Drexl, Andreas; Jørnsten, Kurt

Pricing the multiple-choice nested knapsack problem

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 626

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Suggested Citation: Drexl, Andreas; Jørnsten, Kurt (2007) : Pricing the multiple-choice nested knapsack problem, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 626, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/147679

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 626

Pricing the multiple-choice nested knapsack problem

Andreas Drexl, Kurt Jørnsten

June 2007

Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität, Kiel, Germany,
andreas.drexl@bwl.uni-kiel.de

Norwegian School of Economics and Business Administration, Department of Finance and Management Science, Bergen, Norway, kurt.jornsten@nhh.no

Abstract

The multiple-choice nested knapsack problem (MCKP) is a generalization of the ordinary knapsack problem, where the set of items is partitioned into classes. The binary choice of selecting an item is replaced by taking exactly one item out of each class of items. Due to the fact that the MCKP is an NP-hard integer program dual prices are not readily available. In this paper we propose a family of linear programming models the optimal solution of which is integral for many instances. The models are evaluated experimentally using a well-defined testbed consisting of 9,000 instances. Overall our methodology produces an integral solution for 75 % of the instances considered. In particular, for two out of five distribution types studied at least one of the models produces "almost always" an integral solution. Hence, in most of the cases there exists a linear price function that supports the optimal allocation.

Keywords: Knapsack problem, multiple-choice constraints, integer programming, duality, linear programming, pricing

1 Introduction

In order to define the problem formally consider k classes $N_1, \ldots, N_k$ of items to pack in some knapsack of capacity c. Each item $j \in N_i$ has a profit p_{ij} and a weight w_{ij}. The goal is to choose one item from each class such that the profit sum is maximized without exceeding c. If we define the binary variables x_{ij} which take on value 1 if and only if item j is chosen in class N_i the multiple-choice nested knapsack problem (MCKP) may be formulated as the integer program (1).

$$\max \sum_{i=1}^{k} \sum_{j \in N_i} p_{ij} x_{ij} \tag{1a}$$

$$\text{s.t.} \sum_{i=1}^{k} \sum_{j \in N_i} w_{ij} x_{ij} \leq c \tag{1b}$$

$$\sum_{j \in N_i} x_{ij} = 1 \qquad i = 1, \ldots, k \tag{1c}$$

$$x_{ij} \in \{0, 1\} \qquad i = 1, \ldots, k, j \in N_i \tag{1d}$$

In the sequel we will assume that all coefficients p_{ij}, w_{ij}, and c are nonnegative integers, with class N_i having size n_i for $i = 1, \ldots, k$. The total number of items is $n := \sum_{i=1}^{k} n_i$. Negative coefficients p_{ij} and w_{ij} can be handled by adding the constant $\bar{p}_i = -\min_{j \in N_i} p_{ij}$ to all the profits in class N_i, and by adding $\bar{w}_i = -\min_{j \in N_i} w_{ij}$ to all the weights in class N_i as well as to the capacity c.

To avoid trivial or unsolvable cases we assume that

$$\sum_{i=1}^{k} \min_{j \in N_i} w_{ij} \leq c < \sum_{i=1}^{k} \max_{j \in N_i} w_{ij}.$$

Moreover we assume that every item $j \in N_i$ satisfies

$$w_{ij} + \sum_{h=1,\ldots,k,h\neq i} \min_{k\in N_h} w_{hk} \leq c,$$

because otherwise no feasible solution exists when the item is chosen (and hence it may be discarded).

The MCKP in its minimization form may be transformed into an equivalent maximization problem by finding for each class the values $\bar{p}_i = \max_{j\in N_i} p_{ij}$ and $\bar{w}_i = \max_{j\in N_i} w_{ij}$ and be setting $\tilde{p}_{ij} := \bar{p}_i - p_{ij}$ and $\tilde{w}_{ij} := \bar{w}_i - w_{ij}$ for $j \in N_i$ and $\tilde{c} := \sum_{i=1}^{k} \bar{w}_i - c$. Then the maximization problem is defined in $\tilde{p}, \tilde{w}$ and $\tilde{c}$.

If the multiple-choice constraints (1c) are replaced by $\sum_{j\in N_i} x_{ij} \leq 1$, as considered, for instance, in Johnson and Padberg [11], then this problem can be transformed into the equality form by adding a dummy item $n_i + 1$ with $(p_{i,n_i+1}, w_{i,n_i+1}) := (0, 0)$ to each class N_i.

The MCKP is NP-hard as it contains the ordinary knapsack problem (KP) as a special case. This can be easily shown as follows: Given an instance of KP with n profits p_j, weights w_j and capacity c, construct an instance of MCKP by introducing $m := n$ classes where each class i has two items $(p_{i1}, w_{i1}) := (0, 0)$ and $(p_{i2}, w_{i2}) := (p_j, w_j)$, respectively, while the capacity of the new problem is c.

If we relax the integrality constraint (1d) on x_{ij} we obtain the linear programming relaxation (2), abbreviated as LMCKP (used in section 4 for benchmarking purposes).

$$\max \sum_{i=1}^{k} \sum_{j\in N_i} p_{ij} x_{ij} \tag{2a}$$

$$\text{s.t.} \sum_{i=1}^{k} \sum_{j\in N_i} w_{ij} x_{ij} \leq c \tag{2b}$$

$$\sum_{j\in N_i} x_{ij} = 1 \qquad i = 1, \ldots, k \tag{2c}$$

$$x_{ij} \geq 0 \qquad i = 1, \ldots, k, j \in N_i \tag{2d}$$

Several algorithms for solvig the MCKP have been proposed during the last decades, e.g., by Armstromg, Kung, Sinha and Zoltners [1], Dyer, Kayal and Walker [8], Nauss [16], and Sinha and Zoltners [18], respectively. Most of the algorithms start by solving LMCKP (see, e.g., Dudziński and Walukiewicz [5], Dyer [7] and Zemel [19]) in order to obtain an upper bound followed by some tests to fix several variables in each class to their optimal value. The reduced MCKP is then solved to optimality. Pisinger [17] generalized essential results for the KP related to the so-called core (derived, among others, by Balas and Zemel [2]) and he developed an algorithm (called mcknap.c) which essentially works as follows: A simple algorithm is used for solving LMCKP and for deriving an initial feasible solution for MCKP. Starting from this initial feasible solution dynamic programming is used to solve MCKP, adding new classes to the core by need.

The MCKP has a wide range of applications: Capital budgeting and transformation of nonlinear knapsack problems (see, e.g., Nauss [16]), menu planning (see, e.g., Sinha

and Zoltners [18]), MCKP as subproblem induced by Lagrangian relaxation applied to several integer programming problems (see, e.g., Fisher [9]), MCKP used for solving the generalized assignment problem (see, e.g., Barcia and Jørnsten [3]), MCKP used for limiting power consumption in VLSI design (see, e.g., Mejia-Alvarez, Levner and Mosse [15]), MCKP used for high-level synthesis of VLSI circuits (see, e.g., de Leone, Jain and Straus [4]), and MCKP used in the auction setting (see, e.g., Kelly [13]), respectively.

Without going into details we refer the readers to the surveys Dudziński and Walukiewicz [6], Kellerer, Pferschy and Pisinger [12] and Martello and Toth [14], respectively.

In section 2 we present a family of linear programming models the solution of which is integral in many cases. An instance is used in Section 3 for illustrative purposes. In section 4 we provide the results of a computational study. Section 5 concludes the paper.

2 Variable elimination/aggregation

The formulation of the family of linear programming models can be accomplished in two steps. First, we solve the integer program (1). Then we eliminate all variables corresponding to the chosen items and introduce instead an artificional one by means of column aggregation.

More precisely, assume that we know an optimal solution $(x_{ij}^{(1)})$ of (1). Let

$$X_i^0 = \left\{ j \in N_i : x_{ij}^{(1)} = 0 \right\}$$

denote the set of variables which have been fixed to zero for class $i = 1, \ldots, k$. Likewise,

$$X_i^1 = \left\{ j \in N_i : x_{ij}^{(1)} = 1 \right\}$$

denotes the set of variables which have been fixed to one.

The reformulation is based on the idea that we eliminate the variables contained in X_i^1 for $i = 1, \ldots, k$ and that we introduce instead an artificional item. This item contributes the amount

$$\bar{p} = \sum_{i=1}^{k} \sum_{j \in X_i^1} p_{ij}$$

to the optimal objective function value and it consumes

$$\bar{c} = \sum_{i=1}^{k} \sum_{j \in X_i^1} w_{ij}$$

of the capacity c of the knapsack and, hence, $\delta = c - \bar{c}$ is the portion (slack) of the knapsack capacity not used in the optimal solution.

A linear program based on the idea of variable elimination/aggregation, that is, which solely uses the variables X_i^0 for $i = 1, \ldots, k$ and the variable z associated with the artificional item, is provided in (3).

$$\max \sum_{i=1}^{k} \sum_{j \in X_i^0} p_{ij} x_{ij} + \bar{p} \cdot z \tag{3a}$$

$$\text{s.t.} \sum_{i=1}^{k} \sum_{j \in X_i^0} w_{ij} x_{ij} + \bar{c} \cdot z \leq c \tag{3b}$$

$$\sum_{j \in X_i^0} x_{ij} + z = 1 \qquad i = 1, \ldots, k \tag{3c}$$

$$x_{ij} \geq 0 \qquad i = 1, \ldots, k, j \in X_i^0 \tag{3d}$$

$$z \geq 0 \tag{3e}$$

The objective function (3a) comprises the contribution of the variable z and of the variables contained in X_i^0 for $i = 1, \ldots, k$. Constraint (3b) assures that the capacity c of the knapsack is not exceeded by the weights associated with X_i^0 for $i = 1, \ldots, k$ and by the weights associated with z. Constraint (3c) assures that for each class $i = 1, \ldots, k$ either variables of the set X_i^0 and/or the variable z are chosen. Finally, (3d) and (3e) define the decision variables to be nonnegative.

Apparently, in terms of the optimal solution $(x_{ij}^{(1)})$ in general we have positive slack in constraint (3b). If we reduce the right-hand-side c to $\bar{c}$ we get the linear program (4).

$$\max \sum_{i=1}^{k} \sum_{j \in X_i^0} p_{ij} x_{ij} + \bar{p} \cdot z \tag{4a}$$

$$\text{s.t.} \sum_{i=1}^{k} \sum_{j \in X_i^0} w_{ij} x_{ij} + \bar{c} \cdot z \leq \bar{c} \tag{4b}$$

$$\sum_{j \in X_i^0} x_{ij} + z = 1 \qquad i = 1, \ldots, k \tag{4c}$$

$$x_{ij} \geq 0 \qquad i = 1, \ldots, k, j \in X_i^0 \tag{4d}$$

$$z \geq 0 \tag{4e}$$

Constraint (4b) is more tight than constraint (3b) and, hence, we can expect that model (4) will produce an integral solution more often than model (3). However, the dual price then will not reflect the given right-hand-side (capacity of the knapsack) but only the amount used in the optimal solution.

Fortunately, we can improve model (3) by attaching the "optimum" slack to the capacity $\bar{c}$ used in the optimum solution, that is, make use of the fact that $c = \bar{c} + \delta$ trivially is valid. Doing so we get the linear program (5).

$$\max \sum_{i=1}^{k} \sum_{j \in X_i^0} p_{ij} x_{ij} + \bar{p} \cdot z \tag{5a}$$

$$\text{s.t.} \sum_{i=1}^{k} \sum_{j \in X_i^0} w_{ij} x_{ij} + c \cdot z \leq c \tag{5b}$$

$$\sum_{j \in X_i^0} x_{ij} + z = 1 \qquad i = 1, \ldots, k \tag{5c}$$

$$x_{ij} \geq 0 \qquad i = 1, \ldots, k, j \in X_i^0 \tag{5d}$$

$$z \geq 0 \tag{5e}$$

In section 4 we will show be means of a computational study how the models (3) to (5) behave in terms of integrality of optimal solutions and dual degeneracy. Before we illustrate the idea using an example.

3 Illustrative example

Consider the following instance (of the distribution type SZ generated according to the specification given below) with $k = 5$ classes and $n_i = 5$ items for each class. The instance is characterized by the profit matrix

$$(p_{ij}) = \begin{pmatrix} 150 & 333 & 483 & 643 & 698 \\ 202 & 259 & 751 & 790 & 913 \\ 137 & 708 & 739 & 760 & 919 \\ 93 & 287 & 318 & 636 & 885 \\ 106 & 210 & 391 & 404 & 777 \end{pmatrix},$$

the weight matrix

$$(w_{ij}) = \begin{pmatrix} 179 & 476 & 708 & 868 & 874 \\ 40 & 61 & 539 & 823 & 898 \\ 336 & 378 & 441 & 486 & 913 \\ 290 & 296 & 302 & 342 & 513 \\ 219 & 255 & 276 & 513 & 685 \end{pmatrix},$$

and the capacity $c = 1{,}941$.

An optimal integer solution with objective function value 2,885 is:

$$(x_{ij}^{(1)}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

An optimal solution of the linear programming relaxation with objective function value 2,937.85 is:

$$(x_{ij}^{(2)}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0.86 & 0 & 0.14 \end{pmatrix}$$

An optimal solution of the aggregate model (3) with objective function value 2,921.29 is:

$$z = 0.96, \quad (x_{ij}^{(3)}) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0.04 \\ 0 & 0 & 0 & 0 & 0.04 \\ 0 & 0 & 0.04 & 0 & 0 \\ 0 & 0 & 0 & 0.04 & 0 \\ 0 & 0 & 0 & 0 & 0.04 \end{pmatrix}$$

The aggregate model (4) has an integral optimal primal solution $z = 1.0$ and $x_{ij}^{(4)} = 0$ for all i and j with objective function value 2,885. The corresponding dual solution is

$$u = 0.65$$

and

$$(v_i) = (131.67, 331.12, 453.25, 414.39, 333.14)$$

where u is the dual variable corresponding to the knapsack constraint and v_i is the dual variable corresponding to the ith multiple-choice constraint.

The optimal solution of the aggregate model (5) again is $z = 1.0$ and $x_{ij}^{(5)} = 0$ for all i and j. The corresponding dual solution is

$$u = 0.68$$

and

$$(v_i) = (107.26, 306.04, 440.93, 404.84, 314.01).$$

Using these prices in order to suggest the necessary increase of the objective function coefficients of non-optimal variables leads to the following: For all but the variables x_{21}, x_{22}, x_{51} and x_{52} the price increase suggested is not sufficient. The new objective function coefficient for variable x_{21} determined by the prices is 333. However, raising it to 314 is sufficient in order to let this item be included in the knapsack. For the other variables mentioned the results are about the same, that is, 347 compared to 314, 461 to 460 and 487 to 461. Note that in order to get suggestions for the necessary increase for all non-optimal variables one has to examine all optimal dual solutions which is beyond the scope of this paper.

4 Computational results

In this section we provide the results of an in-depth computational study in order to show that the models (3) to (5) in general have different optimal (primal and/or dual)

solutions. Furthermore, we show that the degree of dual degeneracy associated with the optimal primal solutions in general also differs.

The models described earlier have been implemented in Java using the Cplex callable library (version 9.0; all parameters with default values) on an AMD Athlon with 2 GB RAM and 2.1 Ghz clockpulse running under the operating system Linux.

Following the lines of Pisinger [17] in what follows we will study how the models behave for different problem sizes (k and n_i), data ranges (R), and test instances. Five types of randomly generated test instances are considered (given k, n_i and R):

UC In each class i generate n_i items by choosing w_{ij} and p_{ij} randomly in $[1, R]$ (uncorrelated instances).

WC In each class i generate n_i items where w_{ij} is randomly distributed in $[1, R]$ and p_{ij} is randomly distributed in $[w_{ij} - 10, w_{ij} + 10]$ such that $p_{ij} \geq 1$ (weakly correlated instances).

SC In each class i generate n_i items with weight w'_j randomly distributed in $[1, R]$ and set $p'_j = w'_j + 10$. Order the items for each size n_i according to nondecreasing weights. Set $w_{ij} = \sum_{h=1}^{j} w'_h$ and $p_{ij} = \sum_{h=1}^{j} p'_h$ for $j = 1, \ldots, n_i$ (strongly correlated instances).

SS In each class i the weight w_{ij} of the n_i items is randomly distributed in $[1, R]$ and p_{ij} equals w_{ij} (subset-sum instances).

SZ In each class i generate n_i items such that w'_j and p'_j are randomly distributed in $[1, R]$. Order the profits and weights in nondecreasing order. Set $w_{ij} = w'_j$ and $p_{ij} = p'_j$ for $j = 1, \ldots, n_i$ (Sinha and Zoltners instances).

For each instance the capacity c is defined as follows:

$$c = \frac{1}{2} \sum_{i=1}^{k} \left(\min_{j \in N_i} w_{ij} + \max_{j \in N_i} w_{ij} \right)$$

We have generated 100 instances at random for each of the 5 problem types, 9 sizes and 2 data ranges $R = \{R_1, R_2\}$ considered in what follows. Hence, overall the testbed consists of 100·5·9·2 = 9,000 instances. For each instance an optimal solution of the integer program (1) has been computed using Cplex (and not the more efficient code mcknap.c; see [10]).

The results for the problem type UC are displayed in table 1 for different problem sizes and data ranges. For each of the 4 considered models we present the number of times where the solution of the linear program turned out to be integral (indicated by LP=IP). For the subset of instances with integral solutions we report the percentage average number of dual variables equal to zero (#DV0(%) used as abbreviation) in order to give an indication of the degree of dual degeneracy observed. The last row displays (weighted) average values for each of the columns.

For the problem types WC to SZ the results are displayed in tables 2 to 5 in the same way.

The results can be summarized as follows:

		model (2)		model (3)		model (4)		model (5)	
k	n_i	LP=IP	#DV0(%)	LP=IP	#DV0(%)	LP=IP	#DV0(%)	LP=IP	#DV0(%)
10	10	34	8.8	100	9.1	100	3.3	100	4.9
100	10	8	0.9	100	1.0	100	0.2	100	0.2
1,000	10	0	–	100	0.1	100	0.1	100	0.1
10,000	10	0	–	100	0.0	100	0.1	100	0.1
10	100	52	9.1	100	9.1	100	6.3	100	9.1
100	100	50	1.0	100	1.0	100	0.8	100	1.0
1,000	100	29	0.1	100	0.1	100	0.1	100	0.1
10	1,000	49	9.1	100	9.1	100	6.8	100	9.1
100	1,000	47	1.0	100	1.0	100	0.8	100	1.0
				$R_1 = 1{,}000$					
10	10	35	9.1	99	9.1	100	3.2	100	4.4
100	10	4	1.0	100	1.0	100	0.2	100	0.2
1,000	10	0	–	100	0.1	100	0.1	100	0.1
10,000	10	0	–	100	0.0	100	0.1	100	0.1
10	100	50	9.1	100	9.1	100	5.7	100	9.1
100	100	42	1.0	100	1.0	100	0.3	100	0.7
1,000	100	28	0.1	100	0.1	100	0.0	100	0.0
10	1,000	53	9.1	100	9.1	100	6.1	100	9.1
100	1,000	56	1.0	100	1.0	100	0.6	100	1.0
				$R_2 = 10{,}000$					
averages		29.8	5.0	99.9	3.4	100	1.9	100	2.8

Table 1: Instances of Type UC

Table 2: Instances of Type WC

		model (2)		model (3)		model (4)		model (5)	
k	n_i	LP=IP	#DV0(%)	LP=IP	#DV0(%)	LP=IP	#DV0(%)	LP=IP	#DV0(%)
10	10	0	–	46	12.1	100	11.1	100	11.0
100	10	0	–	48	25.2	100	26.3	100	26.2
1,000	10	0	–	65	26.5	100	26.4	100	26.4
10,000	10	0	–	100	26.7	100	26.7	100	26.7
10	100	0	–	27	0.0	27	0.0	27	0.0
100	100	0	–	31	0.0	70	0.0	59	0.0
1,000	100	0	–	94	0.0	98	0.0	98	0.0
10	1,000	0	–	0	–	0	–	0	–
100	1,000	0	–	13	0.0	28	0.0	13	0.0
				$R_1 = 1{,}000$					
10	10	0	–	27	7.4	91	9.5	89	8.3
100	10	0	–	10	24.4	100	25.1	100	24.3
1,000	10	0	–	23	26.5	100	27.1	100	27.0
10,000	10	0	–	74	27.6	100	27.6	100	27.6
10	100	0	–	3	0.0	10	0.0	3	0.0
100	100	0	–	0	–	4	0.0	1	0.0
1,000	100	0	–	38	0.0	71	0.0	69	0.0
10	1,000	0	–	0	–	0	–	0	–
100	1,000	0	–	0	–	1	0.0	0	–
				$R_2 = 10{,}000$					
averages		0.0	–	33.3	15.4	61.1	16.3	58.8	16.7

		model (2)		model (3)		model (4)		model (5)	
k	n_i	LP=IP	#DV0(%)	LP=IP	#DV0(%)	LP=IP	#DV0(%)	LP=IP	#DV0(%)
10	10	0	–	14	0.0	34	0.0	29	0.0
100	10	0	–	19	0.0	100	0.0	100	0.0
1,000	10	0	–	13	0.0	100	0.0	100	0.0
10,000	10	0	–	0	–	100	0.0	100	0.0
10	100	0	–	0	–	8	0.0	0	–
100	100	0	–	0	–	76	0.0	1	0.0
1,000	100	0	–	0	–	62	0.0	11	0.0
10	1,000	0	–	0	–	34	0.0	0	–
100	1,000	0	–	0	–	99	0.0	0	–
				$R_1 = 1{,}000$					
10	10	0	–	2	0.0	33	0.0	7	0.0
100	10	0	–	3	0.0	99	0.0	39	0.0
1,000	10	0	–	2	0.0	100	0.0	41	0.0
10,000	10	0	–	0	–	100	0.0	96	0.0
10	100	0	–	0	–	8	0.0	0	–
100	100	0	–	0	–	72	0.0	0	–
1,000	100	0	–	0	–	91	0.0	2	0.0
10	1,000	0	–	0	–	45	0.0	0	–
100	1,000	0	–	0	–	10	0.0	0	–
				$R_2 = 10{,}000$					
averages		0.0	–	2.9	0.0	65.1	0.0	29.2	0.0

Table 3: Instances of Type SC

Table 4: Instances of Type SS

		model (2)		model (3)		model (4)		model (5)	
k	n_i	LP=IP	#DV0(%)	LP=IP	#DV0(%)	LP=IP	#DV0(%)	LP=IP	#DV0(%)
10	10	0	–	52	87.1	52	87.1	52	87.1
100	10	1	99.0	56	98.8	100	98.7	56	98.8
1,000	10	0	–	100	99.9	100	99.9	100	99.9
10,000	10	0	–	10	70.4	10	70.4	10	70.4
10	100	5	80.0	3	51.5	3	51.5	3	51.5
100	100	4	99.0	100	42.5	100	42.5	100	42.5
1,000	100	0	–	100	1.9	100	1.9	100	1.9
10	1,000	1	45.5	1	72.7	1	72.7	1	72.7
100	1,000	0	–	87	2.2	87	2.2	87	2.2
				$R_1 = 1{,}000$					
10	10	0	–	10	86.4	53	87.0	10	86.4
100	10	0	–	8	98.3	100	98.6	8	98.3
1,000	10	0	–	54	99.9	100	99.9	54	99.9
10,000	10	0	–	0	–	0	–	0	–
10	100	2	90.9	2	90.9	2	72.7	2	68.2
100	100	0	–	55	64.1	100	66.2	55	64.1
1,000	100	0	–	0	–	0	–	0	–
10	1,000	2	90.9	3	75.8	3	63.6	3	63.6
100	1,000	0	–	26	28.6	71	24.6	26	28.6
				$R_2 = 10{,}000$					
averages		0.8	86.9	37.1	56.0	54.6	64.3	37.1	55.9

k	n_i	model (2)		model (3)		model (4)		model (5)	
		LP=IP	#DV0(%)	LP=IP	#DV0(%)	LP=IP	#DV0(%)	LP=IP	#DV0(%)
10	10	0	–	7	7.8	100	13.0	100	12.9
100	10	0	–	32	30.0	100	30.9	100	30.8
1,000	10	0	–	32	31.8	100	31.7	100	31.7
10,000	10	0	–	59	31.8	100	31.8	100	31.8
10	100	0	–	41	0.0	98	0.0	98	0.0
100	100	0	–	71	0.0	100	0.0	100	0.0
1,000	100	0	–	77	0.0	100	0.0	100	0.0
10	1,000	0	–	47	0.0	49	0.0	48	0.0
100	1,000	2	0.0	89	0.0	100	0.0	100	0.0
				$R_1 = 1{,}000$					
10	10	0	–	2	4.5	100	11.5	100	10.0
100	10	0	–	8	31.6	100	31.7	100	31.7
1,000	10	0	–	4	31.4	100	31.9	100	31.9
10,000	10	0	–	19	31.9	100	31.8	100	31.8
10	100	0	–	6	0.0	99	0.0	99	0.0
100	100	0	–	15	1.1	100	0.8	100	0.7
1,000	100	0	–	33	0.0	100	0.0	100	0.0
10	1,000	0	–	19	0.0	57	0.0	53	0.0
100	1,000	0	–	49	0.0	100	0.0	100	0.0
				$R_2 = 10{,}000$					
averages		0.1	0.0	33.9	8.0	94.6	12.6	94.3	12.0

Table 5: Instances of Type SZ

- Pricing UC instances seems to be most easy (see table 1). Here in many cases the linear programming relaxation (2) is integral. Moreover, model (3) only in one case does not yield an integral solution while models (4) and (5) always produce them. On average the solution of model (3) shows larger dual degeneracy than the other two models do.

- The picture for SZ is very much the same as for UC, except the fact that the linear programming relaxation (2) is only in two cases integral (see table 5). Moreover, model (3) rarely produced integral solutions which the other two reformulations did in most of the cases considered.

- For the remaining three distribution types we do not have a unique picture (except the fact that the linear programming relaxation (2) rarely is integral). Once more models (4) and (5) in general are more powerful in terms of producing integral solutions than model (3).

- Surprisingly, w.r.t. the models (4) and (5) for the distribution type SC dual degeneracy never appeared (see table 3), a fact for which we do not have an explanation. For the other distribution types this phenomenon could be observed in some cases (in particular for most of the larger SZ instances).

- Noteworthy to mention is the particular case where the linear programming relaxation produces an integral solution for five instances while the other models do achieve this only for three: see type SS, R_1=1,000, k=10, n_i=100.

Summarizing overall our methodology produces an integral solution for 6,756 of the 9,000 instances considered, or in other words, in a bit more than 75 % of the cases.

5 Summary and future work

In this paper we have provided a family of linear programming models for the MCKP the solution of which is integral very often. The models are evaluated experimentally using a well-defined testbed consisting of 9,000 instances. Overall our methodology produces an integral solution in roughly 75 % of the cases. In particular, for two out of five distribution types studied at least one of the models produces "almost always" an integral solution. Hence, in most cases dual prices are readily available.

Subsequently we will enhance the linear programs by valid inequalities so as to get linear prices also for the more difficult instances.

Acknowledgement

The authors are indebted to Christof Kluß for professionally coding the algorithms.

References

[1] ARMSTROMG, R.D., KUNG, D.S., SINHA, P., ZOLTNERS, A.A. (1983), A computational study of a multiple-choice knapsack algorithm, ACM Transactions on Mathematical Software, Vol. 9, pp. 184–198

[2] BALAS, E., ZEMEL, E. (1980), An algorithm for large zero-one knapsack problems, Operations Research, Vol. 28, pp. 1130–1154

[3] BARCIA, P., JÖRNSTEN, K. (1990), Improved Lagrangean decomposition: an application to the generalized assignment problem, European Journal of Operational Research, Vol. 46, pp. 84–92

[4] DE LEONE, R., JAIN, R., STRAUS, K. (1993), Solution of multiple-choice knapsack problem encountered in high-level synthesis of VLSI circuits, Journal of Computer Mathematics, Vol. 47, pp. 163–176

[5] DUDZIŃSKI, K., WALUKIEWICZ, S. (1984), A fast algorithm for the linear multiple-choice knapsack problem, Operations Research Letters, Vol. 3, pp. 205–209

[6] DUDZIŃSKI, K., WALUKIEWICZ, S. (1987), Exact methods for the knapsack problem and its generalizations, European Journal of Operational Research, Vol. 28, pp. 3–21

[7] DYER, M.E. (1984), An $O(n)$ algorithm for the multiple-choice knapsack linear problem, Mathematical Programming, Vol. 29, pp. 57–63

[8] DYER, M.E., KAYAL, N., WALKER, J. (1984), A branch and bound algorithm for solving the multiple-choice knapsack problem, Journal of Computational and Applied Mathematics, Vol. 11, pp. 231–249

[9] FISHER, M.L. (1981), The Lagrangian relaxation method for solving integer programming problems, Management Science, Vol. 27, pp. 1–18

[10] `http://www.diku.dk/~pisinger/codes.html`

[11] JOHNSON, E.L., PADBERG, M.W. (1981), A note on the knapsack problem with special ordered sets, Operations Research Letters, Vol. 1, pp. 18–22

[12] KELLERER, H., PFERSCHY, U., PISINGER, D. (2004), *Knapsack Problems*, Springer, Berlin

[13] KELLY, T. (2004), Generalized knapsack solvers for multi-unit combinatorial auctions: analysis and application to computational resource allocation, Working Paper, Hewlett-Packard Labs, Palo Alto

[14] MARTELLO, S., TOTH, P. (1990), *Knapsack Problems: Algorithms and Computer Implementations*, Wiley, Chichester

[15] MEJIA-ALVAREZ, P., LEVNER, E.V., MOSSE, D. (2002), An integrated heuristic approach to power-aware real-time scheduling. In *International Workshop on Power Aware Computer Systems (PACS'02)*, LNCS 2325, Springer

[16] NAUSS, R.M. (1978), The 0-1 knapsack problem with multiple choice constraints, European Journal of Operational Research, Vol. 2, pp. 125–131

[17] PISINGER, D. (1995), A minimal algorithm for the multiple-choice nested knapsack problem, European Journal of Operational Research, Vol. 83, pp. 394–410

[18] SINHA, A., ZOLTNERS, A.A. (1979), The multiple-choice knapsack problem, Operations Research, Vol. 27, pp. 503–515

[19] ZEMEL, E. (1984), An $O(n)$ algorithm for the linear multiple choice knapsack problem and related problems, Information Processing Letters, Vol. 18, pp. 123–128