Hybrid Heuristic Algorithm for the Multidimensional Knapsack Problem

Can Atılgan[1], Urfat Nuriyev[2]
[1,2]Ege University, Izmir, Turkey
[1]*can.atlgan@gmail.com*, [2]*urfat.nuriyev@ege.edu.tr*

Abstract— **In this work, a new hybrid heuristic algorithm for the 0/1 multidimensional knapsack problem is proposed. In the algorithm, Lagrange multipliers for every constraint are determined to reduce the problem to single dimension and some initial solutions are obtained with greedy algorithms. Then, these solutions are improved with iterative procedures. In order to test efficiency of the algorithm, computational experiments were done on some library problems in literature. It was observed that the algorithm has high efficiency in terms of solutions and time.**

Keywords— Knapsack problem; Boolean variables; hybrid heuristic; greedy algorithm; core approach

I. Introduction

The 0/1 Multidimensional Knapsack Problem can be formally expressed as follows [1]:

$$\sum_{j=1}^{n} c_j x_j \to \max , \qquad (1)$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = \overline{1,m} , \qquad (2)$$

$$x_j = 0 \vee 1 \quad j = \overline{1,n} . \qquad (3)$$

Here, b_i, c_j are positive integers and a_{ij} is a nonnegative integer.

This problem has been analyzed and generalized from various angles. New applications and meaningful interpretations of the problem have been continuously emerging [2].

II. Heuristic Algorithms

It is known that the multdimensional knapsack problem is NP-Complete, hence it can not be solved in polynomial time with exact algorithms [3]. For solving these kind of problems, approximation algorithms which can generate near optimal solutions are used. Quality of an approximation algorithm is closely related to how much the characteristics of the problem is considered [4].

Heuristic methods are easy to apply and they generate suboptimal solutions in short time. However, it must be taken into consideration that these suboptimal solutions might not be satisfactory in some instances.

Heuristic algorithms mostly use the following strategy: A preference number is determined to find a subset of variables that might also be in the optimal solution vector. Here, a variable is picked over another if and only if its preference number is greater. The picking process is performed on the set of measurable factors. At every step of this process, a variable is picked in proportion to the preference number which was assigned to it. In other words, a score for every single element is calculated and the elements with the highest scores are picked. The algorithms using this strategy are known as greedy algorithms in literature. It was also showed that greedy algorithms can find good solutions, even in large problems [5].

The multidimensional knapsack problem differs from the classical knapsack problem in number of constraints in the problem. It is obvious that in the multidimensional knapsack problem, there are more than one constraints to be satisfied, thus it is simply harder than the classical problem. Let *m* be the number of the constraints. Some heuristic methods use aggregation of *m* constraints so that the system of constraints is reduced to single dimension [6, 7]. In linear case, coefficients of the aggregated constraint increase exponentially, regardless of the aggregation method. Surrogate constraints are used to deal with this inconvenience. One of the efficient algorithms which use surrogate constraints and aggregation, called *AS*, was proposed in [8].

III. Core Approach

A great variety of computational experiments showed that the final states of the optimal solution vectors are actually determined by selection of a small subset of variables, especially in large problems. In a knapsack problem, variable a_i, that represents an item in the system of constraints, is thought to be advantageous, if c_i / a_i value of the variable is great. The most advantageous variables occur in optimal solution as well as most of good suboptimal solutions. On the other hand, the least advantageous variables occur almost never in a good suboptimal solution, thus they can be neglected at the beginning of the selection process. Taking these conditions into consideration, the focus of core approach is a small subset which consists of variables with mid-level advantage.

In the linear relaxation of the knapsack problem, it is natural to choose a fraction of item a_i among the variables with mid-level advantage to obtain a good solution. However, deciding which items to pick among the variables with close

c_i/a_i values, is a hard objective. A set of these items is called a core.

In previous works, it was observed that number of elements in a core of a problem is quite smaller than the number of variables [9]. Hence, if items that must occur and items that must not occur in the optimal solution can be determined, then the problem may be solved easily by applying branch and bound method or dynamic programming on merely the extracted core vector.

IV. HYBRID HEURISTIC ALGORITHM (HHA)

In this heuristic algorithm we propose, Lagrange multipliers for every constraint is determined first. Then, these multipliers are used to obtain a single reduced constraint. Initinal solutions are produced by a greedy algorithm using the reduced constraints and c_i values. According to their results, initial solutions and c_i/a_i vector are combined into a selection vector. The result is improved with some iterative changes on the selection vector, based on the core approach. Pseudo-algorithm of HHA is given in the following:

<u>Step 1.</u> Calculate λ Lagrange multipliers with the following formula: $\lambda_i = \dfrac{\sum_{j=1}^{n} a_{ij} - b_i}{b_i} \quad i = \overline{1,m}$.

Determine the reduced contraints:

$$a_j = \sum_{i=1}^{m} a_{ij}\lambda_i \ , \ b = \sum_{i=1}^{m} b_i\lambda_i \quad j = \overline{1,n} \ .$$

Generate the selection vector $V_j = \dfrac{c_j}{a_j} \quad j = \overline{1,n}$ and sort in decreasing order ($V_{j_1} \geq V_{j_2} \geq ... \geq V_{j_n}$).

Find $X = (x_1, x_2, ..., x_n)$ solution with greedy selection algorithm.

Determine the p number that satisfies the following conditions:

$$\forall i, \sum_{k=1}^{p} a_{ij_k} \leq b_i \text{ and } \exists i^*, \sum_{k=1}^{p+1} a_{i^* j_k} > b_{i^*} .$$

pn = n – p; R = Result of X solution.

$$V_j = \begin{cases} V_j + 1, & if\ x_j = 1 \\ V_j, & Otherwise \end{cases}$$

<u>Step 2.</u> Sort $c_j \quad j = \overline{1,n}$ values in decreasing order ($c_{j_1} \geq c_{j_2} \geq ... \geq c_{j_n}$) and find X_C solution with greedy selection algorithm.

R_C = Result of X_C solution.

If $R_C > R \Rightarrow R = R_C, X = X_C$.

$t = 0, \ l = 1$.

$$V_j^{(t)} = \begin{cases} V_j + \dfrac{R_C}{R}, & if\ x_j = 1 \\ V_j, & Otherwise \end{cases}$$

<u>Step 3.</u> Sort $V_j^{(t)} \quad j = \overline{1,n}$ vector in decreasing order ($V_{j_1}^{(t)} \geq V_{j_2}^{(t)} \geq ... \geq V_{j_n}^{(t)}$) and find $X^{(t)}$ solution with greedy selection algorithm.

$R^{(t)}$ = Result of $X^{(t)}$ solution.

If $R^{(t)} > R \Rightarrow R = R^{(t)}, X = X^{(t)}$.

Determine the $p^{(t)}$ number that satisfies the following conditions:

$$\forall i, \sum_{k=1}^{p^{(t)}} a_{ij_k} \leq b_i \text{ and } \exists i^{(t)}, \sum_{k=1}^{p^{(t)}+1} a_{i^{(t)} j_k} > b_{i^{(t)}}$$

$$V_j^{(t)} = \begin{cases} V_j^{(t)} + \dfrac{R}{R^{(t)}}, & if\ x_j^{(t)} = 0, \ \ j = p^{(t)} + s, \ \ s = \overline{1,l} \\ V_j^{(t)}, & Otherwise \end{cases}$$

<u>Step 4.</u> $t = t + 1; \ l = \begin{cases} l + 1, & if\ \ l < pn \\ 1, & Otherwise \end{cases}$

<u>Step 5.</u> If $t < n \Rightarrow$ go to Step 3.
Else Print R and X.

When analyzing Hybrid Heuristic Algorithm, it is seen that the selection vector is updated in order to improve the solution, between Step 3 – Step 5. These updates raise the chance of the unselected items with highest selection values to be picked in the next iteration. Core approach is used at this point.

V. Computational Experiments

In order to test efficiency of the proposed algorithm (HHA), computational experiments were done on some library problems. The mentioned problems can be accesed on OR-Library website [10]. Even though optimal solutions are provided for some of the problems in the library, they are unavailable especially for the large problems. For finding these optimal solution values, Windows Quantitative System for Business (WinQSB) package was used.

Comparison of HHA with another heuristic algorithm was thought to be useful for a better observation and examination. The compared algorithm is AS algorithm which uses the surrogate constraints method [8].

Notations in the experiment tables are explained below.

m : Number of constraints,

n : Number of variables,

p : Index of the library problem,

δ : Functional error,

Functional error was computed with the following formula:

$$\delta = ((F_{OPT} - F(\alpha))/F_{OPT})*100 = (1 - F(\alpha)/F_{OPT})*100.$$

TABLE I. Computational Experiments on Small Sized OR-Library Problems

m	n	p	*Optimal Solution Value*	*AS*	*CPU Time in ms.*	*HHA*	*CPU Time in ms.*	δ
10	6	0	3800	3800	0	3800	0	0.0
10	15	2	4015	4015	2	015	3	0.0
10	20	3	6120	6120	6	6120	5	0.0
10	28	4	12400	12400	11	12400	8	0.0
5	29	5	10618	10584	15	10423	12	1.83
5	50	6	16537	16408	20	15897	18	3.87

TABLE II. Computational Experiments on Large Sized OR-Library Problems

m	n	p	*Optimal Solution Value*	*AS*	*CPU Time in ms.*	*HHA*	*CPU Time in ms.*	δ
5	100	0	24381	24003	31	23880	31	2.05
5	100	9	24411	23929	32	24147	31	1.08
5	100	15	42927	42520	32	42541	31	0.89
5	100	29	59965	59650	31	59860	31	0.17
5	250	8	61885	61278	438	61361	116	0.84
5	250	18	109957	109490	437	109059	120	0.81
5	500	22	299796	299370	3344	299168	515	0.20
10	100	9	22702	22149	63	22114	37	2.59
10	250	23	151275	150822	906	150345	117	0.61
10	500	26	304949	304480	7047	304047	601	0.29
30	100	8	22493	21796	234	21912	37	2.58
30	250	11	108338	107622	2953	106455	175	1.73
30	500	15	215762	214601	22265	211984	708	1.75

In Table 1, the problem with index 1 was not used due to noninteger variables it contains.

In Table 2, random integer numbers between 0 – 29 were generated to choose the problems to be demonstrated. For each of the 9 groups with different *m* and *n* values in the library, there are 30 instances. Indexes we assigned to these problems are between 0 – 29.

VI. Conclusion and Future Work

Computational experiments show that HHA produces near optimal solutions. Moreover, the computer program we developed for HHA produced outputs in very short time. Therefore, HHA can be a proper method to use in applications of the 0/1 multidimensional knapsack problem especially where time is significant. On the other hand, it was observed that for particular large problems, the results HHA produces are not as close to optimal results as desired.

In future work, the main goal would be improving the results with a better integration of core approach into the algorithm. Another possibility would be modifying the selection process with genetic like methods.

References

[1] S.Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, New York, J. Wiley, 1990.

[2] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Berlin, Springer, 2004.

[3] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, San Fransisco, W. H. Freeman and Company, 1979.

[4] Dj. A. Babayev, K. Sh. Mamedov, M. G. Mehtiev, "Methods for the Suboptimal Solution of the Multidimensional Knapsack Probem", *Computational Mathematics and Mathematical Physics*, Vol. 18, No: 6, 1978, pp. 1978. 1443-1453, (Russian).

[5] U. G. Nuriyev, Hybrid Method for Solving the Multidimensional Knapsack Problem, Kiev, IK AN USSR, Preprint No. 83-43, 1983, (Russian)

[6] F.Glover, "Surrogate Constraints", *Operations Research*, Vol. 16, No.4, 1968, pp. 741-749.

[7] H. J. Greenberg, W. P. Pierskalla, "Surrogate Mathematical Programming", *Operations Research*, Vol. 18, No. 5, 1970, pp. 924-939.

[8] U. G. Nuriyev, P. Dündar, "A Self-Organizing Algorithm for Solving the Multidimensional Knapsack Problem", Transactions on Operational Research, Vol. 13-14, No: 1-2, 2002, pp. 13-28, (Turkish).

[9] A. I. Nikitin, U. G. Nuriyev, "On the Method for Solving the Knapsack Problem", Kibernetica, No: 2, 1983, pp. 108-109, (Russian).

[10] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html