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. $\sum\limits_{v \in V} w_v \geqslant W$: 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:

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

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

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].

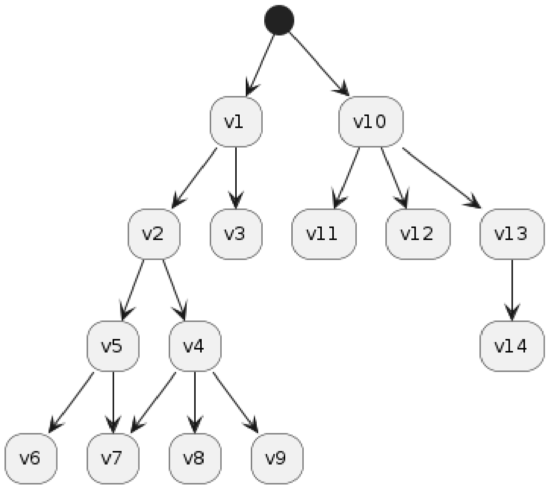

Figure 1: Example of a directed acyclic graph. The black dot indicates the root vertices. For this case, the induced partial order satisfy: $v5 \prec v2$, $v7 \prec v1$, $v14 \prec v10$.

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.

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.