Project - Multidimensional Equal-valued Precedence-constrained Knapsack Problem

Luiz Fernando Bueno Rosa - RA: 221197
Lucas Guesser Targino da Silva - RA: 203534

June 3, 2022

1 checklist what to do

- Título
  - denominação do problema
  - metodologia de solução
- Resumo
  - objetivos
  - informações sobre o problema
  - metodologia de solução
  - como será feita a avaliação dos resultados
- Introdução
  - descrição formal do problema (formulação matemática)
  - revisão bibliográfica do problema (e/ou problemas relacionados)
  - metodologias previamente utilizadas
- Metodologia
  - justificativa
  - descrição das técnicas de otimização
  - descrever as técnicas de otimização contextualizando-as ao problema de otimização combinatória proposto
- Avaliação dos Resultados
  - proposta de experimentos
  - descrição das instâncias
  - como será feita a avaliação dos resultados
- Referências Bibliográficas

Abstract

We present a generalization of the classic 0-1 Knapsack Problem. First, the weight of the items are multidimensional vectors and so is the knapsack capacity. Second, the profit of all items is equal to one. Third, the items are required to be added in a certain order. That problem is reffered as *Multidimensional Equal-valued Precedence-constrained Knapsack Problem (MEPKP)*. Three solution approaches are proposed: 1 an Integer Linear Programming for a exact solution; 2 a greedy algorithm for a fast solution; 3 a Greedy Randomized Adaptive Search Procedures (GRASP) and Tabu Search (TS) for a near optimal solution; The three approaches are compared in terms of quality of the solution and computational time with randomly generated instances. For cases in which the exact solution is not available, a Duality Gap is used to compute the optimality gap.

2 Problem Statement

2.1 Input

1. a directed acyclic graph $G = \langle V, E \rangle$;
2. a multi-dimensional weight function $w : V \to \mathbb{Z}_{>}^{n_w}$, where $n_w \in \natural$;

   We will usually write $w_v = w(v)$
3. a maximum capacity of the knapsack $W \in \mathbb{Z}_{>}^{n_w}$;

Besides that, one requires the input to satisfy the constraints below [1], otherwise the problem would be trivial:

1. $w_v \leqslant W$: the weight of each vertex must be smaller than the knapsack capacity;
2. $\sum_{v \in V} w_v \geqslant W$: the weight of all vertices combined must be greater than the knapsack capacity;

2.1.1 Partial Order

Definition 1 (Partial Order on Directed Acyclics Graph)**.** Given a directed acyclic graph $G = \langle V, E \rangle$, we define the set:

$$\prec = \{\langle v, v' \rangle : \text{there is a path from the first to the second}\} \tag{1}$$

and so $\prec$ is a partial order over the set V.

2.2 Output

A subset $S \subseteq V$ of the vertices which satisfy:

$$\sum_{v \in S} w_v \leqslant W \tag{2}$$

$$\forall v \, (v \in S \to \forall v' \, (v' \prec v \to v' \in S)) \tag{3}$$

Equation (2) is the maximum weight constraint, the total weight of all vertices in the solution set S must not be greater than the weight limit W. It is called Capacity-Constraint.

Equation (3)[1] says that, if a v is included in the solution, then all the v' lower than it (in the sense of the partial order $\prec$) must also be included. It is called Precedence-Constraint.

[1] It is a First-order logic expression [2].

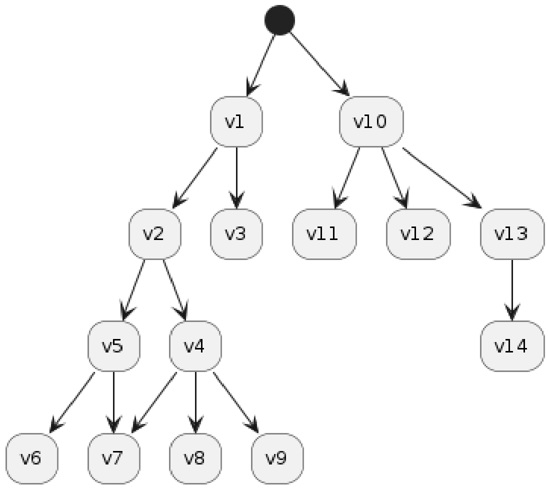

Figure 1: Example of a directed acyclic graph. The black dot indicates the root vertices. For this case, the induced partial order satisfy: $v5 \prec v2$, $v7 \prec v1$, $v14 \prec v10$.

2.3 Objective

Find S that maximizes $|S|$. In other words: find the solution with the maximum number of vertices.

3 Integer Linear Programming Model

3.1 Decision Variables

$$x_v = \begin{cases} 1 & , v \in S \\ 0 & , v \notin S \end{cases} \tag{4}$$

3.2 Mathematical Model

$$\max_{S \subseteq V} \sum_{v \in S} x_v \tag{5}$$

$$s.t. \sum_{v \in S} x_v w_v \leqslant W \tag{6}$$

$$x_v \leqslant x_{v'} \quad \forall v' \prec v \tag{7}$$

$$x \in \{0, 1\}^{n_w} \tag{8}$$

Equation (5) is the objective function: maximize the number of vertices in the solution. Equation (6) is the Capacity-Constraint of Equation (2). Equation (7) is the Precedence-Constraint of Equation (3): if a vertex v is in the solution, then all vertices v' for which there is a path from v must also be in the solution.

4 State of the Art

In [3], the authors proposes a memory based GRASP with restart and a simple tabu search algorithm is proposed to overcome the limitations of local optimality in order to find near optimal solutions. In that paper, numerical tests on benchmark instances demonstrate the effectiveness and efficiency of the proposed methodology which outperform the Mini-Swarm heuristic in terms of the success ratio, relative percentage deviation and computational time.

In [4], the authors reported the implementation of an efficient TS method based on the oscillation strategy and definition of a promising zone, a zone which englobes all factible solutions plus all infactible solutions bordering the infactible solutions, for solving the O-1 MKP which has been tested on standard test problems from [5, 6] and [7]. Optimal solutions were successfully obtained for all instances and the previously best known solutions were improved for five of the last seven instances. These numerical results were claimed to confirm the merit of tabu tunneling approaches to generate solutions of high quality for 0-1 multiknapsack problems. Moreover, these results (like those of [7]) are claimed to establish that the oscillation strategy is efficient to balance the interaction between intensification and diversification strategies of TS.

In [1], the authors used a lagrangean relaxation on the precedence-constraint and the subgradient method to solve the problem faster then use a "pegging" test to guarantee optimality.

4.1 0-1 Knapsack Problem

[8] gives a definition for a 0-1 Knapsack Problem (0-1KP):

$$\begin{aligned} \max \qquad & \sum_{j=1}^{n} p_j x_j \\ \text{subjected to} \qquad & \sum_{j=1}^{n} w_j x_j \leq c \\ & x_j \in \{0,1\} \quad \forall j \in \{1,\ldots,n\} \end{aligned} \tag{9}$$

in which p_j and w_j are know as the profit and the weight of the item j, respectively.

The problem proposed here is a knapsack problem adapted to satisfy two extra constraints: precedence-constrained and multi-dimensional weights. Besides, its profits are all one.

5 Instance Generation

The instances are generated randomly [9], [1], [3]. For that, first the graph is generated, and then the weight of each vertex is chosen. The knapsack capacity is selected so that, on average, X percent of the vertices fit in it. The following subsections analyze each of those aspects.

Consider the parameters:

1. n: number of vertices;
2. K: average number of branches;
3. L: maximum number of leaf vertices;
4. H: the maximum value of an entry of the weight of each vertex;
5. m: fraction of the average number of elements that fit in the knapsack;

5.1 How to Generate the Precedences

The process of generating the precedences is specified in Algorithm 1, which uses Algorithm 2. The Figure 2 has an example of such procedure. The following parameters are used to control the generation:

Algorithm 1 Find-Trees

Require: V: vertices in the 2D plane, K: average number of branches, L: maximum number of leaf vertices

$k \leftarrow$ random number from 1 to K
$\langle R, \mathcal{V} \rangle \leftarrow$ find k clusters in V ▷ R: a set of centers
▷ $\mathcal{V}$: a set with each element being the set vertices of each cluster
$\mathcal{T} \leftarrow \emptyset$
for each pair $r \in R$ and $V' \in \mathcal{V}$ **do**
    if $|V'| \leqslant L$ **then**
        $T \leftarrow$ tree with r as the root node and V' as the leaves
    else
        $T \leftarrow$ tree with r as the root node of the subtree Find-Trees(V', K, L)
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{T\}$
return $\mathcal{T}$

Algorithm 2 Generate-Precedences

Require: n: number of vertices, K: average number of branches, L: maximum number of leaf vertices

$V \leftarrow$ generate n points in the 2D plane randomly
$\mathcal{T} \leftarrow$ Find-Trees(V, K, L)
return $\mathcal{T}$

5.2 How to Generate the Weights

Generate the weights randomly in the interval $[0, H]$.

5.3 How to Generate the Knapsack Capacity

Generate each entry of the knapsack capacity W randomly in the interval $[0, m \cdot n \cdot H]$.

6 How to evaluate the Results

We will generate a table with the result of the experiments in the format below. Graphics are going to be created on demand as we analyze the results. Such results will provide all the information required to see how each method behaves, how different instances impact on each method, how big is the instance they can handle.

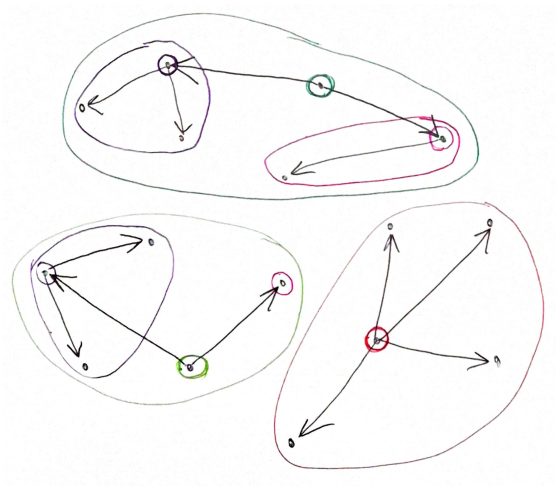

Figure 2: Precedence generation. The root nodes are the green, red and lemon. Red has four leaf vertices. Green has two branches, the pink with one leaf and the purple with two leaves. Lemon has one leaf and one branch with two leaves.

Instance	ILP		greedy		metaheuristic	
	no. items	time	no. items	time	no. items	time
X	10	100	8	14	9	36

Table 1: Results of the methods of solution. The time is given in seconds.

References

[1] Byungjun You and Takeo Yamada. A pegging approach to the precedence-constrained knapsack problem. *European journal of operational research*, 183(2):618–632, 2007.

[2] Cezar A Mortari. *Introdução à lógica*. Unesp, 2001.

[3] Zhen Yang, Guoqing Wang, and Feng Chu. An effective grasp and tabu search for the 0–1 quadratic knapsack problem. *Computers & Operations Research*, 40(5):1176–1185, 2013.

[4] Said Hanafi and Arnaud Freville. An efficient tabu search approach for the 0–1 multidimensional knapsack problem. *European Journal of Operational Research*, 106(2-3):659–675, 1998.

[5] A Freville and G Plateau. *Méthodes heuristiques performantes pour les problèmes en variables 0-1 à plusieurs contraintes en inégalité*. Université de Lille I, UER d'IEEA Informatique, 1982.

[6] Arnaud Freville and Gérard Plateau. An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem. *Discrete applied mathematics*, 49(1-3):189–212, 1994.

[7] Fred Glover and Gary A Kochenberger. Critical event tabu search for multidimensional knapsack problems. In *Meta-heuristics*, pages 407–427. Springer, 1996.

[8] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[9] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.