A Local Branching for the Knapsack Problem with Setup

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

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

Mots-clés : *Knapsack, Local branching, Setup.*

1 Introduction

The Knapsack Problem with Setup (KPS) can be viewed as a variant of the well-known Knapsack Problem (namely KP), where a set of items is considered which is divided into a set of classes. Each class is characterized by both fixed cost and fixed capacity while an item can be selected if the class containing that item is activated. KPS finds its application in many real-world industrial and financial applications, such as order acceptance and production scheduling. The goal of the problem is to maximize the difference between the profits related to the selected items and that related to the fixed costs incurred for setting-up classes without violating the knapsack capacity constraint. The formal description of SKP (noted P_{KPS}) can be written as follows :

$$\max \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} p_{ij}x_{ij} - \sum_{i=1}^{m} f_i y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} w_{ij}x_{ij} + \sum_{i=1}^{m} s_i y_i \leq C \tag{2}$$

$$x_{ij} \leq y_i, \ \forall\ i \in I, \ j = 1, ..., n_i \tag{3}$$

$$x_{ij} \in \{0,1\}, \ y_i \in \{0,1\}, \ \forall\ i \in I, \ j = 1, \ldots, n_i, \tag{4}$$

where C denotes the knapsack capacity, I is the set of m disjoint classes related to all items, n_i is the number of items belonging to the i-th class, p_{ij} and w_{ij} are setup cost and weight respectively of the couple (i, j). Finally f_i and s_i denote the setup cost and the capacity respectively of a selected classe. In this case, x_{ij} is the decision variable that is equal to 1 if item j of class i is selected in the knapsack's solution, 0 otherwise and, y_i denotes the setup decision variable that is equal to 1 if the family i is activated, 0 otherwise.

2 A hybrid method for KPS

A starting solution for KPS can be provided by applying a two-steps procedure :

1. To solve a linear relaxation of the original problem P_{KPS} by using the rounding strategy : to fix step by step the variables y_i to their binary values and round-up (to one) the y_i with fractional value in the relaxed problem. Solve each knapsack problem related to each activated classe (with $y_i = 1$) for getting the values related to x_{ij} variables.
2. To solve a mixed integer relaxation of the original problem by relaxing the variables x_{ij} and, to add a valid constraint to the current relaxation. Hence, each knapsack problem related to each activated classe (with $y_i = 1$) is solved, where a valid constraint is defined as follows : let d denote the number of the decision variables y_i whose values are nonnegative in the resulting knapsack problem ; thus, the following cardinality constraint is added as a valid constraint : $\sum_{i=1}^{m} y_i \leq d$.

In order to adapt LB to KPS (cf. Boukhari *et al.* [2]), we need a starting solution for initializing the first tree, the constraints to use for locally branch on non-searched subspaces and, a state-of-the-art black-box solver for computing the local optimum for each (sub)tree. Let Y be a feasible reference solution of P_{SKP} provided by the constructive method, when the valid cardinality constraint is

added before calling the constructive procedure. Let S_1 (resp. S_0) be the set related to Y containing elements fixed to one (resp. zero), i.e., $S_1 = \{i | i \in I, y'_i = 1\}$ (resp. $S_0 = \{i | i \in I, y'_i = 0\}$). Then, for a given nonnegative integer parameter k, k_{Opt} defines the neighborhood $N(Y', k)$ of the solution Y as the set of the feasible solutions of P_{KPS} satisfying the following additional local branch :

$$\Delta(Y, Y') = \sum_{i \in S_1} (1 - y_i) + \sum_{i \in S_0} y_i \leq k, \tag{5}$$

where the two terms of left-hand side count the number of binary variables flipping their value (with respect to the solution Y) either from 1 to 0 or from 0 to 1, respectively. Herein, as used in Fischetti and Lodi [4], the local branching constraint is applied as a branching criterion within an enumerative scheme for P_{KPS}. Indeed, given the incumbent solution Y', the solution space associated with the current branching node can be partitioned by separately adding the following disjunction :

$$\Delta(Y, Y') \leq k, \quad \text{or} \quad \Delta(Y, Y') \geq k + 1, \tag{6}$$

2.1 Experimental Part

The proposed Local Branching-Based Method (LBBM) was evaluated on a set of 200 instances extracted from the literature (cf., Chebil *et al.* [3]) and Amiri [1]).

TAB. 1 – Performance of LBBM versus Mred and Lag

	Cplex			**Lag**		**Mred**			**LBBM** $k = 7$			**LBBM** $k = 8$		
n_i	z	CPU	Opt	Gap	CPU	Gap	CPU	Opt	Gap	CPU	Opt	Gap	CPU	Opt
500	12522.15	231.21	36	0.1155	4.5	0.0515	0.04	16	0.00053	26.46	39	0.00089	30.19	38
1000	21805.675	587.61	22	0.244	5	0.35969	0.0375	15	0.00408	41.16	34	0.00529	44.56	32
2500	54012.4	996.44	1	0.3365	5	1.06924	0.03	25	0.01336	48.36	32	0.01224	52.2975	31
5000	100699.45	1014.29	12	0.29125	7.75	0.00131	0.01	36	0.0083	37.8	31	0.00411	45.815	26
10000	206632.15	1855.14	8	0.45175	16.5	1.62753	0.02	35	0.00939	40.17	25	0.00411	45.815	26
Av.	79134.365	936.939	79	0.28780	7.75	0.62186	0.03	127	**0.00714**	38.79	**161**	**0.00684**	42.92	**154**

Table 1 reports the results, of the instances, achieved by the four methods tested : Cplex solver, Mred, Lag and LB-BM on all instances. The first column of the table shows the instance's information. Columns from 2 to 4 report the Cplex solver's integer bound (noted z), the runtime limit consumed by the Cplex and the number of optimal solution values matched by the Cplex. Columns 5 and 6 display the average Gap achieved by Lag and its runtime limit (extracted from Amiri [1]). Columns from 7 to 9 tally the average Mred's Gap, its average runtime limit and the number of optimal solution values matched by Mred. Finally, column from 10 to 12 (resp. from 13 to 15) show LB-BM's average Gap with $k = 7$ (resp. $k = 8$), its related average runtime and the number of the optimal solution values matched by the algorithm. According to the Table 1, on can observe that the proposed method remains very competitive, where it outperforms all methods available in the literature.

Références

[1] A. Amiri. A Lagrangean based solution algorithm for the knapsack problem with setups. Expert Systems with Applications, vol. 143, pp. 113-177, 2019.

[2] S. Boukhari, I. Dahmani and M. Hifi. 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.

[3] A. Chebil, R. Lahyani, M. Khemakhem, and L. C. Coelho. Matheuristics for solving the multiple knapsack problem with setup Computers and Industrial Engineering, vol. 129, pp. 76–89, 2019.

[4] M. Fischetti and A.Lodi. Local branching, Mathematical Programming, vol. 98, pp. 23-47, 2003.