International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:3, No:11, 2009

A new heuristic approach for the large-scale Generalized assignment problem

S. Raja Balachandar and K.Kannan

Abstract—This paper presents a heuristic approach to solve the Generalized Assignment Problem (GAP) which is NP-hard. It is worth mentioning that many researches used to develop algorithms for identifying the redundant constraints and variables in linear programming model. Some of the algorithms are presented using intercept matrix of the constraints to identify redundant constraints and variables prior to the start of the solution process. Here a new heuristic approach based on the dominance property of the intercept matrix to find optimal or near optimal solution of the GAP is proposed. In this heuristic, redundant variables of the GAP are identified by applying the dominance property of the intercept matrix repeatedly. This heuristic approach is tested for 90 benchmark problems of sizes upto 4000, taken from OR-library and the results are compared with optimum solutions. Computational complexity is proved to be O(mn^2) of solving GAP using this approach. The performance of our heuristic is compared with the best state-of-the-art heuristic algorithms with respect to both the quality of the solutions. 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, Generalized Assignment Problem, Intercept Matrix, Heuristic, Computational Complexity, NP-Hard Problems.

I. Introduction

The generalized assignment problem (GAP) is a well-known NP-Hard [28] combinatorial optimization problem. It finds the maximum profit or minimum cost assignment of n jobs to m agents such that each job is assigned to exactly one agent and the capacity of each agent without exceeding. Many real life applications can be modeled as a GAP , e.g., the resource scheduling, allocation of memory spaces, design of communication network with capacity constraints for each network node, assigning software development tasks to programmers, assigning jobs to computers in a network, vehicle routing problems, and others. Several algorithms (exact and heuristic) that can effectively solve the GAP have been cited and compared as benchmarks many times in the literature.
This paper, we propose a heuristic algorithm based on dominance principle to solve GAP. The Dominance principle based heuristic algorithm has been implemented successfully to solve 0-1 multi constrained knapsack problem [37]. This heuristic is used here in the first stage, to find optimal or near optimal solution to GAP and second stage is to improve the near optimal solution by using another heuristic called column dominant principle and row dominant principle.

This paper is organized as follows: Section II explains the definition of GAP. A brief survey of various researchers work pertaining to this problem is elucidated in section III. The dominant principle based heuristic and its computational complexity are presented in section IV. The algorithm's utility is illustrated with help of benchmark problems in section V and we have furnished the results obtained for all the benchmark problems in section V. The extensive comparative study of our heuristic with other heuristic approches and salient features of this algorithm are enumerated in section VI, finally the concluding remarks are given in section VII.

S.Raja Balachandar is with the Department of Mathematics, SASTRA University,Thanjavur,INDIA, e-mail: srbala@maths.sastra.edu
K.Kannan is with the Department of Mathematics, SASTRA University,Thanjavur,INDIA, e-mail: kkannan@maths.sastra.edu

II. Generalized assignment problem (GAP)

Let I ={$1, 2, ..., m$} be a set of agents, and let J ={$1, 2, ..., n$} be a set of jobs. For $i \in I$, $j \in J$ define c_{ij} as the cost (profit) of assigning job j to agent i (or assigning agent i to job j), a_{ij} as the resource required by agent to perform job j(profit, if the job j is performed by agent i), and b_i as the resource availability (capacity) of agent i. Also,x_{ij} is a 0-1 variable that 1 if agent i performs job j and 0 otherwise. The mathematical formulation of the GAP is:

Maximize

$$\sum_{i\in 1}\sum_{j\in 1} c_{ij}x_{ij} \tag{1}$$

subject to the constraints

$$\sum_{j\in J} a_{ij}x_{ij} \leq b_i, \forall i \in I \tag{2}$$

$$\sum_{i\in I} x_{ij} = 1, \forall j \in J \tag{3}$$

$$x_{ij} \in \{0,1\}, \forall i \in I, \forall j \in J \tag{4}$$

(3) ensures that each job is assigned to exactly one agent and (2) ensures that the total resource requirement of the jobs assigned to an agent does not exceed the capacity of the agent.

III. previous work

There are many exact algorithms and heuristics developed to solve the GAP. Existing algorithms include branch and bound, branch-and-cut and branch-and-price algorithms [38,33,39]. Ross and Soland [38] proposed a new branch and bound algorithm in 1975. Savelsbergh [39] introduced branch and price approach in 1997. To improve Lagrangian lower bound in his

algorithm, Nauss [33] combined several ideas with cuts suggested by Gottlieb and Rao [11] and proved the performance of this algorithm for GAP instances upto 3000 binary variables. In 2006, he discussed the latest integer programming based algorithms for the GAP[34]. However exact algorithm requires more computation time for finding the optimal solutions of large size GAP. To circumvent this computational difficulty, several researchers started designing heuristic algorithms yet computational attractive algorithms to find optimal or near optimal solutions. Heuristic algorithms are designed to produce near optimal solutions for larger problem instances. Some heuristics use the linear programming relaxation [44]. In the last decade, several algorithms including Lagrangian relaxation (LR) method[9] have been developed. Narciso and Lorena [32] have proposed combining LR with surrogate relaxation, 1999. Haddadi[14] has been applied the Lagrangian decomposition method to solve GAP, 1999. Haddadi and Ouzia [15,16] have been integrated LR and subgradient methods in branch and bound schemes, 2001 and 2004. M. A. S. Monfared [31] has established that the augmented Lagrangian method (neural based combinatorial optimization problems) can produce superior results with respect to feasibility and integrality, 2006. V.Jeet and E.Kutanoglu[20] have combined LR, subgradient optimization, and problem space search techniques to solve GAP, 2007. Others use search techniques such as genetic algorithms, tabu search algorithms and simulated annealing in their meta heuristic approach to solve large size benchmark GAP instances[3] available in literature. Osman [35] has introduced simulated handling methods to solve GAP. Tabu search based heuristic algorithm used by various researchers to solve GAP[18,8,41,25]. Chu and Beasley [7] presented genetic algorithm (GA)- based heuristic for solving the GAP and have shown that the performance of genetic algorithm heuristic holds good, 1996. Harald Feltl [17] introduced a hybrid genetic algorithm which is the improved version of Chu and Beasley algorithm, 2004. Yagiura et al [45,46] has designed an algorithm based on path relinking combined with ejection chain neighbourhood approach and solved a class of GAP. Cattrysse and Wassenhove [5] present an extensive survey of algorithms for the GAP published until 1992. Amini and Racer [1] present a computational comparison of alternative solution methods. More examples of heuristic algorithms for the GAP can be found in [7,5,1,13,21,23,26,27,28]. A comprehensive review on exact and heuristic algorithms is given in [20].

IV. Dominance Principle (DP)

Linear programming (LP) is one of the most important techniques used in modeling and solving practical optimization problems that arise in Industries, 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 analysis and researchers often include all possible constraints and variables although some of them may not be binding with 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 [2,4,12,19,22,24,29,30,40,42,43]. Paulraj.et.al[36] illustrated the intercept matrix of the constraints to identify redundant constraints prior to the start of the solution process in their heuristic approach to solve a LP model.

GAP is a well known 0-1 integer programming problem. Since it is possible to use dominance principle in integer programming problem also, we use the intercept matrix of the constraints (2) to identify the variables of value 1 and 0. The variables of value 0 are known as redundant variables. If the elements of intercept matrix are arranged in decreasing order, the leading element becomes the dominant variable with value 1 and it provides optimum or near optimum solution of GAP. This process of identifying the leading element from intercept matrix is known as dominant principle. The dominant principle focuses at the resource matrix with lower requirement to come forward for maximizing the profit. The intercept matrix of the constraints(2) plays a vital role for achieving the goal in a heuristic manner.

The dominant principle for this problem can be divided into 3 categories namely constraint, column dominance and row dominance. For constraint dominant principle, an intercept matrix is constructed by dividing the right hand side of constraints by corresponding coefficients of constraints said in (2) of the section 2. First, we initialize the solution vector with zero value for all the unknowns(step-(a)). Next we construct the intercept matrix and identify redundant variables through step-(b) and (e). The Values corresponding to column minimum (e_{ij}) are multiplied with corresponding cost coefficients (c_{ij}) and the maximum among this product is chosen $ie., \underbrace{max}_{k}\ e_{ik}c_k$.

If the maximum product falls (r, k) the entry of intercept matrix, then the corresponding x_{rk} assumes the value 1. Next we update the availability (right hand side column vector) by using the relation $b_r = b_r - a_{rk}$, and then the coefficients of constraints a_{ik} are replaced by 0 for all i. This process is repeated n times.

Column dominant principle is used to improve the objective function value by reassigning row i to column j with higher profit. The step-(h) is meant for searching the dominant i for each column satisfying constraint (2) and (3) also focussing the maximal.

Row dominant principle is used to improve the objective function value by reassigning column j to row i with higher profit. The step-(i) identifies for each i, search for maximum profit that satisfies constraints (2) and (3).

We present below, the heuristic algorithm for solving GAP using dominance principle approach.

(a) Initialize the solution by assigning 0 to all x_j.

(b) Intercept matrix D,

$d_{jj} = b_i/a_{ij}$, if $a_{ij} > 0$,

d_{jj} = M, a large value ; otherwise.

(c) Identify 0 value variables(redundant): If any column has <1 entry in D, then the corresponding variable identified as a redundant variable.

(d) Dominant variable: Identify the smallest element (dominant variable) in each column of D.

(e) Multiply the smallest elements with the corresponding cost coefficients. If the product is Maximum in rth row and kth column, then set x_{rk} = 1 and update the objective function value $f(x_1, x_2, ..., x_n)$.

(f) Update the constraint matrix: $b_r = b_r - a_{rk}$ for all r and set a_{rk} = 0 for r = 1 to m.

(g) If a_{ij} = 0 for all i and j, then go to step-(h). Otherwise go to step-(b)

(h) Column dominant principle (to improve the current solution)

For each j (1 to n), identify row i such that x_{i*j} = 1 and satisfies both $c_{ij} > c_{i*j}$, and $\sum_{q=1ton, x_{iq}=1} a_{iq} + a_{ij}$ - $a_{i*j} \leq b_i$, i = 1 to m. If such an i can be found, then set x_{ij}=1 and x_{i*j} = 0.

(i) row dominant principle(to improve the current solution) For each i(m to 1), identify the column j such that x_{ij*} = 1 and satisfies both $c_{ij} > c_{ij*}$, and $\sum_{q=1ton, x_{iq}=1} a_{iq} + a_{ij}$ - $a_{ij*} \leq b_i$, i = 1 to m. If such an i can be found, then set x_{ij}=1 and x_{ij*}=0 (for j* column assign i, to satisfy $a_{ij*} \leq b_i - \sum_{i=1tom, x_{ij}=1} a_{ij}$.

(j) the objective function value is = $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$

Theorem 1. DPHEU can be solved in $O(mn^2)$ time, polynomial in the number of item types and constraints.

proof: The worst -case complexity of finding the solutions of an GAP using DPH can be obtained as follows. Assume that there are n variables and m constraints. The procedure of initialization (step-(a)) requires O(mn) running time. The Formation of D matrix involves n iterations, identification of less than one entry in each column, finding smallest intercept in each column, identification of rows which consists of more than one smallest intercept and updating of constraint matrix A. Since there are m constraints, step-(b), step-(c), step-(d),step-(e), and step-(g), require O(mn) running time each. The step-(f) requires O(n) operations to multiply cost with corresponding smallest intercept and updating the corresponding row of the constraint matrix. The number of iterations required for carrying out all operations in DPH is n. step-(h) and (i) are performed only once in that order.The step-(h) and (i) are attempted to improve the objective function value by reassigning jobs to agents with greater profit. In terms of computational complexity, step-(h) and step-(i) take O(mn) operations. Hence, the heuristic operator has a complexity of $O(n^2 m)$.

We illustrate this procedure to solve the generalized assignment problem given in [10] with m = 3 and n = 8. The iteration wise report is presented in Table I.

The DPH algorithm terminates the iterative process at 8^th iteration, since all the entries are in the constraint matrix are equal to zero. The objective function value is 232. Since step-(h) and step -(i) do not have any effect on columns 2 and 3 in Table I, they have not been shown.

Consider the first problem in GAP1[3], m = 5 and n = 15.The solution to the GAP follows

Stage 1 (step -(a) to step -(g))

Objective function value = 302 The variables x1,5 = x1,7 = x1,13 = x1,14 = x2,2 = x2,8 = x2,11 = x3,3 = x3,6 = x3,15 = x4,10 = x4,12 =x5,1= x5,4 = x 5,9 = 1 and all other variables = 0.

TABLE I
ITERATION WISE REPORT FOR GAP GIVEN IN[10]

Iteration	variables that assumes 1	variables that assumes 0	Objective function value
1	x17	x27,x37	41
2	x25	x15,x35	77
3	x28	x18,x38	111
4	x14	x24,x34	127
5	x26	x16,x36	152
6	x32	x12,x22	186
7	x31	x11,x21	220
8	x13	x23,x33	232

Stage 2 (step-(h))

The following Table II shows the changes of the values of the variables based on column dominant principle (step-(h) of DPH algorithm). At the end of Step-9 DPH returns objective function value as 316.

Stage3 (step-(i))

The following Table III shows the changes of the variables based on row dominant principle (step-(i) of DPH algorithm). Finally DPH gives the objective function value as 336, the optimum one. Thus the total number of iterations required is 22.

TABLE II
CHANGES MADE BY COLUMN DOMINANT PRINCIPLE

variables	value 1 to 0	value 0 to 1
1	X5,1	X2,1
2	X2,2	X5,2
3	X3,3	X4,3

TABLE III
CHANGES MADE BY ROW DOMINANT PRINCIPLE

variables	value 1 to 0	value 0 to 1
1	X5,2 , X5,4	X5,6
2	X4,6	X4,10
3	X3,10	X3,4
4	-	X2,2

V. COMPUTATIONAL RESULTS

The DPH has been coded in C language (DELL Core 2 Duo CPU 1.60GHz). The heuristics were first tested on a set of 12 small instances namely GAP instance 1 to 12 (each instance consists of 5 problems) used in [3], with sizes m x n, for m 5, 8, 10 and n 15, 20, 24, 25, 30, 32, 40, 48, 50, 60. The heuristic is again tested with a set of 30 large scale instances coded in MATLAB7, with sizes m $\in$ 5, 10, 20 and n $\in$ 100, 200,400. The data divided into five classes A, B, C, D and E, (each instance consists of 6 problems) were obtained from the OR - library [3]. Problems of classes A, B and C present increasing knapsacks. Class D and E are specially designed for minimization, the most difficult correlated problems. We considered Type D and E as maximization problem to test our algorithm's performance. The result of this heuristic for small size [3] is listed in Table 4 and detailed comparative study with other state-of-art algorithm is presented in Section 6. The results of DPH, TSDL, and RH for large size [32]

are presented in Table 5. The percentage of deviation of DPH solution from the optimum/near optimum ones has been calculated using the formula

$$PD = \frac{optimum(or)best - DPHsolution}{optimum(or)best} x100 \quad (5)$$

TABLE IV
DPH RESULTS FOR SMALL SIZE GAP

Problem set	m	n	Number of Problems	N.O.P.T	A.P.O.D	average solution time
GAP 1	5	15	5	4	0.35	0.1
GAP 2	5	20	5	5	0	0.2
GAP 3	5	25	5	4	0.03	0.31
GAP 4	5	30	5	3	0.09	0.46
GAP 5	8	24	5	5	0	0.42
GAP 6	8	32	5	4	0.03	0.62
GAP 7	8	40	5	4	0.4	0.71
GAP 8	8	48	5	3	0.03	0.76
GAP 9	10	30	5	3	0.61	0.74
GAP 10	10	40	5	4	0.02	0.83
GAP 11	10	50	5	5	0	0.88
GAP 12	10	60	5	5	0	0.91

A.P.O.D = Average percentage of deviation
N.O.P.T = Number of problems for which the DPH finds the optimal solution

TABLE V
DPH RESULTS FOR LARGE SIZE GAP

prob	optimum* / best solution	DPH solution	TSDL	RH	PD	solution time
A 1	4456*	4456*	4456*	4456*	0	1.36
A 2	8788*	8788*	8788*	8788*	0	1.74
A 3	4700*	4700*	4700*	4700*	0	2.52
A 4	9413*	9413*	9413*	9413*	0	2.02
A 5	4857*	4857*	4857*	4857*	0	1.97
A 6	9666*	9666*	9666*	9666*	0	2.86
B 1	4026	4026	4026	4008	0	1.69
B 2	8502	8502	8505	8502	0	1.19
B 3	4633*	4633*	4633*	4633*	0	1.94
B 4	9255	9255	9255	9255	0	2.33
B 5	4817*	4817*	4817*	4817*	0	2.53
B 6	9682	9682	9682	9670	0	2.76
C 1	4411*	4389	4411*	4411*	0.05	1.79
C 2	8347	8346	8346	8347	0.01	1.46
C 3	4535	4535	4535	4528	0	2.12
C 4	9258	9258	9258	9247	0	1.9
C 5	4790	4790	4790	4784	0	1.94
C 6	9625	9625	9625	9611	0	2.89
D 1	9147*	9147*	9147*	9147*	0	1.63
D 2	18750*	18750*	18750*	18750*	0	1.83
D 3	10349*	10349*	10349*	10349*	0	2.32
D 4	20562*	20562*	20562*	20562*	0	2.32
D 5	10839*	10839*	10839*	10839*	0	2.43
D 6	21733*	21733*	21733*	21733*	0	2.77
E 1	63228*	63228*	-	-	0	1.55
E 2	128648*	128648*	-	-	0	1.73
E 3	81054*	81054*	-	-	0	2.21
E 4	164317*	164317*	-	-	0	2.46
E 5	316844*	316844*	-	-	0	2.57
E 6	94432*	94432*	-	-	0	2.45

The first four colums of Table IV indicate the problem set, number of constraints, number of variables and the number of problems in the set. The next three columns report that DPH algorithm performance, like total number of optimum solution found by DPH, average percentage deviation from optimum solution, and average solution time. It is clear that from Table 4 DPH finds optimal or near optimal in all 60 test problems and the average solution time required by DPH is 0.13 seconds.
The results of DPH, TSDL, and RH for large size [32] are presented in Table 5. The first two columns of Table V indicate that name of the problem and optimum or best solution. The next three columns indicate that DPH, TSDL, and RH solutions respectively. The percentage of deviation of DPH solution from the optimum/best solution is presented in fifth column. The last column indicates that the solution time of DPH. It is clear that, out of 30 large sized problems, 28 problems have reached the optimum/best solution. The DPH has given near optimum solution for the remaining 2 problems with error 0.05 and 0.01 percentage. So the DPH has identified high quality solutions for large instances also.

The application of DPH for GAP D 5 X 100 (D1) is shown in Fig.1 with iterations versus objective function value. 100 is fixed to be the maximum number of iterations and the algorithm is found to reach the best solution (9010). For further iterations, remaining 2 steps-(h) and (i) are executed and the results are depicted in Fig.2. The improved objective function value is 9147, the optimum one.

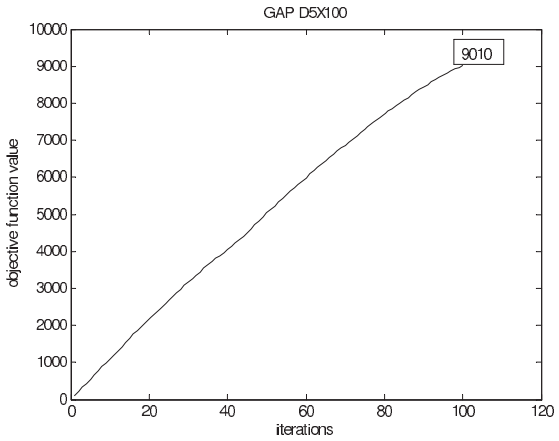

Fig. 1. Performance of DPH algorithm on GAP D5X100(from step-(a) to step-(g)

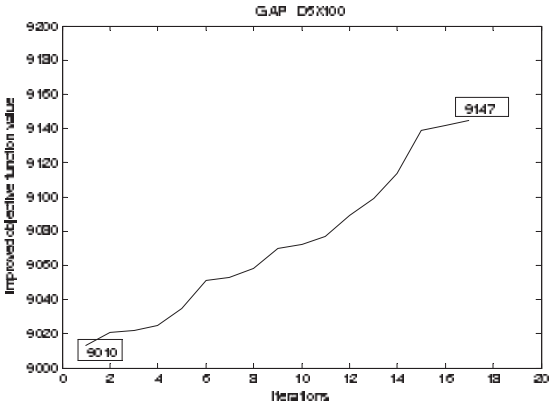

Fig. 2. Performance of DPH algorithm on GAP D5X100 (step-(h) and Step-(i)

As both tables and figures clearly demonstrate, the DPH is able to localize the optimal or near optimal point for all the test problems in quick time. Our approach is used to reduce the search space to find the optimal/near optimal solutions of the GAP. The computational complexity is cubic and the space complexity is O(mn). DPH reaches the optimum or near optimum point in less number of iterations where the

maximum number of iterations is the size of variables. Our heuristic algorithm identifies the zero value variables quickly.

VI. Comparison with other heuristics

The comparative study of DPH with other existing heuristic algorithms (GA, FJ, MTB, RS, TS1, SPH) has been furnished in Table VI in terms of the average deviation for each problem set, the average percentage deviation for all problems, the average percentage deviation of the best solutions and the number of optimal/best solutions obtained (out of a total 60) for each of the small-size problem. The computational times for the other algorithms are not given here since it is difficult to compare different codes and CPU times on different hardware platforms and algorithms with different stopping criteria. Our DPH obtains the results in a single execution like FJ, MTB and SPH, but our algorithm takes maximum of 1 second for small-sized GAP problems. The other heuristics are giving the solution from multiple executions. It can be observed that the proposed DPH heuristic performs the best among all heuristic in terms of the solution quality, being capable of finding the optimal solutions for 49 out of 60 problems.

TABLE VI
SUMMARIZED RESULTS FOR THE SOLUTION QUALITY(SMALL SIZE)

prob set	DPH	GA	FJ	MTB	RS	TS1	SPH
Gap 1	0.35	0	0	0	0	0	0.08
Gap 2	0	0	0	0	0	0.1	0.11
Gap 3	0.03	0	0	0	0	0	0.09
Gap 4	0.09	0	0.83	0.18	0	0.03	0.04
Gap 5	0	0	0.07	0	0	0	0.35
Gap 6	0.02	0.01	0.58	0.52	0.05	0.03	0.15
Gap 7	0.4	0	1.58	1.32	0.02	0	0
Gap 8	0.03	0.05	2.48	1.32	0.1	0.09	0.23
Gap 9	0.61	0	0.61	1.06	0.08	0.06	0.12
Gap 10	0.02	0.04	1.29	1.15	0.14	0.08	0.25
Gap 11	0	0	1.32	2.01	0.05	0.02	0
Gap 12	0	0.01	1.37	1.55	0.11	0.04	0.1
Aver	0.13	0.01	0.84	0.78	0.04	0.03	0.13
No. of opt and best known	60	60	26	24	39	45	40

GA: Genetic Algorithm [7]; FJ: Fisher, Jaikumar and VanWassenhove[10], branch-and-bound procedure with an upper CPU limit; MTB: Martello and Toth[28],vbranch-and-bound procedure with an upper CPU limit; RS: Osman [35], hybrid simulated annealing/tabu search;TS1: Osman [35], long term tabu search with best- admissible selection; SPH: Set Partitioning Heuristic [6]

For large-size problem set, Narciso[32] and Tai-His Wu[41] have run the program for maximization problems. The comparison between DPH, TSDL and RH heuristics has been reported in Table VII. It can be seen from Table VII the RH and TSDL algorithms found 15 optimal solutions, whereas DPH obtains 20 optimal solutions out of 30. Our DPH takes maximum of 3 seconds to reach optimum or near optimum solutions for both large and small-size GAP problems, but TSDL takes maximum of 172.911 CPU time to reach best solution for B(20 X 200) [41] and RPH takes maximum of 278.83 CPU time to complete some of the large-size problems reported in [32].

Features of DPH: The heuristic is used to reduce the search space to find the near-optimal solutions of the GAP. The computational complexity is O(n^2m) and the space complexity

TABLE VII
SUMMARIZED RESULTS FOR THE SOLUTION QUALITY(LARGE SIZE)

problem set	DPH	TSDL	Rh
Gap A	0	0	0
Gap B	0.00	0	0.1
Gap C	0.01	0.01	0.09
Gap D	0	0	0
Gap E	0	-	-
Average	0.0003	0.002	0.05
Total no of problems	30	25	25
No. of optimum	20	15	15
No. of best known	8	6	2

TSDL: Dynamic tabu tenure with long-term memory mechanism [41]; Rh: Lagrangian/surrogate relaxation heuristic for generalized assignment problems [32].

is O(mn). It reaches the optimum or near optimum point in n+k, (k<n) iterations where n is the number of jobs (variables) and k is any integer such that $0 \leq k < n$. Due to dominace principles, this heuristic identifies the zero value variables instantaneously. The maximum CPU time for small-size problem is 1 second and 3 seconds for large-size problem. It concludes that DPH algorithm is the effective one.

VII. Conclusion

In this paper, the dominant principle based approach for tackling the NP-Hard Generalized assignment problem (GAP) is presented. This heuristic has been tested for 90 state-of-art benchmark instances and found to produce optimal or near optimal solutions for all the problems given in literature. For the near optimal instances the average percentage of deviation of DPH solution from the optimum solution is very less. This heuristic is with complexity O(mn^2) and it requires n + k, ($k < n$) iterations to solve the GAP. The wide range of experimental data show that the optimality achieved by this heuristic almost 100 percentage. The basic idea behind the proposed scheme may be explored to tackle other NP-Hard Problems also.

Acknowledgment

The authors would like to thank Prof.T.R.Natesan(late), Anna University, Chennai, INDIA, for his motivation towards to the improvement of this paper.

References

[1] Amini, M.M., Racer, M, A rigorous comparison of alternative solution methods for the generalized assignment problem, Management Science 40, 868-890, 1994.
[2] Anderson, E.D. and K.D. Andersen, Presolving in linear programming. Math. Prog. Series B.,71:221-245, 1995.
[3] Beasley JE. OR-Library; Distributing Test Problems by Electronic Mail, Journal of Operational Research Society 41, 1069-1072, 1990.
[4] Brearley, A.L., G.Mitra and H.P Williams, Analysis of mathematical programming problem prior to applying the simplex algorithm . Math. Prog., 8: 54-83, 1975.
[5] Cattrysse, D.G., Wassenhove, L.N.V., A survey of algorithms for the generalized assignment problem. European Journal of Operational Research 60, 260-272, 1992.
[6] Cattrysse, D., Salomon, M and Van Wassenhove, L.N., A set partitioning heuristic for the generalized problem. European Journal of Operational Research., 72, 167-174, 1994.
[7] Chu, P.C., Beasley, JE., A genetic algorithm for the generalized assignment problem. Computers and Operations Research 24, 17-23, 1997.
[8] Diaz, J.A., Fernandez, E., A tabu search heuristic for the generalized assignment problem, European Journal of Operational Research 132, 22-38, 2001.

[9] Fisher, M.L., The Lagrangian relaxation method for solving integer programming problems. Management Science 27, 1-18,1981.
[10] Fisher, M. L., Jaikumar,R. and Van Wassenhove, L.N, A multiplier adjustment method for the generalized assignment problem. Mgmt Sci., 32, 1095-1103, 1986.
[11] Gottlieb, E.S., Rao, M.R., The generalized assignment problem: Valid inequalities and facets. Mathematical Programming 46, 31-52, 1990.
[12] Gowdzio, J.,Presolve analysis of linear program prior to applying an interior point method .Inform. J.Comput., 9: 73-91, 1997.
[13] Guignard M., Rosenwein M.B. An improved dual based algorithm for the generalized assignment problem, Operations Research 37 (4), 658-663, 1989.
[14] Haddadi, S., Lagrangian decomposition based heuristic for the generalized assignment problem. INFOR 37, 392-402, 1999.
[15] Haddadi, S., Ouzia, H., An effective Lagrangian heuristic for the generalized assignment problem,. INFOR 39, 351-356, 2001.
[16] Haddadi, S., Ouzia, H., Effective algorithm and heuristic for the generalized assignment problem. European Journal of Operational Research 153, 184-190, 2004.
[17] Harald Feltl and Gunther R. Raidl., An improved hybrid genetic algorithms for the generalized assignment problem, SAC '04, Nicosia, Cyprus, march 14-17, 2004.
[18] Higgins, A.J. A dynamic tabu search for large-scale generalized assignment problems, Computers and Operations Research 28 (10), 1039-1048, 2001.
[19] Ioslovich, I., Robust reduction of a class of large scale linear program. Siam J. Optimization, 12: 262-282, 2002.
[20] Jeet V., Kutanoglu E., Lagrangian relaxation guided problem space search heuristics for generalized assignment problems, European Journal of Operational Research 182, 1039-1056, 2007.
[21] Jornsten K., Nasberg M., A new lagrangian relaxation approach to the generalized assignment problem, European Journal of Operational Research 27, 313-323, 1986.
[22] Karwan, M.H., V. Loffi, J. Telgan and S. Zionts, Redundancy in mathematical Programming: A State of the Art Servey (Berlin: Springer-Verlag), 1983.
[23] Klastorin T.D. An effective subgradient algorithm for generalized assignment problem, Computers and Operations Research 6, 155-164, 1979.
[24] Kuhn, H.W. and 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: 107-124, 1962.
[25] Laguna, M., Kelly, J.P., Gonzalez Velarde, J.L., Glover, F. Tabu search for the multilevel generalized assignment problem, European Journal of Operational Research 82, 176-189, 1995.
[26] Lorena L.A.N., Narciso M.G., Relaxation heuristics for a generalized assignment problem, European Journal of Operational Research 91, 600-610, 1996.
[27] Martello, S. P. Toth., An algorithm for the generalized assignment problem, operational Research '81, ed. J.P.Brans. North-Holland, 589-603, 1981.
[28] Martello, S., Toth P. Knapsack Problems: Algorithms and Computer Implementations, Wiley, New York, 1990.
[29] Matthesiss, T.H., An Algorithm for determining irrelevant constraints and all vertices in systems of linear inequalities. Operat. Res., 21: 247-260, 1973.
[30] Meszaros. C. and U.H. Suhl, Advanced preprocessing techniques for linear and quadratic programming, Spectrum, 25: 575-595, 2003.
[31] Monfared. M.A.S and M . Etemadi., The impact of energy function structure on solving generalized assignment problem using Hopfield neural network, European Journal of Operational Research 168, 645-654, 2006.
[32] Narciso, M.G., Lorena, L.A.N. Lagrangian/surrogate relaxation for generalized assignment problems. European Journal of Operational Research 114 (1), 165-177, 1999.
[33] Nauss, R.M., Solving the generalized assignment problem:An optimizing and heuristic approach. INFORMS Journal of Computing 15 (3), 249-266, 2003.
[34] Nauss, R.M., The generalized assignment problem. In: Karlof, J.K. (Ed.), Integer Programming: Theory and Practice. CRC Press, Boca Raton, FL, 39-55, 2006.
[35] Osman, I.H., Heuristics for the generalized assignment problem: Simulated annealing and tabu search approaches. OR Spektrum 17, 211-225, 1995.
[36] Paulraj, S., C. Chellappan and T.R. Natesan, A heuristic approach for identification of redundant constraints in linear programming models. Int. J. Com. Math., 83(8): 675-683, 2006.
[37] 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, 71-77, 2008.
[38] Ross, G.T., Soland, R.M., A branch and bound algorithm for the generalized assignment problem, Mathematical Programming 8, 91-103, 1975.
[39] Savelsbergh, M., A branch-and-price algorithm for the generalized assignment problem. Operations Research 45 (6), 831-841, 1997.
[40] Srojkovic, N.V. and P.S. Stanimirovic, Two direct methods in linear programming. European J. Oper. Res., 131: 417-439, 2001.
[41] Tai- Hsi Wu , Jinn-Yi Yeh, and Yu -Ru Syau., A tabu search approach to the generalized assignment problem, Journal of Chinese institute of Industrial Engineers, vol. 21, no. 3, pp. 301-311, 2004.
Telgan, J., Identifying redundant constraints and implicit equalities in system of linear constraints. Manage. Sci., 29: 1209-1222, 1983.
[42] Tomlin, J.A. and J.S Wetch, Finding duplicate rows in a linear programming model. Oper. Res. Let., 5: 7-11,1986.
[43] Trick, M.A., A linear relaxation heuristic for the generalized assignment problem. Naval Research Logistics 39, 137-152,1992.
[44] Yagiura, M., Ibaraki, T., Glover, F., An ejection chain approach for the generalized assignment problem. INFORMS Journal of Computing 16 (2),133-151, 2004.
[45] Yagiura.M., Ibaraki.T, Glover, F, A path relinking approach with ejection chains for the generalized assignment problem. European journal of Operational Research. 169, 548-549, 2006.
[46] Yagiura.M and T. Ibaraki. Generalized Assignment Problem, in: T.F. Gonzalez, ed., Handbook of Approximation Algorithms and Metaheuristics, Chapman and Hall/CRC in the Computer and Information Science Series, Chapter 48 (18 pages), 2007.