International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:11, 2010

The multi-scenario knapsack problem: an adaptive search algorithm

Mhand Hifi, Hedi Mhalla and Mustapha Michraphy

Abstract—In this paper, we study the multi-scenario knapsack problem, a variant of the well-known NP-Hard single knapsack problem. We investigate the use of an adaptive algorithm for solving heuristically the problem. The used method combines two complementary phases: a size reduction phase and a dynamic 2-opt procedure one. First, the reduction phase applies a polynomial reduction strategy; that is used for reducing the size problem. Second, the adaptive search procedure is applied in order to attain a feasible solution. Finally, the performances of two versions of the proposed algorithm are evaluated on a set of randomly generated instances.

Keywords—combinatorial optimization, max-min optimization, knapsack, heuristics, problem reduction.

I. Introduction

In this paper, the max–min optimization of the multi-scenario knapsack problem (MKP) is investigated. MKP is a variant of the well-known binary knapsack problem (KP), an NP-hard combinatorial optimization problem. An instance of MKP is characterized by a capacity c, a set $\mathcal{N}$ of n items, where each item $j \in \mathcal{N}$ has a fixed weight w_j and the profit p_j^k is evaluated under K different possible configurations ou senarios. The objective of the problem is to determine the subset of items with respect to the capacity constraint c as to maximize the minimal value of a set of linear functions. The mathematical formulation of the MKP can be stated as follows:

$$(MKP)\begin{cases} \text{Maximize} & \min\limits_{1\leq i\leq m} \left\{\sum\limits_{j\in\mathcal{N}} p_j^k x_j\right\} \\ \text{Subject to} & \sum\limits_{j\in\mathcal{N}} w_j x_j \leq c \\ & x_j \in \{0,1\},\ \forall j \in \mathcal{N} \end{cases}$$

where $x_j,\ j \in \mathcal{N}$, denotes the binary decision variable such that $x_j = 1$ if the item j is in the solution set, $x_j = 0$ otherwise. Without any loss of generality, we assume that $\sum_{j\in\mathcal{N}} w_j > c$, and for each item $j \in \mathcal{N}$, $w_j < c$. All the senarios are indexed from by $k = 1, \ldots, K$. We also assume that $w_j,\ p_j^k\ (\forall k = 1, 2, ..., K)$,and c are all nonnegative integers.

The MKP has a wide range of economic and financial applications and, this problem is NP-hard since it is a generalization of the single binary knapsack (Martello and Toth [12], [13]). Few published papers treated the resolution of this problem essentielly to the optimal (Taniguchi *et al* [16], Iida H [8], Kouvelis and Yu [10], Yu G [20],...), and resolved instances with only $|\mathcal{N}| \leq 1000$ and k ≤ 30.

In this paper, we propose an adaptive dynamic search algorithm for the MKP. The proposed algorithm can be viewed as a two-phase approach which combines a reduction instance procedure to a dynamic two-opt procedure. First, and before applying the two main phases of the algorithm, the upper and lower bounds are computed. The first phase of the algorithm consists in reducing the instance size. The second phase of the algorithm tries to determine a feasible solution using an adaptive search algorithm based on a dynamic two-opt procedure. Both phases are complementary and essential to the success of the algorithm, and both can re-iterated to try to improve the given heuristic solution.

The paper is organized as follows. First, Section II presents a brief literature survey of the max-min multi-scenario knapsack problem (MKP). Second, the main steps of the heuristic search algorithm is described in Section III.Third, Section IV evaluates, on a set of a randomly generated instances, the performance of the algorithm is evaluated. Finally, in conclusion, we summarize the main results of the paper.

II. Related literature

A few published papers treated the multi-scenario knapsack problem (MKP). Taniguchi *et al* [16] proposed an approch based a pegging using new upper and lower bounds for the problem, and finally the problem is optimally resolved to the optimal by a branch-and-bound method applyed to the reduced problem. The proposed method is able to resolve instances with only $|\mathcal{N}| \leq 1000$ and k ≤ 30. other approches were developped; essentielly based on a branch-and-bound method ; (Iida H [8], Kouvelis and Yu [10], Yu G [20],...) but these algorithms are limited to resolve instances with very small sizes ($|\mathcal{N}| \leq 90$). The max-min knapsack problems, have been widely studied, another type is the mAx-min allocation problem (Brown [2], Kuno et al. [9], Luss [11], Pang and Yu [14], Tang [15]). A variant of the max- min allocation problem called knapsack sharing problem was also largely studied, and different exact and aproximate approches where developped (Yamada and Futakawa [17], Yamada *et al.* [18], Hifi *et al.* [4], Hifi and Sadfi [3], Hifi *et al.* [6], ...). An other variant of max-min KP, called the *max-min multiple knapsack problem* (M^3KP), where n items to be packed in m knapsacks are considered, was treated in few papers (see Yamada[19], Hifi and Mhalla[5]). Other variants exist and for more details, the reader can refer to Hifi *et al.* [6], and Yamada [19].

M. Hifi is a Professor at Université de Picardie Jules Verne, Equipe ROAD, UR MIS, 33 rue Saint Leu, 80000 Amiens, France (*hifi@u-picardie.fr*).
H. Mhalla is an Associate Professor at Université de Picardie Jules Verne, Equipe ROAD, UR MIS, 33 rue Saint Leu, 80000 Amiens, France. (*hedi.mhalla@u-picardie.fr*)
M. Michraphy, Université de Picardie Jules Verne, Equipe ROAD, UR MIS, 33 rue Saint Leu, 80000 Amiens, France. (*michrafy@fr.ibm.com*)

In this paper, we propose an heuristic algorithm especially for the instances with important sizes, i.e., the number of the items are more than 5000 and t number of scenarios are more 50. The main idea of the paper is to find a first solution with a relatively good quality using a specific lower bound computational procedure. This solution will be improved all along the search process.

III. An adaptive search algorithm

We first define the lower and upper bounds of the multi-scenario knapsack problems. Second, we describe the main steps of the proposed algorithm.

A. Lower bound

The MKP can be viewed as a series of knapsack problems with the same capacity c. So to build a good feasible solution using a greedy algorithm, a specific order will be considered. Let consider $p_j = \min_{1\leq k\leq K} p_j^k$, $\forall\ j \in \mathcal{N}$, and for in what is following items are numbered in the non-increasing order of $r_j = p_j/w_j$. Reposing on this order a greedy solution is built based on the principle of *critical elements*. So, This initial solution is given as follows:

1) let S denotes the binary representation of the solution. Suppose that $s,\ s \leq |\mathcal{N}|$, is a *critical element* of the MKP..
2) Fix all items of the left-critical region (of each class) to "one" and consider that all elements of the right-critical region as "free".

Note that the obtained solution (steps 1 and 2 above) represents a feasible solution for the MKP. Then, the greedy algorithm, noted HEUR, can be described as follows:

Input: An instance of MKP.
Output: An approximate solution.

Starting.
a) Set the initial capacity to zero, i.e. $SumCap = 0$; (cumulate total capacity)
b) Set $j = 1$, $j_{min} = 1$,and for each $k \in \{1, ..., K\}$, $P_i^k = 0$ and $W_i = 0$,
where P_i^k (resp. W_i) is the cumulate profit of scenario k (resp. weight) of items picked.

Iteration.
1) If $SumCap + w_j \leq c$ then set $SumCap = SumCap + w_j$;
2) Set $j_{min} = j_{min} + 1$;
3) Let $min = \min_{1\leq k\leq K}\{P_i^k\}$;
4) Repeat steps 1-3 till $j > |\mathcal{N}|$.

Fig. 1. HEUR: an initial solution for the MKP.

B. Upper bound

Now let consider each scenario as a single knapsack problem. To obtain an upper bound for each subproblem, let consider the relaxed single knapsack problem using an exact polynomial algorithm (Dantzig [**?**], Fayard and Plateau [**?**] and Martello and Toth [12]). In what follow let consider the Dantzig upper bound for the single kanpsack, denoted U_k, with $1 \leq k \leq K$.

Result 3.1: Let $U_{\min}$ be defined as $U_{\min} = \min_{1\leq k\leq K}\{U^k\}$, then $U_{\min}$ is an uupper bound for MKP.

Observe that the upper bound provided by Theorem 3.1 can be obtained quickly by applying a polynomial algorithm to solveat each step a relaxed single knapsack problem, and corresponding to a scenario k, for $1 \leq k \leq K$.

C. Pegging test

To make the algorithm more efficient an instance reduction strategy will be applied. In fact, a pegging test using the previously defined bounds will be very useful to reduce the instance size by fixing some variables either at 0 or 1. Let : N^1 be the set of the elements fixed to 1, N^0 be the set of the elements fixed to 0, and N' be the set of non-fixed elements. Figure 2 describes the main steps of the procedure used for reducing the size of the original problem, noted PP.

Input: *The MKP instance.*
Output: *The sets N'_i, N_i^1 and N_i^0.*

1) Let $Z =$ be the current best solution for the MKP.
2) $\forall j \in \mathcal{N}$ do
   If $UB[x_j = 1] < Z$, set $x_j = 0$ and $j \in N^0$;
   If $UB[x_j = 0] < Z$, set $x_j = 1$ and $j \in N^1$;
3) $N'_i = N_i \setminus \{N_i^1 \cup N_i^1\}$.

Fig. 2. A Pegging Procedure: PP.

D. An adaptive searcht algorithm

To obtain a better solution a two-opt search procedure is locally applied on the set of non fixed items denoted N'.The window width associated to the search zone can be considered as static or dynamic. The static case means that for each considered item by the two-opt procedure, only a fixed number of a neighborhood items can be considered. So, if no neighborhood limits are fixed we are in the case of the ordinary two-opt procedure. The dynamic two-opt procedure, means that the window search is very restricted in the beginning of the algorithm and if no improvement is reached we make this window search larger. To make the search process more efficient; for each item considered by the dynamic two-opt procedure; a short list momory is created and which is representing the best neighborhoods.

It is clear that both of these procedures exit with a feasible solution for the MKP.

If a solution is at hand, the pegging test and the two-opt procedures are applied again, and the algorithm stops if, i) no better solution is reached or ii) if a certain time limit is reached. The performed algorithm considers only the first case, that means that the algorithm exits if the pegging procedure can not fixe more items or if the solution performed by the adaptive two-opt procedure is not better than the current best solution.

Figure 3 describes the main steps of the adaptive heuristic algorithm for the max–min multi-scenario knapsack problem (MKP), noted AH.

Input: an instance of the MKP.
Output: a feasible solution $S(MKP)$ of value $VS(MKP)$.

Phase 1. Initialization step
- Index the scenarios objets from 1 to K;
- Index the set of elements from 1 to $|N|$;

1) Let $S(\mathcal{MKP})$: represents a feasible solution of the problem with value $VS(\mathcal{MKP})$, provided by applying HEUR.
2) Compute the $U_{\min}$.
3) Let $S'(\mathcal{MKP}) = S(\mathcal{MKP})$.

Phase 2. Iterative step
While $(S'(\mathcal{MKP}) > S(\mathcal{MKP}))$ Do

a) $S(\mathcal{MKP}) = S'(\mathcal{MKP})$
b) Apply the *Pegging test procedure* for reducing the problem size, and obtain $N_0, N_1$and Nt;
e) Apply the *Dynamic or static search procedure* for to obtain a better solution $S'(\mathcal{MKP})$;

EndDo
Exit with $S'(\mathcal{MKP})$ realizing the value $VS'(MKP)$.

Fig. 3. An adaptive heuristic algorithm for the MKP (AH).

IV. EXPERIMENTAL PART

In this section we evaluate the performance of the adaptive search algorithm (denoted AH) on a series of randomly generated instances following the problems generator used by Taniguchi *et al*[16]. The algorithm is tested on two set of problem instances with different densities and sizes.. The first set contains the "strongly correlated" instances, and the second set is composed of the "uncorrelated" ones. The optimal solutions of these instances are unknown. To evaluate the behavior of AH, we then compare its solution to the best upper bound computed by the algorithm. The given average runtime of the algorithm is obtained on ten generated instances with The same properties. Both algorithms were tested the static version (with large windows width) and the dynamic ones. Both algorithms were coded in C++ and tested on an UltraSparc-II (450Mhz and with 2Gb of RAM).

Since all the previous algorithms are limited to resolve small sized instances, then we decided to show a comparative study on both medium and large sized instances.

TABLE I
PERFORMANCE OF AH ON THE "STRONGLY CORRELATED" INSTANCES

Groupe	T_{STAT}	T_{DYN}	$\text{GAP}_{\text{STA}\%}$	GAP_{DYN} %
5000.50.C	32	27	0.75	0.76
5000.100.C	34	30	0.50	0.51
10000.50.C	165	163	0.87	0.88
10000.100.C	185	178	0.71	0.71
15000.50.C	228	215	0.93	0.98
15000.100.C	241	233	1.25	1.20
20000.50.C	249	227	0.70	0.70
20000.100.C	276	245	0.31	0.33
25000.50.C	501	453	0.59	0.6
25000.100.C	511	494	0.59	0.59

Table I summarizes the behavior of both exact algorithms on the first set of problem instances. Column indicates the instance's name: we considered the set of instances $\texttt{C}m\texttt{C}$, ..., $\texttt{F}m\texttt{C}$ which correspond to the "strongly correlated" with the number of scenarios k varying in the integer interval $[50, \ldots, 100]$ and the number of items varying in the integer interval $[5000, 25000]$. the instances are denoted as follows: $|N|.k$. Column 2 (resp. colomn 3) contains the runtime (measured in seconds) that needs the static version (resp. the dynamic version) of AH for reaching the performed solution. Column 5 (resp. colomn 3) tallies the relative GAP in percent, between the best solution and the best upper bound given by algorithm This Gap is computed as follows: $(100 * [Z_{best} - U_{\min}]/[Z_{best})$.

From Table I, we can observe that the static version of AH outperforms on almost all the instances if we consider the GAP. both of GAPs are very close, so that means that the produced solutions are very close too. When we consider the CPU, we can clearly see that the dynamic version of the algorithm outperforms the static one on all the instances. Indeed, for all treated "strongly correlated" instances the runtime of the dynamic algorithme is able to realize a significant acceleration.

TABLE II
PERFORMANCE OF AH ON THE "UNCORRELATED" INSTANCES

Groupe	T_{STAT}	T_{DYN}	$\text{GAP}_{\text{STAT}}\%$	GAP_{DYN} %
5000.50	26	24	0.35	0.36
5000.100	24	20	0.35	0.35
10000.50	145	123	0.43	0.43
10000.100	175	151	0.24	0.25
15000.50	200	175	0.13	0.13
15000.100	227	179	0.25	0.25
20000.50	219	187	0.28	0.29
20000.100	227	193	0.34	0.35
25000.50	279	213	0.35	0.36
25000.100	306	249	0.29	0.29

Now let consider, Table II shows the behavior of both versions of the algorithm AH on the second set of instances containing the uncorrelated instances. Table II displays the same informations as for Table I , i.e., the GAPs and the runtime of both versions of the algorithm.

From Table II, we can observe that the same phenomenon is realized. Indeed, we can clearly see that the dynamic version of the algorithm outperforms the static one if we consider the runtime, and the inverse if we consider the GAP.

V. CONCLUSION

We proposed an heuristic algorithm for solving the MKP. First, we showed how bounds can be computed by resolving a series of relaxed single knapsack problems. Second, we presented the procedure of pegging test. Third, we showed how a 2-opt search strategie can be adapted and applied for reaching a high quality solution for the problem. Finally, an experimental part has been presented in which we evaluated the performance of the proposed algorithm on a set of problem instances. On these problem instances, we proved experimentally the effectiveness of the proposed algorithm.

REFERENCES

[1] Brown JR. *The knapsack sharing*, Operations Research, 1979; 27:341-355.

[2] Brown JR. *Solving knapsack sharing with general tradeoff functions*, Mathematical Programming, 1991; 5:55-73.
[3] Hifi M., Sadfi S. *The knapsack sharing problem: an exact algorithm*, Journal of Combinatorial Optimization, 2002;6:35-54.
[4] Hifi M., Sadfi S., Sbihi A. *An efficient algorithm for the knapsack sharing problem*, Computational Optimization and Applications, 2002;23:27-45.
[5] Hifi M., MHalla H. *Variante du knapsack contraint : méthode exacte*, 2010, ROADEF conference.
[6] Hifi M., MHalla H., Sadfi S. *An exact algorithm for the knapsack sharing problem*, 2005, vol. 32, No 5, pp. 1311-1324.
[7] Horowitz E., Sahni S. *Computing partitions with applications to the knapsack problem*, Journal of ACM, 1974;21:277-292.
[8] Iida H. *A note on the max–min 0–1 knapsack problem*, Journal of Combinatorial Optimization 1999;3:89–94.
[9] Kuno T., Konno H., Zemel E. *A linear-time algorithm for solving continuous maximum knapsack problems*, Operations Research Letters, 1991;10:23-26.
[10] Kouvelis P,Yu G. *Robust discrete optimization and its applications*, Dordrecht: Kluwer Academic Publishers; 1997.
[11] Luss H. *Minmax resource allocation problems: optimization and parametric analysis*, European Journal of Operational Research, 1992;60:76-86.
[12] Martello S., Toth P (eds.). Knapsack problems: algorithms and computer implementation, John Wiley and Sons, 1990.
[13] Martello S, Toth P. *Upper bounds et algorithms for hard 0-1 knapsack problems*, Operations Research, 1997;45:768-778.
[14] Pang JS., Yu CS. *A min-max resource allocation problem with substitutions*, European Journal of Operational Research, 1989;41:218-223.
[15] Tang CS. *A max-min allocation problem: its solutions and applications*, Operations Research, 1988;36:359-367.
[16] Taniguchi, F., Yamada, T., Kataoka, S., *Heuristic and exact algorithms for the maxmin optimization of the multi-scenario knapsack problem*, Computers and Operations Research, 2008; 35: 2034–2048.
[17] Yamada T., Futakawa M. *Heuristic and reduction algorithms for the knapsack sharing problem*, Computers and Operations Research, 1997;24:961-967.
[18] Yamada T., Futakawa M., Kataoka S. *Some exact algorithms for the knapsack sharing problem*, European Journal of Operational Research, 1998;106:177-183.
[19] T. Yamada. *Max-Min Optimization of the multiple knapsack problem: An implicit enumeration approch,* E. Kozan, A. Ohuchi eds., "Operations Research/Management Science at Work: Applying Theory in the Asia Pacific Region", Kluwer Academic Publishers, pp. 351-362, 2002.
[20] Yu G. *On the max–min 0–1 knapsack problem with robust optimization applications*, Operations Research 1973;44:407–15.