International Journal of Mathematical Sciences and Applications
Vol. 1 No. 2 (May, 2011)

www.ijmsa.yolasite.com

A new heuristic approach for Knapsack/Covering Problem

S.Raja Balachandar [1] and K.Kannan [2]

[1]Department of Mathematics, SASTRA University, Thanjavur 613401, Tamil Nadu, INDIA.
e-mail: srbala09@gmail.com (corresponding author)

[2]Department of Mathematics, SASTRA University, Thanjavur 613401, Tamil Nadu, INDIA.
e-mail: kkannan@maths.sastra.edu

Abstract

This paper presents a heuristic approach to solve the 0/1 Multidimensional Knapsack/Covering Problem (MKCP) which is NP-hard. The intercept matrix of the constraints, used to find optimal or near optimal solution of the MMKP is proposed. This heuristic approach is tested for benchmark problems of sizes upto (500X30), taken from OR-library and the results are compared with optimum solutions. Space and Computational complexity of solving MKCP using this approach are also presented. The performance of our heuristic is compared with the best state-of-the-art heuristic algorithms with respect to quality of the solutions found and the corresponding execution time. The encouraging results especially for relatively large size test problems indicate that this heuristic approach can successfully be used for finding good solutions for highly constrained NP-hard problems.

Keywords*: Combinatorial Optimization problem; 0/1 Multidimensional Knapsack/covering problem; heuristic; computational complexity; NP-Hard problems.*

1 Introduction

0/1 multidimensional knapsack/covering problem is an extension of multidimensional knapsack problem in which there are greater-than-or-equal-to inequalities, in addition to the standard less-than-or-equal-to constraints. Moreover, the objective function coefficients are not constrained in sign. It can be formulated as

$$\text{Max } z = \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t} \quad \sum_{j=1}^{n} a_{ij} x_j \le b_i , \quad \forall i \in \{1,....,m\} \tag{1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \ge b_i \qquad \forall i \in \{m+1,....,m+q\} \tag{2}$$

$$x_j \in \{0,1\} \ \forall i \in \{1,....,n\} \tag{3}$$

where $b_j > 0 \quad \forall i \in \{1,....,m+q\}, \quad a_{ij} \ge 0 \quad \forall i \in \{1,....,m+q\}, \ \forall i \in \{1,....,n\}$

We also assume that $\sum_{j=1}^{n} a_{ij} > b_i \qquad \forall i \in \{1,....,m+q\}, \quad \max_j \{a_{ij}\} \le b_i$

$\forall i \in \{1,....,m\}$

and $\min_j \{a_{ij}\} < b_i \qquad \forall i \in \{m+1,....,m+q\}$

since any violation of these conditions would result in some x_j 's being fixed to zero or some constraints not being satisfied or being redundant. Each of the m constraints of family (2) is called a knapsack constraint, while we refer to each of the q constraints of family (3) as a cover constraint.

Many practical problems have a structure of MKCP embedded, including portfolio location selection and capital budgeting [2], obnoxious, semi-obnoxious facility and location problem [4] [23] [25]. These problems are NP-hard, and the use of heuristic search methods is highly competitive for even moderately sized instances. In this paper, we propose a heuristic algorithm based on dominance

principle of intercept matrix to solve MKCP. The Dominance principle based heuristic algorithm has been implemented successfully to solve 0-1 multi constrained knapsack problem [24].

This paper is organized as follows: A brief survey of various researchers' works pertaining to this problem is elucidated in section II. The dominant principle based Heuristic (DPH) approach for solving MKCP and its computational complexity are explained in section III. We have furnished the results obtained by DPH for all the benchmark problems in section IV. This section also includes the extensive comparative study of the results of our heuristic with known optimum or best solutions of MKCP and concluding remarks with future direction are also given in section V.

2 PREVIOUS WORK

Most of the researchers are dealt with single constraint knapsack problems (KP), which is not strongly NP-Hard and effective approximation algorithms have been developed for obtaining near-optimal solutions. A good review of the single knapsack problem and its associated exact and approximate algorithm is given by Martello and Toth [16]. Below we review the literature for the knapsack problem variants. We particularly discuss the multidimensional knapsack problem (MDKP), the multiple-choice knapsack problem (MCKP), and the MKCP.

The MDKP is a generalization of the KP for multiple resource constraints. The detailed literature is available in [6][8][23][26]. Another variant of the knapsack problem is the MCKP, where the picking criterion for item is more restricted. For the later variant of the knapsack problem there are one or more disjoint classes of items, for more details, one can refer to Nauss [20]. Finally, the MKCP can be considered as a more generalization of the MDKP and MCKP variants of the 0-1 KP. Most algorithms for optimal solutions of knapsack problem variants are also based upon branch-and-bound procedures [13][20] [22].

A greedy algorithm has been proposed for approximately solving the knapsack problems by Toth [16] [17]). For the classical binary knapsack problem, the approach is composed of two stages (i)sort the items in decreasing order of value - weight ratio and (ii)pick as many items as possible from the left of the ordered list until the resource constraint is violated. By using the same principle for the MDKP, Toyoda [30] used the aggregate resources consumption. The solution of the MDKP needs iterative picking of items until the resource constraint is violated. Shih [26] presented a branch-and-bound algorithm for MDKP. In this method, an upper bound was obtained by computing the objective function value associated with the optimal fractional solution algorithm [7] for the m single constraint knapsack problems and selecting the minimum objective function value among those as the upper bound. In the recent past, great success has been

achieved via the application of local search techniques and metaheuristics to MDKP[6], namely Tabu search, Genetic algorithms, Simulated annealing and hybrid algorithms.

In the literature, very few papers directly dealing with MKCP are available. Cappanera [4] designed a heuristic approach in 2005. They employed a sophisticated two-phase search, with additional use of domination criteria and variable ordering. Their search first attempted to find feasible solutions, and thereafter a local search phase was initiated from each of the feasible solutions found (upto a maximum number), using both single-flip, swap and double-swap neighborhoods, but remaining feasible all the time. Their search was based on strategic oscillation ideas of Glover and Kochenberger[9]. They also showed that the feasible regions may be disconnected for the neighborhoods they use. Finding feasible solutions is a daunting task for some of test problem classes (those with the most demand constraints), with exact solver CPLEX having the most trouble. Arntzen et.al [1] implemented a simple tabu search-based metaheuristic method for MKCP, augmented by adaptive memory and learning principles derived from tabu search ideas. Hvattum et.al [15] have proposed an alternating control tree search (ACT) method for this problem.

In this paper, we propose a heuristic algorithm based on dominance principle of intercept matrix to solve MKCP. The Dominance principle based heuristic algorithm has been implemented successfully to solve 0-1 multi constrained knapsack problem [24]. The main principle of the algorithm is twofold: (i) to generate an initial feasible solution for demand constraints (3) (ii) to improve this feasible solution to satisfy the knapsack constraints (2) and find the optimal or near optimal of (1) by using intercept matrix of (2).

3. DOMINANCE PRINCIPLE BASED HEURISTIC

(DPH)

Linear programming (LP) is one of the most important techniques used in modeling and solving practical optimization problems that arise in industry, commerce and management. Linear programming problems are mathematical models used to represent real life situations in the form of linear objective function and constraints various methods are available to solve linear programming problems. When formulating an LP model, systems analysist and researchers often include all possible constraints and variables although some of them may not be binding at the optimal solution. The presence of redundant constraints and variables does not alter the optimum solution(s), but may consume extra computational effort.

Many researchers have proposed algorithms for identifying the redundant constraints and variables in LP models [3] [10] [11] [12] [14] [18] [19] [27] [28]

[29] [30]. Paulraj [21] used the intercept matrix of the constraints to identify redundant constraints prior to the start of the solution process in his Heuristic approach.

MKCP is a well known 0-1 integer programming problem and many variables have zero values called redundant variables. We use intercept matrix of the constraints (2) to identify the variables of value 1 and 0. The DPH algorithm consists of two phases. The first phase is used to assign the values of the variables from zero to one as long as feasibility is not violated for $\geq$ constraints (3). The second phase is used to assign the values of the variables from zero to one (or) one to zero as long as feasibility is not violated for $\leq$ constraints(2). Both phases are designed by the dominant principle of constraints (2) and (3). The solution obtained from phase one is known as initial feasible solution for $\geq$ constraints. We update the initial feasible solution in phase two by using dominant principle of intercept matrix of $\leq$ constraints (2). Construct the intercept matrix by dividing b_k values by coefficients of (2). The elements of intercept matrix are arranged in decreasing order; the leading element is the dominant variable.

This process of identifying the leading element from intercept matrix is known as dominant principle. We use this dominant variable to improve the current feasible solution and this procedure provides optimum or near optimum solution of MKCP. The dominant principle focuses at the resource matrix with lower requirement come forward to maximize the profit. The intercept matrix of the constraints (2) plays a vital role in achieving the goal, in a heuristic manner.

The various stages of DPH are shown below:-

Step:1 z =0 and x_j = 0 $\forall\ j$

Step:2 While $(b_i \neq 0)$ $(i = m+1,...,m+q)$

$$k = \arg Max\left(\sum_{i=m+1}^{m+q} a_{ij} \right)$$

$$x_k = x_k + 1$$

$$b_i = b_i - a_{ik} \ (i = m+1,...,m+q)$$

end while

While $(a_{ij} \neq 0)$ $(\forall\ j, \quad i = 1\ to\ m)$

Step 3: Intercept matrix $d_{ij} = \begin{cases} b_i / a_{ij}, & a_{ij} > 0 \\ M\,(a\ \ l\arg e\ value) & otherwise \end{cases}$, $\forall i\ \ and\ \forall j$

Step:4 redundant variables rv

$$rv = \arg_{1 \le j \le n} \{\, d_{ij} < 1 \,) \qquad i = 1,\ldots,m$$

$$a_{i\,rv} = 0 \quad \forall i$$

$$x_{rv} = 0$$

Step:5 Dominant variable dv_j

$$dv_j = \min \{d_{ij}\} \qquad \begin{matrix} i = 1,\ldots,m \\ j = 1,\ldots,n \end{matrix}$$

Step 6: $l = \arg\ \max \{dv_j * c_j\}\ \ \forall j$

Step 7: if $x_l = 0$ then check the feasibility; and

$$x_l = 1;$$

$$b_i = b_i - a_{il}$$

$$z = z + c_l$$

$$a_{il} = 0\ \ \forall i$$

end if

end while

3.1. Example

Maximize: $6x_1+14x_2+11x_3$

Subject to

$13x_1+14x_2+18x_3 \leq 34$, $22x1+21x_2+28x_3 \leq 53$,

$33x_1+39x_2+34x_3 \leq 80$, $17x1+11x_2+16x_3 \geq 22$

The algorithm begins with initial solution (0,0,0). We convert this infeasible solution into feasible solution for ≥ constraint by assigning 1 (step - 2) to the larger coefficients as long as feasibility is not violated. At the end of this stage the solution is (1,0,1). We update this solution by using the dominant principle of constraints (2) iteratively.

Iteration 1: The variable x_3 dominates the other two variables (step-3 to step - 7) and this variable is already in the current solution. The objective function value is 11 and solution is (1, 0, 1).

Iteration 2: The variable x_2 dominates the variable x_1 (step- 3 to step - 7) and this variable is not available in the current solution. We modify the current solution by assigning 1 to x_2 and 0 to x_1.

Iteration 3: There is no updation in the solution (step- 3 to step - 7). Thus the new solution is $x_1=0$; $x_2=1$; $x_3=1$; objective function value= 25 which is the optimum value.

Theorem 1: DPH can be solved in $O(mn^2)$ time, polynomial in the number of variables and constraints.

Proof: The Worst-case complexity of finding the solutions of an MKCP using DPH can be obtained as follows. Assume that there are n variables and m constraints. The procedure initialization (Step-1) and (Step-2) require O(n) and O(qn) running time respectively. The formation of D matrix iterates n times, identification of less than one entry in each column, finding smallest intercept in each column, identification rows which consists of more than one smallest intercept and updating of constraint matrix A. Since there are m constraints, Step-3, Step-4, Step-5, Step-7 respectively, O(mn). Step-6 and requires O(n) operations to multiply cost with corresponding smallest intercept. The maximum number of iterations required for DPH is n. So the overall running time of the procedure DPH can be deduced as $O(mn^2)$. Since $m \geq q$.

4. EXPERIMENTAL DESIGN

The heuristics, which were tested on 810 instances, can be found in [4]. These test instances are divided into 2 sets. The first set consists of positive coefficients in the objective function and second consists of both positive and negative coefficients. Each set consists of nine classes; each class contains 45 test instances. The MKCP test instances have the following characteristic values:-

n: the number of variables: 100, 250 or 500.

m: the number of knapsack constraints: 5, 10 or 30.

q: the demand constraints: 1, m/2 or m.

r: 1, if the objective function has negative coefficients; otherwise, 0.

s = 0,1,...,14 is the instance number within the class. we use n-m-q-r-s for each instances and n-m-r for each class.

The computational results are given in TABLE I. The first column in TABLE I indicate the name of the problem set. The next two columns report for the CPLEX, the number of optimum solutions and number of infeasible solutions out of 45 problems in each problem set. The last two columns report for DPH that the number of proven optimal solutions found and the number of failures per class. It is clear that from TABLE I that our DPH finds the optimal in 344 test problems but CPLEX fails to catch the feasible solution in 72 test problems. The application of DPH algorithm for 100-5-1-0-0 is presented in Fig.1. The maximum number of iterations is 100 fixed. DPH reaches the solution 30909 at 32*nd* iterations which is optimum.

Table 1: Computational results of DPH

Class	CPLEX		DPH	
	Opt	Fail	Opt	Fail
100-5-0	45	0	45	0
100-5-1	45	0	45	0
100-10-0	35	0	33	0
100-10-1	39	0	38	0
100-30-0	0	18	12	0
100-30-1	13	22	18	0
250-5-0	17	0	19	0
250-5-1	13	0	15	0
250-10-0	0	0	10	0
250-10-1	0	0	9	0
250-30-0	0	10	12	0
250-30-1	0	11	8	0
500-5-0	0	0	14	0
500-5-1	0	0	16	0
500-10-0	0	0	18	0
500-10-1	0	0	12	0
500-30-0	0	5	9	0
500-30-1	0	6	11	0
Total	207	72	344	0

(Opt = Optimal solution
Fail = Infeasible solution)

The comparative study of DPH with other existing heuristic algorithms has been furnished in TABLE II(N.O.O = number of optimum solution CPLEX: Problem solver; Almha: Adaptive memory search[1]; NT: nested tabu search[4]) in terms of the average deviation for problems and the number of optimal solutions obtained for the test instances. It can be seen from TABLE II that DPH is found to be better than the other heuristics. As both table and figure clearly demonstrate that the DPH is able to catch the optimum or best solution for all the test problems in quick time. The heuristic is used to reduce the search space to find the near optimal solutions of the MKCP.

The computational complexity is $O(mn^2)$ and the space complexity is $O(mn+qn)$. It reaches the optimum or near optimum point in n iterations where n is the

number of groups. Due to dominance principles, this heuristic identifies the zero value variables instantaneously. The maximum CPU Time required for large-size problem are 386 seconds. From the comparison table we observe that our algorithm is the effective one.

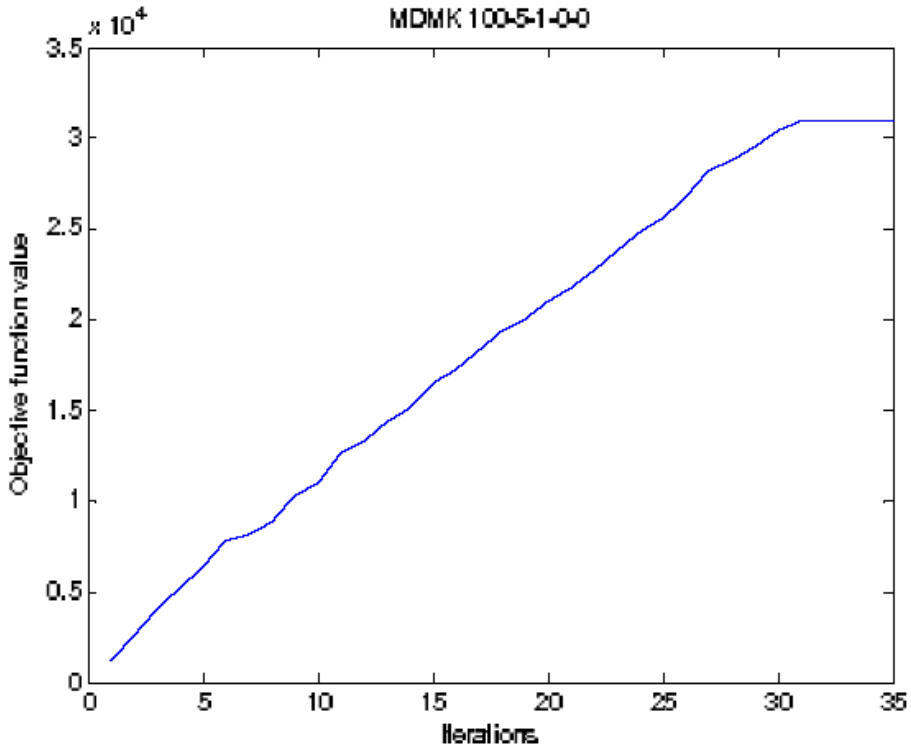

Fig I: Iteration wise Objective function value for 100-5-1-0-0

Table 2: Summarized results for the solution quality of MKCP

Class	CPLEX	Almha	NT	DPH
100-5-0	0.00	0.04	0.07	0.00
100-5-1	0.00	0.12	0.25	0.00
100-10- 0	0.13	0.22	0.47	0.62
100-10-1	0.16	0.30	0.64	1.49
100-30-0	4.44	4.83	4.37	5.99
100-30-1	8.92	12.63	8.23	10.91
250-5-0	0.04	0.09	0.22	0.11
250-5-1	0.20	0.43	1.04	0.44
250-10-0	0.41	0.58	0.92	0.47
250-10-1	0.79	1.10	1.68	0.24
250-30-0	4.38	2.18	2.72	4.83
250-30-1	13.15	4.56	5.55	14.58
500-5-0	0.05	0.11	0.24	0.06
500-5-1	0.17	0.44	0.97	0.06
500-10-0	0.21	0.39	0.64	7.16
500-10-1	0.54	0.93	1.48	0.70
500-30-0	3.56	1.40	1.71	4.46
500-30-1	7.28	2.35	2.91	8.72
N.O.O	207	93	61	344

5. CONCLUSION

In this paper, the dominant principle based approach for tackling the NP-Hard 0/1 Multidimensional knapsack/covering problem (MKCP). This approach has been tested on 810 state-of-art benchmark instances and has led to given optimum solutions for 344 instances given in the literature, whereas CPLEX has failed to catch the feasible solutions for 74 instances. For rest of the instances the average percentage of deviation of DPH solution from the optimum solution is observed to be 3.38. This heuristic is with $O(mn^2)$ complexity and it requires n iterations to solve the MKCP. The basic idea behind the proposed scheme may be explored to tackle other NP-Hard Problems.

References:

[1] Arntzen.H, Hvattum.L.M, Lokketangen A., "Adaptive memory search for multidemand multidimensional knapsack problem", *Computers and Operations Research*, (2006), 2508-2525.

[2] Beaujon GJ, Marin SP, McDonald GC., "Balancing and optimizing a portfolio of R and D projects. Naval Research Logistics 2001;48:(2001), 18- 40.

[3] Brearley,A.L, G.Mitra, Williams H.P., "Analysis of mathematical programming problem prior to applying the simplex algorithm", *Math. Prog.,* 8: (1975), 54-83.

[4] Cappanera P, Trubian M., "A local search based heuristic for the demand constrained multidimensional knapsack problem", *Informs Journal on Computing*, 17(1): (2005), 82-98.

[5] Cappanera P., "Discrete facility location and routing of obnoxious facilities", PhD thesis, University of Milano, Available at http://www.di.unipi.it/ cappaner, (1999).

[6] Chu P, Beasley J.E., "A genetic algorithm for the multi-dimensional knapsack problem", *J. Heuristics*, 4(1998), 63-86.

[7] Dantzig G.B., "Discrete variable extremum problems", *Opns Res,* 5: (1957), 266- 277.

[8] Freville A, Plateau G., "An efficient pre processing procedure for the multidimensional 0-1 knapsac problem", *Discrete Appl Math*, 49: (1994), 189-212.

[9] Glover F, Kochenberger G., "Critical event tabu search for multidimensional knapsack problems". In: Osman IH,Kelly JP, editors. Meta heuristics: theory and applications. Dordrecht: Kluwer Academic Publishers;(1996), 407-427.

[10] Gondzio J., "Presolve analysis of linear programs prior to applying an interior point method", *Inform. J.Comput*, 9: (1997), 73-91.

[11] Ioslovich, I., "Robust reduction of a class of large scale linear program, *Siam J. Optimization*", 12: (2002), 262-282.

[12] Karwan,M.H, Loffi V, Telgan J, Zionts S., "*Redundancy in mathematical Programming"*: A State of the Art Servey (Berlin: Springer-Verlag),(1983).

[13] Khan S., "Quality adaptation in a multi-session adaptive multimedia system: model, algorithms and architecture". PhD Thesis, Department of Electronical

and computer engineering, University of Victoria, Canada,(1998).

[14] Kuhn, H.W, R.E .Quant., "An Experimental Study of the Simplex Method. In: Metropolis", N. et al.(Eds.). *Preceedings of Symposia in Applied Mathematics.* Providence, RI: Am. Math. Soc., 15:(1962), 107-124.

[15] Lars Magnus Hvattum, Halvard Arntzen, Arne Lkketangen, Fred Glover., "Alternating control tree search for knapsack/covering problems", *Journal of heuristics*, published online, November (2008).

[16] Martello S, Toth P., Algorithms for knapsack problems, *Ann Discrete math*, 31(1987), 70-79.

[17] Martello S, Toth P., Knapsac problems : "*Algorithms and computer implementations"*, Wiley,Chichester, England,(1990).

[18] Matthesiss T.H., "An Algorithm for determining irrelevant constraints and all vertices in systems of linear inequalities." *Operat. Res*., 21: (1973), 247- 260.

[19] Meszaros C., Suhl U.H., "Advanced preprocessing techniques for linear and quadratic programming". *Spectrum*, 25:(2003), 575-595.

[20] Nauss MR., "The 0-1 knapsac problem with multiple choice constraints", *Eur J Opl Res*, 2:(1978), 125-131.

[21] Paulraj, S., Chellappan C T.R. Natesan T.R., "A heuristic approach for identification of redundant constraints in linear programming models", *Int. J. Com. Math.,* 83(8):(2006), 675-683.

[22] Pisinger D., "An exact algorithm for large multiple knapsack problems", *Eur J Opl Res*, 114: (1999), 528-541.

[23] Plastria F., "Static competitive facility location: an overview of optimization approaches", *European Journal of Operational Research* 129:(2001), 461-470.

[24] Raja Balachandar .S, Kannan.K, "A new polynomial time algorithm for 0-1 multiple knapsack problems based on dominant principles", *Applied Mathematics and Computation,* 202, (2008), 71-77.

[25] Romero-Morales D, Carrizosa E, Conde E., « Semi-obnoxious location models: a global optimization approach", *European Journal of Operational*

Research 1997;102:(1997), 295-301.

[26] Shih W., A branch and bound method for the multi constraint 0-1 knapsac problem, *J. Opl. Res. Soc*, 30: (1979), 369-378.

[27] Srojkovic, N.V, P.S. Stanimirovic., "Two direct methods in linear programming", *European J. Oper. Res.,* 131: (2001), 417-439.

[28] Telgan, J., "Identifying redundant constraints and implicit equalities in system of linear constraints", *Manage. Sci.,* 29: (1983), 1209-1222.

[29] Tomlin, J.A, J.S Wetch., "Finding duplicate rows in a linear programming model". *Oper. Res. Let*., 5: (1986), 7-11.

[30] Toyoda Y., "A simplified algorithm for obtaining approximate solution to 0-1 programming problems", *Mngt Sci* 21:(1975), 1417-1427.