Multiple Disjunctively Constrained Knapsack Problem

Marco Favorito

Abstract

In this document, we will present a problem, *Multiple Disjunctively Constrained Knapsack Problem* (MDCKP), that is a combination of well-known problems: *Multiple 0-1 Knapsack Problem* (MKP) and *Disjunctively Constrained Knapsack Problem* (DCKP). Linear programming formulation is provided, as well as correlations with other Knapsack problem variants. Finally, we describe a (naive) algorithm.

1 Introduction

Multiple Disjunctively Constrained Knapsack Problem (MDCKP) is a combination of two Knapsack-like problems, namely:

- Multiple 0-1 Knapsack Problem (MKP) [1, 2];
- Disjunctively Constrained Knapsack Problem (DCKP) [3]

Informally, it is about how to assign n items to m knapsacks, each of them with limited capacity, in order to maximize the total profit of assigned items, where:

- each item has a weight;
- there are some items that cannot be assigned to the same knapsack, due to intrinsic incompatibility.

2 Linear Programming

More formally, we have n items, each of them with $p_1, \dots, p_n$ profits and $w_1, \dots, w_n$ weights. Each of them can be assigned to only one of the m knapsacks, which have capacities $c_1, \dots, c_m$. Let E be the set of incompatible pairs, such that $(i, j) \in E$ iff items i and j are incompatible. To avoid duplicates, we define $E \subset \{(i, j) | 1 \leq i \neq j \leq n\}$ and assume E to be a reflective relation.

Now follow a LP formulation of the problem:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{j}^{m} \sum_{i}^{n} p_i x_{ij} & \\
\text{subject to} \quad & \sum_{i}^{n} w_i x_{ij} \leq c_j & j \in M = \{1, \dots m\} \\
& \sum_{j}^{m} x_{ij} \leq 1 & i \in N = \{1, \dots n\} \\
& x_{kj} + x_{hj} \leq 1 & (k, h) \in E, j \in M \\
& x_{ij} \in \{0, 1\} & i \in N, j \in M
\end{aligned}
$$

where:

$$
x_{ij} = \begin{cases} 1 & \text{if item } i \text{ is assigned to knapsack } j \\ 0 & \text{otherwise} \end{cases}
$$

Notice: with $E = \emptyset$ and $m = 1$, the problem reduces to the classic 0-1 Knapsack problem. Hence, MDCKP is $\mathcal{NP}$-hard, since KP is $\mathcal{NP}$-hard too.

3 Naive Approximation Algorithm

The simplest approximation algorithm one can think of is the following:

- sort the items by non-increasing profit per weight, that is:

$$
p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_n/w_n
$$

- start to fill up the knapsacks and, if there is not enough capacity or there is some incompatibility with other items, then go to the next knapsack. If no knapsack is available, then leave the item out.
- perform some local search algorithm by defining the neighborhood of a solution as the following:

> Look for a pair of items, one inside and the outer outside the knapsacks, and see if it is possible to exchange those items. If the exchange yields a better solution, swap the items.

References

[1] Michail G. Lagoudakis. *The 0-1 Knapsack Problem – An Introductory Survey.* Tech. rep. 1996.

[2] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations.* New York, NY, USA: John Wiley & Sons, Inc., 1990. ISBN: 0-471-92420-2.

[3] Aminto Senisuka, Byungjun You, and Takeo Yamada. *Reduction and Exact Algorithms for the Disjunctively Constrained Knapsack Problem.*