Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 183 (2007) 618–632
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

A pegging approach to the precedence-constrained knapsack problem

Byungjun You [1], Takeo Yamada *

Department of Computer Science, The National Defense Academy, Yokosuka, Kanagawa 239-8686, Japan

Received 6 May 2005; accepted 10 October 2006
Available online 13 December 2006

Abstract

The knapsack problem (KP) is generalized to the case where items are partially ordered through a set of precedence relations. As in ordinary KPs, each item is associated with profit and weight, the knapsack has a fixed capacity, and the problem is to determine the set of items to be packed in the knapsack. However, each item can be accepted only when all the preceding items have been included in the knapsack. The knapsack problem with these additional constraints is referred to as the precedence-constrained knapsack problem (PCKP). To solve PCKP exactly, we present a pegging approach, where the size of the original problem is reduced by applying the Lagrangian relaxation followed by a pegging test. Through this approach, we are able to solve PCKPs with thousands of items within a few minutes on an ordinary workstation.

Keywords: Combinational optimization; Pegging test; Knapsack problem; Precedence constraints

1. Introduction

Let $G = (V, E)$ be a *directed graph* [10] with vertex set $V = \{1,2,\ldots,n\}$ and edge set $E \subseteq V \times V$. Here, V represents the set of items that can be included into a knapsack of capacity c. Associated with each $j \in V$ are its weight w_j and profit p_j. Without much loss of generality we assume that parameters w_j, p_j $(j = 1,\ldots,n)$ and c are all positive integers. The set of edges E represents *precedence relations* between the items. That is, $(i,j) \in E$ implies that item j can be accepted only when item i has been included in the knapsack. Throughout the paper, we put $m := |E|$, and assume that G is *acyclic*, i.e., no directed cycle is included in G, since otherwise the precedence relations are not well defined.

* Corresponding author.
E-mail address: yamada@nda.ac.jp (T. Yamada).
[1] Current address: The Republic of Korea Navy, Republic of Korea.

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2006.10.031

The problem can be formulated mathematically as a 0–1 programming problem. Let x_j be a variable such that

$$x_j = \begin{cases} 1, & \text{if item } j \text{ is accepted,} \\ 0, & \text{otherwise.} \end{cases}$$

Then, we have the following *precedence-constrained knapsack problem* [21].

PCKP:

$$\text{maximize} \quad z(x) := \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_j \leqslant c, \tag{2}$$

$$x_i \geqslant x_j, \quad \forall (i,j) \in E, \tag{3}$$

$$x_j \in \{0,1\}, \quad \forall j \in V. \tag{4}$$

Here, without loss of generality, we assume that

$$w_j \leqslant c \quad (\forall j \in V), \quad \sum_{j=1}^{n} w_j > c$$

since otherwise the problem is trivial.

Precedence relations arise naturally as a consequence of logical/physical requirements among items. For example, in a project management activities are usually arranged in the form of a flow chart or a network, and each activity can be initiated only when all the preceding activities have been finished. Or, in open-pit mining [2] we can remove a block only when all the blocks lying immediately above have been removed. Mathematically, these relations are represented in the form of inequality (3). Then, if we wish to complete as many projects as possible, or excavate as many blocks as possible within a fixed time limit, we need to solve PCKP.

PCKP is $\mathcal{NP}$-hard [9]; because without precedence constraint (3), it reduces to the *knapsack problem* (KP, [17,15]), which is already $\mathcal{NP}$-hard. An important subclass of PCKP is the *tree-knapsack problem* (TKP, [5,22,14,12]), where G is a directed tree rooted at node 1. Hirabayashi et al. [11] formulated a tool-module design problem as a PCKP on a bipartite graph. Moriyama et al. [18] generalized this into a PCKP under the name of partially-ordered knapsack problem, and developed a branch-and-bound algorithm with some numerical experiments. Samphaiboon et al. [21] presented a dynamic programming algorithm to solve this problem. If the size of the problem is not so large, it may be solved by commercial or free IP solvers [8]. Using NUOPT [20], a popular IP solver in Japan, we were able to solve most of the randomly generated PCKPs with up to $n = 2000$, but for larger problems we often encountered difficulties in obtaining exact solutions.

In this paper, we propose a novel approach to solve larger PCKPs exactly as follows. First, we eliminate constraints (3) by applying the Lagrangian relaxation, and together with the continuous relaxation of (4), the result is a *continuous knapsack problem*. Then, the pegging test for ordinary KPs can be applied, and if the Lagrangian multipliers are well tuned up, we obtain a PCKP of substantially reduced size. With the precedence constraint (3), we can further derive an improved *block* pegging test, and the reduced PCKPs are often solved by commercial IP solvers. We implement these algorithms, and evaluate the developed method through a series of computational experiments.

2. Upper and lower bounds

This section derives an *upper bound* by applying the *Lagrangian relaxation* [19,23] to PCKP. We also present a *local search* [1] algorithm to obtain a good approximate solution quickly, which gives a *lower bound* to PCKP.