〈研究論文〉

Journal of the Korean Institute of Industrial Engineers
Vol. 14, No. 2, Dec. 1988

An Efficient Heuristic for the Generalized Assignment Problem

Tai Yun Kim*

Abstract

In this paper an efficient heuristic for the generalized assignment problem(GAP) is presented. A new lower bound that is slightly improved by adding the feasibility constraint is used to measure the quality of solutions obtained by the heuristic. The heuristic was tested on a number of large-scale random problems and a large sample of small problems. The heuristic appears to be better than the best previously existing heuristic for GAP. Its effectiveness in comparison of running times and closenesses to lower bound is discussed.

1. Introduction

Consider the following assignment problem, where there are n jobs to be assigned to m agents. The resource of each agent is limited. The total available resource of agent i is bi. Each job requires a amount of resource. The resource required by agent i to do job j is aij. The jobs should be assigned to minimize total operation cost, given the resource availability.

The problem described can be formulated as follows.

$$\text{Minimize} \sum_{i \in I} \sum_{j \in J} CijXij \qquad \text{(GAP)}$$

$$\text{subject to} \sum_{i \in I} Xij = 1 \qquad j \in J \cdots\cdots (1)$$

$$\sum_{j \in J} aijXij \leq bi \qquad i \in I \cdots\cdots (2)$$

$$Xij \in \{0, 1\} \qquad i \in I, j \in J$$

where

* Department of Computer Science, Korea University

$$Xij = \begin{cases} 1, & \text{if job j is assigned to agent i} \\ 0, & \text{otherwise} \end{cases}$$

Cij:cost of having agent i do job j
I:set of agents
J:set of jobs

This is the generalized assignment problem(GAP), which is well-known in Operations Research.

The classical assignment problem is a special case of the generalized assignment problem in which aij=1 for all i and j, and the number of jobs is equal to the number of agents. There are many other applications of the generalized assignment model. Some of these are the assignment of software development tasks to programmers, the assignment of jobs or data-bases to computers in a computer network, the scheduling of variable length television or radio commercials into time slots, the scheduling of payments on accounts where "lump sum" payments are specified, vehicle routing problems for assigning customers to vehicles, and fixed charge plant location models in which customer requirements must be satisfied by a single plant.

In 1975, Ross and Soland[10] developed a branch and bound algorithm for solving GAP, as did Martello and Toth [8] and Fisher et al.[4]. The branch and bound algorithms of Ross and Soland and Fisher et al. have a common feature. In both methods, the lower bounds are obtained from a Lagrangian relaxation in which the constraint set

$$\sum_{i \in I} Xij = 1 \qquad j \in J$$

is dualized, To set the Lagrangian multipliers, Ross and Soland used linear programming, while Fisher et al. used a heuristic multiplier adjustment method.

Ross and Soland provided a lower bound for the generalized assignment problem by solving binary knapsack problems, which will be biscussed in the next section. In the branch and bound algorithm of Martello and Toth[8], lower bounds are computed by using the Ross and Soland scheme at each node of the decision-tree. Later, Ross and Soland[11] modified the GAP algorithm to take advantage of the special structure of the facility location problem. Fisher et al.[4] proved that GAP is NP-complete, which suggests that it may be more reasonable to devise good heuristic procedures to solve large-scale problems than to develop exact algorithms.

Fisher and Jaikumar[3] presented a heuristic for the vehicle routing problem in which an assignment of customers to vehicles is obtained by solving a generalized assignment problem. The heuristic is based on a Lagrangian relaxation in which the multipliers are determined by a multiplier adjustment method. Martello and Toth[8] also provided a heuristic, HGAP, which exhibits good experimental performance.

2. Lower Bound

Ross and Soland[10] introduced a lower bound for the generalized assignment problem, but

the bound can be improved as follows.

$$LB = Z + \sum_{i \in I} z_i$$

where Z is the optimal solution of the following problem.

$$\text{(P)} \quad \begin{aligned} &\text{Minimize} \sum_{i \in I} \sum_{j \in J} C_{ij}X_{ij} \\ &\text{st.} \quad \sum_{i \in I} X_{ij} = 1 \qquad j \in J \\ &\qquad a_{ij}X_{ij} \leq b_i \qquad i \in I,\ j \in J \quad \cdots\cdots\cdots\cdots\cdots\cdots (3) \\ &\qquad X_{ij} \in \{0, 1\} \end{aligned}$$

where

$F_i = \{j \mid X_{ij} = 1$ in the solution of(P), $j \in J\}$, for $i \in I$

$I = \{i \mid \sum_{j \in F_i} a_{ij}X_{ij} > b_i,\ i \in I\}$.

z_i, $i \in I$, is the optimal solution of the followint biinary knapsack problem.

$$\text{Minimize} \sum_{j \in F_i} p_jY_{ij}$$

$$\text{st.} \quad \sum_{j \in F_i} a_{ij}Y_{ij} \geq d_i$$

$$Y_{ij} \in \{0, 1\}$$

where

$$d_i = \sum_{j \in F_i} a_{ij}X_{ij} - b_i$$

$$p_j = \min_{k \in J - \{j(i)\}} \{C_{ik} - C_{ij(i)}\}$$

where $a_{ik} \leq b_i$

$j(i)$ = job j having the minimum C_{ij} for agent i, $i \in I$

The lower bound of Ross and Soland did not take into account the feasibility constraint(3). The lower bound has been slightly improved by adding the feasibility constraint to P and the calculation of pj. This lower bound will be used to measure the quality of solutions obtained by the following heuristic NH.

3. New Heuristic(NH) for the Generalized Assignment Problem

(1) Basic Concept

This algorithm is based on a combination of the transportation simplex method and the

Hungarian assignment method. It consists of two phases, initialization and improvement. In Phase I, four rules are used to attempt to obtain a feasible solution. The first rule successively assigns jobs to the agent having the maximum difference between the minimum cost and the next lowest cost. The second rule(minimal-cost rule) successively assigns each job to the agent having the minimum cost in whoch it fits. The third rule successively assigns each job, by decreasing resource amount required, to the agent having the minimum cost. The fourth rule is the best-fit-decreasing(BFD) rule. Each job is successively assigned to the agent for which the resulting available resource is minimal, in minimal, in order of decreasing(or increasing) resource amount required. This rule disregards the cost element, and therefore may result in the most costly feasible solution. It is useful, however, when the sum of resource amounts required is close to the total resource availability. All four rules may fail to find a feasible solution.

We hope that every job is assigned to the agent that results in the lowest operating cost for the job. However, it may not be possible due to resource limitations. After finding an initial feasible assignment in Phase I, adjustments are made in Phase II to improve the assignment. First of all, a cost matrix is built, in which the row indicates job and the column indicates agent. As this algorithm is used to assign only single jobs, each row has only one assignment, but a column may have more than one according to the resource availability of the agent. Phase II tries to move each job to its best agent; that is, the agent with the lowest cost(or as low as possible). If the agent having the lowest or lower cost element in a row is already taken by another job(s) and that agent's remaining resource is not enough for the new job, this algorithm tries to move the old job(s) out and the new job in. This change is made only if the decrease in cost that results from moving in the new job is greater than the increase that results from moving out the old job(s).

(2) Heuristic NH

Define

$$Xij = \begin{cases} 1, & \text{if dob j is assigned to agent i} \\ 0, & \text{otherwise} \end{cases}$$

b_i = the initial resource availability of agent i
b_i' = the currently available resource of agent i

Phase I. Initialization
Rule 1:
Step 1. Let the set of jobs $N = \{1,2,\cdots, n\}$ and the set of agents $M = \{1,2,\cdots,m\}$. Let $U = N$.
Step 2. For each j, $j \in U$, find

$$Ci'j = \underset{i \in M}{\text{Minimum}} \{Cij\}$$

$$Ci''j = \underset{i \in M-\{i'\}}{\text{Minimum } i} \{Cij\}$$

where $aij \leq b'i$.

If no $C_{i'j}$ exists, then apply Rule 2.

$P_j = C_{i'j} - C_{i^*j}$. $j \in U$

If C_{i^*j} does not exist, $P_j = \infty$.

Step 3. Find(i', j') with

$$P_{j'} = \underset{j \in U}{\text{Maximum}} \{P_j\}$$

Step 4. Assign j' to i'.

$b_{i'} = b_{i'} - a_{i'j'}$

$X_{i'j'} = 1$

Remove j' from U. If U is empty, stop--an initial feasible solution is available. Otherwise, return to Step 2.

Rule 2:

Step 1. Let the set of jobs $N = \{1,2,\cdots,n\}$ and the set of agents $M = \{1,2,\cdots,m\}$.

Let

$T = \{(i,j) \mid i \in M, j \in N\}$.

Step 2. Find(i',j')with

$$C_{i'j'} = \underset{(i,j) \in T}{\text{Minimum}} \{C_{ij}\}$$

If there are ties, choose the one with the smallest aij. When aij's are the same, choose the one with the largest b_j. If $b_{i'} < a_{i'j'}$, go to Step 3. Otherwise, assign j' to i'.

$b_{i'} = b_{i'} - a_{i'j'}$

$X_{i'j'} = 1$

Go to Step 4.

Step 3. Remove($i'j'$) from T. If T is empty, apply Rule 3. Otherwise, Return to Step 2.

Step 4. Remove($i'j'$) from T and remove j from N. If N is empty, stop-- an initial feasible soultion is available. apply Rule 3. If T is empty, apply Rule 3.. Otherwise, return to Step 2.

Rule 3:

Step 1. Let the set of jobs $N = \{1,2,\cdots,n\}$

Step 2. Find j', $j \in N$, with

$$\underset{j \in N}{\text{Maximum}} \left\{ \sum_{i \in M} a_{ij} \right\}$$

Break ties arbitrarily. Let the set of agents $M = \{1,2,\cdots,m\}$.

Step 3. Find i', $i \in M$, with

$$C_{i'j'} = \underset{i \in M}{\text{Minimum}} \{C_{ij'}\}$$

If there are ties, choose the one with the largest b_i. If $b_{i'} < a_{i'j'}$, go to Step 4. Otherwise, assign j' to i'.

$b_{i'} = b_{j'} - a_{i'j'}$

$X_{i'j'} = 1$

Go to Step 5.

Step 4. Remove i from M. If M is empty, apply Rule 4. Otherwise, return to Step 3.

Step 5. Remove j from N. If N is empty, an initial feasible solution is available. Compare the solution from this rule with the one from Rule 2. Choose the better one, and then stop. Otherwise, return to Step 2.

Rule 4:

Step 1. Let the set of jobs $N = \{1,2,\cdots,n\}$

Step 2. Find j', $j \in N$, with

$$\underset{j \in N}{\text{Maximum}} \left\{ \sum_{i \in N} aij \right\}$$

Break ties arbitrarily, Let the set of agents $M = \{1,2,\cdots,m\}$.

Step 3. For each i, $i \in M$,

$b''i' = b'i' - aij'$

If all $b''i's < 0$, apply Rule 4.1. Otherwise, find i', $i \in M$, with

$$b''i' = \underset{i \in M}{\text{Minimum}} \{b''i\} \text{ and } b''i \geq 0$$

Break ties arbitrarily. Assign j' to i'.

$b'i' = b'i' - ai'j'$

$Xi'j' = 1$

Remove j' from N. If N is empty, stop--an initial feasible solution is available. Otherwise, return to Step 2.

Rule 4.1:

Step 1. Let the set of jobs $N = \{1,2,\cdots,n\}$.

Step 2. Find j', $j \in N$, with

$$\underset{j \in N}{\text{Minimum}} \left\{ \sum_{i \in N} aij \right\}$$

Break ties arbitrarily. Let the set of agents $M = \{1,2,\cdots,m\}$.

Step 3. For each i, $i \in M$,

$b''i = b'i - aij'$

If all $b''i's < 0$, stop--a feasible solution cannot be obtained by this algorithm. Otherwise, find i', $i \in M$, with

$$b''i' = \underset{i \in M}{\text{Minimum}} \{b''i\} \text{ and } b''i \geq o$$

Break ties arbitrarily. Assign j' to i'.

$b'i' = b'i' - ai'j'$

$Xi'j' = 1$

Remove j' from N. If N is empty, stop--an initial feasible solution is available. Otherwise, return to Step 2.

Phase II. Improving the Initial Feasible Solution

Step 1. Let the set of jobs $N = \{1,2,\cdots,n\}$ and the set of agents $M = \{1,2,\cdots,m\}$.

$$Xij = \begin{cases} 1, & \text{if job j is assigned to agent i} \\ 0, & \text{otherwise} \end{cases}$$

$$Z = \sum_{i=1}^{m} \sum_{j=1}^{n} CijXij$$

LB = lower bound

$$b'i = bi - \sum_{j \in N} aijXij \quad i \in M$$

Step 2. If Z = LB, stop--the current solution is optimal. Let Vij = Cij − Ci′j , j ∈ N, i∈M − {i′} Xi′j = 1. and P = {(i,j) | Vij<0, i ∈ M, j ∈ N} .
If all Vij ≥0, stop--an optimal solution is available.

Step 3. If P is empty, stop--the final solution is available. If Z = LB, that solution is optimal.

Step 4. Let $Vi'j' = \underset{(i,j) \in P}{\text{Minimum}} \{Vij\}$

where Xi~j′ = 1.
If there are ties, choose one with the smallest aij. If b′i′<ai′j′, go to Step 5. Otherwise, assign j′ to i′. Remove(i′,j′) from P. Let

$$b'i = b'i' - ai'j'$$
$$b'i\sim = b'i\sim + ai\sim j'$$
$$Z = Z - Ci\sim j' + Ci'j'$$
$$Xi\sim j' = 0$$
$$Xi'j' = 1$$

Return to Step 2.

Step 5. Set
R = R′ = {(i,j) | Vij < | Vi′j′ | , b′i′ + ai′j≥ai′j′, Xi′j = 1, i ∈ M, j ∈ N}
If R is not empty, go to Step 7.

Step 6. Remove (i′, j′) from p. Return to Step 3.

Step 7. Let $Vi''j'' = \underset{(i,j) \in R}{\text{Minimum}} \{Vij\}$

where Xi~j″ = 1.
If there are ties, choose one with the largest b′i. If b′ i″≥ai″ j″ , go to Step 8. If Xi″ j′ = 0, go to Step 9. If b′i″ + ai″ j′≥ai″ j″ , go to the next step. Otherwise, go to Step 9.

Step 8. Move j″ from i′ to i″. Transfer j′ to i′. Let

$$b'i = b'i + ai'j'' - ai'j'$$
$$b'i\sim = b'i\sim + ai\sim j'$$
$$b'i'' = b'i'' - ai''j''$$
$$Z = Z - Ci\sim j' + Ci'j' - Ci'j'' + Ci''j''$$
$$Xi\sim j' = Xi'j'' = 0$$
$$Xi'j' = Xi''j'' = 1$$

Return to Step 2.

Step 9. Remove(i'', j'') from R. If R is empty, go to Step 10. Otherwise, return to Step 7.

Step 10. Let $Vi''j'' = \underset{(i,j) \in R'}{\text{Minimum}} |Vij|$

where $Xi^{-}j'' = 1$.
If there are ties, choose one with the largest $b'i$. Let $U = \{ j \mid Xi''j = 1, \in N - |j'| \}$. If U is not empty, go to Step 12.

Step 11. Remove(i'', j'') from R'. If R' is empty, return to Step 6. Otherwise, return to Step 10.

Step 12. Let $U' = \phi$. $SUMV = Vi''j''$. If $Xi''j' = 1$,
$SUML' = ai'j'$ and $SUML'' = ai''j'$. Otherwise,
$SUML' = SUML'' = 0$.

Step 13. Let $Vi'j^{-} = \underset{j \in U}{\text{Minimum}} |Vi'j|$

Set $U = U - |j^{-}|$
$U' = U' \cup |j^{-}|$
$SUML' = SUML' + ai'j^{-}$
$SUML'' = SUML'' + ai''j^{-}$
$SUMV = SUMV + Vi'j^{-}$
Check the following constraints.
$b'i' + ai'j'' \geq SUML'$
$b'i'' + SUML'' \geq ai''j''$
$SUMV < |Vi'j'|$
If those constraints are satified, go to Step 14. If U is empty, return to Step 11. Otherwise, repeat Step 13.

Step 14. .Move j'' from i' to i''. Transfer j' to i'
Move all j, $j \in U'$, from i'' to i'. Let

$$b'i' = b'i' + ai'j'' - \sum_{j \in U'} ai'j - ai'j'$$

$$b'i^{-} = b'i^{-} + ai^{-}j'$$

$$b'i'' = b'i'' + \sum_{j \in U'} ai''j' - ai''j''$$

$$Z = Z - Ci^{-}j' + Ci'j' - Ci'j' + Ci'j'' + Ci''j'' + \sum_{j \in U'} Ci'j - \sum_{j \in U'} Ci''j$$

$$Xi^{-}j' = Xi'j'' = 0$$

$$Xi'j' = Xi\ j'' = 1$$

$$Xi''j = 0,\ j \in U'$$

$$Xi'j = 1,\ j \in U'$$

Return to Step 2.

(3) Optimality Criterion

This is the fundamental criterion for determining whether a particular feasible solution is optimal or not. If a solution satisfies one the following conditions, its optimality is guaranteed.

1) If Vij is always greater than or equal to zero, this feasible solution is optimal.
2) If the solution is the same as the lower bound LB, it is optimal.

Proof:

1) If Vij is always greater than or equal to zero, every job is assigned to the agent that has the smallest cost element for the job. The total operatin cost is the sum of operating costs for all jobs. If each job is assigned to the agent with the lowest operating cost for that job, it is obious that the total operating cost is the lowest. Thus the proof is obvious.
2) Any optimal solution cannot be better than the lower bound LB. The proof is also obvious.

4. Computational Results

The algorithm NH was programed in FORTRAN and executed on the IBM − 3033. The speed and accuracy of NH were tested by comparing its results with the computational results of Martello-Toth [8] and by comparing the solution to the improved lower bound LB. Problems were generated according to the procedure described by Martello and Toth.

Table shows the performance of NH on generalized assignment problems in terms of speed, accuracy, and the number of times that no feasible solution was obtained by NH. Problems were generated as follows.

aij and Cij are integers selected from a Uniform distribution between 5 and 25 and between 1 and 40 respectively, and

$$bi = [0.4(n/m)15 + 0.6 \max_{i \in I} \{\Sigma aij\}]P \quad j \in J$$

Table. Computational Experience—GAP Problems

P	m	n	Average running time		#of times a F.S. Was not found*		Closeness to (%) lower bound	
			HGAP	NH	HGAP	NH	HGAP	NH
1.15	5	50	0.09	0.13	3	1	95.3	98.2
1.30	5	100	0.20	0.47	4	1	99.1	99.5
1.50	5	200	0.72	1.08	0	0	98.0	99.6
.70	10	50	0.06	0.12	0	0	97.4	97.8
.88	10	100	0.48	1.08	5	0	95.8	97.3
1.12	10	200	1.40	2.85	1	0	94.7	98.3
.22	20	50	0.13	0.78	1	0	75.2	82.5
.40	20	100	0.53	1.38	3	0	93.2	94.9
.60	20	200	4.80	6.27	6	1	95.1	97.3

Where CPU time is IBM-3033 seconds. Ten problems were run for each entry. *: feasible solutions were not obtained by the heuristic, but the problem's feasibility is unknown.

where P is a constant, n is the number of jobs, and m is the number of agents.

The results in Table consistently show that NH is superior to the GAP heuristic, HGAP, of Martello and Toth[8]. NH and HGAP were coded and executed at the same time. NH was slightly slower than HGAP, but its accuracy was much better. The slowness of a few seconds in not significant. In addition, HGAP often does not find feasible solutions when they exist. NH, on the other hand, finds a good feasible solution in most cases. In fact HGAP failed to find a feasible solution for 23 problems while HN found a feasible solution for all but 3 problems.

Two experimental designs have been implemented to check the performance of HN. Computational experiments have failed to find a problem class that causes poor results.

5. Conclusion

This paper has presented an efficient heuristic for solving the generalized assignment problem. The algorithm consists of two phases, initialization and improvement. In the first phase, four rules are used to attempt to obtain a feasible solution. If an initial feasible assignment is found, adjustments are made in the second phase to improve the assignment. The algorithm does so in a polynomial effort. The heuristic algorithm was tested in terms of speed and accuracy and found to be effective in solving real, life-size problems. A new lower bound that is slightly improved by adding the feasibility constraint to the previously existing lower bound is used to measure the accuracy of solutions obtained by the heuristic. In computational experiments the heuristic outperforms other heuristics for the generalized assignment problem.

References

1. Bulfin, R.L., Parker, and C.M. Shetty, "Computational Results with a Branch-and-Bound Algorithm for the General Knapsack Problem," Naval Research Logistics Quarterly 26, 41, 1979.
2. Eswaran, K.P., "Placement of Records in a File and File Allocation in a Computer Network," Information Proceedings 1974, Proc. IFIP Congress, Stockholm, 1974.
3. Fisher, M.L. and R. Jaikumar, "A Generalized Assignment Heuristic for Vehicle Routing," Networks, Vol. 11, 109 − 124, 1981.
3. Fisher, M.L., R. Jaikumar, and L.N. Van Wassenhove, "A Multiplier Adjustment Method for the Generalized Assignment Problem, "Management Science, Vol. 32, No. 9, 1095 − 1103, 1986.
5. Jornsten, K. and M. Nasberg, "A New Lagrangian Relaxation Approach to the Generalized Assignment Problem," European Journal of Operational Research 27, North-Holland, 313 − 323, 1986.
6. Karp, R.M., "Complexity of Computer Computations," *Reducibility among Combinatorial Problems*, Plenum Press, New York, 85 − 103, 1972.
7. Kim, T.Y., Optimal File Allocations in Large-Scale Distributed Comptuter Networks, Ph. D. Dissertation, Auburn University, Auburn, AL, 1987.
8. Martello, S. and P. Toth. "Linear Assignment Problems," Annals of Discrete Mathematics, 31, North-Holland, 259 − 282, 1987.

9. Ramamoorthy. C.V. and B.W. Wah, "The Isomorphism of Simple File Allocation," IEEE Trans. Comput., vol. C – 32, No. 3, 221 – 232, 1983.
10. Ross, G.T. and R.M. Soland, "A Branch and Bound Algorithm for the Generalized Assignment Problem," Math. Programming, vol. 8, 92 – 103, 1975.
11. Ross, G.T. and R.M. Soland, "Modeling Facility Location Problems as Generalized Assignment Problem, Management Science, vol. 24, No. 3, 345 – 357, 1977.
12. Suri, R., "A Decentralized Approach to Optimal File Allocation in Computer Networks," IEEE Trans. Comput., 141 – 146, 1979.
13. Urano, U., Ono, and S. Inoue, "Optimal Design of Distributed Networks," Proc. Int, Conf. on Comput. Commun., 2nd, 413 – 420, 1974.