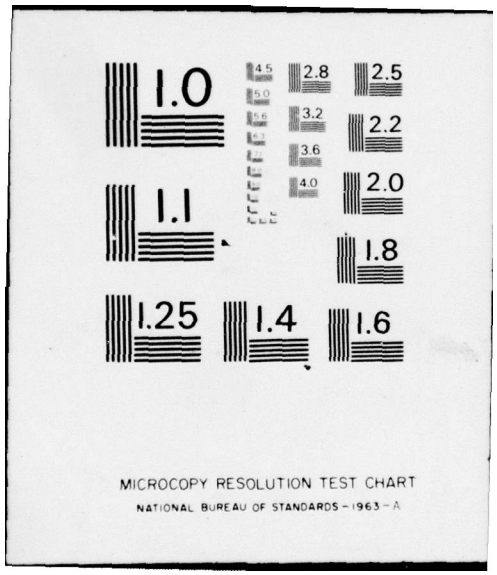
AD-A031 386 TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES F/G 12/2
A NOTE ON THE BOUNDED INTERVAL GENERALIZED ASSIGNMENT PROBLEM.(U)
JUL 76 G T ROSS, R M SOLAND, A A ZOLTNERS N00014-75-C-0616
UNCLASSIFIED CCS-253 NL

END

DATE FILMED
11-76


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

12

CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

Research Report CCS 253

A NOTE ON THE BOUNDED INTERVAL GENERALIZED ASSIGNMENT PROBLEM

by

G. Terry Ross
Richard M. Soland*
Andris A. Zoltners**

July 1976

*Universite de Montreal, Montreal, Canada.

**University of Massachusetts, Amherst, Massachusetts.

This research was partly supported by Project NR047-021, ONR Contracts N00014-75-C-0616 and N00014-75-C-0569 with the Center for Cybernetic Studies, The University of Texas. Reproduction in whole or in part is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 203E
The University of Texas
Austin, Texas 78712
(512) 471-1821

Abstract

The bounded interval generalized assignment problem is exemplified by the problem of assigning tasks to agents so that each task is assigned to exactly one agent and the time required to complete the set of tasks assigned to any one agent falls between prespecified lower and upper bounds. This note describes an efficient algorithm for solving this problem.

1.0 Introduction

Assignment models are characterized by a set of indivisible tasks, a set of agents responsible for completing the tasks, and a criterion function for evaluating pairings of agents and tasks. In many practical problems, the number of tasks exceeds the number of agents, and this characteristic gives rise to "many-for-one" assignment models. In these models, it is possible for an agent to be assigned a set of tasks which collectively require more of some resource (e.g. time) than the agent has available. To accommodate this factor, assignment models and special purpose algorithms have been proposed that explicitly consider the resource burden associated with multiple task assignments [4,11]. An important generalization of these models, known as the generalized assignment problem, additionally considers differences in the agents' abilities to complete the same task [9]. Specific applications of this model include the allocation of inventory items to warehouses [4], the assignment of ships to shipyards for overhaul [8], the assignment of jobs to computers in computer networks [1] and the assignment of software development tasks to programmers.

Analysis of other assignment problems reveals additional restrictions limiting the pairings of agents and tasks. As suggested in [12], some situations require each agent to contribute a minimal amount of his resources to completing the tasks. Such constraints may reflect managerial policies that require an equitable distribution of work. Consideration must be given to the relative strengths and weaknesses of the agents, and the agents should neither be overburdened nor underutilized [5].

Analagous restrictions arise in other contexts. In machine loading models, it may be desirable to balance the machine workloads rather than allowing some heavily loaded and some lightly loaded machines. Similarly, capacity and

configuration constraints for facility location models may restrict both the minimum and the maximum size of a facility or the number of facilities [10]. Finally, territory design procedures for problems of political districting, school districting and sales districting [14] require an equitable distribution of some entity (such as voters, minority students or sales potential) among the districts.

The purpose of this note is to formulate these generalized assignment problems with bounded interval resource limitations and to show how the branch and bound algorithm of [9] may be extended to solve these problems.

2.0 The Bounded Interval Generalized Assignment Problem

The bounded interval generalized assignment problem may be formulated as:

$$(P) \quad \text{minimize} \quad Z = \sum_{i\in I} \sum_{j\in J} c_{ij}x_{ij}$$

$$\text{subject to} \quad \sum_{i\in I} x_{ij} = 1 \text{ for every } j\in J \qquad (1)$$

$$a_i \leq \sum_{j\in J} r_{ij}x_{ij} \leq b_i \text{ for every } i\in I \qquad (2)$$

$$x_{ij} = 0 \text{ or } 1 \text{ for every } i\in I,\ j\in J \qquad (3)$$

The set $J \equiv \{1, 2, \ldots, n\}$ represents the tasks which must be completed, the set $I \equiv \{1, 2, \ldots, m\}$ represents the agents who can be assigned, and the parameters c_{ij} and r_{ij} are, respectively, the cost incurred and the resource required if agent i is assigned task j. The natural interpretation of the decision variable is

$$x_{ij} = \begin{cases} 1 \text{ of agent i is assigned task j} \\ 0 \text{ otherwise} \end{cases}$$

The constraints (1) and (3) insure that each task is uniquely assigned to a single agent. Constraints (2) insure that each agent expends no less than a_i and no more than b_i resource units in accomplishing his assigned tasks. The usual generalized assignment problem [9] is a special case of (P) in which $a_i = 0$ for all $i\in I$. We shall refer to that problem as (P_o).

3.0 An Algorithm For Solving (P)

The branch and bound algorithm of [9] exploits the special mathematical structure of (P_o). Our objective here is to show how that algorithm can be easily extended to solve (P). Moreoever, we shall show how the necessary modifications preserve, as much as possible, the computational efficiency of the Ross and Soland algorithm.

A bounding procedure similar to that of [9] is employed. Its validity can be established using the theory of Lagrangean relaxation [6]. That is, lower bounds for (P) can be calculated by solving the following relaxation of (P):

$$(PR_\lambda) \quad \text{minimize } Z_\lambda = \sum_{i\in I} \sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J} \lambda_j \left(1 - \sum_{i\in I} x_{ij}\right)$$

$$\text{subject to:} \quad a_i \leq \sum_{j\in J} r_{ij}x_{ij} \leq b_i \quad \text{for every } i\in I$$

$$x_{ij} = 0 \text{ or } 1 \quad \text{for every } i\in I,\ j\in J.$$

The problem (PR_λ) relaxes the requirement that every task must be completed, but it incorporates the constraints (1) in the objective function with fixed multipliers to penalize the optimal value whenever these constraints are not satisfied. The objective function of (PR_λ) may also be written as

$$Z_\lambda = \sum_{j\in J} \lambda_j - \text{maximum } \left[\sum_{i\in I} \sum_{j\in J} (\lambda_j - c_{ij})\, x_{ij} \right].$$

With the objective function in this form, it is clear that (PR_λ) separates into a collection of "bounded interval" binary knapsack problems, one for each $i\in I$. Each of these knapsack problems has one constraint of the form (2). Such constraints differ from the usual knapsack constraint in that the summation is bounded both from below and from above. Thus, the efficiency of the bounding procedure rests in part on the fact that bounded interval binary knapsack problems can be solved very efficiently [3, 13].

As in [9], each λ_j in the objective function of (PR_λ) is set equal to c_{2j}, the second smallest value of c_{ij} for all $i\epsilon I$. These λ_j are, in turn, optimal dual multipliers for the trivial bounded variable linear program:

$$(PR_L) \quad \text{minimize} \quad Z_L = \sum_{i\epsilon I} \sum_{j\epsilon J} c_{ij}x_{ij}$$

$$\text{subject to:} \quad \sum_{j\epsilon J} x_{ij} = 1 \text{ for every } j\epsilon J$$

$$0 \leq x_{ij} \leq 1 \text{ for every } i\epsilon I,\ j\epsilon J.$$

Other values for the λ_j could be used, but the ones just described have the advantage of being easy to compute.

Thus, determining a lower bound requires two steps. First, solve (PR_L), then solve (PR_λ). If, however, the primal solution $\bar{X} = (\bar{x}_{ij})$ to (PR_L), which satisfies (1), should also satisfy (2), then $Z = Z_L = Z_\lambda$, and (PR_λ) need not be solved. Ordinarily, $\bar{X}$ will not satisfy (2), and (PR_λ) must be solved to obtain Z_λ.

In addition to providing a lower bound, the solutions to (PR_L) and (PR_λ) are used in selecting a branching (or separation) variable for defining subsequent candidate problems (CP). As noted above, the solution to (PR_L), $\bar{X}$, is usually not feasible to (2). In essence, the solution to (PR_λ), $\bar{\bar{X}} = (\bar{\bar{x}}_{ij})$, may be interpreted as recommending changes in $\bar{X}$ which must be made in order to satisfy (2). That is, it is possible that for some $j\epsilon J$, $\sum_{i\epsilon I} \bar{\bar{x}}_{ij} = 0$ to avoid overloading any agent or $\sum_{i\epsilon I} \bar{\bar{x}}_{ij} > 1$ to insure every agent uses a minimum amount of his resource. Those variables $\bar{\bar{x}}_{ij}$ with an optimal value of one indicate agent-task pairings that should be made whereas those $\bar{\bar{x}}_{ij}$ with an optimal value of zero indicate pairings that should be avoided. Thus, these variable values indicate changes that will reduce the aggregate infeasibility of $\bar{X}$ in (2), and they are helpful in choosing a branching variable.

To formalize the concept of reducing aggregate infeasibility, we define the infeasibility in constraint i prior to taking a branch to be

$$D_i = \max \{0, d_i^+, d_i^-\}$$

where $d_i^+ = \sum_{j\in J} r_{ij}\bar{x}_{ij} - b_i$,

$$d_i^- = a_i - \sum_{j\in J} r_{ij}\bar{x}_{ij},$$

The set $I^+ \equiv \{i\in I \mid d_i^+ > 0\}$ identifies those constraints (2) for which $\bar{X}$ exceeds the upper bound, and $I^- \equiv \{i\in I \mid d_i^- > 0\}$ identifies those constraints (2) for which $\bar{X}$ fails to satisfy the lower bound.

Suppose $I^+ \neq \emptyset$ and $k\in\{j\in J \mid \bar{x}_{ij} = 1$ and $i\in I^+\}$; if x_{ik} is set to 0 then d_i^+ and d_i^- become:

$$d_i^+ = \sum_{j\in J} r_{ij}\bar{x}_{ij} - b_i - r_{ik}$$

$$d_i^- = a_i - \sum_{j\in J} r_{ij}\bar{x}_{ij} + r_{ik}$$

and the resulting infeasibility in constraint i becomes

$$D_i^k = \max \{0, d_i^+, d_i^-\}.$$

Assuming that task k is reassigned to the second least costly agent, (say agent h, where $c_{hk} = \min_{l\neq i} c_{lk}$) then the infeasibility in constraint h becomes

$$D_h^k = \max \{0, d_h^+, d_h^-\}$$

where

$$d_h^+ = \sum_{j\in J} r_{hj}\bar{x}_{hj} - b_h + r_{hk}$$

$$d_h^- = a_h - \sum_{j\in J} r_{hj}\bar{x}_{hj} - r_{hk}.$$

Hence the net difference in total infeasibility is:

$$\Delta D_i^k = (D_i + D_h) - (D_i^k + D_h^k)$$

If $\Delta D_i^k > 0$ then setting $x_{ik} = 0$ yields a reduction in aggregate infeasibility, and if $\Delta D_i^k \leq 0$ then such a branch will not reduce aggregate infeasibility.

Similarly, suppose that $I^- \neq \emptyset$ and $k \in \{j \in J \mid \bar{x}_{ij} = 0 \text{ and } i \in I^-\}$; if x_{ik} were set to 1 then d_i^+ and d_i^- become:

$$d_i^+ = \sum_{j \in J} r_{ij} \bar{x}_{ij} - b_i + r_{ik}$$

$$d_i^- = a_i - \sum_{j \in J} r_{ij} \bar{x}_{ij} - r_{ik}$$

and the resulting infeasibility in constraint i would be

$$D_i^k = \max \{0, d_i^+, d_i^-\}.$$

If x_{ik} is set to 1 then task k is assigned to agent i and agent i_k relinquishes it, where $i_k = \min_i c_{ik}$. Hence the infeasibility for constraint i_k becomes

$$D_{i_k}^k = \max \{0, d_{i_k}^+, d_{i_k}^-\}$$

where

$$d_{i_k}^+ = \sum_{j \in J} r_{i_k j} \bar{x}_{i_k j} - b_{i_k} - r_{i_k k}$$

$$d_{i_k}^- = a_{i_k} - \sum_{j \in J} r_{i_k j} \bar{x}_{i_k j} + r_{i_k k} .$$

The net difference in infeasibility is

$$\Delta D_i^k = (D_i + D_{i_k}) - (D_i^k + D_{i_k}^k)$$

where D_i and D_{i_k} are the infeasibilities in constraints i and i_k prior to any branch. As before, if $\Delta D_i^k > 0$ then there is reduced infeasibility following a branch on variable x_{ik}.

Several rules for selecting the branching variable, x_{i*j*}, are formulated as follows:

I. a) x_{i*j*} is that variable for which

$$\Delta D_{i*}^{j*} = \max_{(i,\, j)\ \varepsilon\ H^+ \cup H^-} \{\Delta D_i^j\}$$

where $H^+ \equiv \{(i, j) \mid \bar{\bar{x}}_{ij} = 0 \text{ and } i \varepsilon I^+\}$

$H^- \equiv \{(i, j) \mid \bar{\bar{x}}_{ij} = 1 \text{ and } i \varepsilon I^-\}$

b) If $\Delta D_{i*}^{j*} = 0$ in a) then x_{i*j*} is that variable for which

$$\Delta D_{i*}^{j*} = \max_{(i,\, j)\ \varepsilon\ (G^+ - H^+)\ \cup\ (G^- - H^-)} \{\Delta D_i^j\}$$

where $G^+ \equiv \{(i, j) \mid \bar{x}_{ij} = 1 \text{ and } i \varepsilon I^+\}$

$G^- \equiv \{(i, j) \mid \bar{x}_{ij} = 0 \text{ and } i \varepsilon I^-\}$.

II. a) x_{i*j*} is that variable for which

$$\rho_{i*j*} = \min \left\{ \min_{(i,\, j) \varepsilon G^+} \left\{ \frac{\lambda_j - c_{ij}}{\Delta D_i^j} \right\} , \min_{(i,\, j) \varepsilon G^-} \left\{ \frac{c_{ij} - c_{ikj}}{\Delta D_i^j} \right\} \right\}$$

b) If $\Delta D_i^j = 0$ for all $(i, j) \varepsilon G^+ \cup G^-$ then x_{i*j*} is that variable for which

$$\rho_{i*j*} = \max_{(i,\, j) \varepsilon E^+} \left\{ \frac{\lambda_j - c_{ij}}{[r_{ij} / (b_i - \sum_{J \varepsilon F_i} r_{ij})]} \right\}$$

where $E^+ \equiv \{(i, j) \mid \bar{\bar{x}}_{ij} = 1 \text{ and } i \varepsilon I^+\}$,

F_i denotes the set of tasks assigned to agent i by prior branching.

Rules Ia and Ib are designed to choose that variable which reduces the post branch aggregate infeasibility by the greatest amount. Rule IIa conditions the choice of branching variable on the additional cost incurred per unit reduction in infeasibility. Rule IIb is the one used in [9]; the variable chosen by this rule repre-

sents an agent-task pairing which should be made considering the penalty for not doing so weighted by the fraction of the agent's remaining free resources consumed by the assignment.

As the algorithm progresses and new candidate problems (CP) are defined by the branching process, the additional steps given below may be taken to facilitate fathoming. These steps are specialized adaptations of more general forcing (or variable fixing) tests suggested by Balas [2] and Glover [7].

In solving any (CP), any $x_{i'j'}$ for which $r_{i'j'} > b_{i'} - \sum_{j\in F_{i'}} x_{i'j}r_{i'j}$ may be set equal to zero. Here $F_{i'}$ denotes those $j\in J$ for which $x_{i'j}$ has been assigned a value of zero or one by prior branching or variable fixing tests. Similarly, if there is an $x_{i'j'}$ for which $a_{i'} - \sum_{j\in F_{i'}} x_{i'j}r_{i'j} > \sum_{j\in J-F_{i'}} r_{i'j}-r_{i'j'}$, then $x_{i'j'}$ must be set equal to one in the solution to (CP). These variable forcing tests may subsequently result in other variables being forced to zero or to one when all of the resultant implications are considered. Moreover, forcing certain variables to zero or to one in the solution to (CP) may affect the values of some of the λ_j obtained from solving (CPR_L). This change may, in turn, increase the value of the lower bound provided by (CPR_λ).

Another test may be used to check the feasibility of (P) (or any candidate subproblem). Summing the constraints (2) together yields the constraint (5):

$$A = \sum_{i\in I} a_i \leq \sum_{i\in I}\sum_{j\in J} r_{ij}x_{ij} \leq \sum_{i\in I} b_i = B \qquad (5)$$

This new constraint, together with constraints (1), implies that for any feasible soltuion to (P), we must have:

$$\sum_{j\in J} r'_j \geq A \text{ and } \sum_{j\in J} r''_j \leq B \qquad (6)$$

where

$$r'_j \equiv \max_{i\in I} \{r_{ij}\} \text{ and } r''_j \equiv \min_{i\in I} \{r_{ij}\}.$$

The values necessary for the tests (6) can be updated easily as part of the branching process in order to apply this test to each (CP).

The algorithm terminates in the usual way when all candidate problems have been fathomed.

4.0 Conclusion

This note has described an efficient branch and bound algorithm for the bounded interval generalized assignment problem. The algorithm serves as a useful tool for solving a large number of applications of this assignment model, a representative sample of which are mentioned in the introduction.

References

1. V. Balachandran, "An Integer Generalized Transportation Model for Optimal Job Assignment in Computer Networks" Working Paper 34-72-3, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pa. (November, 1972).

2. E. Balas, "An Additive Algorithm for Solving Linear Programs with Zero-one Variables," Operations Research, 13 (1965), 517-545.

3. R.S. Barr and G.T. Ross, "A Linked List Data Structure for a Binary Knapsack Algorithm," Research Report CSS-232, Center for Cybernetic Studies, University of Texas, Austin, Texas (August, 1975).

4. A. DeMaio and C. Roveda, "An All Zero-One Algorithm for a Class of Transportation Problems," Operations Research, 19 (1971) 1406-1418.

5. L.E. Freund and G.E. Staats, "Model for Optimally Assigning Nursing Personnel Based on Difficulty," Working Paper, Department of Industrial Engineering, University of Missouri, Columbia, Missouri (September, 1974).

6. A.M. Geoffrion, "Lagrangean Relaxation for Integer Programming," Mathematical Programming Study 2, (1974) 82-114.

7. F. Glover, "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," Operations Research, 13 (1965), 879-919.

8. D. Gross and C.E. Pinkus, "Optimal Allocation of Ships to Yards for Regular Overhauls," Technical Memorandum 63095, Institute for Management Science and Engineering, George Washington University, Washington, D.C. (May, 1972).

9. G.T. Ross and R.M. Soland, "A Branch and Bound Algorithm for the Generalized Assignment Problem," Mathematical Programming 8 (1975), 91-103.

10. G.T. Ross and R.M. Soland, "Modeling Facility Location Problems as Generalized Assignment Problems," Research Report CCS-254, Center for Cybernetic Studies, University of Texas, Austin, Texas (February, 1976).

11. V. Srinivasan and G.L. Thompson, "An Algorithm for Assigning Uses to Sources in a Special Class of Transportation Problems," Operations Research, 21 (1973), 284-295.

12. A. Zoltners, "A Direct Descent Binary Knapsack Algorithm" Working Paper 75-31, School of Business Administration, University of Massachusetts, (September, 1975).

13. A. Zoltners, "The Audit Staff Assignment Problem" Working Paper 74-34, School of Business Administration, University of Massachusetts, (September, 1974).

14. A. Zoltners, "A Unified Approach to Sales Territory Alignment," Working Paper, School of Business Administration, University of Massachusetts, (July, 1976).

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Center for Cybernetic Studies
The University of Texas

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

(9) Research + rept.

3. REPORT TITLE

(6) A Note on the Bounded Interval Generalized Assignment Problem.

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

(10)

(14) CCS-253

5. AUTHOR(S) (First name, middle initial, last name)

G. Terry Ross
Richard M. Soland
Andris A. Zoltners

(15) N00014-75-C-0616
N00014-75-C-0569

6. REPORT DATE

(11) July 1976

7a. TOTAL NO. OF PAGES

12

7b. NO. OF REFS

14

8a. CONTRACT OR GRANT NO.

N00014-75-C-0616;0569

b. PROJECT NO.

NR047-021

c.

d.

(12) 14 p.

9a. ORIGINATOR'S REPORT NUMBER(S)

Center for Cybernetic Studies
Research Report CCS 253

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

(16) NR-047-021

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Office of Naval Research (Code 434)
Washington, D.C.

13. ABSTRACT

The bounded interval generalized assignment problem is exemplified by the problem of assigning tasks to agents so that each task is assigned to exactly one agent and the time required to complete the set of tasks assigned to any one agent falls between prespecified lower and upper bounds. This note describes an efficient algorithm for solving this problem.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Unclassified
Security Classification

A-31408

406194

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Assignment Problems						
Loading Problems						
0-1 Programming						