RAIRO-Oper. Res. 50 (2016) 995–1001
DOI: 10.1051/ro/2015061

RAIRO Operations Research
www.rairo-ro.org

AN IMPROVED BINARY SEARCH ALGORITHM FOR THE MULTIPLE-CHOICE KNAPSACK PROBLEM *

Cheng He[1], Joseph Y-T. Leung[2], Kangbok Lee[3] and Michael L. Pinedo[4]

Abstract. The Multiple-Choice Knapsack Problem is defined as a 0-1 Knapsack Problem with additional disjoint multiple-choice constraints. Gens and Levner presented for this problem an approximate binary search algorithm with a worst case ratio of 5. We present an improved approximate binary search algorithm with a ratio of $3 + (\frac{1}{2})^t$ and a running time $O(n(t + \log m))$, where n is the number of items, m the number of classes, and t a positive integer. We then extend our algorithm to make it also applicable to the Multiple-Choice Multidimensional Knapsack Problem with dimension d.

Mathematics Subject Classification. 68Q25, 90C10, 90C27.

Received July 15, 2015. Accepted September 24, 2015.

1. Introduction

The Multiple-Choice Knapsack Problem (MCKP) can be described as follows. We are given m classes $N_1, N_2, \ldots, N_m$ of items, that are mutually disjoint, and that have to be packed into a knapsack with capacity b. Class N_i contains n_i items and we refer to the jth item of the ith multiple-choice class as item (i, j). Item (i, j) has a profit c_{ij} and a weight a_{ij}, where c_{ij}, a_{ij} ($1 \le i \le m$ and $1 \le j \le n_i$) and b are positive integers. Thus, the total number of items is $n = \sum_{i=1}^{m} n_i$. We are supposed to choose at most one item from each class such that the total profit is maximized and the total weight does not exceed the capacity b. Therefore, the MCKP may be formulated with $X = (x_{ij})$ as:

$$\text{maximize} \quad f(X) = \sum_{i=1}^{m} \sum_{j \in N_i} c_{ij} x_{ij}$$

Keywords. Multiple-Choice Knapsack Problem (MCKP), Approximate binary search algorithm, Worst-case performance ratio, Multiple-choice Multi-dimensional Knapsack Problem (MMKP).

* *This work was supported by NSFC (grant No. 11201121) and CSC (201309895008) and (Young Backbone Teachers of Henan Colleges 2013GGJS-079) and PSC CUNY (Grant TRADA-46-477).*

[1] School of Science, Henan University of Technology, Zhengzhou, Henan 450001, P.R. China. hech202@163.com

[2] Department of Computer Science, New Jersey Institute of Technology, Newark NJ-07102, USA.

[3] Department of Business and Economics, York College, The City University of New York, 94-20 Guy R. Brewer Blvd, Jamaica, New York 11451, USA.

[4] Department of Information, Operations and Management Sciences, Stern School of Business, New York University, 44 West 4th Street, New York 10012-1126, USA.

Article published by EDP Sciences

$$\text{subject to} \quad \sum_{i=1}^{m} \sum_{j \in N_i} a_{ij} x_{ij} \leq b$$

$$\sum_{j \in N_i} x_{ij} \leq 1,\ i = 1, 2, \ldots, m$$

$$x_{ij} \in \{0, 1\},\ i = 1, 2, \ldots, m;\ j \in N_i.$$

MCKP has been extensively studied (see *e.g.*, Armstrong *et al.* [1], Pisinger [9], and Lawler [6]). It has practical applications in various areas such as capital investment and planning choice in transportation.

Lawler [6] developed a fully polynomial time approximation scheme (FPTAS) for the problem which runs in time $O(n \log n + (mn)/\epsilon)$. Thus, an approximation algorithm with the same or of higher time complexity would not be of interest. We propose an approximation algorithm with a lower time complexity and a constant bound.

Gens and Levner [4] presented an approximate binary search algorithm for finding an approximate solution for the MCKP. For every instance I of MCKP, let $f^0(I)$ and $f^*(I)$ be the solution values obtained by the algorithm and by an optimal algorithm, respectively. Gens and Levner proved that $f^*(I)/f^0(I) \leq 5$ and its running time is $O(n \log m)$.

The multiple-choice multidimensional knapsack problem (MMKP) is a generalization of MCKP; the weight of each item is now a vector and the total weight of selected items cannot exceed the capacity which is now also a vector. Since MMKP is also related to the conventional multidimensional knapsack problem it has a variety of applications in practice and is receiving more and more attention lately. Chen and Hao [2] summarized the recent results and categorized them into two groups: exact methods (*e.g.*, branch-and-bound) and heuristic approaches (*e.g.*, local searches, relaxation based heuristics, meta heuristics).

Frieze and Clarke [3] presented a polynomial time approximation scheme (PTAS) for the (single-choice) multidimensional variant of knapsack, and Magazine and Chern [7] showed that obtaining an FPTAS for multidimensional knapsack is NP-hard. Patt-Shamir and Rawitz [8] developed an improved PTAS for MMKP and its time complexity is $O\left((nm)^q\right)$ where $q = \min\{n, d/\epsilon\}$. Thus, by setting ϵ to be d, a $(1+d)$-approximation solution can be obtained in $O(nm)$ time. We propose an approximation algorithm with a lower time complexity and a constant bound.

For more details on the knapsack problem and its variants, we refer the reader to [5].

In Section 2, we provide an improved branching algorithm and, based on this improved branching algorithm, we present in Section 3 an improved algorithm with a ratio of $3 + (\frac{1}{2})^t$ and with a running time $O(n(t + \log m))$, t being a positive integer. Furthermore, in Section 4, we generalize this solution approach to a multidimensional version of the problem and present an approximate binary search algorithm with a ratio of $1 + 2d + (\frac{1}{2})^t$ and a running time of $O(n(t + \log(m - 2d)))$, t being a positive integer.

2. An improved branching algorithm

In the (exact) binary search algorithm, when the optimum objective function value f^* lies within a search interval $[L, U]$, for a given value x, there is a method $\mathbf{M}$ for determining whether $f^* < x$ or $f^* > x$. Thus, if one takes the value $x = (U - L)/2$ and applies method $\mathbf{M}$, the length of the interval $[L, U]$ will be reduced by a factor of 2. The iterative process will then be terminated in no more than $\log_2(U - L)$ steps.

Unlike the exact binary search, in the approximate binary search, we use a rougher computation that determines whether $f^* < x(1 + \epsilon_1)$ or $f^* > x(1 - \epsilon_2)$ for some positive ϵ_1 and ϵ_2.

Branching Algorithm BA(x)
Step 0. Let $L \leq f^* \leq U$, and $x \in [L, U]$ be a given value. Let $i := 1$ and $J := \emptyset$ and $C(x) := 0$.
Step 1. If $i > m$, then STOP. Otherwise let $p_{ij} := c_{ij}/a_{ij}$ for any $j \in N_i$ and $N_i' := \{j | p_{ij} \geq x/b\}$.
Step 2. If $N_i' = \emptyset$, then let $i := i + 1$ and go back to Step 1. Otherwise choose the item j_i with the largest c_{ij_i} from N_i'. Let $J := J \bigcup \{i\}$ and $C(x) := C(x) + c_{ij_i}$ and $i := i + 1$ and go back to Step 1.

Theorem 2.1. *Suppose $C(x)$ is the value obtained by Branching Algorithm $BA(x)$.*

(i) *If $C(x) \geq 0.5x$, then $f^* > 0.5x$.*
(ii) *If $C(x) < 0.5x$, then $f^* < 1.5x$.*

Proof.
(i) Assume that $C(x) \geq 0.5x$. If $\sum_{i \in J} a_{ij_i} \leq b$, then $X = \{x_{ij} | x_{ij} = 1 \text{ if } i \in J \text{ and } j = j_i; \text{ otherwise } x_{ij} = 0\}$ is a feasible solution for the MCKP. So $f^* \geq \sum_{i \in J} c_{ij_i} \geq 0.5x$. On the other hand, if $\sum_{i \in J} a_{ij_i} > b$, then let $I \subset J$ and $k \in J \backslash I$ with $\sum_{i \in I} a_{ij_i} < b$ and $\sum_{i \in I} a_{ij_i} + a_{kj_k} \geq b$. Since $p_{ij_i} = c_{ij_i}/a_{ij_i} \geq x/b$ for any $i \in J$, we have

$$2f^* \geq \sum_{i \in I} c_{ij_i} + c_{kj_k} = \sum_{i \in I} a_{ij_i} p_{ij_i} + a_{kj_k} p_{kj_k} \geq x/b(\sum_{i \in I} a_{ij_i} + a_{kj_k}) > x.$$

Hence $f^* > 0.5x$.

(ii) Assume that $C(x) < 0.5x$. Let X^* be an optimal solution for MCKP instance. Let

$$I_1 = \{(i,j) \mid x^*_{ij} = 1 \text{ and } p_{ij} = c_{ij}/a_{ij} < x/b\},$$

$$I_2 = \{(i,j) \mid x^*_{ij} = 1 \text{ and } p_{ij} = c_{ij}/a_{ij} \geq x/b\}.$$

Then,

$$f^* = \sum_{1 \leq i \leq m} \sum_{j \in N_i} c_{ij} x^*_{ij} = \sum_{(i,j): x^*_{ij} = 1} c_{ij} = \sum_{(i,j) \in I_1} c_{ij} + \sum_{(i,j) \in I_2} c_{ij}.$$

Since

$$\sum_{(i,j) \in I_1} c_{ij} < x/b \sum_{(i,j) \in I_1} a_{ij} < x \quad \text{and} \quad \sum_{(i,j) \in I_2} c_{ij} \leq \sum_{i \in J} c_{ij} < 0.5x,$$

we have $f^* < 1.5x$. This completes the proof. □

3. An improved approximate binary search Algorithm

For a positive integer t, we define the Binary Search Algorithm(t) as follows.

Binary Search Algorithm(t)
Step 0. Let $L := \max_{i,j}\{c_{ij}\}$, $L_0 := L$, $U_0 := mL$, $x_0 := \frac{1}{3}U_0 + L_0$ and $k := 0$.
Step 1. Perform the Branching Algorithm BA(x_k). Determine whether $C(x_k) \geq 0.5x_k$ or $C(x_k) < 0.5x_k$. Let $k := k + 1$.
Step 2. If $C(x_{k-1}) \geq 0.5x_{k-1}$, then let $L_k := 0.5x_{k-1}$ and $U_k := U_{k-1}$ and go to Step 3. Otherwise let $U_k := 1.5x_{k-1}$ and $L_k := L_{k-1}$ and go to Step 3.
Step 3. If $U_k - 3L_k \leq (\frac{1}{2})^t L$, then let $f^0 := L_k$ and STOP; otherwise let $x_k = \frac{1}{3}U_k + L_k$ and go back to Step 1.

Note that if $U_k - 3L_k > (\frac{1}{2})^t L$, then $U_k > 3L_k$. Also, we have $x_k = \frac{1}{3}U_k + L_k > L_k$ and $x_k = \frac{1}{3}U_k + L_k < \frac{1}{3}U_k + \frac{1}{3}U_k = \frac{2}{3}U_k < U_k$. Hence, $L_k < x_k < U_k$. And by Theorem 2.1, we note that $L_k \leq f^* \leq U_k$ at any step k. And when the algorithm terminates, we find an approximation value $f^0 = L_k$.

Theorem 3.1. *The Binary Search Algorithm(t) can find an approximate value of the MCKP with a ratio of at most $3 + (\frac{1}{2})^t$ in $O(n(t + \log m))$ time.*

Proof.
If $C(x_{k-1}) \geq 0.5x_{k-1}$, then $L_k := 0.5x_{k-1}$ and $U_k := U_{k-1}$. Thus, we have

$$U_k - 3L_k = U_{k-1} - 3 \cdot \frac{1}{2} \times \left(\frac{1}{3}U_{k-1} + L_{k-1}\right) = \frac{1}{2}(U_{k-1} - 3L_{k-1}).$$

If $C(x_{k-1}) < 0.5x_{k-1}$, then $U_k := 1.5x_{k-1}$ and $L_k := L_{k-1}$. Thus, we have

$$U_k - 3L_k = \frac{3}{2} \cdot \left(\frac{1}{3}U_{k-1} + L_{k-1}\right) - 3L_{k-1} = \frac{1}{2}(U_{k-1} - 3L_{k-1}).$$

Since $U_k - 3L_k = \frac{1}{2}(U_{k-1} - 3L_{k-1})$, $U_k - 3L_k$ will decrease exponentially and as k approaches infinity, $U_k - 3L_k$ converges to zero. Thus, we have $\lim_{k\to\infty} U_k = 3(\lim_{k\to\infty} L_k)$. Let the number of required iterations be p. Since $U_0 - 3L_0 = (m-3)L$ and $(m-3)L \cdot (\frac{1}{2})^p \leq (\frac{1}{2})^t L$, we have $p \geq t + \log_2(m-3)$. Thus, we can set $p = \lceil t + \log_2(m-3)\rceil$. Therefore, the algorithm terminates with

$$\frac{f^*}{f^0} \leq \frac{U_k}{L_k} \leq \frac{3L_k + (\frac{1}{2})^t L}{L_k} \leq 3 + \left(\frac{1}{2}\right)^t$$

after at most $O(t + \log m)$ iterations. As for the running time of the algorithm, we see that there are at most $O(t + \log m)$ rounds and each round of Branching Algorithm BA(x) needs $O(n)$ time. Hence the total running time is $O(n(t + \log m))$. $\square$

Note that $3 < 3 + \left(\frac{1}{2}\right)^t \leq 4$. Even for $t = 0$, the worst case performance ratio is 4, which is better than the one by Gens and Levner [4]. In order not to increase the time complexity, t should be at most $O(\log m)$.

4. Extension to d-dimensional MMKP

We can generalize the current approach to a d-dimensional problem, where $d \leq (m-1)/2$. This problem is a special case of the Multiple-choice Multidimensional Knapsack Problem (MMKP). The special d-dimensional MMKP can be formulated with $X = (x_{ij})$ as:

$$\begin{aligned} \text{maximize} \quad & f(X) = \sum_{i=1}^{m} \sum_{j \in N_i} c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{m} \sum_{j \in N_i} a_{ij}^h x_{ij} \leq b, \quad h = 1, \ldots, d \\ & \sum_{j \in N_i} x_{ij} \leq 1, \quad i = 1, 2, \ldots, m \\ & x_{ij} \in \{0, 1\}, \quad i = 1, 2, \ldots, m;\ j \in N_i. \end{aligned}$$

We generalize the Branching Algorithm and the Binary Search Algorithm as follows.

Branching Algorithm BA(x)
Step 0. Let $L \leq f^* \leq U$, and $x \in [L, U]$ be a given value. Let $i := 1$ and $J := \emptyset$ and $C(x) := 0$.
Step 1. If $i > m$, then STOP. Otherwise let $p_{ij} := c_{ij}/(\sum_{h=1}^{d} a_{ij}^h)$ for any $j \in N_i$ and $N_i' := \{j | p_{ij} \geq x/(d \cdot b)\}$.
Step 2. If $N_i' = \emptyset$, then let $i := i + 1$ and go back to Step 1. Otherwise choose the item j_i with the largest c_{ij_i} from N_i'. Let $J := J \bigcup \{i\}$ and $C(x) := C(x) + c_{ij_i}$ and $i := i + 1$ and go back to Step 1.

Theorem 4.1. *Suppose $C(x)$ is the value obtained by the Branching Algorithm BA(x).*

(i) *If $C(x) \geq \frac{1}{2d}x$, then $f^* > \frac{1}{2d}x$.*

(ii) *If* $C(x) < \frac{1}{2d}x$, *then* $f^* < (1 + \frac{1}{2d})x$.

Proof.
(i) Assume that $C(x) \geq \frac{1}{2d}x$. We consider two sub-cases (a) and (b):
(a) If $\sum_{i \in J} a^h_{ij_i} \leq b$ for all $h = 1, \ldots, d$, then $X = \{x_{ij} \mid x_{ij} = 1 \text{ if } i \in J \text{ and } j = j_i; \text{ otherwise } x_{ij} = 0\}$ is a feasible solution for the MMKP. So $f^* \geq \sum_{i \in J} c_{ij_i} \geq \frac{1}{2d}x$.
(b) Otherwise, we can define sets of items $\emptyset = I_0 \subset I_1 \subset I_2 \subset \ldots \subset I_K \subset J$ for some $2 \leq K \leq d$ such that

$$\left|\left\{h \mid \sum_{i \in I_k} a^h_{ij_i} > b\right\}\right| \geq \left|\left\{h \mid \sum_{i \in I_{k-1}} a^h_{ij_i} > b\right\}\right| + 1 \quad \text{and}$$

$$\sum_{i \in I_k \setminus I_{k-1}} a^h_{ij_i} \leq b \quad \text{for} \quad k = 1, \ldots, K.$$

Since $I_k \setminus I_{k-1}$ corresponds to a feasible solution, we have

$$f^* \geq \sum_{i \in I_k \setminus I_{k-1}} c_{ij_i} \quad \text{for} \quad k = 1, \ldots, K,$$

and thus we have

$$Kf^* \geq \sum_{k=1}^{K} \sum_{i \in I_k \setminus I_{k-1}} c_{ij_i}.$$

Since $c_{ij_i}/(\sum_{h=1}^{d} a^h_{ij_i}) \geq x/(d \cdot b)$ for any $i \in J$, we have

$$\sum_{k=1}^{K} \sum_{i \in I_k \setminus I_{k-1}} c_{ij_i} \geq \frac{x}{d \cdot b}\left(\sum_{k=1}^{K} \sum_{i \in I_k \setminus I_{k-1}} \sum_{h=1}^{d} a^h_{ij_i}\right)$$

$$> \frac{x}{d \cdot b}(b + 2b + \ldots + (K-1)b) = \frac{x}{d \cdot b}\left(\frac{K(K-1)}{2}b\right) = \frac{K(K-1)}{2d}x.$$

By combining the above two inequalities, we have

$$Kf^* \geq \sum_{k=1}^{K} \sum_{i \in I_k \setminus I_{k-1}} c_{ij_i} > \frac{K(K-1)}{2d}x.$$

Hence, since $K \geq 2$,

$$f^* > \frac{K-1}{2d}x \geq \frac{1}{2d}x.$$

(ii) Assume that $C(x) < \frac{1}{2d}x$. Let X^* be an optimal solution for the MMKP, and

$$I_1 = \left\{(i,j) \mid x^*_{ij} = 1 \text{ and } c_{ij}/\left(\sum_{h=1}^{d} a^h_{ij_i}\right) < x/(d \cdot b)\right\},$$

$$I_2 = \left\{(i,j) \mid x^*_{ij} = 1 \text{ and } c_{ij}/\left(\sum_{h=1}^{d} a^h_{ij_i}\right) \geq x/(d \cdot b)\right\}.$$

Then,

$$f^* = \sum_{i=1}^{m} \sum_{j \in N_i} c_{ij} x_{ij}^* = \sum_{(i,j): x_{ij}^* = 1} c_{ij} = \sum_{(i,j) \in I_1} c_{ij} + \sum_{(i,j) \in I_2} c_{ij}.$$

Since

$$\sum_{(i,j) \in I_1} c_{ij} < \frac{x}{d \cdot b} \sum_{(i,j) \in I_1} \sum_{h=1}^{d} a_{ij_i}^h = \frac{x}{d \cdot b} \left\{ \sum_{(i,j) \in I_1} \sum_{h=1}^{d} a_{ij}^h \right\} \leq \frac{x}{d \cdot b}(d \cdot b) = x$$

and

$$\sum_{(i,j) \in I_2} c_{ij} \leq \sum_{i \in J} c_{ij} < \frac{1}{2d} x,$$

we have $f^* < \left(1 + \frac{1}{2d}\right) x$. This completes the proof. □

For a positive integer t, we define the following Binary Search Algorithm(t).

Binary Search Algorithm(t)
Step 0. Let $L := \max_{i,j}\{c_{ij}\}$, $L_0 := L$, $U_0 := mL$, $x_0 := \frac{d}{1+2d} U_0 + dL_0$ and $k := 0$.
Step 1. Apply the Branching Algorithm BA(x_k). Determine whether $C(x_k) \geq \frac{1}{2d} x_k$ or $C(x_k) < \frac{1}{2d} x_k$. Let $k := k + 1$.
Step 2. If $C(x_{k-1}) \geq \frac{1}{2d} x_{k-1}$, then let $L_k := \frac{1}{2d} x_{k-1}$ and $U_k := U_{k-1}$ and go to Step 3. If $C(x_{k-1}) < \frac{1}{2d} x_{k-1}$, then let $U_k := \left(1 + \frac{1}{2d}\right) x_{k-1}$ and $L_k := L_{k-1}$ and go to Step 3.
Step 3. If $U_k - (1 + 2d)L_k \leq (\frac{1}{2})^t L$, then let $f^0 := L_k$ and STOP; otherwise let $x_k := \frac{d}{1+2d} U_k + dL_k$ and go back to Step 1.

Note that if $U_k - (1 + 2d)L_k > (\frac{1}{2})^t L$, then $U_k > (1 + 2d)L_k$. Also, $x_k = \frac{d}{1+2d} U_k + dL_k > 2dL_k > L_k$ and $x_k = \frac{d}{1+2d} U_k + dL_k < \frac{d}{1+2d} U_k + \frac{d}{1+2d} U_k = \frac{2d}{1+2d} U_k < U_k$. Thus, $L_k < x_k < U_k$.

Theorem 4.2. *The Binary Search Algorithm(t) can find an approximate value of the d-dimensional MMKP with a ratio of at most* $1 + 2d + (\frac{1}{2})^t$ *in* $O(n(t + \log(m - 2d)))$ *time.*

Proof.
If $C(x_{k-1}) \geq \frac{1}{2d} x_{k-1}$, then $L_k := \frac{1}{2d} x_{k-1}$ and $U_k := U_{k-1}$. Thus, we have

$$U_k - (1 + 2d)L_k = U_{k-1} - (1 + 2d) \frac{1}{2d} \left(\frac{d}{1 + 2d} U_{k-1} + dL_{k-1} \right)$$

$$= \frac{1}{2} \left\{ U_{k-1} - (1 + 2d)L_{k-1} \right\}.$$

If $C(x_{k-1}) < \frac{1}{2d} x_{k-1}$, then $U_k := (1 + \frac{1}{2d}) x_{k-1}$ and $L_k := L_{k-1}$. Thus, we have

$$\begin{aligned} U_k - (1 + 2d)L_k &= \left(1 + \frac{1}{2d}\right) \left(\frac{d}{1 + 2d} U_{k-1} + dL_{k-1} \right) - (1 + 2d)L_{k-1} \\ &= \frac{1}{2} \left\{ U_{k-1} - (1 + 2d)L_{k-1} \right\}. \end{aligned}$$

Since $U_k - (1 + 2d)L_k = \frac{1}{2}\{U_{k-1} - (1 + 2d)L_{k-1}\}$, $U_k - (1 + 2d)L_k$ will decrease exponentially and as k approaches infinity, $U_k - (1 + 2d)L_k$ converges to zero. Thus, we have $\lim_{k \to \infty} U_k = (1 + 2d)(\lim_{k \to \infty} L_k)$.

Let the number of required iterations be p. Since $U_0-(1+2d)L_0=(m-(1+2d))L$ and $(m-(1+2d))L\cdot(\frac{1}{2})^p\leq(\frac{1}{2})^tL$, we have $p\geq t+\log_2(m-(1+2d))$. Thus, we can set $p=\lceil t+\log_2(m-(1+2d))\rceil$. Therefore, the algorithm terminates with

$$\frac{f^*}{f^0}\leq\frac{U_k}{L_k}\leq\frac{(1+2d)L_k+(\frac{1}{2})^tL}{L_k}\leq 1+2d+\left(\frac{1}{2}\right)^t$$

after at most $O(t+\log(m-2d))$ iterations.

As for the running time of the algorithm, we see that there are at most $O(t+\log(m-2d))$ rounds and each round of Branching Algorithm BA(x) needs $O(n)$ time. Hence the total running time is $O(n(t+\log(m-2d)))$. $\square$

References

[1] R.D. Armstrong, D.S. Kung, P. Sinha and A.A. Zoltners, A computational study of a multiple-choice knapsack algorithm. *ACM Trans. Math. Software* **9** (1983) 184–198.

[2] Y. Chen and J.-K. Hao, A "reduce and solve" approach for the multiple-choice multidimensional knapsack problem. *Eur. J. Oper. Res.* **239** (2014) 313–322.

[3] A.M. Frieze and M.R.B. Clarke, Approximation algorithms for them-dimensional 0-1 knapsack problem: worst-case and probabilistic analyses. *Eur. J. Operat. Res.* **15** (1984) 100–109.

[4] G. Gens and E. Levner, An approximate binary search algorithm for the multiple-choice knapsack problem. *Inf. Process. Lett.* **67** (1998) 261–265.

[5] H. Kellerer, U. Pferschy and D. Pisinger, *Knapsack problems*. Springer (2004).

[6] E.L. Lawler, Fast Approximation Algorithms for Knapsack Problems. *Math. Oper. Res.* **4** (1979) 339–356.

[7] M.J. Magazine and M.-S. Chern, A note on approximation schemes for multidimensional knapsack problems. *Math. Oper. Res.* **9** (1984) 244–247.

[8] B. Patt-Shamir and D. Rawitz, Vector bin packing with multiple-choice. *Discrete Appl. Math.* **160** (2012) 1591–1600.

[9] D. Pisinger, A minimal algorithm for the Multiple-Choice Knapsack Problem. *Eur. J. Oper. Res.* **83** (1995) 394–410.