Improved Fully Polynomial time Approximation Scheme for the 0-1 Multiple-choice Knapsack Problem

Mukul Subodh Bansal* V.Ch.Venkaiah
International Institute of Information Technology
Hyderabad, India

Abstract

In this paper the 0-1 Multiple-Choice Knapsack Problem (0-1 MCKP), a generalization of the classical 0-1 Knapsack problem, is addressed. We present a fast Fully Polynomial Time Approximation Scheme (FPTAS) for the 0-1 MCKP, which yields a better time bound than known algorithms. In particular it produces a $(1+\epsilon)$ approximate solution and runs in $\mathbf{O}(nm/\epsilon)$ time, where n is the number of items and m is the number of multiple-choice classes.

Keywords: Approximation Algorithm, FPTAS, 0-1 Multiple-choice Knapsack.

1 Introduction

The 0-1 Multiple-Choice Knapsack Problem (0-1 MCKP) is a generalization of the classical 0-1 Knapsack problem. In this problem, we are given m classes $N_1, N_2, \ldots, N_m$ 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}, and the problem is to choose at most one item from each class such that the profit sum is maximized without the weight sum exceeding c. The 0-1 MCKP may thus be formulated as:

$$\text{maximize} \quad z = \sum_{i=1}^{m} \sum_{j \in N_i} p_{ij} x_{ij}$$

$$\text{Subject to} \quad \sum_{i=1}^{m} \sum_{j \in N_i} w_{ij} x_{ij} \leq c,$$

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

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

All coefficients p_{ij}, w_{ij} and c are positive integers, and the classes $N_1, \ldots, N_m$ are mutually disjoint, with class N_i having size n_i. The total number of items is thus $n = \sum_{i=1}^{m} n_i$.

*`Email:bansal@cs.iastate.edu`

To avoid unsolvable or trivial situations we can assume:

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

0-1 MCKP is NP-hard [4] as it contains the 0-1 Knapsack problem as a special case [5, 13] but it can be solved in pseudo-polynomial time through dynamic programming [7, 2]. The problem has numerous applications to the design of industrial and communication systems. Other applications arise in Capital Budgeting [14], Menu Planning [16], transforming nonlinear Knapsack Problems (KP) to MCKP [14], determining which components should be linked in series in order to maximize fault tolerance [16] etc.

Several algorithms for MCKP have been presented during the last three decades [15, 14, 16, 9, 7]. The MCKP is defined as follows: Given m classes $N_1, N_2, \ldots, N_m$ 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}, then the MCKP is to choose exactly one item from each class such that the profit sum is maximized without the weight sum exceeding c. Note that algorithms for the MCKP can easily be used to solve the 0-1 MCKP by introducing a dummy object having zero profit and zero weight in each multiple-choice class. Approximation algorithms for the 0-1 MCKP are presented in [12, 10]. Lawler, in [12], gave a very efficient and classic FPTAS for the problem which runs in time $\mathbf{O}(n \log n + nm/\epsilon)$. Although it has been over two decades and many of the results presented in [12] have been improved, the FPTAS of Lawler for the 0-1 MCKP has remained the fastest. While the running time of Lawler [12] is very efficient, it has been a nagging question whether the running time can be made to match $\mathbf{O}(nm/\epsilon)$. In particular, can we give an approximation scheme without even sorting the objects and using the technique of d'Atri [6] (explained in [12]) as used by Lawler.

We answer this question in the affirmative. We give an approximation scheme which has a running time of $\mathbf{O}(nm/\epsilon)$. Our source of improvement in the running time is a novel technique where we replace the technique of d'Atri used in [12] by a more involved procedure involving finding the solution to a continuous relaxation of the 0-1 MCKP and deriving efficient upper and lower bounds on the optimal profit value based on the solution to the continuous problem. We believe that this idea is quite general and might find other uses elsewhere.

This paper is organized as follows. In the next subsection we introduce the terminology for approximation algorithms. In section 2, we develop the FPTAS for the 0-1 MCKP in a step by step manner, while maintaining the correctness and analyzing the complexity of the algorithm at each step. In section 3 we present a summary of the algorithm, prove its correctness, analyze the complexity and prove that the algorithm is indeed an FPTAS for the MCKP. Concluding remarks are made in section 4.

1.1 Terminology for Approximation Algorithms

Let Π be an optimization problem with objective function f_Π, with optimal solution S^*, and optimal value OPT(Π).

An algorithm $\mathbf{Z}$ is an *approximation scheme* for the problem Π if for input (I, ϵ), where I is an instance of Π and $\epsilon > 0$ is an error parameter, $\mathbf{Z}$ outputs a solution S_Z satisfying

$$f_\Pi(I; S_Z) \le (1+\epsilon)\mathrm{OPT}(\Pi)$$

if Π is a minimization problem and,

$$f_\Pi(I; S_Z) \ge (1-\epsilon)\mathrm{OPT}(\Pi)$$

if Π is a maximization problem.

The approximation scheme **Z** is called a *polynomial time approximation scheme* (or PTAS) if for all fixed input (I, ϵ), the running time of **Z** is polynomial in the size of I.

Z is a *fully polynomial time approximation scheme* (or FPTAS) if it is a PTAS with running time bounded by a polynomial in the size of I, $\frac{1}{\epsilon}$. In a very technical sense, an FPTAS is the best one can hope for when considering an **NP**-hard optimization problem assuming $\mathbf{P} \ne \mathbf{NP}$.

2 A Fast FPTAS for the 0-1 MCKP

In this section we develop the algorithm in a step by step fashion, successively improving the algorithm in each step, while maintaining the correctness of the algorithm.

Let the *unary size* of instance I, denoted by $|I_u|$, be defined as the number of bits needed to write I_u in unary. An algorithm for problem Π whose running time on instance I is bounded by a polynomial in $|I_u|$ is called a *pseudo-polynomial time algorithm.*

The 0-1 MCKP, being **NP**-hard, does not admit a polynomial time algorithm. However, it does admit a pseudo-polynomial time algorithm. This fact is used critically in obtaining an FPTAS for the 0-1 MCKP. All known pseudo-polynomial time algorithms for **NP**-hard problems are based on dynamic programming.

2.1 An Efficient, Exact Pseudo-Polynomial Time Algorithm for the 0-1 MCKP

Given m subsets (classes) of items (or objects), $N_1, \ldots, N_m$, to be packed in some knapsack of capacity c. Each item $j \in N_i$, denoted by a_{ij}, has a profit p_{ij} and a weight w_{ij}, and the problem is to choose one item from each class such that the profit sum is maximized without the weight sum exceeding c. Let class N_i have size n_i. The total number of items is thus $n = \sum_{i=1}^{m} n_i$.

Let P^* denote the optimal profit. In order to make the FPTAS efficient it is essential to compute an effective upper-bound on the value of the optimal profit. The $n \log n$ term in the running time of the algorithm of [12] comes from this step of estimating an upper bound P_0 such that $P_0 \le P^* \le 2P_0$. We shall now describe how to efficiently compute an upper-bound on the value of P^* such that $P_0 \le P^* \le 3P_0$. Dyer [8] and Zemel [18] developed $\mathbf{O}(n)$ time algorithms for the Continuous MCKP (C(MCKP)). The C(MCKP) is a relaxation of the MCKP. C(MCKP) can be formulated as

$$\text{maximize} \quad z = \sum_{i=1}^{m} \sum_{j \in N_i} p_{ij} x_{ij}$$

$$\text{Subject to} \quad \sum_{i=1}^{m} \sum_{j \in N_i} w_{ij} x_{ij} \le c,$$

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

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

As mentioned earlier the MCKP can be converted to the 0-1 MCKP by introducing an object having zero profit and zero weight in each multiple-choice class. Thus the algorithm for C(MCKP) is easily modified (by adding a dummy object with profit and weight equal to zero to each multiple-choice class) to solve C(0-1 MCKP). We shall now describe the Dyer-Zemel algorithm for the C(0-1 MCKP) and show how it can be used to obtain P_0 satisfying $P_0 \leq P^* \leq 3P_0$. We assume that the dummy objects have been added for this computation. The Algorithm follows:

Dyer-Zemel Algorithm

1. For all classes N_i: pair the items two by two as (ij_1, ij_2). Order each pair such that $w_{ij_1} \leq w_{ij_2}$ breaking ties such that $p_{ij_1} \geq p_{ij_2}$ when $w_{ij_1} = w_{ij_2}$. If item j_1 dominates item j_2 then delete item j_2 from N_i and pair item j_1 with another item from the class. Continue this process till all items in N_i have been paired (apart from the last one item if $|N_i|$ is odd). Set $P = 0$ and $W = 0$. Note: If two items r and s in the same class N_i satisfy $w_{ir} \leq w_{is}$ and $p_{ir} \geq p_{is}$, then we say that item r dominates item s.

2. For all classes N_i: if the class has only one item j left, then set $P = P + p_{ij}$, $W = W + w_{ij}$ and fathom class N_i.

3. For all pairs (ij_1, ij_2) derive the slope $\gamma_{ij_1ij_2} = \frac{p_{ij_2} - p_{ij_1}}{w_{ij_2} - w_{ij_1}}$.

4. let γ be the median of the slopes $\{\gamma_{ij_1ij_2}\}$.

5. Derive $M_i(\gamma)$ and ϕ_i, ψ_i for $i = 1, \ldots, m$ according to:

   $$\phi_i = \arg \min_{j \in M_i(\gamma)} w_{ij}$$

   $$\psi_i = \arg \max_{j \in M_i(\gamma)} w_{ij}$$

   and

   $$M_i(\gamma) = \left\{ j \in N_i : (p_{ij} - \gamma w_{ij}) = \max_{\ell \in N_i} (p_{i\ell} - \gamma w_{i\ell}) \right\}.$$

6. If γ is optimal, i.e if $W + \sum_{i=1}^{m} w_{i\phi_i} \leq c < W + \sum_{i=1}^{m} w_{i\psi_i}$ then set $W = W + \sum_{i=1}^{m} w_{i\phi_i}$ and $P = P + \sum_{i=1}^{m} p_{i\phi_i}$. An optimal solution to C(0-1 MCKP) is $z^* = P + (c - W)\gamma$. Stop.

7. if $\sum_{i=1}^{m} w_{i\phi_i} \geq c$ then for all pairs (ij_i, ij_2) with $\gamma_{ij_1j_2} \leq \gamma$ delete item j_2.

8. If $\sum_{i=1}^{m} w_{i\psi_i} < c$ then for all pairs with $\gamma_{ij_1j_2} \geq \gamma$ delete item j_1.

9. Go to step 1.

Theorem 2.1.1 *An optimal solution x^* to C(0-1 MCKP)satisfies the following: 1) x^* has at most two fractional variables x_{ab_a} and $x_{ab'_a}$. 2) If x^* has two fractional variables they must be adjacent variables within the same class N_a. 3) If x^* has no fractional variables, then the break solution is an optimal solution to 0-1 MCKP.*

For proofs and further details refer [15, 8, 18, 19].

From the theorem it follows that the optimal solution to C(0-1 MCKP) does not contain more than two fractional variables. Let P_0 be the Maximum of the following three values: 1) profit when the fractional variables are discarded from z^*, 2) the profit from the object corresponding to fractional variable x_{ab_a}, and 3) the profit from the object corresponding to fractional variable $x_{ab'_a}$. Clearly $P_0 \le P^*$. Also, note that, the profit value of the optimal solution to C(0-1 MCKP) must be greater than or equal to the profit value associated with an optimal solution of the 0-1 MCKP. And hence, $P^* \le 3P_0$. Thus, $P_0 \le P^* \le 3P_0$.

For each $i \in \{1, \dots, m\}$ and $p \in \{1, \dots, 3P_0\}$ let $S_{i,p}$ denote a feasible subset of minimum cardinality of elements of $\{N_1 \bigcup \dots \bigcup N_i\}$ (i.e. it does not contain more than one item from each class) whose total profit is exactly p and total weight is minimized. Let $F(i,p)$ denote the weight of the set $S_{i,p}$. $F(i,p) = \infty$ if no such set exists. We assume that $F(i,0) = 0$ for each $i \in \{1, \dots, m\}$. Clearly $F(1,p)$ is known for every $p \in \{1, 2, \dots, 3P_0\}$. It is the weight of that item in class N_1 that has profit exactly p and minimum weight. The following recurrence helps compute all values $F(i,p)$.

$$F(i+1,p) = \min\{F(i,p), \min_{j \in N_{i+1}} \{w_{(i+1)j} + F(i, p - p_{(i+1)j})\}\} \text{ if } p_{(i+1)j} \le p,$$

and, $F(i+1,p) = F(i,p)$ otherwise, (i.e. if for all $j \in N_{i+1}$, $p_{(i+1)j} > p$).

And we have,

$$P^* = \max\{p | F(m,p) \le c\}.$$

2.1.1 Complexity Analysis

Note that there are m values for i and $3P_0$ ($\mathbf{O}(P^*)$) values for p. Computing $F(i,p)$ for each value of i and p takes time depending on the number or elements in N_i. Therefore the number of steps carried out is $c\sum_{i=1}^{m} n_i = n$. The time required to compute P_0 is $\mathbf{O}(n)$. Hence the time complexity of the algorithm is $\mathbf{O}(n + nP^*)$, equivalent to $\mathbf{O}(nP^*)$. Note, however, that this does not mean that the time is polynomial in n, the size of the instance, since P^* itself may not be polynomially bounded by n.

2.1.2 Constructing the Solution by Backtracking

Up to this point we have ignored the problem of constructing the solution. The solution can be easily constructed by trivially modifying the algorithm so that it supports backtracking. This can be easily done using suitable data-structures while implementing the algorithm. In this paper we shall concentrate on the FPTAS, and not on the problem of constructing the solution.

2.2 Scaling of Profits

We can make the computation more efficient by reducing the number of distinct p values for which $F(i,p)$ is computed. Let us replace each profit value p_{ij} by

$$q_{ij} = \left\lfloor \frac{p_{ij}}{K} \right\rfloor,$$

where K is a suitably chosen scale factor. We wish to make K as large as possible while making sure that the solution we obtain does not differ from the optimum by more than ϵP^*. Note that

$$Kq_{ij} \le p_{ij} < K(q_{ij}+1).$$

It follows that for any set S,

$$\sum_{j\in S} p_{ij} - K\sum_{j\in S} q_{ij} < K|S|.$$

Let S^* be an optimal solution set. Therefore K will be a valid scale factor if we ensure that

$$K|S^*| \le \epsilon P^*.$$

It is clear that $|S^*| < m$ and $P^* \ge P_0$ where P_0 is the value computed in the previous sub-section. Hence we may choose

$$K = \frac{1}{m}\epsilon P_0.$$

Note that $P^* \le 3P_0$, hence

$$\frac{P^*}{K} \le \frac{3m}{\epsilon}.$$

Thus the computation can now be carried out in $\mathbf{O}(nm/\epsilon)$ time, exclusive of the time required to compute P_0.

2.3 Further Observations

For a special class of the 0-1 MCKP, satisfying the following additional constraint

$$c \le \sum_{i=1}^{m} \min_{j\in N_i} w_{ij},$$

we can obtain P_0 satisfying $P_0 \le P^* \le 2P_0$ (note the tighter upper bound) in a straightforward manner without affecting the complexity of the overall algorithm. It can be done as follows. Select those items in each subset, $N_1,\ldots,N_m$, which have the maximum profit density (profit/weight). Let they be denoted by o_i for each subset of items, i.e $i=1,\ldots,m$. This procedure can be carried out in $\mathbf{O}(n_1)+\ldots+\mathbf{O}(n_m)$ time. i.e in $\mathbf{O}(n)$ time. We shall now find P_0 such that $P_0 \le P^* \le 2P_0$. One way to do this is to follow the following procedure: Order the items o_i, $i=1,\ldots,m$ according to decreasing profit densities. Let the profits of the sorted items be $\dot{p}_1,\dot{p}_2,\ldots,\dot{p}_m$ respectively. And let their weights be $\dot{w}_1,\dot{w}_2,\ldots,\dot{w}_m$. Find the largest integer $1 \le l \le m$ such that $W = \dot{w}_1+\cdots+\dot{w}_l \le c$. (Note that if $W = c$ then $P^* = \dot{p}_1+\cdots+\dot{p}_l$). Set $P_0 = \max\{\dot{p}_1+\cdots+\dot{p}_l, \dot{p}_{l+1}\}$. (Clearly $P_0 \le P^* \le 2P_0$ since $\dot{p}_1+\cdots+\dot{p}_l \le P^* \le \dot{p}_1+\cdots+\dot{p}_l+\dot{p}_{l+1}$ and $P^* \ge \max\{\dot{p}_1+\cdots+\dot{p}_l,\dot{p}_{l+1}\}$). However this takes $\mathbf{O}(m\log m)$ time.

But sorting is not necessary to compute P_0. This can be done in $\mathbf{O}(m)$ time by employing a median-finding algorithm as follows. We find the median of the profit/weight (profit density) ratios for the m items. (All ratios are considered to be distinct; if ties occur, the item with the smaller index is considered to have smaller ratio.) Let p_i and w_i denote the profit and weight of item m_i. Let the median ratio be r and let

$$H = \{h : p_h/w_h \geq r\}.$$

If $\sum_{h\in H} w_h > c$, find the median ratio in H until the largest set H is found such that $\sum_{h\in H} w_h \leq c$. If $\sum_{h\in H} w_h < c$, find the median ratio in the complement of H until the largest set H is found such that $\sum_{h\in H} w_h \leq c$.

$$P_0 = \max\left\{\sum_{h\in H} p_h, p_{max}\right\}.$$

Note that $\dot{p}_{l+1}$, required to compute P_0, is simply the profit of an item with the highest profit density in the complementary set of H. Thus $\dot{p}_{l+1}$ can be obtained trivially in $\mathbf{O}(m)$ time.

There are median finding routines which require only $\mathbf{O}(m)$ time [3]. This procedure requires $\mathbf{O}(\log m)$ applications of such a routine. However these are carried out over sets which contain $m, m/2, m/4, \ldots$ elements. The computation of P_0 thus requires only $\mathbf{O}(m)$ time,with the overall time complexity of calculating P_0 being $\mathbf{O}(n+m)$ or $\mathbf{O}(n)$ since $m \leq n$.

3 Summary and Analysis of the Algorithm

We now summarize the steps of the approximation algorithm for the 0-1 MCKP problem:

1. Compute P_0 as shown in section 2.
2. compute the scale factor $K = \epsilon P_0/m$.
3. For each profit value p_{ij} set $q_{ij} = \lfloor \frac{p_{ij}}{K} \rfloor$.
4. Create instance $I' = \langle a_{ij}, w_{ij}, p'_{ij}; c\rangle$. Use the Exact Pseudo-polynomial time dynamic programming algorithm to get a solution S' to I'.
5. Set $\widehat{S}$ to be the set corresponding to the profit P_0.
6. Return the more profitable of $\widehat{S}$ and S'.

Lemma 3.0.1 *Let A denote the output set of the above algorithm, and P^* the optimal profit. Then, $f_\Pi(I; A) \geq (1-\epsilon)P^*$.*

Proof: Let A^* be an optimal solution, i.e. $f_\Pi(I; A^*) = P^*$. For every item a_{ij} we have

$$p_{ij} - K \leq Kq_{ij} \leq p_{ij},$$

and so therefore, considering all items in the optimal solution, we have,

$$P^* - Kf_\Pi(I'; A^*) \leq mK.$$

Since A is optimal in I' we know that $f_\Pi(I'; A) \geq f_\Pi(I'; A^*)$ and applying the above inequality we see that

$$Kf_\Pi(I'; A) \geq Kf_\Pi(I'; A^*) \geq P^* - mK.$$

By the definition of p_{ij} we also know that $f_\Pi(I; T) \geq Kf_\Pi(I'; T)$ for any feasible solution T. We thus have

$$f_\Pi(I; A) \geq Kf_\Pi(I'; A) \geq P^* - mK = P^* - \epsilon P_0.$$

Note that the algorithm returns a solution which is guaranteed to have value at least P_0. Therefore, we have

$$f_\Pi(I; A) \geq P^* - \epsilon P_0 \geq P^* - \epsilon f_\Pi(I; A),$$

which gives us

$$f_\Pi(I; A) \geq \frac{1}{1+\epsilon} P^*.$$

Since for all $\epsilon > 0$ it follows that $1 - \epsilon \leq (1+\epsilon)^{-1}$, the algorithm is in fact better than $(1-\epsilon)$OPT.

3.1 Complexity Analysis of the Algorithm for the 0-1 MCKP

The time required to calculate P_0 is $\mathbf{O}(n)$. And, as shown above, the time complexity of the remaining part of the algorithm is $\mathbf{O}(nm/\epsilon)$. The total time complexity of the FPTAS is thus $\mathbf{O}(n + nm/\epsilon)$ which is equal to $\mathbf{O}(nm/\epsilon)$.

Theorem 3.1.1 *The algorithm is an FPTAS for the 0-1 MCKP.*

Proof: By the lemma, the solution is $(1-\epsilon)$-optimal. The running time is

$$\mathbf{O}\left(\frac{nm}{\epsilon}\right)$$

which is polynomial in n and m, the size of the instance, and $\frac{1}{\epsilon}$. The algorithm is therefore an FPTAS.

4 Conclusion

In the paper, we gave an improved FPTAS for 0-1 MCKP whose running time is $\mathbf{O}(nm/\epsilon)$. This improvement is obtained because of an improved method to calculate tight upper and lower bounds on the value of the optimal solution. We believe that our technique of replacing the sorting step and the method of d'Atri [6] by a method involving finding the solution to a continuous relaxation of the 0-1 MCKP and deriving efficient upper and lower bounds on the optimal profit value based on the solution to the continuous problem could find potential uses elsewhere.

Acknowledgements. The authors wish to thank Daniel Camps Mur, research scientist at NEC Laboratories Europe, for pointing out a missing case in the dynamic programming formulation for the 0-1 MCKP in an earlier version of this manuscript.

References

[1] Mukul Subodh Bansal, V. Ch. Venkaiah, “Improved Fully Polynomial time Approximation Scheme for the 0-1 Multiple-choice Knapsack Problem”, *Technical Report Number: IIIT-H/TR/2004/003, IIIT, Hyderabad, India,* (2004).

[2] R.E. Bellman, “Dynamic Programming”, *Princeton University Press, Princton, NJ,* (1957).

[3] M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, R.E. Tarjan, “Time Bounds for Selection”, *J. Comput. System Sci.,* (7), (1973): 448–461.

[4] S.A. Cook, “The Complexity of Theorem Proving Procedures”, *Conf. Record of Third ACM Symposium on Theory of Computing,* (1970): 151–158.

[5] M.R. Garey, D.S. Johnson, “Computers and Intractability: A Guide to the Theory of NP-Completeness”, Freeman, Newyork, (1979).

[6] G. d’Atri, “The Generalised Knapsack Problem”, *Communication at the annual meeting of CNR-GNIM, Rimini, Italy.*

[7] K. Dudzinski, S. Walukiewicz, “Exact Methods for the Knapsack Problem and its Generalizations”, *European Journal of Operational Research,* (28), (1987): 3–21.

[8] M.E. Dyer, “An O(n) Algorithm for the Multiple-choice Knapsack Linear Program”, *Mathematical Programming,* (29), (1984): 57–63.

[9] M.E. Dyer, N. Kayal, J. Walker, “A Branch and Bound Algorithm for Solving the Multiple-Choice Knapsack Problem”, *Journal of Computational and Applied Mathematics,* (11),(1984): 231–249.

[10] George Gens, Eugene Levner, “An approximate binary search algorithm for the multiple-choice knapsack problem”, *Information Processing Letters,* (67), (1998): 261–265.

[11] O.H. Ibarra, C.E. Kim, “Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems”, *Journal of the ACM,* (22), (1975): 462–468.

[12] Eugene L. Lawler, “Fast Approximation Algorithms for Knapsack Problems”, *Mathematics of Operations Research,* (Vol.4, No.4), (November 1979).

[13] S. Martello, P. Toth, “Knapsack Problems: Algorithms and Computational Implementations”, Wiley, Chichester, (1990).

[14] R.M. Nauss, “The 0-1 Knapsack Problem with Multiple Choice Constraint”, *European Journal of Operational research,* (2),(1978): 125–131.

[15] D. Pisinger “A minimal Algorithm for the multiple Choice Knapsack Problem”, *European Journal of Operational Research,* (83), (1995): 394–410.

[16] A. Sinha, A.A. Zoltners, “The Multiple-Choice Knapsack Problem”, *Operations Research,* (27), (1979): 503–515.

[17] V. V. Vazirani, “Approximation Algorithms”, *Springer-Verlag,* (2001).

[18] E. Zemel, “An O(n) Algorithm for the Liner Multiple choice Knapsack Problem and Related Problems”, *Information Processing Letters,* (18), (1984): 123–128.

[19] E. Zemel, “The Liner Multiple choice Knapsack Problem”, *Operations Research,* (28), (1980): 1412–1423.