European Journal of Operational Research 199 (2009) 339–348

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

An iterative variable-based fixation heuristic for the 0-1 multidimensional knapsack problem

Christophe Wilbaut [a,*], Saïd Salhi [b], Saïd Hanafi [a]

[a] LAMIH-SIADE, UMR CNRS 8530, Université de Valenciennes, Le Mont Houy, 59530 Valenciennes Cedex 9, France
[b] Centre for Heuristic Optimisation, Kent Business School, University of Kent, UK

ARTICLE INFO

Article history:
Received 4 October 2007
Accepted 18 November 2008
Available online 6 December 2008

Keywords:
Heuristics
Exact methods
Fixation techniques
Knapsack

ABSTRACT

An iterative scheme which is based on a dynamic fixation of the variables is developed to solve the 0-1 multidimensional knapsack problem. Such a scheme has the advantage of generating memory information, which is used on the one hand to choose the variables to fix either permanently or temporarily and on the other hand to construct feasible solutions of the problem. Adaptations of this mechanism are proposed to explore different parts of the search space and to enhance the behaviour of the algorithm. Encouraging results are presented when tested on the correlated instances of the 0-1 multidimensional knapsack problem.

1. Introduction

The 0-1 multidimensional knapsack problem (**MKP**) is a generalization of the well-known knapsack problem with the presence of more than one constraint. The MKP can be formulated as follows:

$$(\text{MKP})\begin{cases}\max & \sum_{j\in N} c_j x_j,\\ \text{subject to}: & \sum_{j\in N} a_{ij}x_j \leqslant b_i, \quad \forall i\in M=\{1,\ldots,m\},\\ & x_j\in\{0,1\}, \quad j\in N=\{1,\ldots,n\},\end{cases}$$

where $c_j, \forall j \in N$, $a_{ij}, \forall i \in M$ and $j \in N$ and $b_i, \forall i \in M$ are positive integers. Without loss of generality, we can assume that the following constraints, as defined by (1), are satisfied. If this is not the case, one or more variables could be fixed to 0 or 1

$$\max\{a_{ij} : j \in N\} \leqslant b_i < \sum_{j\in N} a_{ij} \quad \forall i \in M. \tag{1}$$

For simplicity we also use the following matrix notation for the MKP:

$$(\text{MKP}) \quad \max\{c^T x : Ax \leqslant b, x \in \{0,1\}^n\}.$$

Many applications of this NP-hard problem are resource allocation-based, see for instance the first references of Lorie and Savage (1955) and Weingartner (1966). Recently, Meier et al. (2001) used the MKP as a subproblem in a new capital budgeting model. There are other applications such as cutting stock (Gilmore and Gomory, 1966), loading problems (Shih, 1979), and the daily management of a satellite (Vasquez and Hao, 2001a). Efficient algorithms have been proposed for solving the MKP. Several of those are metaheuristic-based algorithms like tabu search (see for instance Glover and Kochenberger, 1996 and Hanafi and Fréville, 1998), and genetic algorithm (Chu and Beasley, 1998). The review article by Fréville and Hanafi (2005) and the book by Kellerer et al. (2004) are informative and provide interesting and useful references. Very recently Wilbaut et al. (2008) produced a survey paper in this area with an emphasis to effective heuristics and their applications.

Several preprocessing techniques are often used to develop efficient integer programming-based approaches. It is well-known that if we are able to reduce the size of the problem to a reasonable level, even NP-hard problems can then be solved optimally with reasonable computational effort. Such a reduction process can be achieved by setting variables, identifying infeasibility and constraint redundancy, and tightening the linear programming (**LP**) relaxation. The latter includes modifying coefficients and generating strong valid inequalities. Some of these tools are described in Nemhauser and Wolsey (1999) and also in Savelsberg (1994). For instance, for the MKP techniques, the idea to reduce the number of variables was exploited by Babayev and Mardanov (1994) and also Zhu and Broughan (1997). Though the above techniques help to reduce the size of the problem in several cases, there is however no guarantee of their efficiency when tested on a given instance. For example, Wilbaut et al. (2006) applied a classical technique to fix variables in a global intensification algorithm, including a dynamic programming method for the MKP, and showed empirically

* Corresponding author. Tel.: +33 327511945.
E-mail addresses: christophe.wilbaut@univ-valenciennes.fr (C. Wilbaut), s.salhi@kent.ac.uk (S. Salhi), said.hanafi@univ-valenciennes.fr (S. Hanafi).

that it is difficult to fix any variable for those instances with large values of m.

In this paper, we propose an algorithm which is based on a heuristic fixation of the variables for solving the MKP. This is achieved by using an iterative scheme to generate useful information from LP-relaxation. This knowledge is then used to fix iteratively a subset of the decision variables and hence to generate feasible solutions. This method has the advantage of generating both lower and upper bounds of the problem. To explore the diversity of the solutions efficiently, we put forward three variants of the algorithm.

The remainder of this paper is organised as follows: in Section 2 we present the iterative scheme which we use in our algorithm to generate memory information. We describe in Section 3 the different strategies we implemented to fix the variables and to generate bounds of the problem. Section 4 is devoted to the computational results. We summarize our conclusions and point out some research avenues in the last section.

2. An iterative scheme

Several exact methods designed to find the optimal value of the problem were successfully applied to small sized instances. Such a success is unfortunately not repeated for problems with moderate and large size due to memory and computational time requirements. It is well-known that heuristics that use relaxation-based techniques are among the efficient ways to provide both upper and lower bounds for large and difficult combinatorial optimization problems. Glover (2005) proposed a general iterative method for pure and mixed integer programming. This method, which is referred to as the Adaptive Memory Projection (**AMP**), consists of four steps: (i) from an initial solution apply a heuristic to define a subset of the free variables; (ii) use an exact method to solve the sub problem associated with these remaining variables; (iii) re-launch the heuristic used in (i) from the solution obtained in (ii) with the introduction of restrictions generated from the memory; and finally (iv) introduce diversification processes to visit unexplored regions of the search space.

Some of the ideas of this general method do also exist in other efficient algorithms that are based on the exploration of small neighborhoods around the incumbent solution. For instance in Fischetti and Lodi (2003), a constraint, called local branching constraint, is added at every iteration to define a k-OPT neighborhood of the incumbent solution. The motivation is to explore better solutions quickly during the search. In the relaxation induced neighborhood search of Danna et al. (2005), variables which happen to have the same values in both the incumbent solution and in the solution of the current linear programming relaxation are made fixed, and the corresponding remaining sub problem is then optimally solved. For the MKP, Volgenant and Zwiers (2007) recently used partial enumeration. This approach can be viewed as a particular case of the AMP in which steps (i) and (ii) are only applied. Even if this method generates lower bounds of the MKP quickly (i.e. a few seconds/minutes by instance in general), it is clearly outperformed by other methods.

The method described in this paper is based on the scheme proposed by Wilbaut and Hanafi (2008) for solving the 0-1 mixed integer programming problem. For completeness this is shown in Fig. 1.

This interesting method though it guarantees an optimal solution, it was observed that its convergence can be really difficult to achieve in practice especially for larger sized instances. In other words, the results obtained for the MKP showed that even if this scheme was the basis in developing efficient algorithms for generating good bounds, the computational effort associated with these techniques can be significantly high. This can be due to either the large number of reduced problems to solve or the high level of difficulty in solving some of the reduced problems.

In this paper we propose a new algorithm that attempts to overcome the above drawbacks by using an iterative-phase to generate useful information in fixing some variables of the problem heuristically. This iterative process is described in Fig. 2.

The addition of a new constraint in the current problem in Step 2 appears to be useful in generating a better solution of the LP-relaxation. The following two propositions explain the construction of the constraint and show how it only cuts off the current

Step 1: Solve one or more relaxation(s) of the current problem (P) and record the corresponding optimal solution(s).

Step 2: Generate and solve one or more reduced problem(s) induced from the previous solution(s) to obtain one or more feasible solution(s) of (P).

Step 3: Update the best lower bound $\underline{v}^*$ of (P) if necessary and the best upper bound $\bar{v}$ of (P).

Step 4: If a stopping criterion is satisfied then return $\underline{v}^*$ and $\bar{v}$, else add one or more constraint(s) generated from the solution(s) of the relaxation(s) to (P) and return to Step 1.

Fig. 1. A general iterative scheme.

Step 1: Solve the LP-relaxation of the current problem (P) and keep an optimal solution $\bar{x}$ of this relaxation. Update the upper bound $\bar{v}$ of (P).

Step 2: Generate a constraint from $\bar{x}$ which eliminates this solution without eliminating any other solution of the initial problem and add this constraint to (P).

Step 3: If a chosen number of iterations is reached then return $\bar{v}$, otherwise go to Step 1.

Fig. 2. The iterative-phase.