Computers & Operations Research 37 (2010) 2017–2026

Contents lists available at ScienceDirect

Computers & Operations Research

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

Kernel search: A general heuristic for the multi-dimensional knapsack problem

Enrico Angelelli [a], Renata Mansini [b,*], M. Grazia Speranza [a]

[a] *Department of Quantitative Methods, University of Brescia, Italy*
[b] *Department of Information Engineering, University of Brescia, Italy*

ARTICLE INFO

Available online 6 February 2010

Keywords:
Multi-dimensional knapsack problem
Binary variables
Kernel search
Heuristics

ABSTRACT

In this paper we apply the kernel search framework to the solution of the strongly NP-hard multi-dimensional knapsack problem (MKP). Kernel search is a heuristic framework based on the identification of a restricted set of promising items (*kernel*) and on the exact solution of ILP sub-problems. Initially, the continuous relaxation of the MKP, solved on the complete set of available items, is used to identify the initial kernel. Then, a sequence of ILP sub-problems are solved, where each sub-problem is restricted to the present kernel and to a subset of other items. Each ILP sub-problem may find better solutions with respect to the previous one and identify further items to insert into the kernel. The kernel search was initially proposed to solve a complex portfolio optimization problem. In this paper we show that the method has general key features that make it appropriate to solve other combinatorial problems using binary variables to model the decisions to select or not items. We adapt the kernel search to the solution of MKP and show that the method is very effective and efficient with respect to known problem-specific approaches. Moreover, the best known values of some MKP benchmark problems from the MIPLIB library have been improved.

1. Introduction

The multi-dimensional knapsack problem (MKP) is a strongly NP-hard combinatorial optimization problem arising in many application contexts such as capital budgeting, cargo loading [15], cutting stock problems [8] and asset-backed securitization [12]. In the last years the MKP turned out to be one of the favorite playgrounds for experiments with heuristics and metaheuristics, in particular tabu search and genetic algorithms (see Kellerer et al. [11]).

A common interpretation of the model considers a set $N=\{1,\ldots,n\}$ of items and a set $M=\{1,\ldots,m\}$ of resources with availability c_i, $i\in M$. Each item j has a value p_j and consumes resource i at a rate w_{ij}. It is assumed that all p_j, w_{ij} and c_i are non-negative integer values. More precisely, w_{ij} can be zero for some i, j, as long as $\sum_{i\in M} w_{ij}\geq 1$ holds for all items $j\in N$. The problem is to find a subset of items such that their total value is maximum while the total requirement for each resource i does not exceed the availability c_i.

The decision of selecting or not an item j is modeled by a binary variable x_j. The MKP can be formulated as follows:

$$MKP(N) \quad z:=\max\sum_{j\in N} p_j x_j, \tag{1}$$

$$\sum_{j\in N} w_{ij}x_j \leq c_i, \quad i\in M, \tag{2}$$

$$x_j\in\{0,1\}, \quad j\in N. \tag{3}$$

Several heuristics have been proposed to solve the MKP. These include simulated annealing [6], tabu search [9,10], genetic algorithms [4] and hybrid methods [16]. Some of them are strongly problem-dependent. A minor change in the model would require the re-design of the heuristic. Some others spend a large part of the computational time examining solutions that are very unlikely to be optimal or close to an optimal one. The method Kernel Search described here is a general heuristic framework which tries to overcome both these drawbacks.

The main idea of the Kernel Search is to obtain a solution, of hopefully high quality, from a small set of promising items called the *kernel*. The kernel is initially built using information provided by the solution of the linear relaxation of the original problem. Then, new promising items are identified and added to the kernel by means of the solution of a sequence of small/moderate size ILP sub-problems. One of the main issues addressed in this paper concerns the size of the ILP sub-problems. This value should be small enough to limit the computational time required to solve each ILP sub-problem. At the same time it has to be large enough to allow possibly correlated items to be jointly selected.

Although initially designed to solve a portfolio optimization problem (see Angelelli et al. [1]), the Kernel Search has general key features that make it applicable to other combinatorial problems where binary variables are used to model the selection or rejection

* Corresponding author.
E-mail addresses: angele@eco.unibs.it (E. Angelelli), rmansini@ing.unibs.it (R. Mansini), speranza@eco.unibs.it (M. Grazia Speranza).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.02.002

of items. The portfolio optimization problem represents a combinatorial problem extremely suitable for the Kernel Search since the number of selected items (securities) in an optimal solution is quite small independently of the initial size of the problem. On the contrary, the MKP is a combinatorial problem far from being ideal for the proposed method because of the possibly large number of variables set to 1 in the optimal solution. Nevertheless, in this paper we adapt the Kernel Search to the solution of the MKP and show that the performance of the Kernel Search, compared to known problem-specific approaches, on benchmark instances is extremely good. We frequently reach the best known solutions and in a few cases even improve them, whereas, on average, we solve the instances in a much shorter computational time.

The idea of considering a problem of small/moderate size in order to intensify the search only on a promising region of the solution space is not new in the literature and has been used in different problem-specific approaches (see references in Angelelli et al. [1]). Solution approaches inspired by this idea were proposed for the knapsack problems (KPs) where the core algorithm, initially introduced for the classical 0–1 knapsack problem by Balas and Zemel [2], is based on the idea of identifying a small subset of items (called the core) around the critical item on which to solve a restricted problem exactly. Along this idea, all the proposed methods are mainly exact algorithms and need to exhaustively examine all the possible cores. Pisinger [13] introduced an expanding method where the size of the core problem is modified during the algorithm execution. Gomes da Silva et al. [5] analyzed the core concept for the bi-criteria KP, while, only recently, such concept has been extended to the multi-dimensional knapsack problem by Puchinger et al. [14]. The authors studied this new core concept extensively and provided an interesting analysis on different efficiency measures for the MKP. The results of their empirical analysis were used to develop new concepts for solving the MKP using ILP-based and memetic algorithms.

The paper is organized as follows. Section 2 describes the Kernel Search and the way it is adapted to the solution of the MKP, whereas Section 3 is devoted to the computational results on benchmark instances. Finally, conclusions and future developments are drawn.

2. Kernel Search

The Kernel Search was introduced in [1] and applied to the solution of a portfolio optimization problem. In order to make the paper self-contained, in this section we describe the main structure of the Kernel Search, adapted to the solution of the MKP. We recall that even if MKP is a pure ILP problem, the Kernel Search as described in [1] refers to a more general MILP problem containing binary variables modeling items selection as well as other integer and continuous variables.

Table 1
Benchmark instances with n=250, m=5.

Problem	BestVal	*Fixed-bucket-I*(1)			*Fixed-bucket-I*(0.2)			*Fixed-bucket-I*(0.1)		
		Gap LP (%)	Gap (%)	Time (s)	Gap LP (%)	Gap (%)	Time (s)	Gap LP (%)	Gap (%)	Time (s)
250.5_1	59,312	0.219	0.000	53	0.219	0.000	9	0.297	0.078	36
250.5_2	61,472	0.255	0.000	135	0.255	0.000	82	0.272	0.016	24
250.5_3	62,130	0.208	0.000	13	0.208	0.000	8	0.208	0.000	4
250.5_4	59,463	0.193	0.000	755	0.193	0.000	119	0.244	0.050	60
250.5_5	58,951	0.216	0.000	312	0.216	0.000	89	0.216	0.000	15
250.5_6	60,077	0.269	0.000	327	0.294	0.025	205	0.295	0.027	48
250.5_7	60,414	0.185	0.000	208	0.214	0.030	84	0.231	0.046	67
250.5_8	61,472	0.270	0.000	204	0.299	0.029	150	0.299	0.029	70
250.5_9	61,885	0.238	0.000	65	0.238	0.000	27	0.238	0.000	18
250.5_10	58,959	0.155	0.000	20	0.155	0.000	7	0.155	0.000	3
Average		**0.221**	**0.000**	**209**	**0.229**	**0.008**	**78**	**0.245**	**0.025**	**34**
250.5_11	109,109	0.102	0.000	293	0.102	0.000	154	0.102	0.000	44
250.5_12	109,841	0.109	0.000	56	0.109	0.000	32	0.109	0.000	10
250.5_13	108,508	0.130	0.000	178	0.130	0.000	218	0.130	0.000	71
250.5_14	109,383	0.117	0.000	260	0.117	0.000	133	0.117	0.000	37
250.5_15	110,720	0.103	0.000	738	0.103	0.000	602	0.103	0.000	350
250.5_16	110,256	0.100	0.000	257	0.100	0.000	154	0.105	0.005	82
250.5_17	109,040	0.103	0.000	129	0.103	0.000	89	0.103	0.000	44
250.5_18	109,042	0.088	0.000	509	0.088	0.000	496	0.088	0.000	107
250.5_19	109,971	0.138	0.000	215	0.138	0.000	130	0.138	0.000	102
250.5_20	107,058	0.097	0.000	162	0.097	0.000	143	0.097	0.000	76
Average		**0.109**	**0.000**	**280**	**0.109**	**0.000**	**215**	**0.109**	**0.000**	**92**
250.5_21	149,665	0.067	0.000	135	0.067	0.000	141	0.067	0.000	152
250.5_22	155,944	0.090	0.000	87	0.090	0.000	86	0.090	0.000	66
250.5_23	149,334	0.067	0.000	157	0.067	0.000	212	0.067	0.000	139
250.5_24	152,130	0.075	0.000	92	0.075	0.000	102	0.075	0.000	59
250.5_25	150,353	0.077	0.000	138	0.077	0.000	146	0.077	0.000	87
250.5_26	150,045	0.058	0.000	10	0.058	0.000	8	0.058	0.000	7
250.5_27	148,607	0.061	0.000	7	0.061	0.000	4	0.061	0.000	4
250.5_28	149,782	0.083	0.000	173	0.083	0.000	286	0.083	0.000	115
250.5_29	155,075	0.087	0.000	47	0.087	0.000	23	0.087	0.000	17
250.5_30	154,668	0.097	0.000	149	0.097	0.000	148	0.097	0.000	84
Average		**0.076**	**0.000**	**99**	**0.076**	**0.000**	**116**	**0.076**	**0.000**	**73**
Tot. average		**0.135**	**0.000**	**196**	**0.138**	**0.003**	**136**	**0.144**	**0.008**	**67**