Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 186 (2008) 63–76
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

A dynamic programming based reduction procedure for the multidimensional 0–1 knapsack problem

Stefan Balev [a,*], Nicola Yanev [b], Arnaud Fréville [c], Rumen Andonov [d]

[a] LITIS, Le Havre University, 25 rue Ph. Lebon BP 540, 76058 Le Havre Cedex, France
[b] Faculty of Mathematics and Computer Science, Sofia University, 5, J. Bouchier str., 1126 Sofia, Bulgaria
[c] LAMIH/ROI, University of Valenciennes, Le Mont Houy, 59313 Valenciennes Cedex 9, France
[d] IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France

Received 22 January 2003; accepted 16 February 2006
Available online 13 February 2007

Abstract

This paper presents a preprocessing procedure for the 0–1 multidimensional knapsack problem. First, a non-increasing sequence of upper bounds is generated by solving LP-relaxations. Then, a non-decreasing sequence of lower bounds is built using dynamic programming. The comparison of the two sequences allows either to prove that the best feasible solution obtained is optimal, or to fix a subset of variables to their optimal values. In addition, a heuristic solution is obtained. Computational experiments with a set of large-scale instances show the efficiency of our reduction scheme. Particularly, it is shown that our approach allows to reduce the CPU time of a leading commercial software.

Keywords: Dynamic programming; Integer programming; Multidimensional knapsack problem; Variable reduction; Heuristics

1. Introduction

In this paper, we present a preprocesssing scheme for the 0–1 Multidimensional Knapsack Problem (MKP), which can be formulated as

$$\begin{aligned} \max \quad & \sum_{j\in N} c_j x_j \\ \text{s.t.} \quad & \sum_{j\in N} a_{ij} x_j \leqslant b_i, \quad i \in M, \\ & x_j \in \{0,1\}, \quad j \in N, \end{aligned}$$

where $N = \{1, 2, \dots, n\}$ is the set of items, $M = \{1, 2, \dots, m\}$ is the set of knapsack constraints with capacities b_i, associated weights a_{ij} and profits c_j. The objective is to find a subset of items that yields a maximum profit.

* Corresponding author. Tel.: +33 232 744 321; fax: +33 232 744 314.
E-mail addresses: stefan.balev@univ-lehavre.fr (S. Balev), choby@math.bas.bg (N. Yanev), arnaud.freville@univ-valenciennes.fr (A. Fréville), randonov@irisa.fr (R. Andonov).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2006.02.058

We assume that all the data a_{ij}, b_i, c_j are non-negative integers and, without loss of generality, that $c_j > 0$, $b_i > 0$, $a_{ij} \leqslant b_i$ for all $j \in N$ and all $i \in M$ and $\sum_{j \in N} a_{ij} > b_i$ for all $i \in M$.

The MKP is typically encountered in the areas of capital budgeting and resource allocation. The paper by Manne and Markowitz [27] is probably one of the earliest references to this problem. Other applications include project selection, cutting stock, loading problems, determining the optimal investment policy for the tourism sector of a developing country, and, more recently, delivery of groceries in vehicles with multiple compartments, approval voting, multi-project scheduling, satellite communications. It also appears in a collapsing problem and as a subproblem in large models for allocating processors and data bases in a distributed computer system. Finally, the MKP model is more and more frequently used as a benchmark to compare general purpose methods as metaheuristics.

In this paper we will often use shortcut notations for the problem like

$$\max\{cx : a_i x \leqslant b_i,\ i \in M,\ x \in B^n\},$$

or

$$\max\left\{\sum_{j \in N} c_j x_j : \sum_{j \in N} A_j x_j \leqslant b,\ x_j \in B,\ j \in N\right\},$$

or simply

$$\max\{cx : Ax \leqslant b,\ x \in B^n\}.$$

2. Related work

The multidimensional knapsack problem generalizes the well-known Knapsack Problem (KP) which deals with only one constraint. As the single constraint case, the MKP is $\mathcal{NP}$-hard but not strongly $\mathcal{NP}$-hard. Polynomial approximation schemes exist for the single knapsack problem and some of them are generalized for the MKP [6,13]. But while there are fully polynomial approximation schemes for $m = 1$, finding fully polynomial approximations for $m > 1$ is $\mathcal{NP}$-hard [15,26].

Most of the research on knapsack problems deals with the much simpler single constraint case ($m = 1$). This problem is very well studied and efficient exact and approximate algorithms have been developed for obtaining optimal and near-optimal solutions. An extensive overview of exact and heuristic algorithms is given by Martello and Toth [29]. Randomly generated instances up to 250 000 variables may be solved to optimality. Important recent advances can be found in [30,34].

2.1. Exact methods

In contrast, the MKP is significantly harder to solve in practice than the KP. As soon as the number of knapsack constraints increases, exact algorithms usually fail to provide an optimal solution of moderate size instances in a reasonable amount of time. For example, one of the recent versions of CPLEX (6.5.2) is not able to solve difficult problems with 100 variables and 5 constraints to optimality, because of the memory requirements of the search tree [33].

All general 0–1 integer programming techniques may be applied to the MKP [8,31,32]. Only the non-negativity of the coefficients distinguishes this problem from the general 0–1 integer programming problem. The dense constraint matrix and the absence of special constraints, such as generalized upper bounds, special-ordered sets, etc., complicate the development of efficient algorithms for the MKP. That is why relatively few special-purpose algorithms address the MKP.

The development of exact algorithms for 0–1 integer programming began several decades ago [2,3,16,18]. Typically, these approaches start with preprocessing phase, finding lower and upper bounds of the objective value and trying to reduce the problem size by fixation of variables and elimination of constraints. The second phase is an implicit enumeration, which uses the preprocessing information.

The first special-purpose branch and bound algorithm for the MKP is published by Shih [38]. An upper bound is obtained using the minimum of the LP-relaxation values associated with each of the m single