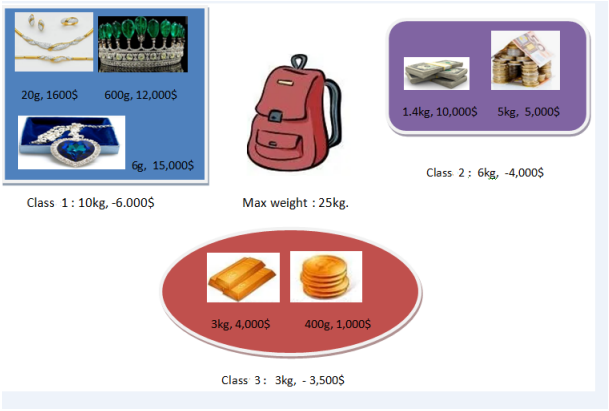

FIG. 1 – An instance of KPS.

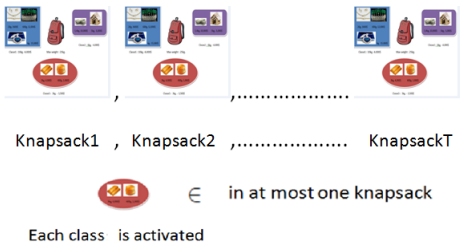

FIG. 2 – An instance of MKPS.

Let x_{ijt} be the decision variable that is equal to 1 if item j of family (class) i is placed in the knapsack t, 0 otherwise, and y_{it} be the setup binary variable which is equal to 1 if family i is placed in the knapsack t, 0 otherwise. Then, the formal description of MKPS (noted P_{MKPS} for the rest of the paper) can be stated as follows :

$$P_{MKPS}: \quad \max \quad \sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{j=1}^{n_i} p_{ijt}x_{ijt} - \sum_{t=1}^{T}\sum_{i=1}^{m} f_{it}y_{it} \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} w_{ij}x_{ijt} + \sum_{i=1}^{m} s_i y_{it} \leq r_t \qquad \forall t \in \{1,...,T\} \tag{2}$$

$$x_{ijt} \leq y_{it} \qquad \forall\ i \in \{1,...m\},\ \ \forall\ j \in \{1,...,n_i\},\ \ \forall t \in \{1,...,T\} \tag{3}$$

$$\sum_{t=1}^{T} y_{it} \leq 1 \qquad \forall\ i \in \{1,...m\} \tag{4}$$

$$x_{ijt} \in \{0,1\},\ y_{it} \in \{0,1\}, \forall\ j \in \{1,...,n_i\},\ \forall\ i \in \{1,...m\},\ \forall t \in \{1,...,T\}, \tag{5}$$

where the objectif function (1) maximizes the profit of selected items minus the fixed setup costs of selected families(classes). Constraints (2) ensure that the weight of selected items in the knapsack t, including all setup capacities related to the activated classes, does not exceed the knapsack capacity r_t. Constraints (3) ensure that an item j is selected in the knapsack t iff it belongs to a family i with a setup in the knapsack t. Finally, constraints (4) indicate that any family i is setup at most in one knapsack.

3 Hybrid algorithm for P_{MKPS}

Let RP_{MKPS} denote the problem provided by relaxing all x_{ijt} variables in P_{MKPS}, providing the relaxed problem RP_{MKPS}. In order to propose a constructive solution procedure, we first

solve RP_{MKPS} for providing a first solution $(\overline{X}, \overline{Y})$; then, fixing step by step the variables $\overline{y_{it}}$ to their binary values in PR_{MKPS}. Second, let α be the number of variables $\overline{y_{it}}$ fixed to one ; thus, let α knapsacks problems whose variables x_{ijt} correspond to the classes $\overline{y_{it}} = 1$. Thus, the optimal solution related to the series of knapsack achieves a complete feasible solution for MKPS. The procedure used, for providing such a solution, can be described as follows :

— Solve RP_{MKPS}(P_{MKPS} with relaxing X variables) using a black-box solver, and let $(\overline{X}, \overline{Y})$ be its optimal solution.
— Let $Y' = \overline{Y}$, such as :
— Set $S = \{i \in I \mid y'_{it} = 1\}$ and $K = \{t \mid y'_{it} = 1\}$, Let $r'_h = r_h - \sum_{i \in S} s_i, h \in K$.
— Let P_{KP_h}, be the knapsack problems with capacity r'_h and $\underline{X_h}$, $h = 1, \ldots, |K|$, be the achieved optimal solution. Set $X' = \cup_{h \in K} \underline{X_h}$ and $Y' = \overline{Y}$, return (X', Y').

$$
\begin{aligned}
P_{KP_h}: \quad \max \quad & \sum_{i \in S} \sum_{j=1}^{n_i} p_{ijh} x_{ijh} \\
\text{s.t.} \quad & \sum_{i \in S} \sum_{j=1}^{n_i} w_{ij} x_{ijh} \leq r'_h \\
& x_{ijh} \in \{0,1\}, \ \forall\, j = 1, ..., n_i, \ \ \forall\, i \in S.
\end{aligned}
$$

4 Highlighting the hybrid algorithm

Herein, we propose to enhance the hybrid method by including a series of valid constraints to the original problem :

$$
\sum_{i=1}^{m} y_{it} \leq d_t, \ \forall t \in \{1, ..., T\} \qquad (6)
$$

In fact, constraint (6) is related to each knapsack t, where the number of classes fixed in the solution must not exceed d_t.

— We optimize the linear relaxation of P_{MKPS} by using the simplex method. In this case, let (X^0, Y^0) be the provided solution, and D be the set related to the values d_t, which is associated to the number of y^0_{it} nonnegative values in the t^{th} knapsack.
— Let RP_{MKPS-V} be the new model obtained by combining RP_{MKPS} and the new additional constraints (6). We optimize RP_{MKPS-V} by using a black-box solver. Let (X', Y') be the provided solution.
— Set $S = \{i \in I \mid y'_{it} = 1\}$ and $K = \{t \mid y'_{it} = 1\}$, Let $r'_h = r_h - \sum_{i \in S} s_i, h \in K$.
— Let P_{KP_h}, be the knapsack problems with capacity r'_h and $\underline{X_h}$, $h = 1, \ldots, |K|$, be the achieved optimal solution. Set $X'' = \cup_{h \in K} \underline{X_h}$ and $Y'' = Y'$, return (X'', Y'').

Algorithm 1 describes the main steps of the enhanced algorithm, where the valid constraints are added to the original problem.

Algorithm 1 Enhancing the Hybrid Method

1: Solve the linear relaxation of P_{MKPS} with the simplex method and let (X^0, Y^0) be its optimal solution.
2: Let Y^1 be the solution extracted from Y^0 such that
3: **if** $\left(y_{it}^0 > 0\right)$ **then**
4: Set $y_{it}^1 = 1$
5: **else**
6: Set $y_{it}^1 = 0$
7: **end if**
8: Set $D = \left\{d_t \mid d_t = \sum_{i=1}^{m} y_{it}^1,\ t = \{1, ..., T\}\right\}$ and, let (X', Y') be the optimal solution of RP_{MKPS-V}.
9: Set $S' = \left\{i \mid y'_{it} = 1\right\}$, $K' = \left\{t \mid y'_{it} = 1\right\}$ and, $r'_h = r_h - \sum_{i \in S} s_i, h \in K'$.
10: Let PL_{KP_h}, $h = 1, \ldots, |K'|$, be the knapsack of capacity r'_h and $\underline{X_h} (h = 1, ...|K'|)$ be an optimal solution.
11: Set $X" = \cup_{h \in K'} \underline{X_h}$ and $Y" = Y'$.
12: **return** $(X", Y")$.

5 Experimental Part

The objective of the computational investigation is to assess the performance of the Hybrid Method with and without adding the valid constraints and, by comparing their provided bounds to the best bounds available in the literature. Because two versions are considered, we then noted the fist version as HM (Hybrid Method) and EHM (Enhanced Hybrid Method) for the second version employing a series of valid constraints. Both versions of the proposed method are evaluated on the set containing 360 instances extracted from Chebil *et al.* [4]. Note that all proposed solution procedures were coded in C and performed on a computer with an Intel Pentium Core i3 with 2 GHz.

#Inst		Hybrid Method		Enhanced Hybrid Method		
T	m	LB_{HM}	t_{HM}	LB_{EHM}	t_{EHM}	t_{Gap}
5	$m \in \{10, 20, 30\}$	338919.033	3.613	**338919.166**	3.576	**-0.110**
10	$m \in \{10, 20, 30\}$	578932.033	21.493	578932.033	27.670	**-7.670**
15	$m \in \{10, 20, 30\}$	799021.733	112.556	**799021.900**	21.710	**-90.850**
20	$m \in \{10, 20, 30\}$	982156.400	186.076	**982156.466**	55.106	**-130.933**
Average		674757.320	80.940	**674757.390**	22.020	**-58.920**

TAB. 1 – Behavior of the hybrid approach with and without adding the valid constraints : ten instances of the group with objects in $[40, 60]$.

#Inst		Hybrid Method		Enhanced Hybrid Method		
T	m	LB_{HM}	t_{HM}	LB_{EHM}	t_{EHM}	t_{Gap}
5	$m \in \{10, 20, 30\}$	815420.566	8.730	815420.566	3.970	**- 4.763**
10	$m \in \{10, 20, 30\}$	1581942.566	37.126	1581942.566	13.140	**-23.986**
15	$m \in \{10, 20, 30\}$	2192423.666	704.760	2192423.666	28.406	**-676.356**
20	$m \in \{10, 20, 30\}$	2641255.066	1254.356	2641255.066	147.906	**-1106.446**
Average		1807760.470	501.240	1807760.470	48.350	**-452.890**

TAB. 2 – Behavior of the hybrid approach with and without adding the valid constraints : ten instances of the group with objects in $[60, 90]$.

Table 1 (resp. Tables 2 and 3) reports the results achieved by both HM and EHM on all

#Inst		Hybrid Method		Enhanced Hybrid Method		
T	m	LB_{HM}	t_{HM}	LB_{EHM}	t_{EHM}	t_{Gap}
5	$m \in \{10, 20, 30\}$	534176.033	13.093	**534176.133**	7.616	**- 5.476**
10	$m \in \{10, 20, 30\}$	1018157.500	158.956	**1018157.600**	16.840	**-142.116**
15	$m \in \{10, 20, 30\}$	1411569.033	1443.953	**1411711.500**	43.085	**-1400.866**
20	$m \in \{10, 20, 30\}$	1816032.333	1881.383	**1816672.133**	370.143	**-1511.343**
Average		1194983.730	874.350	**1195179.340**	109.400	**-764.950**

TAB. 3 – Behavior of the hybrid approach with and without adding the valid constraints : ten instances of the group with objects in $[90, 110]$.

tested instances : LB_{HM} and LB_{EHM} are the average lower bound achieved by HM and EHM, t_{HM} and t_{EHM} are the runtime needed by HM and EHM for reaching the final bound . finaly, t_{Gap} denotes the gap between both versions (the negative value means that EHM performs better than HM and, the value of bold-space means that the method performance better than the other one) . In what follows, we comment on the results of Tables 1, 2 and 3 :

— From Table 1 (the group of instance with the $n_i \in [40, 60]$), one can observe that EHM performs better than HM. Indeed, on the one hand, it is able to provide an global average bound of 674757.39, which is better than that achieved by HM, i.e., 674757.32. On the other hand, adding the valid constraints induces a powerful method since the average runtime limit is considerably decreased (the gap between both versions is equal to -58.92). Finally, more the size of the instance increases, more the average gap (t_{Gap}) increases.

— From Table 2 (the group of instance with $n_i \in [60, 90]$), we can observe that both versions of the method match the same average lower bounds while the average runtime is much smaller for the enhanced version EHM ; in this case, t_{EHM} =48.35 sec while t_{HM} =501.24 sec, which means that EHM is 10.37 faster than HM.

— From Table 3 (the group of instance with $n_i \in [90, 100]$), we can notice that for large-scale instances, EHM definitely performs better than the initial version. Indeed, more the instances are complex to solve, more EHM surpasses the initial version. In this case, t_{EHM} varies from 7.616 sec (line 1, column 6) to 370.143 sec (line 4, column 6) while t_{HM} varies from 13.093 sec (line 1, column 4) to 1881.383 sec (line 4, column 4).

#Inst	Cplex solver	MTS		VND&IP		EHM	
	LB_{Cplex}	LB_{MTS}	Gap_{MTS}	$LB_{\text{VND\&IP}}$	$Gap_{\text{VND\&IP}}$	LB_{EHM}	Gap_{EHM}
$n_i \in [40, 60]$	674541.08	674756.96	-0.032003	674756.15	-0.031883	**674757.39**	**-0.032067**
$n_i \in [60, 90]$	1806945.75	/	/	1807760.33	-0.045080	**1807760.47**	**-0.045088**
$n_i \in [90, 110]$	1193582.25	1195178.33	-0.133722	1195178.90	-0.133770	**1195179.34**	**-0.133807**

TAB. 4 – Computational power of EHM versus MTS, VND&IP and the Cplex solver.

The results provided by EHM are compared to those reported in Adouani *et al.* [**?**] (VND&IP : VND combined with IP), in Lahyani *et al.* [4] (a two phase matheuristic, noted MTS) and the state-of-the-art Cplex solver.

In what follows, we comment on the results reported in Tables 4 and 5 :

— Cplex versus other methods : MTS, VND&IP and EHM dominate the Cplex solver despite several tunings applied to the solver settings.

— EHM versus MTS Tables 4 and 5 : one can observe that EHM outperforms MTS. Indeed, EHM is able to achieve better global average lower bounds than those reached by MTS ; EHM provides two (new) greatest average values for the instances whose objects belong to $[40, 60]$ and $[90, 100]$ than those achieved by MTS.

— EHM versus VND&IP : over all tested instances Tables 4 and 5, it globally matches all the best solution values achieved by VND&IP. The average improvement achieved by

EHM varies from 0.032067 to 0.133807 (Table 4).

#Inst	Cplex solver (S)	MTS (S)	VND&IP (S)	EHM (S)
	t_{Cplex}	t_{MTS}	$t_{\text{VND\&IP}}$	t_{EHM}
$n_i \in [40, 60]$	2940.83	81.57	4.30	22.02
$n_i \in [60, 90]$	2836.00	/	12.69	48.35
$n_i \in [90, 110]$	3135.17	138.17	14.58	108.63

TAB. 5 – EHM runtime versus MTS, VND&IP and the Cplex solver.

6 Conclusion

In this paper the multiple knapsack problem with setup was tackled with a hybrid algorithm. The proposed algorithm was designed for efficiently solving large-scale problem instances. The performance of the proposed algorithm and its enhanced version were evaluated on benchmark instances of the literature, where their provided results are compared to those reached by the Cplex solver and the best methods available in the literature. According to the experimental part, the computational power of the enhanced hybrid algorithm showed its ability to remain very competitive.

Références

[1] Boukhari S, Dahmani I, and Hifi M. Effect of valid cardinality constraints in local branching : The case of the knapsack problem with setup. Information Technology in Industry, vol. 8(3), pp. 8-20, 2020.

[2] Chebil K, and Khemakhem M. A dynamic programming algorithm for the knapsack problem with setup. Computers and Operations Research, 64, 40–50, 2015.

[3] Khemakhem M and Chebil K. A tree search based combination heuristic for the knapsack problem with setup. Computers and Industrial Engineering, 99, 280–286, 2016.

[4] Lahyani R, Chebil K, Khemakhem M, Coelho C, Matheuristics for solving the Multiple Knapsack Problem with Setup, Computers and Industrial Engineering 129 76–89, 2019.

[5] McLay, L. A. Designing aviation security systems : Theory and practice, 2006.

[6] Yanchun Y. Knapsack problems with setup. Dissertation, USA : Auburn University, 2006.