© 1978

HENRY CRAWFORD THIBAULT, II

ALL RIGHTS RESERVED

HEURISTIC SOLUTION METHODS FOR MULTI-RESOURCE GENERALIZED ASSIGNMENT PROBLEMS

By

HENRY CRAWFORD THIBAULT, II

Bachelor of Science in Business Administration
University of Arkansas
Fayetteville, Arkansas
1975

Master of Science in Operations Research
University of Arkansas
Fayetteville, Arkansas
1976

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF PHILOSOPHY
December, 1978

OKLAHOMA STATE
UNIVERSITY
LIBRARY

HEURISTIC SOLUTION METHODS FOR MULTI-RESOURCE

GENERALIZED ASSIGNMENT PROBLEMS

Thesis Approved:

Kenneth E. Case
Thesis Adviser

Joe H. Mize

James E. Shamblin

Donald W. Grace

Wayne Turner

Philip M. Wolfe

Norman N. Durham
Dean of the Graduate College

1032399

ACKNOWLEDGMENTS

This dissertation could not have been written without the contributions of a whole series of people. The most direct contributions were made by my wife, Nina, and by Kenneth E. Case, the chairman of my advisory committee.

Nina took in sewing in a cramped apartment where the temperature was twenty degrees warmer than any she had ever experienced. In spite of her unfamiliarity with American bureaucrats, banks, and automobile mechanics, she ran nearly all the errands that otherwise would have stolen a great deal of time from learning and research. Her imagination and creativity with limited resources made our apartments cozy places to work. Unlike most of her contemporaries, she had to leave her homeland and family six years ago, knowing that we could not afford even a short visit during the time I was a student. She expressed her love and confidence in many other ways that encouraged me when I needed it most.

Professor Case and I arrived here at the same time and were assigned to find a way to assign artillery units to enemy targets. He was the best boss I ever had, and I could not have had a better adviser. He went out of his way to make my situation more enjoyable and financially rewarding. He minimized his own considerable contributions to our efforts while making sure of recognition for me. He had some of the key ideas in developing the methods discussed in this dissertation. Meeting him was one of the most fortunate things that ever happened to me.

I asked the other professors to serve on my advisory committee because they are superb teachers and researchers. Each of them has been a model to me in some way, and each suggested valuable avenues of research to pursue. I am especially grateful to them for making it so easy to arrange meetings, even on short notice.

My parents gave me a home where love of learning was taken for granted. They kept believing in me even when I no longer did so.

Mr. and Mrs. Roy Wilson, who were my high school math teacher and first grade teacher, along with Mr. B. L. Garner, my high school science teacher, gave me basic skills and research insights that I have used every day thereafter.

Mrs. Linda Howard took the hideous job of typing this dissertation at the last minute, worked around my disorganized schedule, and used her experience and wisdom to make a cohesive whole out of a pile of messy papers.

To all of these and to many others who have helped me along the way--thank you. I cannot repay you for what you did for me, but I will try to do it for others.

This work was supported in part by the AFSC Optimal Systems Design program, contracts F08635-76-C-0003 and F08635-76-C-0309. These contracts were overseen by the Joint Munitions Effectiveness Manual/Surface to Surface Methodology and Evaluations Working Group.

TABLE OF CONTENTS

Chapter | Page

Chapter	Page

Chapter Page

LIST OF TABLES

Table		Page

LIST OF FIGURES

CHAPTER I

PROBLEM SUMMARY AND RESEARCH OBJECTIVES

Introduction

The objective of this dissertation is to develop and evaluate heuristic solution methods for multi-resource generalized assignment models, including some variations and complications. These problems belong to a class for which efficient optimal solutions probably cannot be developed. Without using references or symbols, this chapter summarizes the problem, justifies and develops the approaches and objectives of the research, and reports briefly the results that have been obtained and the contributions that have been made.

Problem Summary

General

All assignment models are similar in seeking the best assignments of a set of "agents" to a set of "tasks." Typical applications are assigning machines or workers to jobs, factories to production orders, merchandise types to warehouse spaces, deployment of medical resources in catastrophic situations ("triage"), and many others. For example, the original motivation for this research was assignment of artillery to military targets.

Previous Approaches

In the "classical" assignment problem, the number of agents and taks is, perhaps after a simple augmentation, the same. Each agent is assigned to exactly one task as some objective function is optimized. The "generalized" assignment model allows the assignment of several tasks (or none at all) to each agent, so long as the tasks do not exceed the agent's capacity of some resource.

Multi-Resource Models

The primary concern of this research is the extension of generalized assignment models to consider several resources for each agent. The need for this is illustrated by an example in Chapter II, where an elegant solution of a single-resource model is (invalidly) used for a multi-resource problem.

Complications and Reality

Secondary consideration is given to some of the complications that arise in actual problem situations. These include variations on the model, such as:

Scheduling the execution of the assignments, including prior restrictions on the schedule.

Incorporating discretionary resources for some agents; that is, a decision must be made as to which category of a given type of resource would be used for a given task.

Allowing mixed assignments, in which agents can share some tasks.

Task distribution leveling, an attempt to avoid solutions where very efficient agents may be overloaded, while others are idle or nearly so, even though no constraints are violated.

Combinations of some or all of the above variations.

Other complications arise in the problem-solving environment:

Limited computer resources may be all that are available.

Conversational response times are often required.

Simplicity of use is very important.

Summary and Justification of Solution Approaches

Optimal Approaches

There is probably no hope of obtaining a nonenumerative optimal solution to a multi-resource problem of realistic size, where the number of agents times the number of tasks may be well over a thousand. Branch-and-bound logic has always been the most efficient enumerative way to attack this kind of problem. For some single-resource problems this has been fairly satisfactory, approaching conversational speed, but the problems lacked variations. Also, the fastest results were associated with problems where the number of agents was very small compared to the number of tasks. This combined in fortunate coincidence with the single-resource characteristic to allow especially rapid solution. Current computer technology will probably not allow optimal solution of multi-resource problems in conversational time, especially if complications are present.

Heuristic Approaches

This dissertation describes and evaluates heuristic solution methods. Certain characteristics of multi-resource problems bear on the development of these methods.

Unlike classical assignment or transportation problems, these problems cannot readily be checked for possession of a feasible solution (i.e., one covering all tasks). It is probably just as difficult to devise a procedure that can always detect a feasible solution (if one exists) as it is to develop an optimal algorithm. For this reason, the best any heuristic procedure can do is to frequently find an excellent feasible solution. Also, the only way of testing any solution for optimality consitutes an optimal solution method for the entire problem. Further, without re-solving the problem from the beginning, it is a matter of guesswork to determine how resource limitations should be changed in order to improve a solution or perform sensitivity analysis.

Justifying the Heuristic Approach

Why, then, is it desirable to develop heuristic approaches at all? This is answered by examining justifications for use of heuristic methods (1) in general, and (2) with this class of problems.

Solution Time and Its Variability

One justification has already been mentioned: solution time. Up to this point, however, only the duration itself was emphasized, and not its variability. In management planning or systems design, it is helpful to be able to predict response time. Heuristic methods can frequently be designed to require a fixed (or bounded) amount of time (thus enabling the use of worst-case analysis), but a branch-and-bound algorithm's time usage can vary through a vast range. This variability also applies to storage requirements.

General Usefulness

Heuristics are useful in spite of the aforementioned difficulty in finding a feasible solution. The fact is that in actual practice, many feasible solutions usually exist, so a good one will be obtained by a well-designed heuristic. Management will usually be willing to allot additional resources or reduce the number of tasks if a normally reliable method has failed to find a feasible solution. Sometimes it is sufficient to minimize the number of unassigned tasks, as in triage. Also, several different heuristics can be used on a problem. Perhaps one will find an answer where others do not.

Multiple Alternatives

Heuristics can be designed to provide several attractive solutions, from which the most suitable can be chosen according to secondary objective requirements that may be impossible to codify. This is not true of most optimal procedures.

Inexact Data

Data are almost always so inexact that a good approximate solution cannot be called inferior to a solution obtained by optimal methods. Also, the difference between optimal and approximate objective values will often be less than the incremental cost of the optimal solution.

Flexibility

Heuristics are typically far more adaptable to changing requirements than are optimal methods. The former are not required to be as precisely formulated (in a mathematical sense) as are the latter. Indeed, as will be seen, some of the more successful methods developed by this research descend directly from heuristics developed for quite different problems. Although branch-and-bound methods are relatively easy to adapt compared to other optimal methods, they do not approach the flexibility of heuristic methods.

Improving Optimal Methods

Branch-and-bound methods themselves provide two other justifications for heuristic solution methods. A very good bound on the optimal solution can be obtained, thus enabling early elimination of large numbers of nodes. Also, the branching process uses heuristic rules to find promising branches.

Choosing Heuristic Approaches

Whatever the justification for use of heuristic methods, it must eventually be decided which of the literally infinite number of possible approaches to take. This is, of course, determined to some extent by the design objectives and performance standards that will be specified. One cannot, however, escape the fact that designing a heuristic is an intuitive process in which inspiration comes from experience and investigation of the work of others.

Construction Heuristics

The first heuristic approaches that will be described here are those that construct a solution. Most of them attempt to progressively augment a partial solution by adding an especially attractive agent-task combination. This process is guided by some intuitively developed intermediate logic that seeks a better solution than would be achieved by simply assigning successive tasks to the cheapest available agent. The intermediate logic is where experimentation has been done. The approaches of this research include:

Random intermediate logic, where many complete solutions are generated at random.

Penalty methods, quite similar to Vogel's approximation method.

"LP-guided" methods, where successive assignments are based on variable values in a linear programming solution.

Improvement Heuristics

Additionally, ways have been developed to improve an existing solution. Two strategies try to obtain a savings by switching the assignment of two tasks to different agents:

"Greedy" methods, which make the first profitable switch found.

CRAFT-type methods, motivated by the well-known layout procedure, which make the most profitable switch found after examining all possibilities.

Objectives

Development

Specific design objectives come from analysis in which desirable performance characteristics are determined by (a) the requirements and limitations derived from the operating environment, and (b) cost-effectiveness versus other possible approaches.

Requirements and Limitations

The most important requirements involve:

The problem definition in terms of size and complexity. The size of a realistic problem (in tasks times agents) can vary from about ten to thousands. Many applications deal with multiple resources, and complicating variations may be present.

Response time, measured in real elapsed time. This requirement may vary considerably. It might be a few minutes in emergency or wartime situations, or "on-the-spot" in a factory. An hour or two would satisfy most managers. Problems involving large, long-term investments could justify much slower response, if a solution could be sufficiently improved or shown to be nearly optimal. This leads to the next type of requirement.

Accuracy, in terms of nearness to the optimum solution (if one exists and can be found, or if a reasonable set of bounds can be determined). As has been mentioned, problem data are usually inaccurate. However, for the previously mentioned investment situation, or for a procedure that will be used many times, there may be reason to strive for high accuracy. Very good data will be needed, though, if the added effort is to be cost-effective.

Feasibility, or coverage of all tasks. This can be the most important requirement. As has been noted, however, there is probably no way to be sure of finding a feasible solution, and it is equally difficult to determine what should be done to introduce feasibility. Since feasibility is so important, it is necessary to detect when (a) it is certain that no feasible solution exists, and (b) it is probable that none exists. Heuristic rules for slack analysis can help guide the relaxation of constraints.

Limitations, besides those noted in conjunction with data accuracy, arise from the resources available for implementation:

Personnel resources are limiting in that any solution method is more useful if it is as simple as possible to implement, maintain, use, and modify.

Computer resources can be limited in speed, storage, peripheral devices, and programming languages. Many of the methods described are compatible with some of the smallest microcomputers.

Cost-Effectiveness

Note that the requirements of response time, accuracy, and feasibility bear directly on cost-effectiveness. There must, however, be some basis for comparison. What would a user do if this research had not been undertaken? The incremental improvement would have to be measured against the incremental cost.

No reasonable alternative is known to be available. It is estimated that the best optimal branch-and-bound algorithm that could be developed for a typical multi-resource problem with two resources, 15 agents, and 100 tasks, with no complicating variations, would have a response time of about thirty minutes and would require about a million bits of storage, using existing computer technology. The storage requirement is reasonable only for fairly large computers, and the response time would be suitable for only a few applications.

Based on the above paragraphs and earlier discussion, three points can be made about the cost-effectiveness of this research:

(1) There is apparently no other way to obtain a solution quickly enough. This means that the limiting value of the method is the value of the solution, for which users are willing to bear development costs of five to seven digits.

(2) The incremental cost of a single heuristic problem solution is at most a few dollars.

(3) Refinements to approach optimality should be made only if the improvement is of greater value than the cost of the refinement. No refinement is justified that produces a solution closer to the optimum than the amount of error in the data, which is usually very difficult to measure.

Objectives

The objectives given below are based on the requirements and limitations encountered in the assignment of air and artillery units to military targets. This is the application where the most taxing requirements occur ("worst case" philosophy), and actual problems are available. Many complications are present, response times on the order of five minutes are desired, multiple daily use places some premium on accuracy (although data are often estimated), problems are frequently so highly constrained that feasibility is the most important consideration, and personnel will usually be familiar only with input-output characteristics. The computer, for which specifications are currently sketchy, will use fairly recent technology. Total storage will probably be limited to 500,000 bits. (As was noted, methods suitable for microcomputers are also included).

The precise objectives of this research can now be stated: To devise heuristic solution methods substantially fulfilling the aspiration levels given below for realistic multi-resource generalized assignment problems. A realistic problem is defined as one whose size (tasks times agents) is on the order of ten to a thousand, possibly including one or more variations. Primary emphasis is placed on the multiple-resource model without variations. This model contains the features believed to be common to most applications, thus warranting the most thorough investigation. Variations may or may not apply to specific problems. Those that apply may be present in widely varying forms and severities. Therefore, procedures for handling variations are demonstrated to the extent that they have been identified in actual problems and dealt with to the user's satisfaction. It is emphasized that procedures for solving the basic multiple-resource problem have been planned for adaptability to variations encountered in practice. Suggestions are made for dealing with the variations.

Aspiration Levels

The first category of secondary objectives is evaluation of the methods that have been developed according to the following aspiration levels and qualitative criteria:

Coverage (feasibility): A single aspiration level cannot be set. For problems appearing to be fairly loosely constrained, it is not unreasonable to hope that solutions covering all tasks would be found in at least 90 percent of the cases tested (some of which, despite appearances, probably do not possess feasible solutions). The deterioration of this performance becomes more severe as constraints tighten, since more problems are probably actually infeasible.

Response time: A reliable response time on the order of five minutes is the aspiration level.

Accuracy/Optimality: The aspiration level for this factor is to produce a solution within 15 percent of the optimum in 90 percent of the cases where a feasible solution is found and the optimum is known or can be adequately bounded.

Computer Storage: The aspiration level is to use an amount of storage (bits) that does not exceed 300 times the product of the numbers of resources, agents, and tasks.

Other: Qualitative evaluation criteria include:

(a) Adaptability to introduction of variations, which is necessary for any method to be of general applicability.

(b) Availability of multiple solution alternatives subject to virtually instant access, which would be highly desirable in order to better satisfy additional secondary or transient objective criteria.

(c) Ease of implementation, operation, and maintenance, which would be critical to actual usefulness.

(d) Predictability of response time.

Evaluation Techniques

Another category of secondary objectives is to determine whether the above criteria have been satisfied. It is not intended to evade the usual research technique of evaluating an approximation by comparing it to the value being approximated, but the ill-conditioned nature of this class of problems makes it impractical to obtain exact information about optimality and feasibility. Therefore, the following techniques are used to overcome these difficulties:

Special heuristics enable probabilities to be calculated for obtaining a solution within a certain quantile of all solutions.

Continuous methods (linear programming) give additional information about existence and bounds of solutions.

Tests on smaller problems give some intuitive support while enabling more thorough use of special heuristics and continuous methods.

Summary of Results

Where measurements were possible, objectives were usually satisfied beyond the aspiration levels by one or more methods. This section summarizes the results for each category of objectives.

Realistic Problems: A method was developed that will be used by the U. S. Marine Corps in a conversational implementation to solve artillery problems containing every variation that has been described. It is described in Chapter V. Elsewhere in Chapter V, some ways are suggested for considering variations in basic methods, even when the methods are implemented on a microcomputer.

Coverage: A solution covering all tasks was always found unless known not to exist. If no solution existed, about 90 percent confidence could be associated with covering as many tasks as possible.

Response Time: Response times under five minutes could be guaranteed with the best methods on most computers.

Accuracy/Optimality: Ninety-four percent of the answers were within 15 percent of the optimum, under stricter conditions than aspired to. Results support very high confidence of obtaining a solution superior to all but a few other solutions.

Computer Storage: Depending on the output and user options desired, storage requirements were well within the aspiration level. Also, special methods for saving storage are discussed in Chapter V.

Evaluation Techniques: Basic methods, either modified or used in slightly different ways, gave most of the information needed.

Contributions

General

This research has made several contributions. Besides the solution methods themselves, these include problem definition, evaluation methodology, and realistic applications.

Heuristic Methods

Considerable effort and inspiration were necessary to combine methods used with other classes of problems. Powerful heuristics were produced by adapting such methods to the characteristics of multi-resource generalized assignment problems.

Problem Definition

Although these problems are frequently encountered, no discussion of their multi-resource aspect was found in the literature. Researchers have used algorithms that are "optimal" for <u>single-resource</u> problems. Such an approach is itself heuristic. This dissertation establishes the need to consider multiple resources explicitly.

Evaluation Methodology

It was necessary to develop most of the evaluation methodology. The literature is weak in describing evaluation methodology for heuristics in general. Therefore, this dissertation may well serve as one of the more comprehensive sources of ideas for evaluating <u>any</u> heuristic.

Realistic Applications

Researchers confronted with actual problems will seldom find pre-existing solution methods that can be applied unchanged. This dissertation describes the adaptation of some of its heuristics to fit specific applicational requirements, thus serving as a possible source of inspiration.

CHAPTER II

THE PROBLEM IN LITERATURE AND PRACTICE

Introduction

The classical assignment model occurs in almost every textbook (see [17, 31 and 33]). Agents and tasks are interchangeable because of the assumption that each agent has enough resources for exactly one task. Ross and Soland [26] point out that a model would be more useful if it allowed the assignment of several tasks to a single agent, so long as these tasks do not use more of some resources than the agent has available. However, they and others [3, 4, 9 and 29] did not go beyond one resource. This chapter presents mathematical models and discusses applications, beginning with the single-resource problem, but primary emphasis is placed on the extension to multiple resources, with additional discussion of problems with variations.

Single-Resource Problems

Models

Figure 1 is a model of the single-resource problem. It was adapted from Ross and Soland [26], to whom the terms "agent," "task," and "generalized assignment problem" are also due. A similar model is given by Balachandran [3, 4].

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1-1}$$

Subject to:

$$\sum_{j=1}^{n} a_{ij} x_{ij} \leq b_i \qquad i=(1,2,\ldots,m) \tag{1-2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j=(1,2,\ldots,n) \tag{1-3}$$

$$x_{ij} = 0 \text{ or } 1 \tag{1-4}$$

where

m = number of agents

n = number of tasks

c_{ij} = cost incurred if agent i is assigned to task j

a_{ij} = resource required by agent i to do task j

b_i = amount of resource available to agent i

x_{ij} = 1 if agent i is assigned to task j

x_{ij} = 0 if otherwise

Figure 1. Mathematical Model of Single-Resource Generalized Assignment Problem

Figure 1 reduces to the classical assignment model if we let $a_{ij} = b_i = 1$. De Maio and Roveda [9] and Srinivasan and Thompson [29] discuss the special case that can be interpreted as a generalized transportation model where each destination must be supplied from a single source. This can be represented by allowing a_{ij} to be a_j in Figure 1.

Applications

Many specific applications have been cited, especially in Ross and Soland [26]. They include assignment of software development tasks to programmers, assignment of jobs in computer networks (Ross and Soland [26] cite a working paper for Balachandran [3], assignment of contractual payments or television commercials to time periods, along with fixed charge plant location models (Ross and Soland [26] cite Geoffrion [13] and Gross and Pinkus [16] here) where each customer must be supplied by one plant, and communication network design models with node capacity constraints (Ross and Soland [26] cite Grigoriadis et al. [15]).

Multi-Resource Problems

Justification

Actually, many of the applications cited above may be multi-resource situations that have been simplified in order to make them analytically tractable. For example, Balachandran [3, 4], in discussing the assignment of jobs to computers in a network, states that each job requires resources such as CPU time, memory, software, or peripherals. Later, the problem is simplified dramatically by associating an infinite cost with combinations for which the job's requirements for one or more resources exceed the total capacity of the computer. The only constrained resource is "processing time," giving a model like Figure 1. It is not clear whether "processing time" is CPU time or elapsed time, but the multi-programming capabilities of the computers involved appear to invalidate the single-resource model in either case. This example shows why it is often necessary to consider multiple resources in generalized assignment problems. All of the models discussed below would require modification to adequately describe Balachandran's problem (which could probably be said of most applications), but the need for investigation of multi-resource problems seems well-established.

The Basic Multi-Resource Model

Figure 2 was derived from a model developed during preliminary research dealing with assignment of artillery units to engage enemy targets [6]. (Note that Figure 2 can be reduced to Figure 1 by letting the number of resources (p) be one.) In the artillery problem, two resources are involved: ammunition and time. The computer network [3, 4] problem dealt with resources of five types, most of which should have been considered explicitly, although software can be handled with Balachandran's infinite-cost approach. This technique has been used elsewhere [6, 26], and is mentioned in standard texts [17, 31, 33].

The Unconstrained Optimum

Definition

If the resource constraints (1-2) and (2-2) are disregarded in Figures 1 and 2, an optimal solution becomes readily available by

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2-1}$$

Subject to:

$$\sum_{j=1}^{n} a_{ijk} x_{ij} \leq b_{ik} \qquad i=(1,2,...,m); \quad k=(1,2,...,p) \tag{2-2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j=(1,2,...,n) \tag{2-3}$$

$$x_{ij} = 0 \text{ or } 1 \tag{2-4}$$

where

p = number of resources, indexed by k

a_{ijk} = amount of resource k required by agent i to do task j

b_{ik} = amount of resource k available to agent i

(Other notation is identical to that in Figure 1)

Figure 2. Mathematical Model of Multiple-Resource Generalized Assignment Problem

simply assigning each task to the cheapest agent. Such a solution, which has also been called the "trivial solution" [26], will be referred to in this dissertation as the "unconstrained optimum." Strictly speaking, of course, the problem has become "unconstrained" only in terms of resources. The other restrictions remain because these could otherwise no longer be called "assignment problems."

Complicated Multi-Resource Problem

Introduction

Despite the extended generality of the basic multi-resource model in Figure 2, it would need to be modified for most applications. Although it is neither possible nor practical to construct a model that will be of complete generality, it seems to be a worthwhile example to expand the basic model to cover several variations, especially since such an application has been identified.

The expanded mathematical model, however, is quite complex, which limits its usefulness. Therefore, this section begins with a Model Summary, followed by the model itself and a discussion of its components.

Model Summary

The meaning of each expression in the model is given below:

(3-1) (Objective function) Minimize a weighted combination of:
(a) total cost
(b) disparity in task distribution
(c) deviation from desired mixed assignments.

(3-2a) (Ammunition constraints) No unit may use more of a particular type of ammunition than is available.

(3-2b) (Time constraints) Units may not exceed the specified amount of time available.

(3-3a) (Binary coverage constraints) Targets for which mixed assignment is not desired must have exactly one unit assigned to cover them completely.

(3-3b) (Mixed coverage constraints) Units in a mixed assignment must provide aggregate coverage that is sufficient for the target.

(3-4a) (Mixed assignment restrictions) For a unit firing a given type of ammunition at a given target in a mixed assignment:
- (a) Each gun in the unit must fire at least one shell.
- (b) The unit's fractional coverage of the target is equal to the number of shells fired divided by the number the unit would need to fire to cover the whole target.
- (c) A record must be kept of the particular combination of unit, target, and ammunition type.

(3-4b) (Binary assignment restrictions) In "unmixed" assignments, a unit either covers all of a target or none of it.

The Model

Figure 3 includes the variations for the most complicated version of the artillery problem, in which the agents are "units" and the tasks are "targets." The notation is given in Figure 4. Figure 3 does not include the scheduling variation, for which the additional constraints and notation are given in Figure 5.

Priority

The model does not consider target priority, which is handled by solving a subproblem (of the form given in Figure 3) for each priority class in decreasing order of importance. Each subproblem has access

$$\text{Minimize} \quad h_1 \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{p_i} c_{ijk}(x_{ijk} + y_{ijk}) + h_2(\max_i B_i - \min_i B_i)$$

$$+ h_3 \sum_{j=1}^{n} c'_j(M_j, M'_j, G_j, G'_j) \tag{3-1}$$

Subject to:

$$\sum_{j=1}^{n} a_{ijk}(x_{ijk} + y_{ijk}) \leq A_{ij} \qquad i=(1,2,\ldots,m);\ k=(1,2,\ldots,p_i) \tag{3-2a}$$

$$B_i = \sum_{j=1}^{n} \sum_{k-1}^{p_i} t_{ijk}(x_{ijk}, y_{ijk}) \leq T \qquad i=(1,2,\ldots,m) \tag{3-2b}$$

$$\sum_{i=1}^{m} \sum_{k=1}^{p_i} x_{ijk} = 1 \qquad j \in J_b \tag{3-3a}$$

$$\sum_{i=1}^{m} \sum_{k=1}^{p_i} y_{ijk} \geq 1 \qquad j \in J_m \tag{3-3b}$$

$$\left.\begin{array}{l} \left.\begin{array}{l} y_{ijk} = 0 \\ z_{ijk} = 0 \end{array}\right\} \text{ or } \left\{\begin{array}{l} y_{ijk} \geq q_i/a_{ijk} \\ z_{ijk} = 1 \end{array}\right. \\ x_{ijk} = 0 \\ a_{ijk}y_{ijk} \in (0, q_i, q_i+1), \ldots) \end{array}\right] \quad \left\{\begin{array}{l} i=(1,2,\ldots,m) \\ j \in J_m \\ k=(1,2,\ldots,p_i) \end{array}\right. \tag{3-4a}$$

$$\left.\begin{array}{l} x_{ijk} = 0 \text{ or } 1 \\ y_{ijk} = z_{ijk} = 0 \end{array}\right\} \quad \left\{\begin{array}{l} i=(1,2,\ldots,m) \\ j \in J_b \\ k=(1,2,\ldots,p_i) \end{array}\right. \tag{3-4b}$$

Figure 3. Mathematical Model of Artillery Problem

h Combining weight for objective function ($\Sigma h=1$; all $h \geq 0$).

m Number of friendly units (agents); indexed by i.

n Number of enemy targets (tasks); indexed by j.

p_i Number of ammunition types (discretionary resources) available to unit i; indexed by k.
(NOTE: k and p are used differently than in Figures 1 and 2.)

$\left.\begin{matrix} c_{ijk} \\ a_{ijk} \\ t_{ijk} \end{matrix}\right\}$ Cost, ammunition usage, and time needed if unit i engages target j using ammunition type k.
(NOTE: c and a are coefficients on the sum of x and y, but t is a function of x and y.)

x_{ijk} Binary assignment variable; =1 if unit i alone engages target j using ammunition type k; =0 otherwise, even if unit i participates in mixed engagement of target j.

y_{ijk} Mixed assignment variable; value is fraction of target j that unit i engages using ammunition type k.

B_i Total amount of time unit i is firing (actually, busy).

c'_j Cost due to deviation from mixed assignment specifications; a function of M_j, M'_j, G_j, and G'_j.

M_j Number of units requested for mixed assignment for target j.

M'_j Number of units actually assigned in mixed assignment to target j.

G_j Set of units requested for primary consideration for mixed assignment to target j.

G'_j Set of units actually assigned to target j.

A_{ik} Supply of kth ammunition type at unit i.

T Time horizon; must be in same units as t.

J_b Set of indices to tasks requiring binary assignment.

J_m Set of indices to tasks requiring mixed assignment.
(NOTE: $J_b \cup J_m = \{1,2,\ldots,n\}$; $J_b \cap J_m = \emptyset$).

q_i Number of guns located at unit i.

z_{ijk} Binary indicator variable; =1 if $y_{ijk} > 0$; $=0=y_{ijk}$ otherwise.

Figure 4. Notation for Mathematical Model of Artillery Problem

only to those resources not allocated in an earlier subproblem. This concept of absolute priority was the result of a user specification, but also occurs elsewhere, e.g., in the operating systems for IBM 360 and 370 computers. Other viewpoints exist, such as the "goal programming" approach of maximizing the number of assigned tasks as long as no tasks remain unassigned in a final solution if sufficient resources for them can be diverted from tasks of lower priority. The distinction between these two concepts of priority is rather fine--the first optimizes in groups; the second optimizes the entire problem (and would always achieve coverage at least as wide as the first). The second concept, besides being difficult to understand (which is regarded by Woolsey [34] as a fatal flaw), is computationally unwieldy and could prevent assignment of the most efficient units to the most important targets.

Objective Function

Figure 3 incorporates only one of many possible formulations for the four objective criteria:

(1) Coverage: Maximizing the number of targets covered.

(2) Cost Minimization: Maximizing target value requires only a simple transformation.

(3) Mixed Assignments: Minimizing overall deviations from the numbers and types of units specified.

(4) Task Distribution Leveling: Minimizing the maximum disparity between any two units in fraction of available time used.

Coverage is not reflected in Figure 3, because coverage can be made a consequence of cost minimization by adding to the problem a fictitious unit with unlimited resources. Any targets that could not be assigned elsewhere could be assigned to this unit. However, the associated cost would be so great that any solution actually covering n + 1 targets would be of lower cost than if n or fewer targets were covered. This approach is also used by Balachandran [3, 4]. The objective function has been formulated as a simple linear combination of the other three criteria. Balachandran [3, 4] justifies this by noting that (a) various theoretical appraoches [12, 24, 27] would not be economically feasible because of the computation time required, and (b) the linear combination is adequate if management can assign utilities for use as combining weights. Woolsey [34] describes a procedure for obtaining and refining such weights through interaction with the user. In summary, there is little evidence that a more elaborate formulation would better represent the largely intuitive decision standard that a user would employ.

It is quite possible that a heuristic will obtain an answer that will satisfy a model without operating explicitly on the model's specifications. This is true in the case of the decision variables of Figure 3, which the heuristic considers only indirectly. Also, the user has not yet decided on the final form of all objective criteria, which may portend changes in the final heuristic even though the model does not change.

Mixed Assignments and Discretionary Resources

The model in Figure 3 also contains nonbinary variables (y_{ijk}) to reflect the mixed assignment variation. For those targets defined by the user as requiring simultaneous engagement by more than one unit, y_{ijk} represents the "fraction" of target j that unit i will cover using ammunition type k (the use of different ammunition types is the discretionary resource variation). The restrictions of y_{ijk} and $a_{ijk}y_{ijk}$ to the sets of discrete values defined in Figure 3 (3-4a) are derived from a further requirement ("one-volley-minimum") that each participating unit must fire at least one round from each gun, with the total number of rounds fired by each unit being, of course, an integer. Note that the indicator variable z_{ijk} is a count of the number of units participating in a mixed assignment on target j.

Correspondence Between Models

To help understand the correspondence between models, Figures 1, 2 and 3 have had their components numbered according to equivalent function. For example, (1-1), (2-1), and (3-1) are the objective functions; (1-2), (2-2), (3-2a) and (3-2b) are resource constraints; (1-3), (2-3), (3-3a), and (3-3b) are complete coverage constraints.

Scheduling

The additional constraints and notation for the scheduling variation are given in Figure 5. The meaning of each constraint is given below:

(5-1) The duration of an assignment must be at least as great as the time required to execute it.

(5-2) An assignment to a target with a specified "start time" must be scheduled with an allowance for set-up time.

(5-3) A specified "end time" becomes the actual end time.

(5-4) If (a) only the "end time" or (b) only the "start time" is specified, the assignment must (a) start as late as possible, or (b) end as early as possible.

$$e_{ij} - s_{ij} \geq t_{ijk}(x_{ijk},y_{ijk}) \quad (5\text{-}1)$$

$$s_{ij} = S_j - u_i \qquad j \in J_s \quad (5\text{-}2)$$

$$e_{ij} = E_j \qquad j \in J_e \quad (5\text{-}3)$$

$$s_{ij} \leq E_j - t_{ijk}(x_{ijk},y_{ijk}) \qquad j \notin J_s;\ j \in J_e \quad (5\text{-}4a)$$

$$e_{ij} \geq S_j + t_{ijk}(x_{ijk},y_{ijk}) \qquad j \in J_s;\ j \notin J_e \quad (5\text{-}4b)$$

$$\left.\begin{array}{ll} s_{ij} \geq 0 & (5\text{-}5) \\ e_{ij} \leq T & (5\text{-}6) \\ D_{ij_1} \cap D_{ij_2} = \emptyset;\ j_1 \neq j_2 & (5\text{-}7) \end{array}\right\} j=(1,2,\ldots,n)$$

NOTE: In (5-1) through (5-7), $i=(1,2,\ldots,m)$

$$\left.\begin{array}{ll} s_{i_1j} - u_{i_1} = s_{i_2j} - u_{i_2} & (5\text{-}8) \\ e_{i_1j} = e_{i_2j} & (5\text{-}9) \end{array}\right\} j \in J_m;\ i_1 \neq i_2;\ z_{ij}=1$$

where

S_j = specified "start time" (first shell falls on target j).

E_j = specified "end time" (last shell falls on target j).

J_s = set of targets for which "start times" are specified.

J_e = set of targets for which end times are specified.

u_i = set-up time for unit i. (NOTE: t_{ijk} includes u_i.)

s_{ij} = scheduled time for unit i to begin setting up to fire on target j.

e_{ij} = scheduled end of unit i's engagement of target j.

D_{ij} = interval from s_{ij} to e_{ij}.

Other notation is as in Figures 3 and 4.

Figure 5. Additional Constraints and Notation for Scheduling Variation in Artillery Problem

(5-5), (5-6) Assignments must occur within the specified time horizon.

(5-7) Assignments for a given unit may not overlap.

(5-8), (5-9) In a mixed assignment on a given target, shells from all participating units must start and stop falling on the target simultaneously.

These constraints come from user specifications. A problem from a different area might use entirely different scheduling constraints.

The complexity of the problem modeled in Figure 5 can be appreciated by imagining an exercise in project management where the network cannot be constructed in advance except for fragments derived from specified start and end times for some activities. Durations, costs, and materials requirements are not initially known, because it is not known who will execute each activity. Some of the usual flexibility has been removed by prior restrictions on activities that may or may not be on the critical path. Thus, the scheduling variations make the problem very difficult indeed.

Difficulty of Optimal Solution

General

As was stated in Chapter I, generalized assignment problems are known [28] to belong to a class (called "P-complete") of problems for which it is believed that no nonenumerative optimal solutions can be obtained. The artillery problem is doubly complicated. If we regard the units as "jobs" to be scheduled for processing on "machines" representing targets, it can be seen to be an extension (mixed assignments, schedule restrictions) of the jobshop problem, which is also known [11] to be an unpromising ("NP-complete") problem. Indeed, all problems that are NP-complete are also P-complete, but the converse does not necessarily hold [28].

"P-complete" stands for "polynomial-complete," a term derived from a formal definition of efficiency. Garey et al. [11] defines an efficient algorithm as one for which some constant c exists such that the amount of time required for a problem with n variables will never be above $O(n^c)$. ($O(n^c)$ denotes a quantity that is "on the order of n^c.") Such an algorithm is said [11, 28] to run in "polynomial time." In other words, an efficient algorithm is one capable of being executed at worst in an amount of time on the order of a constant power of the number of variables. (This definition of efficiency appeared only recently, and thus lacks wide acceptance.) P-complete problems are believed not to be solvable in polynomial time, thus requiring enumerative solutions, for which the number of iterations is on the order of c^n, which is greater than n^c as long as c is less than n and c is greater than 2, so enumerative solutions can be very tedious. Even the branch-and-bound methods that have been developed for single-resource problems [3, 4, 9, 26, 29] cannot be guaranteed to examine fewer nodes than on the order of m^n, although the fastest algorithms [3, 4, 26] never needed excessive CPU time, for randomly generated problems of 500 to 5000 variables.

Multi-Resource Problems

Unfortunately, the optimal methods for single-resource problems offer almost no hope of extension to multiple resources. Only the algorithm of Ross and Soland [26] appears compatible with multi-resource problems, but response times would probably be too great for most applications. Running time should be many times that of the single-resource version, which on seven 20 x 50 (1000-variable) randomly generated problems used between 0.199 and 1.568 minutes of CPU time on a CDC 6600, excluding input-output and editing of the data. For multi-resource problems (using the data given by Glover et al. [14] for comparative speeds of different computers in solving transportation problems) these times could increase by thousands of times if the programs were run on a more typical computer. Storage requirements would also be very great--probably several million bits.

Attempts have been made to model single-resource problems in terms of network flows, but Balachandran [3] reported that such algorithms did not appear to be amenable to guarnateeing the binary characteristics of the variables. Ross and Soland [26] compared their algorithm to two others, one of which was a network model [19] that repeatedly exceeded a 50-minute time limit (four of seven 500-variable problems) on the CDC 6600.

A study by Glover et al. [14], reveals that it is difficult to equitably compare speeds of algorithms. However, it seems clear that any optimal algorithm would be too unwieldy for most applications.

CHAPTER III

BASIC HEURISTIC METHODS

Introduction

History and Classification

Heuristic methods are not new. Michael's lengthy review [21] reports that heuristics were once grouped with philosophy, psychology, and logic. He says the Romans recognized heuristic approaches as early as 300 A.D., and notes that both Descartes and Leibnitz tried to develop a classification system.

Michael also attempts to classify heuristic methods, as have others [5, 18, 23]. The various classifications have little in common, which may be due to each author's concentration on methods in his own field. One idea, however, that seems to fit into all systems is the concept of "construction" and "improvement" heuristics. These terms, due to Parker [23], are practically self-explanatory. Construction heuristics attempt to *generate* a complete solution, usually trying to proceed toward a solution that is especially attractive according to some objective criterion. Improvement heuristics operate on pre-existing complete solutions in an attempt to *improve* the value of the objective function.

Ubiquity

Examples of heuristics abound in everyday life. Michael gives several, such as the golfer who uses an old ball on a hole with a water hazard, or the motorist selecting a route through a city based on perceived traffic conditions.

Games (Michael mentions chess) constitute a familiar area where heuristic analysis is the only practical approach. Ignizio [18] cites remarks about the ability of humans to play ticktacktoe, in which most players generate a strategy to guide them through thousands of outcomes. Although chess is vastly more complex, there exists for either of these deterministic games an optimal strategy (which may be impractical to determine). Other games are complicated by stochastic elements that add possibilities for the use of heuristics. Startling similarity to the language of academic discussion of the philosophy behind heuristic strategies can be found in discussions between tournament bridge players.

Design Process

It seems, then, that heuristics are everywhere. Everyone has an intuitive feeling for developing and using them without being able to describe exactly what is happening. Michael [21] says that the process of developing a heuristic should be based on a study of "cognitive processes," and cites Polya [25] as recommending that the basis be experience in solving problems and watching problems be solved. A more structured philosophy is difficult to achieve. Ignizio [18] points out that the infinite number of possibilities makes it easy to criticize any one choice versus the others that were possible, and that it is probably impossible to explain the design to everyone's satisfaction. How does a painter know which brushstroke completes the canvas? These last considerations should be kept in mind when considering the methods described and evaluated in the remainder of this dissertation.

Background and Development of Specific Methods

Sahni and Gonzalez [28] have shown that P-complete problems can be as ill-suited for heuristics as for optimal methods. They conclude that any heuristic that runs in polynomial time must occasionally produce arbitrarily bad results. Therefore, neither optimal nor near-optimal results can be guaranteed to be obtainable in a reasonable amount of time. With this in mind, several heuristics were developed for this research in the hope that some may perform well when others do not.

Figure 6 outlines the heuristic methods that were developed. Many were inspired by examples described in the literature for use with problems of similar structure, such as traditional assignment and transportation models [17, 31, 33], as well as plant layout [10, 20, 23], facilities location [10, 31], covering [18], knapsack [34], and project-scheduling [8] models.

Construction Heuristics

The construction heuristics used here all fit a classification due to Ignizio [18]. They use "add" logic, in which all variables are initially set to zero, then selectively set to one in the hope that an acceptable complete solution will result. They differ according to the type of intermediate logic that decides which variable is "added."

Some are motivated by the popular method which makes assignments at random [8, 20, 23]. This procedure has the advantage of simplicity. In pure scheduling applications [8], it has produced significantly

I. Construction Heuristics

A. Random Intermediate Logic

1. RANDR: Random column, random row
2. RANDC: Random column, cheapest row

B. Penalty-based (VAM) Logic (all assign cheapest row)

1. VAMC: Column from VAM on costs
2. VAMI: Same, but on resource-biased costs

C. LP-guided Logic

1. LPMAX: Random column, row of max LP variable

II. Improvement Heuristic

A. GREEDY: <u>First</u> profitable switch

B. CRAFTY: <u>Most</u> profitable switch

Figure 6. Outline of Basic Heuristic Methods

better results than more refined heuristics. McRoberts [20] has done work in determining sample size and estimating the distribution of solution values. The speed and simplicity of randomly-guided layout heuristics has also been mentioned [23]. Two heuristics of this type will be described: RANDR and RANDC. Both can be used to obtain evaluation standards, and RANDC is a very good problem-solver.

Another form of intermediate logic used in this research was motivated by the Vogel approximation method (VAM), a textbook [17, 31, 33] heuristic giving good initial solutions for transportation problems. Preliminary research [7] produced two VAM-based heuristics that gave excellent results: VAMC and VAMI.

The third type of construction heuristic (LPMAX) has been used in many integer-constrained problems. Variable values from a continuous (linear programming) solution are adjusted to integers. As often noted [17, 30, 31, 33], adjustment must be judicious, or infeasibility or unacceptable suboptimality can occur. The continuous solution can also give information about bounds and existence of the discrete optimum. Unfortunately, obtaining the continuous solution to a problem of realistic size requires much storage and time, and there is little room for discretion in adjusting the variables.

Improvement Heuristics

Parker [2] distinguishes between "greedy" methods and the well-known CRAFT [1] technique in a class that Brockelhurst [5] calls "bivariate searches." Parker and others he cited found that (for layout problems) CRAFT gave the best objective function values, but greedy methods were faster. The adaptations used here, GREEDY and CRAFTY, run so slowly that their usefulness is limited to evaluating other methods' performances on relatively small problems.

Specific Methods

Introduction

This section describes in detail each of the methods given in Figure 6. Construction heuristics are described in a brief narrative followed by a detailed outline. The same logic is used to optimize a task in RANDC, VAMC, and VAMI, so it is given in detail only for RANDC. Problem data are assumed to be given. Figure 7 explains the notation used in the outlines, some of which is repeated from Figure 2. VAMC and VAMI will be described and outlined together because VAMC is implemented as a special case of VAMI.

Improvement heuristics are flowcharted rather than outlined. The flowchart makes the logic clearer by avoiding the subscripts on subscripts that an outline would use. Only one flowchart is used because of the similarity of the logic of GREEDY and CRAFTY.

Narrative Description of RANDC

The user specifies how many solutions are to be generated ("sample size"). A solution is generated simply by "optimizing" all tasks in random order. The best solutions are printed.

"Optimizing" a task means assigning it to the cheapest agent having sufficient remaining resources. If no agent is resource-feasible, a flag is set to indicate that the task remains unassigned. The "cost" of an unassigned task is set to a value (see II.D.3.c. below) that is so

Symbol	Method(s) Where Used; Meaning
a_{ijk}	All; Amount of resource k required by agent i to do task j
A	RANDR; Vector for shuffling agent indices
b_{ik}	All; Amount of resource k originally available to agent i
B_{ik}	All; Amount of resource k remaining for agent i
c_{ij}	All; Cost incurred if agent i is assigned to task j
C	All; Contribution of current assignment to objective function
d	RANDR, RANDC; Random number seed
F	VAMI; Factor to balance cost-inefficiency combination
H	VAMI; Vector of penalties $\{H_j\}$
i,j,k	All; Indices of agents, tasks, and resources, respectively
I,J	All; Indices of assignment currently being constructed
m	All; Number of agents, indexed by i
n	All; Number of tasks, indexed by j
N	RANDR, RANDC; "Sample Size," or number of trial solutions to be generated
p	All; Number of resources, indexed by k
P	VAMI; Matrix of inefficiency-biased costs $\{P_{ij}\}$
Q	VAMI; Combining weight for constructing P
q	VAMI; Number of values of Q to use
S	VAMI; Matrix of resource inefficiencies $\{S_{ij}\}$
T	RANDR, RANDC, LPMAX; Vector for shuffling task indices $\{T_j\}$

Figure 7. Notation Used in Outlines of Construction Heuristics

Symbol	Method(s) Where Used; Meaning
U	All, Count of tasks that could not be assigned
W	RANDC, VAMI, LPMX; Vector for seeking ith - smallest element in jth column of c_{ij}'s (modified x_{ij}'s in LPMAX) $\{W_i\}$
x_{ij}	All; = 1 if agent i is assigned to task j; = 0 otherwise
X	All; Assignment vector $\{X_j\}$: X_j=i means x_{ij}=1; X_j=-1 means task j could not be assigned
Z	All; Current value of objective function
Z_{min}	All; Minimum Z among complete solutions found so far

Figure 7. (Continued)

large that maximizing the number of assigned tasks is a direct consequence of minimizing total cost.

Outline of RANDC

I. Acquire N and d; set T_j=j for all j and A_i=i for all i.

II. Generate N solutions:

A. (Re)set u and Z to zero.

B. (Re)set B_{ik} to b_{ik} for all i and k.

C. Use random numbers to shuffle T (task indices).

D. For all j:

1. Set J = T_j (i.e., pick a task at random).

2. Set $W_i = c_{iJ}$ for all i.

3. For all i:

a. Set I to index of ith - smallest W_i.

(1) If a_{IJk} exceeds B_{Ik} for some k, go to II.D.3.b.

(2) If not, subtract a_{IJk} from B_{Ik} for all k.

(3) Set C = c_{IJ} and go to II.D.4.

b. If i < m, go to II.D.3.a. for next i.

c. If not, set $C = n \max_{i,j} (c_{ij})$; set I = -1; Add 1 to u.

4. Add C to Z, set X_J=I.

E. Print solution if new best solution or one of first five solutions.

F. Go to II.A. until N solutions have been generated.

II.D.2., 3., and 4. constitute a procedure that will be referred to as "Optimize task J" in describing VAMI/VAMC.

Example Solution Using RANDC

The following example problem will also be used to illustrate VAMI, as well as being quite similar to the problem solved in the computer runs of Appendix D.

Suppose the problem is to minimize

$$40x_{11} + 87x_{12} + 60x_{13} + 79x_{14} + 89x_{21} + 63x_{22} + 58x_{23} + 10x_{24}$$

subject to:

$$61x_{11} + 16x_{12} + 72x_{13} + 43x_{14} \le 140 \quad i=1,\ k=1$$

$$19x_{11} + 16x_{12} + 46x_{13} + 50x_{14} \le 150 \quad i=1,\ k=2$$

$$48x_{21} + 28x_{22} + 49x_{23} + 67x_{24} \le 150 \quad i=2,\ k=1$$

$$36x_{21} + 62x_{22} + 51x_{23} + 81x_{24} \le 130 \quad i=2,\ k=2$$

Corresponds to (2-2) in Figure 2.

$$x_{11} + x_{21} = 1 \quad j=1$$

$$x_{12} + x_{22} = 1 \quad j=2$$

$$x_{13} + x_{23} = 1 \quad j=3$$

$$x_{10} + x_{24} = 1 \quad j=4$$

Corresponds to (2-3) in Figure 2.

$$x_{ij} = 0 \text{ or } 1$$

Corresponds to (2-4) in Figure 2.

Note that m=2, n=4, and p=2.

Expressing the problem data as matrices and vectors to correspond with the notation of Figure 2 gives:

c_{ij}:		j=1	j=2	j=3	j=4
	i=1	40	87	60	79
	i=2	89	63	58	10

a_{ijk}:		j=1	j=2	j=3	j=4	b_{ik}:
	i=1, k=1	61	16	72	43	140
	i=1, k=2	19	16	46	50	150
	i=2, k=1	48	28	49	67	150
	i=2, k=2	36	62	51	81	130

e.g., $a_{132} = 46$, $a_{241} = 67$, etc.

This example will not exactly trace the outline of RANDC. Rather, it seeks to communicate the concept of repeated optimization of tasks in random order which is the main idea of RANDC. Three solutions will be generated.

Suppose the vector T is first shuffled to give the order 4, 1, 2, 3 for optimizing the tasks. Task 4 is assigned to agent 2 (the cheapest agent) at a cost of 10. The resource supplies for agent 2 are reduced from 150 and 130 to 83 and 49. Note that it is no longer possible to assign tasks 2 and 3 to agent 2 because they would require more of resource 2 (62 or 51) than is available (49).

Task 1 is the next to be optimized. Agent 1 is cheapest at a cost of 40 and is resource-feasible. The data matrices, annotated to show the effect of the first two assignments, are:

$$c_{ij}: \quad \begin{array}{cccc} 40 & 87 & 60 & 79 \\ 89 & 63^* & 58^* & 10 \end{array} \qquad \text{slack}$$

$$a_{ijk}: \quad \begin{array}{cccc} 61 & 16 & 72 & 43 \\ 19 & 16 & 46 & 50 \\ 48 & 28^* & 49^* & 67 \\ 36 & 62^* & 51^* & 81 \end{array} \qquad b_{ik}: \begin{array}{c} 79 \\ 131 \\ 83 \\ 49 \end{array}$$

Circled elements are those associated with assignments that have been made; those marked with an asterisk indicate that the corresponding assignment has become infeasible because of resource limitations.

The third task to be optimized is task 2. The annotated data matrices are:

$$c_{ij}: \quad \begin{array}{cccc} 40 & 87 & 60^* & 79 \\ 89 & 63 & 58^* & 10 \end{array}$$

						slack b_{ik}:
	61	16	72*	43		63
	19	16	46*	50		115
a_{ijk}:	48	28	49*	67		83
	36	62	51*	81		49

Note that task 3 cannot be assigned to either agent. Agent 1 would require 72 units of resource 1 and only 63 are available. A similar situation exists for agent 2's second resource, of which 51 units are needed, but only 49 units remain.

This first solution is thus complete, with a total cost of 137 (40 + 87 + 10) with one task remaining unassigned.

Suppose the second RANDC solution begins by shuffling the vector T to obtain the order 1, 3, 4, 2 for optimizing the tasks. When task 1 is optimized by assigning it to agent 1 at a cost of 40, not enough resources are used to interfere with any potential assignment of another task. However, after optimizing task 3 via assignment to agent 2 at a cost of 58, the potential assignment of task 4 to agent 2 becomes infeasible:

c_{ij}:	40	87	60	79
	89	63	58	10*

					slack b_{ik}:
	61	16	72	43	79
	19	16	46	50	131
a_{ijk}:	48	28	49	67*	101
	36	62	51	81*	79

Task 4 is next to be optimized, and only agent 1 has sufficient resources. This assignment, at a cost of 79, does not reduce resource supplies enough to affect any potential assignment of task 2. This is therefore made to agent 2, which is cheapest at a cost of 63. This gives a complete solution in which no tasks remain unassigned:

$$c_{ij}: \quad \begin{array}{cccc} 40 & 87 & 60 & 79 \\ 89 & 63 & 58 & 10 \end{array}$$

$$a_{ijk}: \quad \begin{array}{cccc} 61 & 16 & 72 & 43 \\ 19 & 16 & 46 & 50 \\ 48 & 28 & 49 & 67 \\ 36 & 62 & 51 & 81 \end{array} \qquad \begin{array}{c} \text{slack} \\ b_{ik}: \\ 36 \\ 81 \\ 73 \\ 17 \end{array}$$

This, as can be seen by inspection or enumeration, is the optimum solution, with a total cost of 240.

A third RANDC solution is generated by shuffling the elements of the vector T to obtain, for example, an order of 4, 2, 3, 1 for optimizing tasks:

$$c_{ij}: \quad \begin{array}{cccc} 40 & 87 & 60 & 79 \\ 89 & 63 & 58 & 10 \end{array}$$

$$a_{ijk}: \quad \begin{array}{cccc} 61 & 16 & 72 & 43 \\ 19 & 16 & 46 & 50 \\ 48 & 28 & 49 & 67 \\ 36 & 62 & 51 & 81 \end{array} \qquad \begin{array}{c} \text{slack} \\ b_{ik}: \\ 52 \\ 88 \\ 35 \\ 13 \end{array}$$

The total cost of this complete solution is only 245, so it represents a useful alternative to the optimal solution obtained earlier.

Narrative Description of RANDR

This heuristic generates solutions by assigning tasks in random order to randomly chosen agents. Tasks are assigned only to resource-feasible agents, however.

Outline of RANDR

This is identical to RANDC except for II.D.2. and 3. which are replaced by the following:

II.D.2. Shuffle A (agent indices)

3. For all i:

a. Set $I = A_i$ (i.e., pick an agent at random).

The remainder of II.D.3. is the same as given for RANDC.

Narrative Description of VAMI/VAMC

The logic of this heuristic can probably best be understood by tracing its development. VAMC, the first heuristic developed in this research, is essentially identical to the Vogel Approximation Method, except that penalities ("H") are calculated for columns (tasks) only, and not additionally for rows as with transportation problems. The task associated with the largest penalty is optimized. Any penalties that could have changed (by some assignment becoming infeasible) are recalculated.

VAMC often produced bad results in preliminary research. It could not avoid assignments that were especially inefficient uses of resources if the relative cost was low. VAMI attempts to overcome this by combining the cost of a prospective assignment with its resource inefficiency (which is a sort of "resource cost"--the fraction of the agent's remaining supply of the scarcest resource). Different combinations are tried, each with more weight (Q) on inefficiency (s_{ij}) and less (1-Q) on cost (c_{ij}).

For each value of Q between zero and one, a "P-matrix" of the combined cost and inefficiency elements is built. A balancing factor (F) must first be applied to make the average inefficiency equal to the average cost, because these averages usually differ by several magnitudes. Penalties are calculated from the P-matrix.

Otherwise, VAMI is the same as VAMC. In fact, VAMI is equivalent to VAMC when Q is zero, because p_{ij} is then equal to c_{ij} (see IV.B.2. of the following outline).

VAMI resembles (and was motivated by) the optimization of a La Grangian function, with Q playing the role of a multiplier. No claim is made, however, that this resemblance justifies any expectation of near-optimal results.

Great efforts have been made to find a way to predict the best values of Q and q. Unfortunately, only the following impressions were produced:

(1) The best results were <u>usually</u> obtained for small (but nonzero) values of Q, unless constraints were very tight.

(2) The best value for q was <u>usually</u> between 3 and 25, with larger values of q being needed for tight constraints.

The results of these observations were incorporated into VAMI as follows:

(1) The steps taken in Q (see IV.C. and D.) from 0 to 0.25 are only a third as large as those taken from 0.25 to 1, but equal in number. Allowing for the VAMC trial (Q=0) means q must be odd.

(2) q can be acquired as a user input, or as a value calculated

from the data (say, 20 times average S_{ij}), or as a constant (11 usually works well). One will be added if q is even.

Outline of VAMI/VAMC

I. Acquire q.

II. For all i and j where agent i is feasible for task j:

   A. Set $S_{ij} = \max_{k} (a_{ijk} \div b_{ik})$.

   B. Accumulate Σc_{ij} and ΣS_{ij}.

III. Calculate balancing factor and initialize Q:

   A. Set $F = \Sigma c_{ij} \div \Sigma S_{ij}$ (Sums calculated above).

   B. Set Q = 0.

IV. Generate the number of solutions specified by q:

   A. Set u = 0 and Z = 0.

   B. For all i:

      1. (Re)set B_{ik} to B_{ik} for all k.

      2. Set $P_{ij} = (1-Q)c_{ij} + Q \cdot F \cdot S_{ij}$ for all j.

   C. If Q < .25, add 1/(2q - 2) to Q.

   D. If not, add 3/(2q - 2) to Q.

   E. Set H_j = difference between two smallest P_{ij} for all j.

   F. For all j:

      1. If $j \neq 1$, recalculate H_j if possibly affected by the previous assignment.

      2. Set J to index of jth - largest H_j.

      3. Optimize task J.

   G. Print first 5 solutions and all new best solutions.

   H. If Q exceeds 1, stop. If not, go to IV.A.

Example Solution Using VAMI

The same problem is used as with RANDC:

$$c_{ij}: \begin{matrix} 40 & 87 & 60 & 79 \\ 89 & 63 & 58 & 10 \end{matrix}$$

$$a_{ijk}: \begin{matrix} 61 & 16 & 72 & 43 \\ 19 & 16 & 46 & 50 \\ 48 & 28 & 49 & 67 \\ 36 & 62 & 51 & 81 \end{matrix} \qquad b_{ik}: \begin{matrix} 140 \\ 150 \\ 150 \\ 130 \end{matrix}$$

Before generating any solutions, a matrix $\{S_{ij}\}$ of resource inefficiencies must be calculated:

$$S_{ij}: \begin{matrix} .44 & .11 & .51 & .31 \\ .32 & .48 & .39 & .62 \end{matrix}$$

As stated in the Outline of VAMI/VAMC,

$$S_{ij} = \underset{k}{\text{Max}} \; (a_{ijk} \div b_{ik}).$$

For example, the value of .44 for S_{11} was obtained as follows:

$$S_{11} = \text{Max}\; \left(\frac{a_{111}}{b_{11}}, \frac{a_{112}}{b_{12}}\right) = \text{Max}\; \left(\frac{61}{140}, \frac{19}{150}\right) = \text{Max}\; (.44,\ .13) = .44$$

The costs and inefficiencies are summed:

$$\sum_i \sum_j c_{ij} = 40 + 87 + \ldots + 58 + 10 = 486$$

$$\sum_i \sum_j S_{ij} = .44 + .11 + \ldots + .39 + .62 = 3.18$$

Their ratio is calculated to use as a balancing factor in later calculations, in which it is desirable to transform the inefficiencies so that their average magnitude will be equal to average cost:

$$F = \frac{\sum_i \sum_j c_{ij}}{\sum_i \sum_j S_{ij}} = \frac{486}{3.18} = 152.83$$

which is rounded to 153 for convenience in this example.

In the iterative portion of VAMI, the number of solutions generated is given by q. Q is started at zero and is increased to 1 in q steps, not all of which will be given here. Every solution is guided by VAM-style penalties developed from a matrix $\{P_{ij}\}$ whose elements are functions of Q and the corresponding cost and balanced inefficiency values:

$$P_{ij} = (1 - Q)\ c_{ij} + QFS_{ij}$$

Note that when $Q = 0$, $P_{ij} = c_{ij}$ and VAMI is equivalent to VAMC (i.e., penalties are calculated from costs alone, without considering potential resource problems).

Thus, for $Q = 0$, penalties will be calculated from the matrix

$$P_{ij}: \begin{matrix} 40 & 87 & 60 & 79 \\ 89 & 63 & 58 & 10 \end{matrix}$$

VAMI-style penalties are calculated by subracting the smallest element in each column from the second-smallest. When this is done for the above matrix, the penalty vector $\{H_j\}$ is obtained:

$$H_j: \quad 40 \quad 24 \quad 2 \quad 69$$

The largest penalty is 69, associated with task 4, which is then optimized:

$$c_{ij}: \begin{matrix} 40 & 87 & 60 & 79 \\ 89 & 63* & 58* & 10 \end{matrix}$$

$$a_{ijk}: \begin{matrix} 61 & 16 & 72 & 43 \\ 19 & 16 & 46 & 50 \\ 48 & 28* & 49* & 67 \\ 36 & 62* & 51* & 81 \end{matrix} \qquad \begin{matrix} \text{slack } b_{ik}: \\ 140 \\ 150 \\ 83 \\ 49 \end{matrix}$$

Penalties must be recalculated, because with only two agents in the problem, any assignment must affect either the cheapest or second-cheapest agent. There is no change in the penalty for task 1, but the cheapest agents have become infeasible for tasks 2 and 3. Since only one agent is still available for these two tasks, the penalty is arbitrarily calculated by subtracting the corresponding P_{ij} from 99998. Task 4 is already assigned, so no penalty calculation will be made for it, which is indicated by "**" in the following vector of recalculated penalties:

H_j: 49 99911 99938 **

The largest penalty is associated with task 3, which is assigned to agent 1. This does not consume enough resources to further affect feasibility, so recalculation will not change the penalties associated with tasks 1 and 2:

H_j: 49 99911 ** **

Task 2 is assigned to agent 1. This makes agent 1 infeasible for task 1, which will thus be assigned to agent 2. This gives the same near-optimum (total cost: 246) as the third RANDC solution.

Taking further arbitrary steps of 0.1 in Q will not change the solution until Q reaches 0.4, where VAMI will not yield a feasible solution. The next example uses Q = 0.5 to obtain a new alternative solution that is only 10 percent worse than the optimum. The resource-biased costs are:

$$P_{ij}: \begin{array}{cccc} 53 & 52 & 69 & 63 \\ 69 & 68 & 59 & 52 \end{array}$$

These figures were obtained from the formula given earlier. For example,

$$P_{11} = (1 - Q)c_{11} + QFS_{11} = (.5)(40) + (.5)(153)(.44) = 53$$

The P_{ij} values have been truncated to integers for convenience (this is also done in the program to allow use of integer arithmetic to improve execution speed). From them a vector of penalties is calculated:

H_j: 16 16 10 11

There is a tie for the largest penalty between tasks 1 and 2. Such ties are arbitrarily broken in favor of the lower-numbered task, so task 1 is assigned to agent 1, because P_{11} is less than P_{21}. This does not affect any potential assignment of another task, so the recalculated penalties show no change:

H_j: ** 16 10 11

This means that task 2 is the next to be assigned. It is assigned to agent 1, which is associated with the lowest P_{ij}, even though the corresponding c_{ij} is not the lowest currently feasible for task 2. This shows how, as Q increases, VAMI becomes increasingly biased toward assignments that make especially good use of resources. Thus, the status of the problem is:

					slack b_{ik}:
c_{ij}:	40	87	60*	79	
	89	63	58	10	
a_{ijk}:	61	16	72*	43	63
	19	16	46*	50	115
	48	28	49	67	150
	36	62	51	81	140

Since agent 1 has become infeasible for task 3, the penalties are recalculated as:

H_j: ** ** 99939 11

and task 3 is assigned to agent 2. This forces the assignment of task 4 to agent 1 because of resource limitations, giving:

$$c_{ij}: \quad \begin{array}{cccc} 40 & 87 & 60 & 79 \\ 89 & 63 & 58 & 10 \end{array}$$

$$\begin{array}{lccccc} & & & & & \text{slack} \\ & & & & & b_{ik}: \\ & 61 & 16 & 72 & 43 & 20 \\ & 19 & 16 & 46 & 50 & 65 \\ a_{ijk}: & 48 & 28 & 49 & 67 & 101 \\ & 36 & 62 & 51 & 81 & 79 \end{array}$$

The total cost of this solution is 264, which compares well with the optimum of 240.

Increasing Q above 0.7 causes a solution to be generated that is similar to the above except that task 1 is assigned to agent 2. The cost of that alternative would be an unattractive 313.

VAMI did not find the optimum for this example (as RANDC did), but it did produce three feasible solutions, two of which were very near the optimum.

Narrative Description of LPMAX

Despite the apparent complexity of LPMAX, the basic logic is fairly simple. Any $x_{ij}=1$ indicates that the corresponding assignment can be made immediately. Tasks that remain unassigned are optimized in random order exactly as in RANDC, except that elements of W corresponding to nonzero x_{ij} are set equal to x_{ij} instead of c_{ij}.

Outline of LPMAX

It is assumed that a continuous optimum solution is available for a problem identical to Figure 2 except for relaxation of the zero-one constraint (2-4) to allow x_{ij} to take on any value from zero to one.

I. Initialization.

A. Acquire N, d, and x_{ij}'s for all i and j.

B. For all j:

1. For all i;

a. If $x_{ij} = 1$:

(1) Store j in right-hand end of T (starting at T_n).

(2) Set X_j=i.

(3) Go to I.B.1. for next j.

2. (All x_{ij} known to be $\neq 1$ for this j): Store j in left-hand end of T (starting at T_1).

II. Generate N solutions.

A. Set u and Z to zero, set $B_{ik} = B_{ik}$ for all i and k.

B. Shuffle left-hand indices in T.

C. For all j = (n, n-1,..., 2, 1) (note right-to-left order).

1. Set J = T_j.

2. If right-hand j, go to II.C.4.

3. If left-hand j:

a. For all i:

(1) Set $W_i = 1000(1-x_{ij})$.

(2) If $W_i = 0$, set $W_i = 1000 + c_{ij}$.

b. For all i:

(1) Set I to index of ith-smallest W_i.

(a) If a_{IJk} exceeds B_{Ik} for some k, go to II.C.3.b.(2).

(b) If not, subtract a_{IJk} from B_{Ik} for all k.

(c) Set $C = c_{IJ}$ and go to II.D.4.

(2) If $i < m$, go to II.C.3.b.(1) for next i.

(3) If not, set $C = n \max_{i,j} (c_{ij})$; set $I = -1$; add 1 to u.

4. Add C to Z, set $X_J = I$.

D. Print first five solutions and all new best solutions.

E. Go to II.A.

GREEDY/CRAFTY

These two methods are flowcharted together in Figure 8, where reference is made to "RH" (right-hand) and "LH" (left-hand) tasks, which are the two tasks being considered for changes in agent assignment. The methods terminate when a complete cycle through all possible changes produces none that are feasible and profitable. A cycle addresses all (left-hand) tasks from 1 to n-1. For each of these, a trial agent is chosen. Then, each (right-hand) task of higher index than the left-hand task is examined to see if it is feasible for its assignment to be switched to some trial agent giving a lower objective function value in conjunction with the trial agent for the other task. In GREEDY, the change is made immediately, but CRAFTY makes the best change found in the entire cycle. Both methods then begin a new cycle.

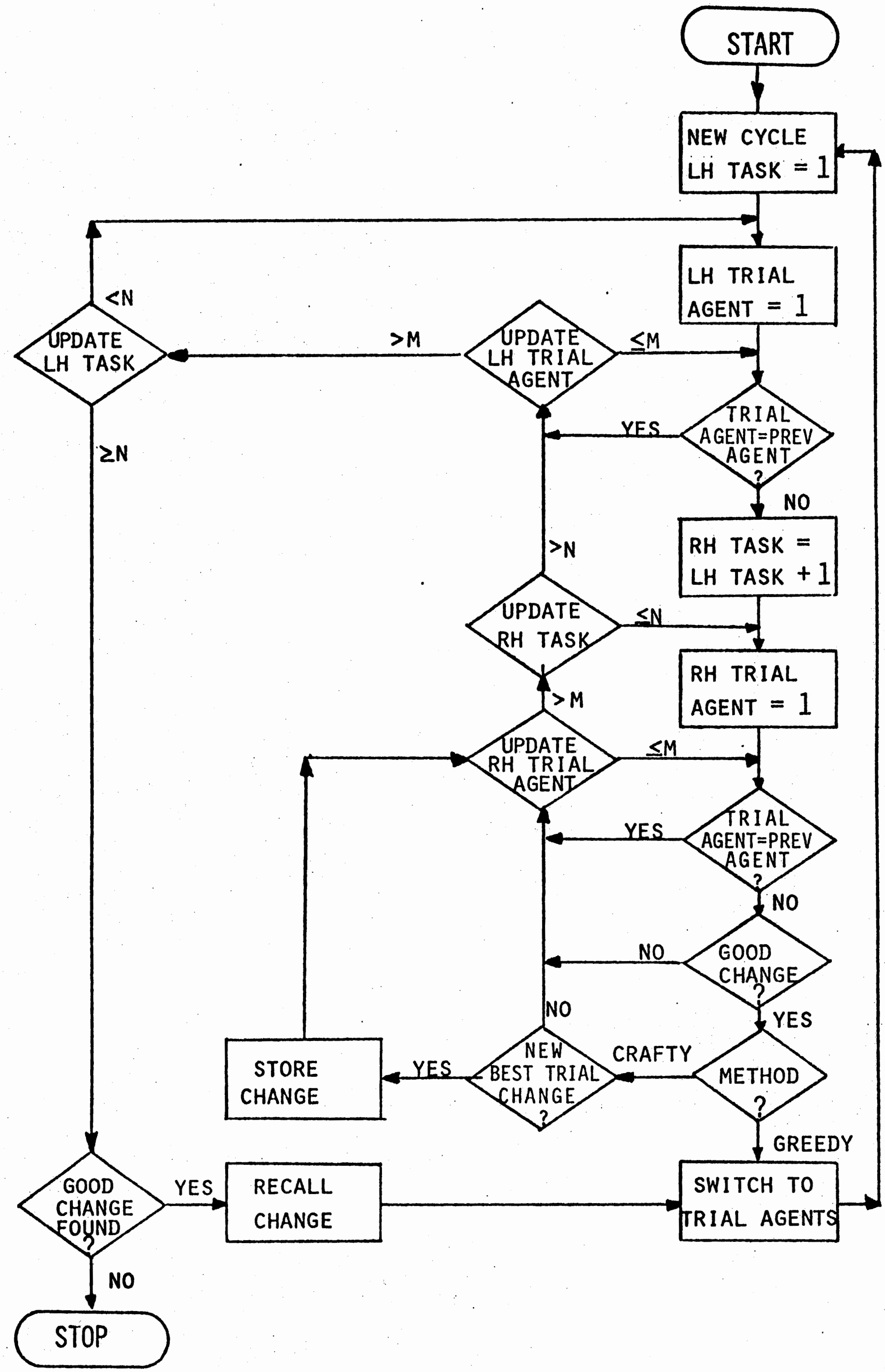

Figure 8. Flowchart of GREEDY/CRAFTY

CHAPTER IV

BASIC METHODS PROGRAMMED AND TESTED

Introduction

This chapter describes the programming and testing of the basic methods of Chapter III, as well as auxiliary routines written to facilitate testing.

Programs

Languages

All programs are written in FORTRAN IV, except that continuous solutions are produced by IBM's MPS (Mathematical Programming System).

Organization

Each solution method is programmed as a subroutine named SOLVER, which is called as part of an overall testing scheme which is flow-charted in Figure 9. A small main program directs the first step of the scheme through a housekeeping and control routine SOLOOP from which SOLVER is called. Before calling SOLOOP, the main program uses other subroutines to randomly generate (MATGEN) and print (MATPRT--optional) problems. After SOLOOP, another optional subroutine (MPSGEN) can be called to create a data set for input to MPS in the second step of the scheme. SOLVER calls SWAPPR, which is optionally GREEDY or CRAFTY.

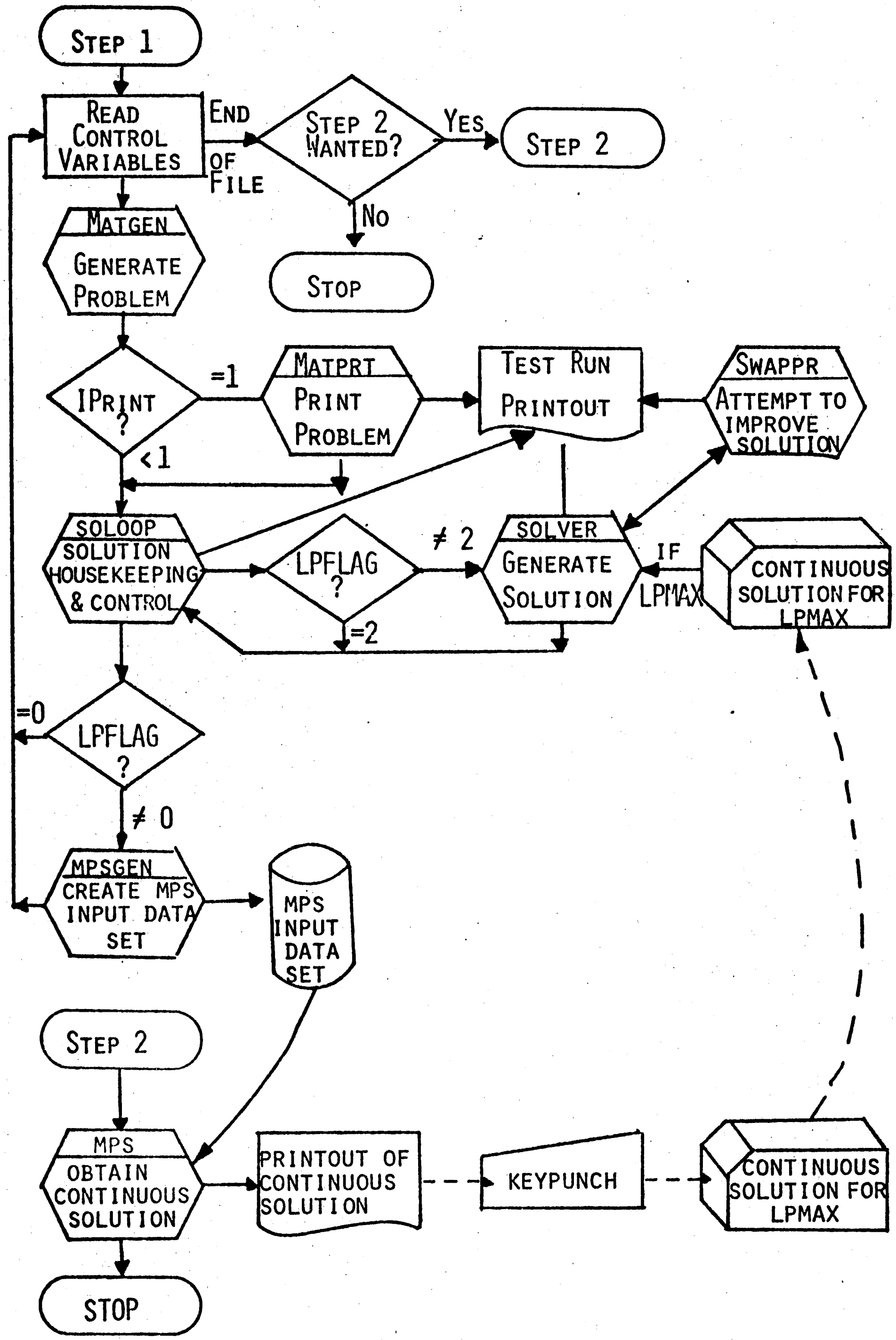

Figure 9. Flowchart of Testing Scheme for Programmed Basic Solution Methods

The following paragraphs outline or describe each test routine.

Outline of Main Program

I. Read control variables:

A. NOVBLS: Indicates end-of-file if greater than 9000.

B. ISEED: Seed for random-number function (RANDU).

C. IPRINT: Print switch; controls degree of detail in printout.

D. NBIGQS: "N" or "q" from Figure 7, depending on method used by SOLVER.

E. MTEST: Passed to SOLOOP to control number of solutions produced (one for each set of b_{ik} right-hand-side values), and (optional--used with LPMAX) reading of x_{ij} values from a previous continuous solution.

F. LPFLAG: Controls calling of MPSGEN (see below).
LPFLAG = 0: MPSGEN not called.
LPFLAG = 1: MPSGEN called after SOLVER runs.
LPFLAG = 2: Prevents SOLOOP from calling SOLVER; only MPSGEN is called.

G. IGREED: Controls method used in SWAPPR.
IGREED = 0: No improvement is attempted.
IGREED = 1: GREEDY.
IGREED = 2: CRAFTY.

H. MM,NN,PP: Problem dimensions (m,n,p in Figure 2).

II. Call MATGEN to generate problem.

III. Call MATPRT if IPRINT = 1.

IV. Call SOLOOP to call SOLVER for several sets of b_{ik} values.

V. Call MPSGEN to generate MPS problem data (unless LPFLAG is zero).

Outline of MATGEN

I. Generate c_{ij} and a_{ijk} values as integers distributed U(1,1000).

II. Generate number of infeasibilities as an integer distributed U(1,mn/3).

III. Generate indices of infeasibilities as integers distributed (row) U(1,m) or (column) U(1,n).

IV. Flag infeasibilities: $c_{ij} = 9999$; $a_{ijk} = 0$.

Outline of MATPRT

I. Print matrix of c_{ij} values.

II. For each k, print matrix of a_{ijk} values.

Outline of SOLOOP

I. If all agents are infeasible for some task, restore feasibility for a randomly chosen agent.

II. Find and print unconstrained optimum and resources required for it by each agent. This determines maximum b_{ik} value (IBSTOP) to be tried in V. below.

III. Return to Main Program if LPFLAG = 2 (i.e., MPS data are only output wanted; see I.F. in Outline of Main Program, above).

IV. Calculate cost of unassigned task as $n \cdot \max_{i,j} (c_{ij})$.

V. Control generation of solutions:

A. Check MTEST to control handling of b_{ik} values and (optional; used with LPMAX) input of optimal continuous x_{ij} values produced by MPS. All b_{ik} will be equal (variable name: IB) to facilitate testing.

MTEST = 0: Takes 11 steps in IB from 50p(n/m + 1) to IBSTOP (see II. above).

MTEST > 0: MTEST is the number of values of IB that are tried. Each IB is read from a card.

MTEST < 0: The negative of MTEST is again the number of IB's that are tried. However, after each IB, a deck of cards is read which contains i,j, and [1000 x_{ij}] for each nonzero x_{ij} in an earlier MPS solution for the IB just read.

B. For each IB:

1. Finds and prints unconstrained optimum when IB < 1000. Because a_{ijk} is distributed U(1,1000), this gives a tighter bound on the optimum than calculations in II. above.

2. Calls SOLVER to obtain a solution for all b_{ik} = IB.

Description of MPSGEN

The flow of MPSGEN is determined by the sequence required for MPS input data, an example of which can be found in Appendix B. The output of MPSGEN can be related to Figure 2 as follows:

MPS Data Item	Notation in Figure 2
Row R00000	Objective Function (2-1)
Row R1iiik	Resource Constraints (2-2)
Row R2jjj	Coverage Constraints (2-3)
Column X1iiijj(n<100)	x_{ij}
Column X1iijjj(n>99)	X_{ij}

The unmodified output of MPSGEN can be used by MPS in the next job step.

Description of SOLVER

SOLVER is coded using symbolic names that are either self-explanatory or coincide as closely as possible with Figures 2 and 7. Figure 10 establishes correspondence between Figures 2 and 7 and the code of SOLVER (see Appendix A). Four versions of SOLVER were prepared: RANDC, RANDR, LPMAX, VAMI. Each version of SOLVER uses logic that is similar to the corresponding outline in Chapter III. The main exception is the use of IB for all b_{ik}, which greatly facilitates testing without (because a_{ijk} are random variables) introducing undesirable bias into the testing process. Each SOLVER can be easily recoded to use b_{ik} values passed in an array. The solutions found by SOLVER will be printed with a degree of detail that depends on the value stored in IPRINT:

Symbol in Figures 2 and 7	Variable Name(s) in Appendix A
a	AV(vector form), A(matrix)
A	AB
b	IB
B	BV,B
c	CV,C
C	CBIG
d	ISEED
F	F
H	H
i	I
I	IBIG
j	J
J	JBIG
k	K
m	MM(object-time dimension, M(operational)
n	NN,N
N	NBIGQS
p	PP,P
P	PS
q	NBIGQS
Q	Q
S	S
T	T
u	U
W	W
X	XB
Z	Z
Z_{min}	MINZ

Figure 10. Symbols From Figures 2 and 7 Corresponding to Variable Names in Appendix A

IPRINT = 1: The first 5 solutions and all new best solutions are printed in long form. This includes the value of the objective function ("COST"), the number of tasks remaining unassigned ("NO UNASGD TASKS"), the sum of the C_{ij} values for the assigned tasks ("COST OF ASGD TASKS"), and the number of trials necessary for SOLVER to obtain the solution ("TRIAL NO."), all on a single line. The next line begins with the words "ASSIGNMENT VECTOR:" followed by X_1 through X_{20}, with additional lines being used as needed for X_{21} through X_n. Then the slacks (each agent's remaining supply of each resource) are printed.

IPRINT = 0: Identical to IPRINT = 1, except that new best solutions are the only ones printed.

IPRINT = -1: This also causes output to be printed only for new best solutions, but in short form, where the slacks are not printed.

SWAPPR is called to try to improve any new best solution. If IGREED = 0 SWAPPR will take no action. However, even if IGREED = 0, it will be set to 1 to let GREEDY attempt to improve the best solution found by SOLVER for each value of b_{ik}. VAMI uses a subroutine named PENCOL to obtain or recalculate the penalty for each task.

Description of SWAPPR

This subroutine follows the logic of Figure 8. The code of SWAPPR in Appendix A refers to six important indices:

JL (JR) Index of left(right)-hand task

IL (IR) Index of agent to which left(right)-hand task is currently assigned

IL2(IR2) Index of trial agent for left(right)-hand task

The fundamental decision of SWAPPR is to determine if a cost savings can be attained without violating any resource constraints if the assignment of task JL is switched from agent IL to agent IL2 while switching task JR from agent IR to agent IR2.

Improvement methods can be used in a "stand-alone" mode if an initial solution is made available to SWAPPR for improvement.

Description of RANDU

RANDU is a multiplicative congruential generator of pseudorandom variates distributed U(o,1). It was adapted as a FORTRAN FUNCTION from the well-known subroutine RANDU found in IBM's Scientific Subroutine Package. The modification used in Appendix A was designed for maximum speed, but retains the statistical characteristics of the original RANDU. RANDU is machine-dependent, as are almost all such routines, and will probably need to be rewritten if not implemented on a computer similar to the IBM 360/370 series.

Continuous Solutions with MPS

MPS is implemented in a straight-forward manner, as can be seen from the code in Appendix A. The only extension beyond the simplest minimization of a linear program is the use of the "BOUND" option to "SETUP" the relaxation of the zero-one constraint to bounded variables. In this work, the output of MPSGEN has always been passed to MPS as a temporary data set. This is easily accomplished using Job Control cards, and is much more convenient than handling the thousands of data cards required to describe the continuous form of a thousand-variable program with several resources.

Passing the Results of MPS to LPMAX

Usually, almost all variable values produced by MPS are zeros. A typical problem with 50 tasks might have only 55-65 nonzero x_{ij}'s in its continuous solution, depending on tightness of constraints, even though the total number of variables might be 1000 or more. This makes it fairly convenient to manually prepare input cards for testing LPMAX.

Testing the Programs

Preliminary Testing

Initially, several problems of various dimensions were run in order to decide on the design of further testing procedures. For most problems, RANDC, VAMI (which includes VAMC), and LPMAX were allowed to produce several solutions each, with their best solutions being improved by GREEDY and CRAFTY. GREEDY and CRAFTY were also used in the "stand-alone" mode by allowing RANDC to generate one solution which was then passed to SWAPPR for improvement. Finally, as aids to evaluation, RANDC and RANDR were run for large values of N and succeeded by GREEDY. The continuous optimum produced for LPMAX and the unconstrained zero-one optima found by SOLOOP also served as evaluation standards. Several general observations were made.

Execution Time

Not surprisingly, this seemed to be a function of the number of variables (mn), the "shape" (ratio of m to n), and the number of resources (p). Different methods appeared to be affected quite differently by these factors, however. GREEDY and CRAFTY are too slow to use on large problems, even for test purposes.

Objective Function Values

In this respect, the construction heuristics were consistently closer (in percentage) to a bound on the optimum for large (mn = O(1000)) problems than for small, which was unexpected. However, this became less surprising after calculations revealed that the average difference among all possible objective values is many magnitudes less for a large problem than for a small one. Consider the following example of two problems of the same shape but different size:

Dimensions (m x n):	7 x 5	35 x 25
No. Solutions (m^n):	1.6×10^4	4.0×10^{38}
Worst Solution (All Tasks Unassigned):	25000	625000
Best Solution (Unconstrained Optimum):	1639	580
Average Difference Between Solutions:	1.4	1.6×10^{-33}

Of course, many of the m^n possible solutions are usually infeasible, but a similar analysis based only on feasible solutions is not a reasonable undertaking, and it is doubtful if the results would differ significantly.

Feasibility

Where feasible solutions were known to exist, construction heuristics seemed to be a bit better at finding them for large problems than for small ones. Again, there are probably enormously greater numbers of feasible solutions to a large problem.

Problem Characteristics

It was clearly impossible to test all methods thoroughly with several problems in each category of characteristics. Suppose five different problem sizes were tested for six different shapes with five different sets of b_{ik} values for from one to four resources, using each basic method, with GREEDY and CRAFTY being applied to the final result of each construction heuristic, along with the use of RANDC and RANDR for very large values of N (2000) to obtain a solution that would be 99.7 percent sure to lie in the .997 quantile of all solutions. Even without multiple replication, thousands of computer runs would be required, many of which would cost over $100 each. The testing of programs would require several years, and several rooms could be filled with the printouts.

From the preliminary testing, it appeared that there were pronounced performance differences between the methods. Therefore, it was decided that an extensive testing procedure as described above would reveal very little that could not be inferred from an abbreviated scheme. Each problem characteristic was considered from the standpoint of its importance in revealing differences in the performance of methods relative to each other.

Problem Size

This characteristic had great effect on performance during preliminary testing, but the effect appeared to be purely linear (construction) or quadratic (improvement). Relative performance between methods seemed to be almost the same for small and large problems.

Therefore, it was decided to do almost all further testing for problems with approximately (1) 50, or (2) 1000 variables. Other problem sizes would only be "spot-checked."

Problem Shape

This seemed to be a very important characteristic, so it was decided to try five or six shapes for each problem size. However, it appeared that "tall" problems (m/n of, say, three or more) gave identical objective values with any method. There were usually strong indications that these results were optimal. Therefore, more emphasis was placed on "wide" (m/n about 0.1) problems than on "tall" ones.

Number of Resources

This affected LPMAX, CRAFTY, and GREEDY strongly, but made less difference with other methods. Also, it made little difference in the relative performance of the methods. Therefore, various values of p were tried for most problems, with p being held constant for an occasional specialized test.

Tightness of Constraints

Relative performance of methods appeared to depend on the degree to which problems were constrained, so it was decided to try several values of b_{ik}. To increase the chances of interesting results, one method (usually RANDC or VAMI) was run with MTEST = 0, which caused 11 values of b_{ik} to be tried. The other methods were then used with the (usually) four b_{ik} values which appeared to be most likely to cause differences in relative performance of methods.

Characteristics of Methods

These make it impossible to devise a "fair" way to compare methods. One obvious appraoch would be to allow each method equivalent time and storage (perhaps combined, e.g., kilobyte-hours) to work on identical problems. Also, methods could be allowed to run until equivalent solutions were produced. Neither of these approaches is fair because methods vary in their performance characteristics:

(1) Some methods (RANDC) can make better use of additional time than others (VAMI).

(2) Some methods (LPMAX) require a high initial investment of storage and time for the first solution, but subsequent solutions are produced very rapidly.

(3) There is no way to be sure that each method has been coded to use individual logic features as efficiently as possible.

(4) Each method is designed to use different amounts of time and/or storage in the hope of obtaining a solution whose quality is related to its cost.

Glover et al. [14] also concluded that no "fair" comparison can be devised.

Some approach, however, had to be chosen. The considerations discussed in the last several pages led to the final test design, in which the "equal time" approach allowed RANDC the same time as needed by VAMI, while other methods were run so as to reveal if their results justified their cost. Results were compared from many viewpoints.

Test Design

RANDC was run for approximately the time needed by VAMI for $q = 11$.

LPMAX and GREEDY were run as seemed "natural" for them:

(1) LPMAX was run for N = 10 after being allowed the tremendous overhead of MPS.

(2) GREEDY was run to completion with an initial solution produced by RANDC for N = 1. GREEDY was not tested for large problems.

Most of each test run was devoted to obtaining evaluation standards. Problem size determined what could be done:

(1) Small problems, where mn = 0(50):

(a) RANDR and RANDC were run for N = 2000.

(b) GREEDY was used to attempt to improve the results obtained by each other method.

(2) Large problems, where mn = 0(1000):

(a) RANDR and RANDC were usually run for N = 500, although several runs were made for N = 2000.

(b) No improvement with GREEDY was attempted. A single run would have cost about $200.

(3) SOLVER calculated the unconstrained optimum.

(4) MPS gave the continuous optimum. Not only the solution value was used, but also the fraction of nonzero x_{ij} that were equal to one. This fraction seemed to be a good indicator of constraint severity, since no method (in preliminary testing) ever found a way to cover all tasks when this fraction was below one-half.

Summary of Test Runs

("Run" means one execution of the scheme described above in Test Design for one set of b_{ik} values.) A total of 107 runs were made--66 for small problems (only 47 and 53 of these included LPMAX and GREEDY, respectively) and 41 for large problems. Of the 107 test runs, 42 were intended to produce results for detailed tabulation to allow direct comparison of relative performances of the various methods. These 42 runs were made for eleven different problems by using four (two with Problem 5) sets of b_{ik} values for each problem. The results are displayed in Tables II through XII and summarized in Table I.

TABLE I

SUMMARY OF PROBLEM RESULTS

Problem/Table	Size (mn)	m	n	p	Number of b_{ik} Values
1/II	48	4	12	4	4
2/III	50	5	10	1	4
3/IV	49	7	7	3	4
4/V	48	6	8	1	4
5/VI	48	8	6	3	2
6/VII	48	3	16	4	4
7/VIII	1000	10	100	2	4
8/IX	1000	20	50	3	4
9/X	992	31	32	1	4
10/XI	1000	40	25	3	4
11/XII	1000	50	20	4	4

The other 65 runs were used in part to investigate special performance characteristics of methods. All runs were used in summary tabulations.

Test Results

General

The following paragraphs present and discuss summary tabulations of test results based on all 107 test runs. Discussions of special

TABLE II

TEST RESULTS FOR PROBLEM 1

No. Variables 48(4 x 12)	No. Resources 4 $\ddot{u},Z_{min}$ obtained for:				Seed (RANDU) 7001
Methods/Trials	$b_{ik}=$ 1370	$b_{ik}=$ 1940	$b_{ik}=$ 2130	$b_{ik}=$ 2510	Avg. CPU Time (sec.) per b_{ik}
RANDC/30	3,38613	0,4222	0,3385	0,3381	0.3
+ GREEDY	Same	0,4107	Same	Same	0.5
VAMC	5,62413	1,15220	0,3385	0,3381	--
VAMI/11	4,51676	0,4347	0,3385	0,3381	0.3
+ GREEDY	3,39753	Same	Same	Same	0.5
LPMAX/10	3,40101	1,14524	0,3385	0,3381	5.3
+ GREEDY	Same	Same	Same	Same	0.4
GREEDY	4,49188	0,4107	0,3385	0,3381	0.6
RANDR/2000	2,30401	0,4756	0,3922	0,3801	15.1
+ GREEDY	2,28571	0,3913	0,3385	0,3466	0.5
RANDC/2000	3,38613	0,4107	0,3385	0,3381	16.2
+ GREEDY	Same	Same	Same	Same	0.4
CONTINUOUS OPTIMUM $\#x_{ij}=1/\#x_{ij}\neq 0$	5760.4 3/21	3374.7 9/15	3338.2 10/14	3318.6 11/13	5.0
UNCONSTRAINED OPTIMUM	0,3284	0,3284	0,3284	0,3284	

TABLE III

TEST RESULTS FOR PROBLEM 2

No. Variables 50(5 x 10)	No. Resources 1				Seed (RANDU) 1122334455
	u, Z_{min} obtained for:				Avg. CPU Time (sec.) per b_{ik}
Methods/Trials	$b_{ik}=$ 310	$b_{ik}=$ 470	$b_{ik}=$ 550	$b_{ik}=$ 630	
RANDC/30	3,30706	1,11777	0,2844	0,2482	0.2
+ GREEDY	Same	1,11759	0,2810	Same	0.4
VAMC	3,30531	2,20580	0,2996	0,2482	
VAMI/11	3,30631	1,12194	0,2844	0,2482	0.2
+ GREEDY	Same	Same	0,2810	Same	0.4
LPMAX/10		2,22376	0,2844	0,2842	3.6
+ GREEDY		1,12862	0,2810	0,2482	0.4
GREEDY	3,30531	1,11759	1,11618	1,11759	0.8
RANDR/2000	3,30531	1,11777	0,2844	0,2482	14.3
+ GREEDY	Same	1,11759	0,2810	Same	0.4
RANDC/2000	3,30531	1,11777	0,2844	0,2482	16.2
+ GREEDY	Same	1,11759	0,2810	Same	0.4
CONTINUOUS OPTIMUM $\#x_{ij}=1/\#x_{ij}\neq 0$	Infeasible	2711.5 6/14	2342.0 7/13	2228.3 8/12	3.2
UNCONSTRAINED OPTIMUM	0,2370	0,2183	0,2175	0,2175	

TABLE IV

TEST RESULTS FOR PROBLEM 3

No. Variables 49(7 x 7)	No. Resources 3 u, Z_{min} obtained for:				Seed (RANDU) 1001
Methods/Trials	$b_{ik}=$ 510	$b_{ik}=$ 720	$b_{ik}=$ 1350	$b_{ik}=$ 1980	Avg. CPU Time (sec.) per b_{ik}
RANDC/30	2,15148	1,9309	0,1344	0,1146	0.2
+ GREEDY	Same	1,8541	Same	Same	
VAMC	2,15148	1,9313	0,1344	0,1146	--
VAMI/11	2,15148	1,9313	0,1344	0,1146	0.2
+ GREEDY	Same	1,8615	Same	Same	0.4
LPMAX/10		2,15861	0,1344	0,1146	3.7
+ GREEDY		1,8927	Same	Same	0.5
GREEDY	2,15148	2,15152	0,1344	0,1157	0.5
RANDR/2000	2,15148	1,9309	0,1355	0,1463	13.1
+ GREEDY	Same	1,8541	0,1344	0,1146	0.4
RANDC/2000	2,15148	1,9309	0,1344	0,1146	16.5
+ GREEDY	Same	1,8541	Same	Same	0.4
CONTINUOUS OPTIMUM $\#x_{ij}=1/\#x_{ij}\neq 0$	Infeasible	2649.5 5/9	1195.4 6/8	1131.3 6/8	3.5
UNCONSTRAINED OPTIMUM	0,2677	0,2563	0,1120	0,1120	

TABLE V

TEST RESULTS FOR PROBLEM 4

No. Variables 48 (6 x 8)	No. Resources 1 u,Z_{min} obtained for:				Seed (RANDU) 3001 Avg. CPU Time (sec.) per b_{ik}
Methods/Trials	$b_{ik}=$ 580	$b_{ik}=$ 740	$b_{ik}=$ 1220	$b_{ik}=$ 1380	
RANDC/30	1,10246	0,2503	0,1439	0,1439	0.2
+ GREEDY	Same	Same	Same	Same	0.3
VAMC	1,10246	0,2503	0,1439	0,1439	--
VAMI/11	1,10246	0,2503	0,1439	0,1439	0.2
+ GREEDY	Same	Same	Same	Same	0.3
LPMAX/10		1,9963	0,1494	0,1439	3.3
+ GREEDY		0,2503	0,1439	Same	0.4
GREEDY	1,10246	0,2503	0,1439	0,1439	0.4
RANDR/2000	1,10246	0,2503	0,1494	0,2048	8.9
+ GREEDY	Same	Same	0,1439	0,1508	0.4
RANDC/2000	1,10246	0,2503	0,1439	0,1439	13.8
+ GREEDY	Same	Same	Same	Same	0.3
CONTINUOUS OPTIMUM $\#x_{ij}=1/\#x_{ij}\neq 0$	Infeasible	2499.3 7/9	1273.7 7/9	1218.1 7/9	3.1
UNCONSTRAINED OPTIMUM	1,9914	0,2489	0,1137	0,1137	

TABLE VI

TEST RESULTS FOR PROBLEM 5

No. Variables 48(8 x 6)	No. Resources 3 — u,Z_{min} obtained for:		Seed (RANDU) 1357
Methods/Trials	b_{ik}= 790	b_{ik}= 870	Avg. CPU Time (sec.) per b_{ik}
RANDC/30	0,1963	0,1635	0.2
+ GREEDY	Same	Same	0.5
VAMC	0,1963	0,1635	
VAMI/11	0,1963	0,1635	0.2
+ GREEDY	Same	Same	0.5
LPMAX/10	0,1963	0,1635	3.9
+ GREEDY	Same	Same	0.5
GREEDY	0,1963	0,1635	0.5
RANDR/2000	0,1963	0,1635	15.1
+ GREEDY	Same	Same	
RANDC/2000	0,1963	0,1635	21.7
+ GREEDY	Same	Same	0.5
CONTINUOUS OPTIMUM	1962.2	1554.9	3.7
$\#x_{ij}=1/\#x_{ij}\neq0$	5/7	4/8	
UNCONSTRAINED OPTIMUM	0,1958	0,1499	

TABLE VII

TEST RESULTS FOR PROBLEM 6

No. Variables 48 (3 x 16)	No. Resources 4 u,Z_{min} obtained for:				Seed (RANDU) 13579 Avg. CPU Time (sec.) per b_{ik}
Methods/Trials	b_{ik}= 2700	b_{ik}= 3300	b_{ik}= 3600	b_{ik}= 3900	
RANDC/30	2,36291	1,20996	0,5121	0,5121	0.3
+ GREEDY	2,35202	Same	Same	Same	0.6
VAMC	3,51799	0,5624	0,5121	0,5121	--
VAMI/11	3,51777	0,5624	0,5121	0,5121	0.9
+ GREEDY	2,36209	0,5454	Same	Same	0.6
LPMAX/10		1,20996	0,5121	0,5121	4.4
+ GREEDY		Same	Same	Same	0.6
GREEDY	3,50603	0,6001	0,5154	0.5121	0.6
RANDR/2000	2,35921	0,6263	0,5727	0,5313	23.9
+ GREEDY	Same	0,5361	0,5295	0,5163	0.6
RANDC/2000	2,35651	0,5945	0,5121	0,5121	24.7
+ GREEDY	1,20305	Same	Same	Same	0.6
CONTINUOUS OPTIMUM $\#x_{ij}=1/\#x_{ij}\neq 0$	Infeasible	5206.9 14/18	5084.1 15/17	5027.7 15/17	4.1
UNCONSTRAINED OPTIMUM	0,4989	0,4989	0,4989	0,4989	

TABLE VIII

TEST RESULTS FOR PROBLEM 7

No. Variables 1000 (10 x 100)	No. Resources 2 u,Z_{min} obtained for:				Seed (RANDU) 1007
Methods/Trials	b_{ik}= 4140	b_{ik}= 4900	b_{ik}= 5660	b_{ik}= 7180	Avg. CPU Time (sec.) per b_{ik}
RANDC/500	3,320139	0,11867	0,9694	0,8856	30.8
VAMC	7,714355	0,10979	0,9466	0,8856	30.1
VAMI/11	0,13800	0,10148	0,9320	0,8856	
LPMAX/10	1,113622	0,12156	0,9388	0,8856	63.8
RANDR/2000	1,145341	0,422101	0,41651	0,40820	123.2
RANDC/2000	2,217867	0,11715	0,9551	0,8856	205.0
CONTINUOUS OPTIMUM $\#x_{ij}=1/\#x_{ij}\neq 0$	12968.3	9916.5	9272.4	8854.3	60.4
UNCONSTRAINED OPTIMUM	0,8829	0,8829	0,8829	0,8829	

TABLE IX

TEST RESULTS FOR PROBLEM 8

No. Variables 1000 (20 x 50)	No. Resources 3 u, Z_{min} obtained for:				Seed (RANDU) 121341
Methods/Trials	$b_{ik}=$ 1440	$b_{ik}=$ 1770	$b_{ik}=$ 2430	$b_{ik}=$ 2760	Avg. CPU Time (sec.) per b_{ik}
RANDC/125	0,7184	0,4445	0,3075	0,2914	12.2
VAMC	0,7149	0,3874	0,2958	0,2880	--
VAMI/11	0,5587	0,3632	0,2958	0,2880	12.0
LPMAX/10	0,7027	0,3910	0,3115	0,2880	82.3
RANDR/2000	0,20501	0,19012	0,17401	0,17936	83.8
RANDC/2000	0,6107	0,4138	0,3010	0,2880	198.0
CONTINUOUS OPTIMUM $\#x_{ij}=1/\#x_{ij}\neq 0$	5203.3 43/57	3444.6 45/55	2902.2 48/52	2876.5 48/52	75.7
UNCONSTRAINED OPTIMUM	0,2761	0,2761	0,2761	0,2761	

TABLE X

TEST RESULTS FOR PROBLEM 9

No. Variables 992 (31 x 32)	No. Resources 1 u, Z_{min} obtained for:				Seed (RANDU) 380225
Methods/Trials	b_{ik}= 100	b_{ik}= 290	b_{ik}= 860	b_{ik}= 1620	Avg. CPU Time (sec.) per b_{ik}
RANDC/30	4,137693	0,5135	0,1519	0,1088	5.0
VAMC	3,106635	0,4988	0,1572	0,1088	--
VAMI/11	3,106635	0,4988	0,1556	0,1088	5.1
LPMAX/10		0,5217	0,1581	0,1088	60.6
RANDR/500	3,109018	0,12591	0,11693	0,10823	19.8
RANDC/500	3,106635	0,5021	0,1490	0,1088	83.5
CONTINUOUS OPTIMUM $\#x_{ij}=1/\#x_{ij}\neq 0$	Infeasible	4597.2 28/36	1468.7 31/33	1086.2 31/33	55.4
UNCONSTRAINED OPTIMUM	3,104636	0,4281	0,1230	0,1070	

TABLE XI

TEST RESULTS FOR PROBLEM 10

No. Variables 1000 (40 x 25)	No. Resources 3 u,Z_{min} obtained for:				Seed (RANDU) 50359
Methods/Trials	$b_{ik}=$ 350	$b_{ik}=$ 550	$b_{ik}=$ 950	$b_{ik}=$ 1550	Avg. CPU Time (sec.) per b_{ik}
RANDC/30	5,132222	0,5196	0,1014	0,732	7.1
VAMC	5,132222	0,5196	0,1002	0,698	--
VAMI/11	5,132222	0,5196	0,1002	0,698	6.7
LPMAX/10	--	0,5196	0,1019	0,698	91.9
RANDR/500	5,132313	0,8583	0,7329	0,7781	26.8
RANDC/500	5,132222	0,5196	0,1002	0,698	122.0
CONTINUOUS OPTIMUM $\#x_{ij}=1/\#x_{ij}\neq0$	Infeasible	5195.4 24/26	972.6 22/28	697.3 24/26	84.8
UNCONSTRAINED OPTIMUM	3,83054	0,4995	0,867	0,648	

TABLE XII

TEST RESULTS FOR PROBLEM 11

No. Variables 1000 (50 x 20)	No. Resources 4 — u,Z_{min} obtained for:				Seed (RANDU) 121567
Methods/Trials	b_{ik}= 520	b_{ik}= 840	b_{ik}= 1320	b_{ik}= 1640	Avg. CPU Time (sec.) per b_{ik}
RANDC/45	0,6020	0,891	0,372	0,366	7.2
VAMC	0,6020	0,891	0,372	0,366	--
VAMI/11	0,6020	0,891	0,372	0,366	7.1
LPMAX/10	1,25721	0,891	0,372	0,366	99.5
RANDR/500	0,7515	0,6139	0,6823	0,6717	16.7
RANDC/500	0,6020	0,891	0,372	0,366	78.9
CONTINUOUS OPTIMUM $\#x_{ij}=1/\#x_{ij}\neq 0$	5903.2 15/25	887.6 18/22	371.3 19/21	365.8 19/21	94.3
UNCONSTRAINED OPTIMUM	0,5478	0,809	0,362	0,362	

performance characteristics exhibited by individual methods are supported by results from selected runs.

Tables II Through XII

The values tabulated are in the form u, Z_{min} with Z_{min} including the costs charged for the number of unassigned tasks (u). Several things stand out:

(1) VAMI and RANDC (allowed the same amount of time as VAMI) consistently gave better objective values and used less CPU time than LPMAX or GREEDY.

(2) Objective values found by all methods usually have a high probability of being in the uppermost percentile of all possible solutions, based on the value achieved by RANDR for N = 2000 or N = 500.

(3) All methods usually obtained feasible solutions, given existence, and near-optimal solutions, given bounds. GREEDY could not improve many of the solutions found by the construction heuristics.

(4) Problem characteristics (size, shape, number of resources, tightness of constraints) have great effect on absolute and relative performance of methods. RANDC gives much better results with small problems than with large, for example.

Specific measures of performance will be discussed in more detail below, based on results from all test runs.

Pairwise Comparison on Solution Values

Table XIII shows the outcomes of comparing each pair of methods in terms of the objective function values achieved. The data below the diagonal are for large problems, where mn = 0(1000). Each entry in Table XIII consists of three numbers in the form T, L, U:

T: Total number of runs in which both methods were tested on a problem of a given size category.

L: Number of runs in which the Left-hand (row heading) method gave a better objective value.

U: Number of runs in which the Upper (column heading) method gave a better objective value.

For example, the entry 47, 17, 2 at the intersection of the row labeled VAMI and the column labeled LPMAX means that both VAMI and LPMAX were tested on 47 runs of small problems, with VAMI obtaining a better solution than LPMAX in 17 runs, and LPMAX giving a better value than VAMI twice. Obviously the two methods gave the same solution 28 times.

TABLE XIII

OUTCOMES OF PAIRWISE COMPARISONS OF METHODS ON SOLUTION VALUES OBTAINED

			SMALL PROBLEMS		
	RANDC	VAMC	VAMI	LPMAX	GREEDY
RANDC	--	(66,18,7)	66,14,8	47,20,0	(53,18,7)
VAMC	41,20,4	--	66,* ,15	(47,16,6)	53,16,8)
VAMI	41,20,4	41,14,*	--	47,17,2	53,16,7
LPMAX	41,14,14	(41,6,16)	41,0,24	--	47,9,17
	LARGE PROBLEMS				

In interpreting Table XIII, it should be noted that VAMC is the same as VAMI with Q = 0, so VAMC can never give a better solution than VAMI. This is indicated by asterisks where appropriate. Further, GREEDY was not used on large problems, as stated earlier.

Nonparametric sign tests were performed on the data of Table XIII. Each underlined entry indicates an observed significance level (OSL) of 0.05 or less. Parentheses denote an OSL between 0.05 and 0.10. No sign test was performed for the VAMC/VAMI comparison, since VAMI will always perform at least as well as VAMC.

From Table XIII, it is clear that VAMI is best for large problems using this criterion. For small problems, RANDC seems to be best, although it does not differ significantly from VAMI.

A weakness in this comparison technique is that solution values and differences between solutions are not quantified. This makes RANDC and LPMAX seem to perform equally on large problems. In fact, when LPMAX is better than RANDC, it is usually only a little better, but when it is worse, it is often much worse. This can be seen in Tables II through XII.

Best Heuristic Solution

Table XIV shows how often each method gave the best objective value, including ties. Each entry in Table XIV is in the form B/T (P%):

B: Number of Best solutions (or ties) produced by a given method.

T: Total number of runs involving all methods.

P: Percentage of T represented by B.

For example, the entry "36/47 (77%)" for RANDC on a small problem means that in 36 of the 47 runs in which all methods were involved, RANDC gave a solution at least as good as the best obtained by any other method.

Table XIV can be seen as a table of estimates of the probability that one method will outperform or equal any other in terms of the objective value produced. Again, VAMI stands out for large problems, while the distinction between methods is not at all clear for small problems. The best and worst methods (RANDC and LPMAX/GREEDY) for small problems differ by only 22 percent. This is less than the 24 percent difference between the <u>two best</u> methods for large problems (VAMI and VAMC), whose outcomes are not even independent of each other.

TABLE XIV

FREQUENCIES AND PERCENTAGES FOR BEST SOLUTIONS FROM INDIVIDUAL METHODS

	Small Problems	Large Problems
RANDC	36/47 (77%)	14/41 (34%)
VAMC	32/47 (68%)	29/41 (71%)
VAMI	34/47 (72%)	39/41 (95%)
LPMAX	26/47 (55%)	14/41 (34%)
GREEDY	26/47 (55%)	not tested

VAMC appears to have performed well, since it found as good a solution as any other method for more than two-thirds of both large and small problems. However, many of its less-than-best solutions were very poor indeed, especially when constraints were tight (see Tables II, III, and VIII).

From the preceding paragraph, it is clear that the criterion of Table XIV, like that of Table XIII, has the shortcoming of not considering solutions quantitatively. How should quantitative results be reported?

Accuracy/Optimality

Tabulating raw solution values as in Tables I through XII can give some quantitative indication of relative performance. However, the objective values have more meaning if they can be related to the optimal solutions. This is done by comparing them to the optimum, by bounding their percentage difference from the optimum, and by determining some minimum probability of their being in some very small best fraction of all solutions. Three tabulations are used to do this:

(1) Runs finding a known or suspected optimum (Table XV).

(2) Runs within certain percentages of a bound on the optimum (Table XVI).

(3) Runs giving solutions very likely to be in a very small best quantile of all solutions (Table XVII).

In some runs, the optimum was either known or suspected, usually based on comparison of the continuous optimum to the best heuristic solution. Examples of this can be seen in Table V (suspected optimum for b_{ik} = 740 of 2503 where the continuous optimum was 2499.3) and Table VI (known optimum for b_{ik} = 790 of 1963, the next integer above the continuous optimum of 1962.2).

The entries in Table XV are in the form F/T (P%):

F: Number of known or suspected optima Found by a given method.

T: Total number of problems attempted by the method where the optimum was known or suspected.

P: Percentage of T represented by F.

VAMI and VAMC gave identical results for this criterion, so their entries are combined in Table XV.

Clearly, the best results from this point of view were produced by VAMI/VAMC in finding every known or suspected optimum. This does not

TABLE XV

FREQUENCIES AND PERCENTAGES OF METHODS FINDING KNOWN OR SUSPECTED OPTIMAL SOLUTION

Method	Problem Size	Known Optimum	Suspected Optimum	Combined Results
RANDC	Large	14/17 (82%)	3/10 (30%)	17/27 (63%)
	Small	9/9 (100%)	18/18 (100%)	27/27 (100%)
VAMI/	Large	17/17 (100%)	10/10 (100%)	27/27 (100%)
VAMC	Small	9/9 (100%)	18/18 (100%)	27/27 (100%)
LPMAX	Large	12/12 (100%)	4/7 (57%)	16/19 (84%)
	Small	2/6 (33%)	12/12 (100%)	14/18 (78%)
GREEDY	Small	7/7 (100%)	9/14 (64%)	16/21 (76%)

TABLE XVI

CUMULATIVE FREQUENCIES AND PERCENTAGES OF RUNS WITHIN VARIOUS TOLERANCES OF THE BEST BOUND ON THE OPTIMUM

	Method				
	RANDC	VAMC	VAMI	LPMAX	GREEDY
Large Problems					
Total Runs	39	39	39	39	None
2%	16 (41%)	21 (54%)	29 (74%)	15 (38%)	--
5%	25 (64%)	25 (64%)	33 (85%)	16 (41%)	--
10%	27 (69%)	35 (90%)	37 (95%)	23 (58%)	--
15%	29 (74%)	37 (95%)	37 (95%)	27 (69%)	--
Small Problems					
Total Runs	45	45	45	32	35
2%	22 (49%)	22 (49%)	22 (49%)	12 (38%)	14 (40%)
5%	27 (60%)	27 (60%)	27 (60%)	17 (53%)	21 (60%)
10%	27 (60%)	30 (67%)	30 (67%)	17 (53%)	21 (60%)
15%	39 (89%)	42 (93%)	42 (93%)	21 (66%)	28 (80%)

TABLE XVII

FREQUENCIES AND PERCENTAGES OF SOLUTIONS EQUALING OR EXCEEDING VALUES OBTAINED BY RANDR OR RANDC WITH LARGE VALUES OF N

Problem Size	Evaluation Standard, N	RANDC	VAMC	VAMI	LPMAX	GREEDY
Large	RANDR, 2000	15/17 (88%)	15/17 (88%)	17/17 (100%)	17/17 (100%)	--
	RANDR, 500	22/24 (92%)	24/24 (100%)	24/24 (100%)	18/24 (75%)	--
	RANDC, 2000	2/17 (12%)	13/17 (76%)	17/17 (100%)	11/17 (65%)	--
	RANDC, 500	14/24 (58%)	22/24 (92%)	22/24 (92%)	12/24 (50%)	--
Small	RANDR, 2000	54/66 (82%)	51/66 (77%)	57/66 (86%)	32/47 (68%)	36/53 (68%)
	RANDC, 2000	54/66 (82%)	48/66 (73%)	51/66 (77%)	23/47 (49%)	34/53 (64%)

mean that VAMI/VAMC will always find the optimum. Problems for which the optimum is easily found are not at all typical, and VAMI/VAMC, as can be seen under b_{ik} = 1370 in Table II, "must occasionally produce arbitrarily bad approximations," as noted in Chapter III, page 33.

Probably the most meaningful statistics for judging the relative capabilities of the methods in finding good solutions are given in Table XIV. For problems where feasible solutions were known to exist (usually because they were found by some heuristic), frequencies and percentages are tabulated to show how often each method produced a solution that was within (a) 2 percent, (b) 5 percent, (c) 10 percent, and (d) 15 percent of the greatest lower bound (usually the continuous optimum) on the optimal solution.

The frequencies and percentages in Table XIV are cumulative. For example, RANDC gave a solution within 20 percent of the best bound on 16 (41 percent) of 39 large problems, while 25 (64 percent) of 39 RANDC solutions were within 5 percent of the bound. This, of course, implies that 9 solutions from RANDC were between 2 percent and 5 percent greater than the bound.

The best results for large problems were again produced by VAMI, where 95 percent of all solutions to problems known to possess a feasible solution were within 10 percent of the optimal solution, and 85 percent were within 5 percent. For small problems, VAMI, VAMC, and RANDC did not differ significantly, although it should be noted that RANDC ranks behind VAMI/VAMC according to this criterion, which is the reverse of what was reported in Tables XIII and XIV. Again, this happens because solutions from VAMI that were superior to those from RANDC were sometimes very superior, but the reverse was seldom true. RANDC and

VAMI/VAMC also produced many identical solutions, especially with fairly loose constraints.

Another view of optimality is the statistical approach of Table XVII. McRoberts [20] pointed out that it is easy to calculate the number (N) of equally likely solutions that must be randomly generated to have a specified confidence (C) that the best solution obtained will be within a given best fraction (P) of all solutions:

$$1 - C \doteq (1 - P)^N \qquad \text{so} \qquad N \doteq \frac{\log (1 - C)}{\log (1 - P)}$$

N must thus be 459 or more to be 99 percent confident of obtaining a solution from the 99th percentile (P = .01) of all solutions, and if C = .997 and P = .003, N = 1944. Tests were run using RANDR with N = 500 or N = 2000.

Besides being convenient round numbers, 500 and 2000 are conservative, because they could actually be associated with larger values of C and/or smaller values of P. Also, RANDR itself is conservatively biased because it will not assign an infeasible agent to any task, which makes most good solutions much more probable than most bad solutions.

Unfortunately, there are m^n solutions to each problem. In a 50-variable problem, m^n is of the order of 10^5 to 10^7, so there are hundreds or thousands of solutions within the upper fraction P of all solutions, even when P = .003. As can be seen from Tables II - VII, RANDR with N = 2000 ($C \doteq .997$, $P \doteq .003$) produces solutions that are usually worse than those found by the methods being evaluated. The situation deteriorates dramatically for larger problems. If $mn \doteq 1000$, m^n will be of the order of 10^{50} or 10^{100}, so enormous numbers of solutions would be implied by the smallest fraction of all solutions associated with reasonable values of C and P. The starkly inferior solutions to large problems produced by RANDR with N = 500 or 2000 are evident in Tables VIII - XII.

RANDC, however, produces good solutions even for small values of N, as has been seen. RANDC is much more biased toward good solutions than RANDR, so running RANDC with a large N should give great confidence of obtaining one of the very best solutions.

A drawback of RANDC, especially as an evaluation tool, is that it is biased <u>against</u> good solutions in some highly-constrained problems. The best coverage for such problems is often achieved by assigning many tasks to agents that are expensive, but especially resource-efficient. It is possible to devise examples where RANDC would <u>never</u> find an obvious optimum, because of its rule of assigning each task to the cheapest available agent. It is believed, however, that actual problems will rarely exhibit this difficulty.

Table XVII is useful despite the difficulties set out in the preceding paragraphs. It gives strong intuitive support to the contention that some methods are extremely likely to find one of the few very best solutions, even though the likelihood and quantile cannot be determined.

The entries in Table XVII are in the form N/T (P%):

N: <u>N</u>umber of runs giving a solution at least as good as that found by the evaluation standard.

T: <u>T</u>otal number of runs compared to the evaluation standard.

P: N expressed as a <u>P</u>ercentage of T.

VAMI again stands out for large problems. RANDC does well only for small problems, although it is certainly not "fair" to evaluate it against an "advantaged" version of itself. VAMC does very well, considering that it requires so little time.

Although the objective value is severely penalized when the solution does not cover all tasks, there remains a need to test the ability of each method to find solutions covering as many tasks as possible.

Coverage (Feasibility)

Table XVIII is intended to estimate the probability that a given method will find a solution covering all tasks, provided a continuous solution exists. The entries are frequencies and percentages from among 45 small problems and 39 large ones possessing continuous solutions. All methods find feasible solutions fairly reliably, but VAMI, with 100 percent success for both problem sizes, is clearly superior.

TABLE XVIII

FREQUENCIES AND PERCENTAGES OF FINDING FEASIBLE SOLUTION WHEN CONTINUOUS SOLUTION EXISTED

Problem Size	Number of Runs	RANDC	VAMC	VAMI	LPMAX	GREEDY
Large	39	37 (95%)	37 (95%)	39 (100%)	33 (85%)	--
Small	45	42 (93%)	42 (93%)	45 (100%)	36 (80%)	39 (87%)

Up to now, the solution itself has been the only information from the test runs to be investigated. The computer time and storage required to produce these solutions also need to be considered, especially since this research was motivated by a need to conserve these resouces.

Response Time

The computers used for almost all this research were large, fast IBM 370-series systems. The CPU time required by these machines is only about a fourth of that needed by more typical equipment. Tables II through XII show the amount of CPU time required for one run using each method. For large problems, RANDC and VAMI/VAMC are clearly the only methods that can be counted on to provide response times suitable for conversational use on most computer systems. CPU times listed for LPMAX include the time required by MPS, but they do not include time for interfacing the three-step program sequence (MPSGEN, MPS, LPMAX), which admittedly can be refined beyond what was done here, but would always be costly. Requirements for data interface also plague LPMAX. RANDC and VAMI/VAMC generate all solutions in main storage, so CPU time is the only determinant of response time. However, the linear programming formulation of a large generalized assignment problem (mn variables; mp + n constraints; mn upper bounds) forces MPS (or whatever) to use peripheral storage, which lengthens response time considerably.

Execution (CPU) time was observed to be affected by problem size (mn), the number of resources (p), and problem shape (m/n). It was impractical to make the number of runs necessary to investigate this thoroughly, so it was decided to place the most emphasis on effects that were unexpected or otherwise especially interesting.

Problem Size (mn)

During preliminary testing, results were at first confusing until it was noted that execution time was affected by the shape as well as the size of the problems. Then, by holding m/n relatively constant while varying mn, results were obtained that were quite as expected:

(1) For the construction heuristics, execution time was a linear function of mn. This is not surprising, since for each of n tasks, a maximum of m agents are considered by these methods, without any combinatorial complications between tasks. The MPS overhead for LPMAX also contributed linearly to execution time as mn increased, which is normal for linear programming algorithms.

(2) With improvement heuristics, however, execution time was a quadratic function of mn. This is to be expected since they consider assignments in pairs, and there are $O(m^2n^2)$ possible pairs.

Number of Resources (p)

This was held constant (usually at 2, 3, or 4) while investigating the effect of problem size. When p was varied under constant problem dimensions, effects were observed that were quite as expected:

(1) Execution times of RANDR, RANDC, and VAMI/VAMC did not change much. The time spent checking resources is small compared to the time spent seeking minima in columns, calculating penalties, etc.

(2) LPMAX (actually the MPS phase) was strongly affected. Adding one to p increases the number of constraints by m. This means that the CPU time required by an improvement heuristic will therefore be multiplied by a factor of about p to become $O(m^2n^2p)$.

Problem Shape (m/n)

The most interesting results were produced by varying this factor. Unlike the factors discussed above, problem shape affects the fast methods RANDC and VAMI/VAMC, but it has little effect on other methods. The most interesting thing about problem shape is that it affects execution time of RANDC in a way opposite to the effect on VAMI. These opposite effects are graphed in Figures 11 and 12.

Why is VAMI slower for "wide" or "tall" problems than for "square" ones? For "wide" problems, recalculation of penalties must be done for more tasks than with other shapes. As problems become "tall," the search for the two smallest elements in a column begins to require more time.

The reason why RANDC is slowest for "square" problems is less obvious. RANDC uses time for choosing the next task to optimize, and for finding the cheapest available agent. Fewer tasks must be chosen in a "tall" problem, but finding the cheapest agent takes less time in a "wide" problem. Apparently the combined effect is worst for "square" problems.

Storage Requirements

All the methods were well within the capacity of a fairly small computer, except LPMAX. The MPS package requires far more storage (about 2,000,000 bits) than a user-written routine for the continuous solution, but the latter would still be very large and costly to develop.

As will be seen in Chapter V, it is possible to sharply reduce the amount of storage used by packing two or more numbers into the space normally used for one, at a slight cost in execution time. However, most users will have sufficient storage available to avoid packing. Under the assumption that each numeric value uses one "word" of storage, the various methods require array storage as follows:

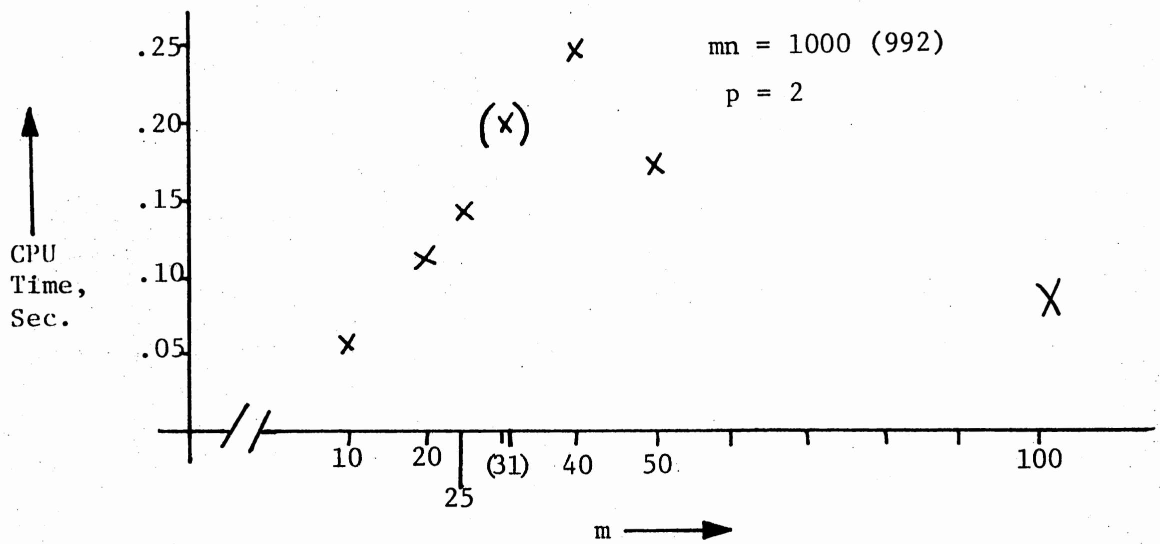

Figure 11. Effect of Shape on Average CPU Time Required for One RANDC Solution

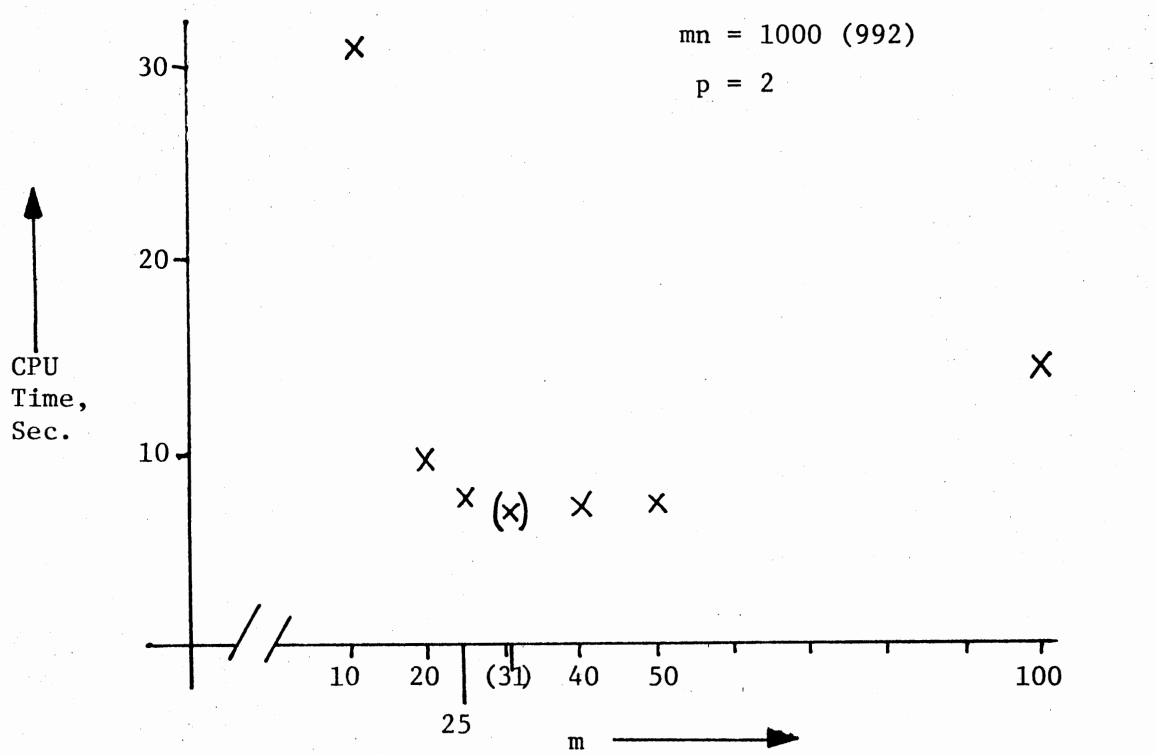

Figure 12. Effect of Shape on CPU Time Required for 11 VAMI Solutions

	Storage Words	Increment
RANDR	$(p + 1)(n + 1)(m) + n$	--
RANDC, LPMAX	$(p + 1)(n + 1)(m) + 2n$	n
VAMC	$(p + 1)(n + 1)(m) + 3n$	n
VAMI	$(p + 3)mn + (p + 1)m + 3n$	2mn

The storage requirement for LPMAX does not include the overhead of MPS. The above requirements can be halved with many computers by using half-word integer storage.

The methods require storage for the program logic, also. This will differ between computers, but the implementations in Appendix A used program storage (for subroutine SOLVER, less array storage) in the following approximate amounts (in bits):

RANDR	24000
RANDC	25000
LPMAX	26000
GREEDY/CRAFTY	35000
VAMI/VAMC	40000

For a fairly large problem ($m = 100$, $n = 15$, $p = 4$), it should be possible to implement VAMI in 200,000 to 500,000 bits of storage, depending on word length, which is well within the capacity of almost any computer. RANDC would require slightly more than half as much storage as VAMI.

Summary

It is clear that RANDC and VAMI/VAMC are superior to the other methods, but the conclusions and recommendations to be drawn from the results presented in this chapter will be developed in Chapter VI, after Chapter V describes two implementations.

CHAPTER V

TWO IMPLEMENTATIONS

Introduction

The methods described in Chapter III must be modified to fit most applications. This is due to constraints of the solution environment as well as complications of the problem itself. This chapter describes ways of dealing with (a) an environmental constraint (limited computer resources), and (b) a complicated problem (the artillery problem of Figures 3 through 5).

Limited Computer Resources

Background

Until recently, the size and cost of computers made them impractical for many on-the-job applications. Almost overnight, miniaturized equipment that is startingly sophisticated has become available at about the same cost as an electric typewriter or a forklift truck. Computers can now be located in industrial environments where assignment problems are encountered. Deciding which machine or worker does which job need no longer be a haphazard process. There is great potential here for improved productivity. Most of the methods described in this dissertation can be used with microcomputers, especially RANDC and VAMI. This section, adapted from Thibault et al. [32], shows how machines are assigned to jobs in an operational situation. The discussion will be in terms of "machines" and "jobs" instead of agents and tasks. Two resources will be considered: "material" and "time."

Simplifications in Methods

The logic of RANDC remains essentially unchanged. VAMI considers only five "Q" values (0, 0.1, 0.2, 0.4, 0.8), and calculates penalties only for the two lowest-cost machines for each job, with no penalties being recalculated during the solution process.

Simplifications in Problem Data

Besides allowing for only two resources, it is assumed that costs and resource requirements can be predefined as part of the program, since the set of machines and the set of possible jobs, along with the corresponding cost and resource data, usually do not change often. The items that change frequently are:

(1) Which machines or jobs are to be considered from the set of those possible, and

(2) The available supplies of material and time.

This is the only information the user must specify. (RANDC also requires a random number seed and the sample size.)

The user-specified subsets of cost/resource data are moved into the "northwest corner" of the corresponding main arrays before the solution phase of the program begins.

Saving Time and Storage

The use of predefined data allows the further simplification of using presorted and pre-indexed costs. The machine indexes and the costs are packed (to save storage) in the form ccccii, where cccc indicates the cost and ii the index. These indexed costs are presorted by the user into descending order for each task. They are part of the source program. This simplifies and speeds up both RANDC and VAMI by making it very easy to find the "next cheapest machine" for a given job.

Similarly, requirements for material and time are packed (mmmttt), but they are entered for each job in the order in which machines are numbers.

Of course, using predefined data may not always be appropriate. Programs can easily be coded to read costs and resource requirements, but they are rather error-prone and tedious to use for problems of realistic size.

For example, suppose job two's costs and resource requirements were as shown in Table XIX. The data for job 2 would be entered in the source program as follows. (Note that leading zeros are not necessary and that ** is entered as a cost of 9999 with material and time requirement of zero.)

```
1080 REM JOB 2
1090 DATA 4804,5060,6303,6805,8701,9607,999902
1100 DATA 16016,0,28062,38089,12069,96019,50033
```

The packing techniques require only six-digit precision. (Costs and resource requirements must be scaled if necessary.)

Programs

Appendix C contains sample programs written in a subset of BASIC that will work on most microcomputers. Identical code (through statement

1970) is used in all programs to initialize and acquire all data except the special items needed by RANDC. The solution routines are as alike as possible, given their differing logic.

TABLE XIX

EXAMPLE OF DATA TO BE ENTERED IN MICROCOMPUTER PROGRAM

	Machine No.						
	1	2	3	4	5	6	7
Cost	87	**	63	48	68	56	95
Material	16	**	28	38	12	95	50
Time	16	**	62	89	69	19	33

**Means that the job cannot be done on the corresponding machine.

Program Outputs

As can be seen from the examples in Appendix D, both programs give the user an opportunity after completion (NEW RUN?) to either restart completely (YES) or try some other set of resource supplies (RHS -- for "Right-Hand-Side"). Before completion, RANDC asks if the user wants additional trials to be made (MORE TRIALS?). If YES is input, RANDC asks HOW MANY?

The program output is otherwise largely self-explanatory, except for the following notes:

(1) The total cost of a solution will be printed with one asterisk to the left of the word COST for each job remaining unassigned to any machine. This, in addition to the listing of UNASSIGNED JOBS, is designed to alert the user that a particular solution is incomplete (which may be a natural result of limited supplies of resources).

(2) Slack data are printed to help guide the user to a successful reallocation of resource supplies. However, the mathematical properties of this problem can make reallocation a tricky process.

(3) Both programs occasionally produce duplicate solutions in an effort to provide the user with multiple alternatives.

Testing and Evaluations

RANDC and VAMI were run against 180 randomly generated problems as indicated in Table XX. RANDC was allowed ten trials to the five (one for each Q) allowed to VAMI, because it was estimated to take about twice as long to generate penalties as random numbers. Ten different sets of dimensions were used, varying from 6 x 3 to 3 x 10 to represent most problem "shapes" that would be possible in the 7 x 10 program arrays. For each set of dimensions, three different sets ("Problems 1, 2, and 3") of cost and resource coefficients were used. For each of these, tight, medium, and loose ("T, M, and L") resource supplies were tried. In Table XX, the method performing best for a given problem is denoted by "R" for RANDC and "V" for VAMI with "-" indicating a tie.

It is clear from Table XX that there is no significant difference between RANDC and VAMI for a basic problem. RANDC can be easily run for a very large number of trials to obtain a more realiable estimate of the true optimum, and it is much easier to understand and explain than VAMI. However, VAMI does have certain advantages if many complications are present, as will be seen.

TABLE XX

SUMMARY OF TEST RESULTS FOR MICROCOMPUTER IMPLEMENTATION

Dimensions	Problem 1			Problem 2			Problem 3		
	T	M	L	T	M	L	T	M	L
6x3	-	V	-	-	-	-	-	-	-
7x4	-	-	-	-	-	-	-	R	-
3x3	-	-	-	-	-	-	V	V	-
5x5	-	-	-	R	R	-	R	-	-
7x7	-	-	-	R	R	-	R	-	-
3x5	R	R	-	-	-	-	R	V	-
5x8	R	R	-	R	-	-	R	V	-
7x10	R	R	-	R	-	R	R	V	V
2x6	R	-	-	-	R	-	-	-	-
3x10	V	V	-	V	R	-	R	V	-

Complications in General

Complications make the task of optimizing a job much more difficult. However, VAMI-type penalties can still usually be calculated in a straightforward way, so the extra time they take diminishes in importance, because the process of optimizing a job may take much longer than choosing a job to optimize. Thus, requiring fewer trials than RANDC can be a big advantage. Below are ideas for dealing with specific complications, mostly adapted from the artillery problem.

Job Priorities. As suggested in Chapter II, each group of jobs of a given priority could be treated as a separate subproblem, solved in order of decreasing importance. VAMI (or RANDC, if this is the only complication) is suitable for this. Another way would be to transform the costs for job j according to priority (being careful to ensure highest costs and penalties for the most important jobs) before solving with VAMI ro RANDC.

Alternative Resources. A good example of this complication is the availability of several types of ammunition to an artillery unit. This can be handled by defining a group of multiple machines, one for each alternative resource, and assigning as usual, decrementing time for all "machines" in the group.

Shared Jobs. Sophisticated approaches are very difficult to fit into a microcomputer, but it may be possible to specify a fictitious mmachine, representing two or more others in combination, and decrementing resources for all machines involved in the assignment. This complication does not combine well with alternative resources. Such a combination might best be handled with VAMI.

Multiple Objectives. If this is the only complications, RANDC with modified objective evaluation is probably best. Otherwise, VAMI is likely to be better, if the penalty calculations are based on the multiple objectives.

Notation

BASIC severely limits variable names, so the code in Appendix C is not the same as that used in Appendix A. Table XXI establishes notational correspondence between Figure 7, Appendix A, and Appendix C for important variables with different names.

The Artillery Problem

Introduction

The basic solution approach descends directly from VAMI. Targets are optimized in a sequence based on penalties calculated from weighted combinations of costs and resource inefficiencies. Optimizing a target is a very complicated process, however. Also, extreme measures were taken to save time and storage while allowing the use of variable problem dimensions (m and n) without recompiling the program. This makes the program (Appendix E) almost indecipherable, in spite of detailed documentation with comment cards.

It is most unlikely that the program of Appendix E could be used in another application without extensive modification. The following Summary Flowchart and Narrative Outline of the Solution Routine present the procedure in a form that is easier to understand and more likely to

TABLE XXI

NOTATION IN APPENDIX C

Name in Appendix C	Contents	Appendix A	Figure 7
A	Assignment vector	XB	X
B,T	Initial resource supplies	IB	b
C5	Current objective function value	Z	Z
G1	Best objective function value yet found	MINZ	Z_{min}
L	Index of task currently being optimized	JBIG	J
M7,N7	Maximum array dimensions	-	-
P	RANDC: vector for shuffling task indices	T	
	VAMI: vector of task penalties	H	H
R	Packed resource coefficients	A	a
R9	RANDC: Random number seed	ISEED	d
U,V	(a) temporary storage for moving cost and resource data into "northwest" corners of main arrays	-	-
	(b) amounts of resources remaining	B	B
U6	Count of tasks that could not be assigned	U	u
Y	Indices of user-specified subset of agents	-	-
Z	(a) indices of user-specified subset of tasks	-	-
	(b) RANDC: number of trials (BASIC arrays and scalars can share a name)	NBIGQS	N

provide inspiration. (Appendix F contains an output example for a small problem.)

Summary Flowchart

After the "Input Phase" is complete, a "Control Routine" supervises the generation and printing of solutions by the "Solution Routine" and the "Output Routine." The overall logic of this process is given in the Summary Flowchart in Figure 13.

Table XXI refers to a "P-matrix" and "Q." These items are similar to those of the same names used in VAMI.

In interpreting Figure 13, it should be noted that the "Output Routine" is designed to produce different lists (emulating video outputs), depending on the codes passed to it by the "Control Routine" and the "Solution Routine."

Narrative Outline of the Solution Routine

I. For each target priority class:

   A. Obtain penalties for each target in priority class. Penalties depend on number of units desired (one for normal assignment, more for mixed assignment) versus number of units available.

      1. No units available: Penalty is -1 (lower than any other penalty) because nothing can be gained by making an early assignment.

      2. Number desired exceeds number available: Penalty is 500,000 + 100,000 x (shortage).

      3. Number desired equals number available: Penalty is 500,000.

      4. Number desired is one less than number available: Penalty is 100,000 + largest difference between two successive (in size) P-matrix values for target.

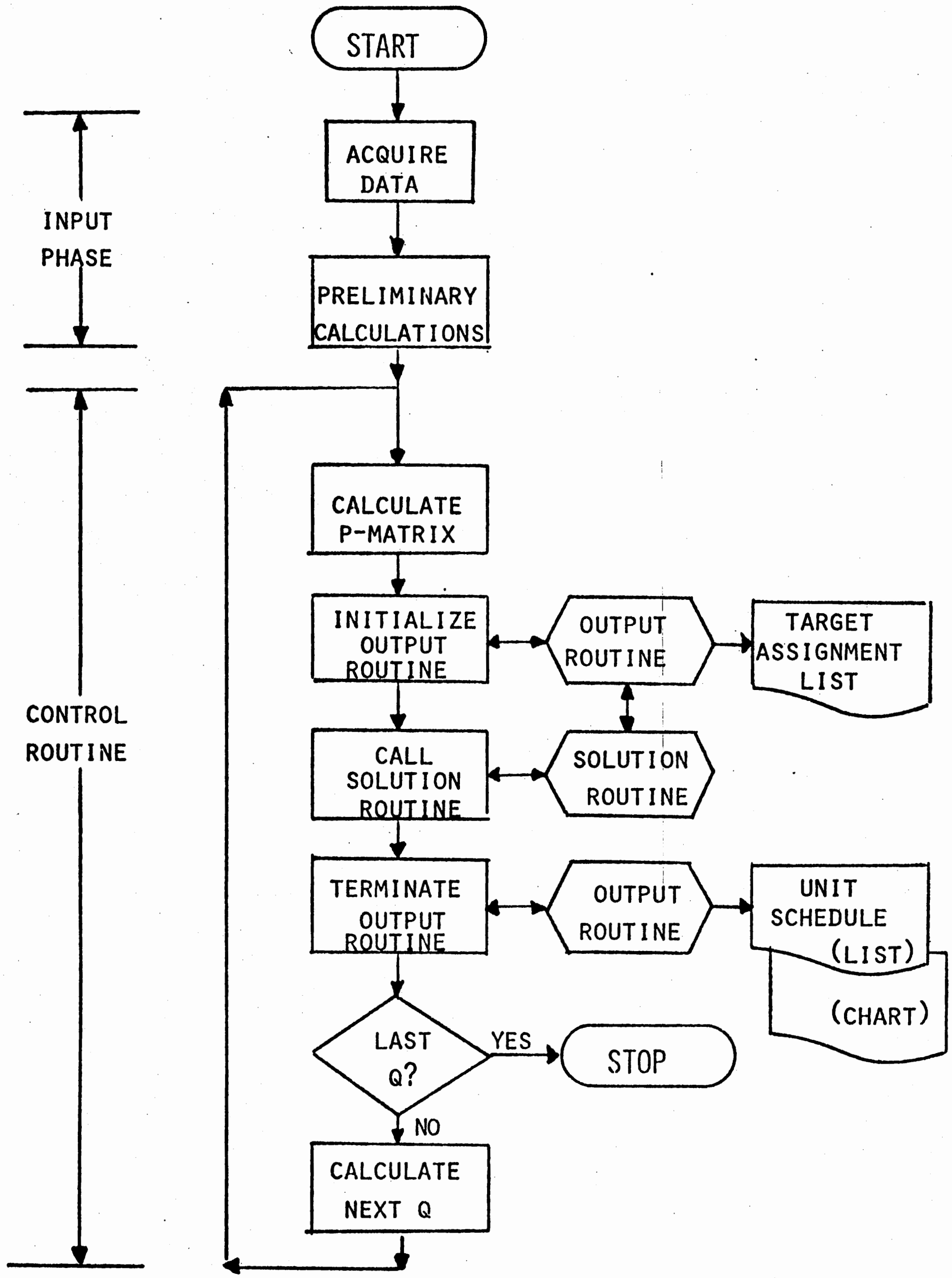

Figure 13. Summary Flowchart for Solution of Artillery Problem

5. Otherwise: Find, from each unit, the type of ammunition having the lowest P-value for this target. Penalty is the greatest difference between two such values over all units.

B. Optimize targets in order of highest-to-lowest penalties. Logic depends on whether mixed assignment and/or start/stop times are specified. (Note: "cheapest" as used below means having smallest P-value for a given target.)

1. Unmixed assignments with start/stop times specified: Make cheapest feasible assignment that fits start/stop period. If only start (or only stop) specified calculate other end of firing period as specified in 5-4a and 5-4b of Figure 5.

2. Unmixed assignments with no start or stop time: Make cheapest feasible assignment, starting as early as possible in the unit's shortest satisfactory schedule gap.

3. Mixed assignment: Calculate "Mixing Limits" as the maximum (TMASMX in Appendix E) and minimum (TMASMN) over all units of the time required for each unit to cover its "ideal share" of a mixed assignment. This "ideal share time" is the amount of time the unit would need to fire a number of shells that would be the smallest integer not exceeded by a_{ijk}/M_j (see Figure 5).

a. Both start and stop times specified:

(1) Check every unit (in order of ascending P-value for this target) to determine if the unit's ammunition supply and schedule permit it to contribute anything to the coverage of this target. If so, add it to list of prospective mixed assignment participants.

(a) If desired number (or more) of units is in list, check if coverage is complete. If it is, go assign. Otherwise, check next unit.

(b) If no more units are available and coverage is complete, go assign. If coverage is not complete, target remains unassigned.

b. Start or stop time (not both) specified:

(1) Determine the set of units able to fire at least one volley. From all such units find the maximum (AVAMAX) and minimum (AVAMIN) length of a schedule gap bounded on one end by the specified start or stop time.

(2) Try to fit mixed assignments in the following order of length: TMASMX, TMASMN, AVAMAX, AVAMIN. Exclude cases known in advance not to fit, e.g., when TMASMX is greater than AVAMAX. The procedure for trying to fit each of these trial lengths is similar to that used when both start and stop times are specified. If complete coverage cannot be achieved with one trial length, then try the next, but assign as soon as any possibility for complete coverage is found. If no trial length gives full coverage, target remains unassigned.

(3) If possible, shorten the successful trial length until at least one participating unit has only exactly enough time for its share. Then assign.

c. Neither start nor stop time specified:

(1) Determine set of trial lengths to be specified according to TMASMX minus TMASMN:

TMASMX minus TMASMN	No. Steps from TMASMX to TMASMN
under 1 min.	1
1 min. - 2 min.	2
over 2 min.	4

(2) Scan schedules of units, starting with cheapest, looking for gaps. Each time a gap is found, try to fit a mixed assignment of the current trial length in it. If scan produces a gap suitable for a "perfect mixed assignment" (number of participating units exactly as specified; each unit can cover a "perfect share"), assign immediately after attempting to shorten length as in I.B.3.b.(3) above. Otherwise, save best gap yet found.

(3) If no "perfect mixed assignment" is found for the current trial length, start scanning again for next trial length.

(4) If no trial length produces a "perfect mixed assignment," check the best gap yet found. If full coverage is not possible in that gap, target remains unassigned. Otherwise, assign target in this "best gap," attempting to shorten length as in I.B.3.b.(3).

C. "Assigning" means adding elements to scheduling arrays, updating counters and pointers, decrementing ammunition supplies, etc. Also, an output routine is called to print a summary of the assignment for this target on the "Target Assignment List."

Interpreting Appendix E

The notation of Appendix E does not correspond exactly to Figures 3 through 5. There are two main reasons for this:

(1) The program, typical of many heuristics, does not operate on the model explicitly. There are elements in the program that are not present in the model, and vice versa.

(2) The sponsor of the research preferred that notation developed in preliminary research be continued.

Also, Appendices E and F refer to "massed fire," etc. instead of "mixed assignments." This was again a sponsor preference.

The Narrative Outline mentions discretionary resources only once. This is because each unit is broken up into several fictitious units (one for each ammunition type). Thus, the term "unit" actually can be read as "row," or "distinct ammunition/unit combination." When an assignment is made, the schedules and remaining time supplies are, of course, updated for all rows associated with the assigned unit.

Appendix E also deals with "primary" and "secondary" rows. The user has the option of specifying a set of rows that receive primary consideration for assignment to a given target. The secondary rows are those specified for consideration if sufficient primary rows are unavailable. This is handled by treating the primary rows as though they were "cheaper" than the secondary rows.

Interpreting Appendix F

These lists are designed to be self-explanatory. The term "ALPHA" is used for "Q" because of the sponsor preference noted above.

The solution routine is intended for use in a conversational environment. Appendix F can therefore be seen as data that will eventually be kept in peripheral storage and displayed on a screen as needed, instead of being printed immediately after generation.

Testing and Evaluation

The user supplied only one set of problem data for test purposes. The problem was too large (37 rows, 27 targets) for initial debugging. Output from this problem is not included in this dissertation for these reasons:

(1) The problem contained logical conflicts that prevent any feasible solution from being found.

(2) No basis has been established by the user for evaluating the objective function. The procedure for determining cost coefficients is currently under revision. The form of the objective function has not been fixed.

(3) The current Appendix F is less cumbersome to use but does not omit any important information.

A few smaller problems were randomly generated and used in debugging the program code, but no formal testing was done, since the objective function remains undefined. However, even for the 37 x 27 problem, the program ran quickly enough for conversational use, and the answers that were obtained appeared to satisfy well the admittedly hazy objective criteria of Figure 3, given any unremovable infeasibility. Also, program size was such that the user could expect to implement the application in 500,000 bits of storage.

Conclusions

This application points up the adaptability of VAMI. RANDC (for an N of reasonable size) would have required too much time because of the complex process of optimizing a target. It is difficult to imagine any way to implement LPMAX. GREEDY and CRAFTY use too much time for smaller and much simpler problems. VAMI, however, is again a standout, as was so often seen in Chapter IV.

CHAPTER VI

SUMMARY, CONCLUSIONS, CONTRIBUTIONS, AND RECOMMENDATIONS

Summary

The Problem

Multi-resource generalized assignment problems have been identified in many applications. Unfortunately, these problems are very difficult to solve, especially if complications are present.

Optimal Solutions Unavailable

Apparently, no cost-effective optimal solution method can be developed. Optimal methods also have several disadvantages per se:

(1) They are difficult to adapt to changing requirements.

(2) They do not produce several solution alternatives.

(3) They use much storage and computer time, which is usually unjustified, since data are often inexact.

Heuristic Approaches

This research has produced several heuristic solution methods. Construction heuristics use various forms of logic to build a solution from the problem data. These forms of logic, along with the corresponding heuristics developed in this research, are:

(1) Random (RANDR, RANDC)

(2) Penalty-Guided (VAMI/VAMC)

(3) Adjusted Continuous Solution (LPMAX)

Improvement heuristics try to make an existing solution better. GREEDY and CRAFTY do this by switching the assignment of two tasks to different agents.

Objectives

This research has sought methods for realistic aspirations for:

(1) Quantitative performance measures: Task coverage, response time, accuracy/optimality, computer storage.

(2) Qualitative performance measures: Adaptability, alternate solutions, ease of use, predictability of response time.

Testing

The methods have been programmed and tested on a number of problems. A number of criteria were used to compare the relative performances of the heuristics. Distinct differences in performance were observed, along with some interesting characteristics of individual methods.

Implementations

Two demonstrations have been developed of implementations under extreme circumstances:

(1) Limited computer resources

(2) Extremely complicated problem

Results have been fairly satisfactory, so far as interpretation is possible.

Conclusions

Evaluation of Test Results

Each of the objectives of this research will be considered to determine:

(1) whether it was achieved, and

(2) which method(s) performed best in achieving it.

Additionally, a tabulation is made of conclusions about the performance of individual methods in achieving research objectives.

Realistic Problems

Chapter V makes it clear that the basic methods VAMI and, to a lesser extent, RANDC can be adapted to fit a variety of realistic problem situations.

Coverage

The aspiration level given in Chapter I was to find a solution covering all tasks in 90 percent of ". . . the cases tested." It is only reasonable to add the qualification that the continuous solution must exist, since there can be no full coverage otherwise. Table XVIII shows that VAMI, VAMC, and RANDC exceeded this level (VAMI scored 100 percent!), with both LPMAX and GREEDY at or above 80 percent. For the 28 problems where full coverage was apparently impossible, only three cases were encountered where VAMI failed to cover as many tasks as believed possible.

Response Time

The aspiration level of five minutes can be guaranteed for large problems only by RANDC and VAMI/VAMC. One might reduce the value of N or q to speed up these methods, if an increased chance of a bad solution could be tolerated.

Interesting effects on the response times of RANDC and VAMI/VAMC were observed to be caused by changing the "shape" of a problem of a given size. Qualitative considerations of response time also involved "shape," as will be seen.

Accuracy/Optimality

Given existence of a feasible solution and knowledge of a bound on the optimum, the objective was to find a solution within 15 percent of the bound in 90 percent of the cases tested. VAMI/VAMC was the only heuristic to satisfy this criterion. Indeed, for large problems, VAMI was within <u>10 percent</u> of the bound in <u>95 percent</u> of the test runs! This result is even more remarkable if it is noted that the bound was usually the continuous optimum. There is, of course, no guarantee that the continuous optimum is anywhere near the actual zero-one optimum.

Consistent with the findings of Sahni and Gonzelez [28], every method occasionally produced terrible results. The probability of this could be substantially reduced by using two different methods on the same problem, provided the outcomes of the methods were very nearly independent.

It is debatable whether bad outcomes of VAMC and RANDC are independent events, since the probability that such events will occur simultaneously appears to be greater than the product of their individual probabilities of occurrence. VAMC and RANDC both optimize a task in the same way and are thus both likely to miss good solutions in problems where that strategy is a poor one. This may have occurred in the results reported in the leftmost data columns of Tables VIII and IX.

VAMI may produce results more nearly independent of the outcome of RANDC, since VAMI was designed specifically to avoid the problem just described. However, it may well be that another set of problems exists where VAMI and RANDC do not produce bad answers independently. Not enough runs were made in this research to thoroughly investigate this proposition empirically. However, on only one test run (see Table II) did both RANDC and VAMI fail to find an appealing solution. This gives some intuitive support to the contention that the probability of such an event is very small indeed.

In summary, the following conclusion can be drawn regarding accuracy/optimality:

(1) VAMI is very powerful, often where other methods fail.

(2) RANDC is a useful supplement to VAMI.

Computer Storage

All the methods except LPMAX use less than half the amount of storage aspired to. There are ways of reducing storage requirements still further, however. Happily, the greatest reductions can be realized with VAMI, which uses more storage than other methods. Besides the use of halfword storage described in Chapter IV and the packing technique of Chapter V, VAMI can be programmed to store the cost and inefficiency matrices on a direct-access storage device. They would be needed only at the beginning of the process of generating a new solution. Each recall would require only a fraction of a second, so the response time would probably suffer little. Direct-access devices are widely available even for microcomputers.

While this research was being done, developments in computer technology have made the consideration of storage much less vital. However, it remains comforting to conclude that heuristics are available that will enable almost any computer to be used on generalized assignment problems with an excellent chance of success.

Qualitative Criteria

As has been noted, VAMI is the most adaptable method, chiefly because its intermediate logic is not executed as often as that of other methods, and thus can be extensively redefined without costing much time. RANDC, however, is also quite adaptable, as long as the process of optimizing a task does not become too complicated.

All methods except VAMC produce multiple solution alternatives by design. RANDC and RANDR obviously offer more variety than other methods, although it is not clear that this is significant.

All methods are sufficiently easy to implement, operate, and maintain. It is estimated that one week or less would be required to implement any of the basic methods. Operation requires only that basic problem data be available. This could be generated by some automated process, or predefined, or at worst, keypunched. VAMI, the most complicated method to implement, might, in the long run, be the easiest to maintain in the face of changing requirements because of its aforementioned flexibility. However, RANDR and RANDC would allow simple changes to be made readily. GREEDY/CRAFTY and LPMAX do not appear to be robust with respect to changes; an apparently minor change in the problem definition might mean that the method would become useless (which is true, to some extent, of all methods).

As pointed out in Chapter IV, response times of RANDC and VAMI were a function of problem "shape" as well as size. The following expressions give highly approximate estimates of the CPU time in seconds required by RANDC and VAMI to produce one solution on the CDC-modified IBM 370-155-II at the University of Arkansas:

RANDC: $(Nmn/1000)(.2 - .1(\log_{10}(m/n))^2)$

VAMI: $(Nmn/1000)(1 + 1.5(\log_{10}(m/n))^2)$

The actual CPU time requirements are pseudorandom variables. Response time depends not only on CPU time but also on other parameters of the solution environment. Nevertheless, on most systems, response times of RANDC and VAMI will probably be:

(1) A linear function of mn, and

(2) An approximately quadratic function of m for a given mn.

These relationships should hold well enough for most planning purposes.

Performance of Individual Methods

The conclusions drawn above are ordered by objective, which makes it difficult to extract information regarding the overall performance of each method. Also, some less important conclusions have not been mentioned. Therefore, all conclusions have been tabulated in Table XXII.

Table XXII leaves little doubt that VAMI is far superior to the other methods. VAMI is the only method that could conceivably be regarded as an all-round problem solver. No other method achieved all

TABLE XXII

TABULATED PERFORMANCE OF BASIC METHODS IN ACHIEVING OBJECTIVES

Objectives	Method								
	VAMI		RANDC		VAMC		LPMAX		GREEDY
	L	S	L	S	L	S	L	S	S
Realistic Problems	*	*	M	M	*	*	U	U	U
Coverage	*	*	S	S	S	S	U	U	U
Response Time	S	S	S	S	*	*	U	U	S
Accuracy/Optimality	*	S	U	S	S	S	U	U	U
Adaptability	*	*	M	M	*	*	U	U	U
Multiple Solutions	S	S	*	*	U	U	S	S	S
Implementation	S	S	*	*	S	S	M	M	M
Operation	S	S	S	S	S	S	M	M	M
Maintenance	S	S	S	S	S	S	M	M	M
Predictable Time	S	S	S	S	S	S	S	S	U

* = Outstanding

S = Satisfactory

M = Marginal

U = Unsatisfactory

research objectives.

In spite of its failure to achieve all research objectives, RANDC does perform very well. It has advantages in areas not addressed by the research objectives:

(1) RANDC resembles the approach a decision-maker would be likely to devise. Therefore it is easy to explain, and has considerable ("infinite-number-of-monkeys") intuitive appeal.

(2) RANDC can make good use of additional computer time to increase the probability that it will find a good solution.

(3) Its results seem likely to be almost independent of those of VAMI, thus enabling the use of a powerful combination for especially intractable problems.

VAMC, although tested here as a special case of VAMI, could be implemented on its own if, for instance, it were known that constraints would never be particularly tight, but that rapid response time and minimal computer storage requirements were very important. Also, VAMI is not difficult to convert to VAMI, should the need arise.

LPMAX, GREEDY, and CRAFTY are not at all cost-effective, although LPMAX's by-product of a tight bound on the optimum and an index of constraint tightness are certainly useful for evaluating other methods. As will be seen under Recommendations, it even appears to be possible to improve performance of GREEDY and CRAFTY.

Contributions

Introduction

This research has made several contributions. The most important of these was the development of powerful heuristic solution methods, but other valuable contributions include problem identification and definition, development of evaluation methodology, and demonstration of specific applications.

Heuristic Methods

All of these were inspired to some degree by one or more existing solution techniques. However, the ultimate development of the heuristics required two main creative inputs:

(1) Combination of techniques used with apparently unrelated classes of problems.

(2) Modification and extension of such techniques to fit the special structure of generalized assignment problems.

The process can be compared to the development of the rotary lawn mower from the previously known principles of scissors and the wheel.

VAMI is the outstanding example of this process of combination and modification. The Vogel Approximation Method (VAM) was combined with the LaGrangian approach of appending weighted resource considerations to the objective function. Modifications included the use of discrete values of Q in place of the continuous-valued LaGrange multiplier, while only columns were optimized, instead of rows and columns as in the original VAM. Further modifications to fit extreme circumstances were described in Chapter V.

Problem Identification and Definition

This contribution had to be made in order to justify this research. The most important aspect was recognition of the need to give explicit consideration to multiple resources in formulating models and devising solution methods. A further contribution in this category was the identification of the need to develop heuristic methods, not only because of the probable unavailability of optimal methods, but also because of inherent disadvantages of optimal methods. Finally, as is evident in the artillery problem and elsewhere, assignment problems and scheduling problems often need to be solved <u>simultaneously</u>. The usual approach of first making assignments and then scheduling their execution is not always satisfactory.

Evaluation Methodology

Many ideas were taken from the literature. Some, such as the pairwise comparison of methods using the nonparametric sign test [23], were changed little. Others, e.g. the use of long runs of RANDC to obtain an evaluation standard, were developed independently. Some traditional methodology (using the continuous optimum as a bound on the zero-one optimum) was, however, much more complicated to develop. Finally, using the $\#x_{ij} = 1/\#x_{ij} \neq 0$ ratio (x_{ij} from the continuous solution) as an index of constraint tightness was an idea that occurred spontaneously during the development of LPMAX, but did not require any developmental work.

Whatever the source, the evaluation methodology used in this dissertation is more comprehensive than any that could be found in the literature. Accepting the reservation of Glover et al. [14] that there can be no "fair" evaluation standards, Chapter IV of this dissertation is likely to be one of the more comprehensive available sources of techniques for evaluating <u>many</u> types of heuristics.

Realistic Applications

A researcher who is faced with a realistic problem will probably be unable to apply one of the methods of this dissertation without modification. It is, of course, impossible for this research to examine every plausible variation that might occur. However, it is hoped that the researcher can be helped by the demonstrations and guidelines that are given in Chapter V for handling extreme but dissimilar requirements.

Recommendations

Introduction

This section makes recommendations for using and explaining the heuristics developed in this research and for further avenues of research to pursue, specifically as regards development of better heuristics.

Using the Heuristics

It would be a mistake to discard all methods except VAMI. There are situations where the use of VAMI would be inadvisable. RANDC is simpler and quicker to implement, and would probably be the method of choice if only a few loosely-constrained problems were to be solved, especially if the problems were not especially large. The advantage of RANDC's tendency to produce results independent of those of VAMI has already been noted. Again, VAMC might be best if speed were important and either constraints were loose or an occasional poor solution could be tolerated.

In conjunction with applications, some experience has been gained with explaining VAMI. It is crucial for the user to understand the method being used, because even such a powerful method may be otherwise rejected, perhaps covertly. The best reception has been given an approach that roughly parallels the development of VAMI:

(1) Description of an unsuccessful method, usually optimization of tasks in order from 1 to n.

(2) Discussion of the need to find a better-than-sequential order in which to optimize tasks.

(3) Introduction of the penalty concept and description of VAMC.

(4) Introduction of a simple example where VAMC fails because it does not adequately consider resources.

(5) Discussion of the concept of a "resource cost," or "resource inefficiency."

(6) Introduction of the combination of costs and inefficiency. It is not usually advisable to explain the concept of variable combining weight in great detail, since it is difficult to handle questions about predicting an "optimal" Q.

As an alternative, one might begin by explaining the concept of VAMC, and then discussing the incorporation of resource inefficiency as a cost, but in far less detail than suggested above.

Explanations of <u>any</u> method should be done in terms of the simple examples using small tables of numbers. Jargon such as "objective function" or "decision variable" should be regarded as taboo except with entirely academic audiences.

<u>Further Research</u>

One glaring need for further investigation is certainly to <u>test the heuristics with actual problems</u>. Some preliminary research was done with hypothetical artillery data displaying highly nonrandom characteristics such as many equal or nearly equal cost or resource coefficients in a column or row. Results were too sketchy to interpret properly, but such problems may be more difficult than those used in this research. <u>Tests in further realistic applications</u> are likewise desirable as a means of revealing more about general applicability of the various methods. There may be situations in which VAMI/VAMC is much less adaptable than some other method. Indeed, further testing per se is needed, both using problem sizes not considered in Chapter IV, and attempting to duplicate the results of this dissertation, especially using different computers.

Development and refinement of evaluation methodology would contribute not only to research with this class of problems, but also with heuristics in general. No claim is made that this dissertation is the last word on such methodology.

Next, it should be possible to refine the heuristic methods themselves. A preliminary attempt to improve the execution speed of GREEDY and CRAFTY appears to have chances of success. The modification used was to attempt to swap the two agents already assigned to a pair of tasks, instead of trying all possible new ways of assigning the tasks.

Perhaps RANDC should be slightly biased toward assigning the second cheapest agent to the task being optimized when constraints are tight. Alternatively, RANDC could be made to consider resource inefficiency in some way. Both ideas are attempts to suggest a way for RANDC to find good solutions to problems where assigning the cheapest agent is a poor strategy for optimizing a column.

Even VAMI may be subject to improvement. It may, for instance, be possible to avoid the time-consuming process of recomputing penalties by considering in penalty calculation the third-smallest element in the column. Also, it may be possible to get better results by not considering inefficiencies where an agent is well-supplied with resources. This technique would also increase execution speed.

LPMAX might give better results if it were guided in some way by the dual variables from the continuous solution. Sensitivity analysis of the continuous solution can give information about the consequences of elevating some variables to one and reducing others to zero.

All construction heuristics make one assignment at a time. It certainly would seem worthwhile to investigate the usefulness of making two or more assignments at a time. The generalized form of this approach is to divide the overall problem into subproblems (sets of tasks) to be solved optimally or heuristically in some sequence. Defining and sequencing subproblems was done on the basis of task priorities in Chapter V. However, many actual problems do not deal with priorities, so some other approach would need to be developed, possibly from VAM.

Finally, most assignment problems are also scheduling problems. This research has concentrated on assignment methods. The scheduling logic for the artillery problem was not the result of a thorough investigation. Therefore, further research is necessary to develop methods for dealing with problems where assignment and scheduling must be done together, with emphasis on scheduling techniques.

BIBLIOGRAPHY

1. Armour, G. and Buffa, E. "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities." Management Science, 9, 1 (1963), 294-309.

2. Balas, E. and Ivanescu, P. "On the Generalized Transportation Problem." Management Science, 11, 1 (1964), 188-202.

3. Balachandran, V. "An Integer Generalized Transportation Model for Optimal Job Assignment in Computer Networks." Operations Research, 24, 4 (1976), 742-759.

4. Balachandran, V. and Deshmukh, S. "Storage Decisions in Information Systems." AIIE Transactions, 8, 3 (1976), 358-364.

5. Brockelhurst, E. "A Heuristic Algorithm for Integer Linear Programming Problems." NPL Report NAC 18, National Physical Laboratory (Division of Numerical Analysis and Computing) United Kingdom Department of Trade and Industry, (June, 1972).

6. Case, K. and Thibault, H. A Heuristic Allocation Algorithm for Conventional Weapons for the Marine Integrated Fire and Air Support System. (Research Report to Methodology and Evaluation Group, Joint Munitions Effectiveness Manual.) Oklahoma State University, (July, 1976).

7. Case, K. and Thibault, H. A Heuristic Allocation Algorithm with Extensions for Conventional Weapons for the Marine Integrated Fire and Air Support System. (Research Report to Methodology and Evaluation Group, Joint Munitions Effectiveness Manual.) Oklahoma State University, (September, 1977).

8. Cooper, D. "Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation." Management Science, 22, 11 (1976), 1186-1194.

9. De Maio, A. and Roveda, C. "An All Zero-one Algorithm for a Certain Class of Transportation Problems." Operations Research, 19, 6 (1971), 1406-1418.

10. Francis, R. and White, J. Facility Layout and Location, Prentice-Hall, Inc., Englewood Cliffs, N. J. (1974).

11. Garey, M., Johnson, D., and Sethi, R. "The Complexity of Flowshop and Jobshop Scheduling." Mathematics of Operations Research, 1, 2 (1976), 117-129.

12. Geoffrion, A. "Solving Bicriterion Mathematical Programs." Operations Research, 15, 1 (1967), 39-54.

13. Geoffrion, A. and Graves, G. "Multicommodity Distribution Systems Design by Benders Decomposition." Management Science, 20, 5 (1974), 822-844.

14. Glover, F., Karney, D., Klingman, D., and Napier, A. "A Computation Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems." Management Science, 20, 5 (1974), 793-813.

15. Grigoriadis, M., Tang, D., and Woo, L. "Considerations in the Optimal Synthesis of Some Communications Networks." 45th Joint National Meeting of ORSA and TIMS, Boston, Mass., April 22-24, 1974.

16. Gross, D. and Pinkus, C. "Optimal Allocation of Ships to Yards for Regular Overhauls." Technical Memorandum No. 63095, Institute for Management Science and Engineering, The George Washington University, (May, 1972).

17. Hillier, F. and Lieberman, G. Operations Research. Holden-Day, Inc., San Francisco, Cal., (1967).

18. Ignizio, J. Heuristic Solution to Generalized Covering Problems. Unpublished Ph.D. dissertation, Virginia Polytechnic Institute and State University, (March, 1971).

19. Klingman, D. and Stutz, J. "Computational Testing on an Integer Generalized Network Code." 45th Joint National Meeting of ORSA and TIMS, Boston, Mass., April 22-24, 1974.

20. McRoberts, K. "A Search Model for Evaluating Combinatorially Explosive Problems." Operations Research, 19, 6 (1971), 1331-1349.

21. Michael, G. "A Review of Heuristic Programming." Decision Sciences, 3, 3 (1972), 74-100.

22. Nauss, R. "An Efficient Algorithm for the Zero-One Knapsack Problem." Management Science, 23, 1 (1976), 27-31.

23. Parker, C. "An Experimental Investigation of Some Heuristic Strategies for Component Placement." Operational Research Quarterly, 27, 11 (1976), 71-81.

24. Pasternak, H. and Passy, U. "Bicriterion Mathematical Programs with Boolean Variables." Mimeograph Series #106, Technion, Israel Institute of Technology, Haifa, Israel (1972).

25. Polya, G. How To Solve It. Doubleday Anchor Books, New York (1957).

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

27. Roy, B. "Problems and Methods with Multiple Objective Functions." Mathematical Programming, I (1971), 239-266.

28. Sahni, S. and Gonzalez, T. "P-Complete Approximation Problems." J. Association for Computing Machinery, 23, 3 (1976), 555-565.

29. Srinivasan, V. and Thompson, G. "An Algorithm for Assigning Uses to Sources in a Special Class of Transportation Problems." Operations Research, 21, 1 (1973), 284-295.

30. Taha, H. Integer Programming. Academic Press, New York (1975).

31. Taha, H. Operations Research. MacMillan, New York (1971).

32. Thibault, H., Case, K., and Dorland, E. "Improved Productivity Through Assignment Planning: Techniques Suited for Implementation with Limited Computer Resources." Proceedings, AIIE 1977 Systems Engineering Conference, Kansas City, Missouri, November 1-4, 1977.

33. Wagner, H. Principles of Operations Research. Prentice-Hall, Englewood Cliffs, N. J. (1969)

34. Woolsey, R. Operations Research for Immediate Application: A Quick and Dirty Manual. Harper and Row, New York (1975).

APPENDIX A

PROGRAMS FOR TESTING BASIC METHODS

```
C*****   THIS PROGRAM GENERATES, PRINTS (IF DESIRED), AND SOLVES (FOR MANY
C*****   RIGHT-HAND-SIDES) A MULTI-RESOURCE GENERALIZED ASSIGNMENT PROBLEM
C*****   WITH RANDOM DIMENSIONS,COEFFICIENTS,NO. RESOURCES, AND INFEASIBILITIES.
C*****   IF DESIRED, AN INPUT DATA SET FOR LP SOLN BY MPS CAN BE GENERATED
         INTEGER CV(1500),AV(6000),BV(2000),PS(1500),S(1500)
         COMMON /XBCOM/IOPT(750)
         INTEGER PP
         COMMON /SEEDC/ ISEED
         COMMON /PRNTC/ IPRINT
         COMMON/RNVMC/ NBIGQS
         COMMON /SWAPC/ IGREED
         COMMON /TESTC/ MTEST
         COMMON /LPCOM/ LPFLAG
C*****   READ APPROX NO VRLS WANTED, RANDOM SEED,PRINT SWITCH, AND SAMPLE
C*****   SIZE (FOR RANDOM-GUIDED METHODS) OR NO Q'S WANTED (FOR VAMI).
C*****   MTEST TELLS IF NEXT CARD(S) ARE FOR LPMAX OR FOR PRESPECIFIED RHS'S
C*****   MTEST:  N = SOLUTION ROUTINE CALLED FOR N RHS VALUES FROM FOL CARDS
C*****          -N = LPMAX USED N TIMES ON SETS OF DATA CARDS CONSISTING OF
C*****               AN RHS VALUE AND THE XIJ VALUES FROM THE LP SOLN FOR
C*****               THAT PARTICULAR RHS..
C*****           0 = NORMAL OPERATION (RHS'S ARE GENERATED FROM DATA; ANY
C*****               EXTRA DATA CARD WILL BE USED AS RHS IN GENERATING
C*****               DATA DECK FOR MPS TO USE TO GET LP SOLN.
C*****   LPFLAG TELLS IF DATA DECK FOR LP SOLN BY MPS IS WANTED.
C*****   LPFLAG:  0 = NO MPS DECK; 1 = DECK & ALL ELSE; 2 = DECK & NOTHING ELSE
C*****   IGREED CONTROLS IMPROVEMENT HEURISTIC: 0 = NONE, 1 = GREEDY, 2 = CRAFTY
    1000 READ(5,1)NOVBLS,ISEED,IPRINT,NBIGQS,MTEST,LPFLAG,IGREED
         IF(NOVBLS.GT.9000) GO TO 98
       1 FORMAT(8I10)
         WRITE(6,223) ISEED
     223 FORMAT('ISEED =',I11)
C*****   READ DIMENSIONS AND NO. RESOURCES, THEN PRINT THEM
         READ(5,1)MM,NN,PP
         WRITE(6,2233) MM,NN,PP
    2233 FORMAT('0MM,NN,PP =',3(I5,', '))
C*****   CALL SUBROUTINE TO GENERATE COEFF'S & INFEASIBILITIES.
         CALL MATGEN(CV,AV,MM,NN,PP)
C*****   PRINT PROBLEM DATA IF DESIRED.
         IF(IPRINT.LE.0)GO TO 33
         CALL MATPRT(CV,AV,MM,NN,PP)
C*****   CALL SUBROUTINE TO USE ONE SOLN METHOD WITH SEVERAL RHS'S
      33 CALL SOLOOP(CV,AV,BV,MM,NN,PP,PS,S)
         IF(LPFLAG.EQ.0)GO TO 1000
C*****   GENERATE DATA SET FOR LP SOLN (RHS MUST BE READ FROM A CARD)
         READ(5,1)IRHS
         ISTOP=MM*PP
         DO 3333 I=1,ISTOP
    3333 BV(I)=IRHS
         CALL MPSGEN(CV,AV,BV,MM,NN,PP)
         GO TO 1000
      98 WRITE(6,99)
      99 FORMAT('1',50X,'***  NORMAL END OF JOB  ***')
         STOP
         END
```

OPTIONS IN EFFECT ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
OPTIONS IN EFFECT NAME = MAIN   , LINECNT =   60

```
      SUBROUTINE MATGEN(C,A,MM,NN,PP)
      INTEGER PP,P,C,A
      DIMENSION C(MM,NN),A(MM,NN,PP)
      COMMON /SEEDC/ ISEED
      M=MM
      N=NN
      P=PP
C*****   GENERATE COEFFICIENTS
      DO 10 I=1,M
      DO 10 J=1,N
      C(I,J)=1000*RANDU(ISEED)+1
      DO 10 K=1,P
      A(I,J,K)=1000*RANDU(ISEED)+1
   10 CONTINUE
C*****   GENERATE NO OF INFEASIBILITIES
      NOINFS=M*N/3*RANDU(ISEED)+1
C*****   GENERATE INDICES OF INFEASIBILITIES; FLAG INFEASIBILITIES
      DO 20 IX=1,NOINFS
      IZAP=M*RANDU(ISEED)+1
      JZAP=N*RANDU(ISEED)+1
      C(IZAP,JZAP)=9999
      DO 20 K=1,P
      A(IZAP,JZAP,K)=0
   20 CONTINUE
      RETURN
      END

*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = MATGEN  , LINECNT =       60
*STATISTICS*    SOURCE STATEMENTS =        23,PROGRAM SIZE =     1224
*STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
FORTRAN IV G LEVEL  21                MATPRT            DATE = 78295         16/18/22                PAGE 0001

 0001               SUBROUTINE MATPRT(C,A,MM,NN,PP)
 0002               COMMON /SEEDC/ ISEED
 0003               INTEGER C,A,PP,P
 0004               DIMENSION C(MM,NN),A(MM,NN,PP)
 0005               M=MM
 0006               N=NN
 0007               P=PP
 0008               WRITE (6,1)
 0009             1 FORMAT('1',40X,'MATRIX OF C(I,J) COEFFICIENTS (9999=INFEASIBLE)')
 0010               WRITE (6,2)(J,J=1,N)
 0011             2 FORMAT(///,59X,'TASK NUMBERS:',//,(17X,20I5))
 0012               DO 5 I=1,M
 0013               WRITE(6,3) I,(C(I,J),J=1,N)
 0014             3 FORMAT('0AGENT NO.',I4,':  ',20I5,/,(17X,20I5))
 0015             5 CONTINUE
 0016               DO 10 K=1,P
 0017               WRITE(6,6)K,K
 0018             6 FORMAT('1',33X,'MATRIX OF A(I,J,K) COEFFICIENTS FOR K=',I2,
                   *' (I. E., RESOURCE NO.',I2,')',///)
 0019               WRITE(6,2) (J,J=1,N)
 0020               DO 10 I=1,M
 0021               WRITE(6,3) I,(A(I,J,K),J=1,N)
 0022            10 CONTINUE
 0023               RETURN
 0024               END

  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = MATPRT  , LINECNT =       60
  *STATISTICS*    SOURCE STATEMENTS =       24,PROGRAM SIZE =     1184
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
           SUBROUTINE SOLOOP(C,A,B,MM,NN,PP,PS,S)
     C*****   CALCULATES UNCONSTRAINED OPTIMUM, THEN OBTAINS SOLUTIONS FOR SEVERAL
     C*****   DIFFERENT RHS'S BY REPEATED CALLS TO A SOLUTION SUBROUTINE THAT
     C*****   USES ONE OF THE HEURISTICS.
           COMMON /XBCOM/IOPT(750)
           COMMON /SEEDC/ ISEED
           COMMON /TESTC/ MTEST
           COMMON /LPCOM/ LPFLAG
           INTEGER PP,C,A,B,P
           INTEGER PS,S
           DIMENSION PS(MM,NN),S(MM,NN)
           DIMENSION C(MM,NN),A(MM,NN,PP),B(MM,PP)
           N=NN
           M=MM
           P=PP
           WRITE(6,5)
         5 FORMAT('1',30X,'***  INFO ABOUT UNCONSTRAINED OPTIMUM  ***')
     C*****   CLEAR B'S FOR ACCUMULATING RESOURCES REQD BY UNCONSTR OPT
           DO 10 I=1,M
           DO 10 K=1,P
        10 B(I,K)=0
     C*****   FIND UNCONSTR OPT AND ACCUM ITS RESCE REQTS; FIND RHS NEEDED TO MAKE
     C*****   UNCONSTR OPT FEAS; FIND MAX FEAS C(I,J)
           IBSTOP=0
           MNCSUM=0
           MXCOST=0
           DO 20 J=1,N
           MINC=C(1,J)
           IMC=1
           DO 15 I=2,M
           ICIJ=C(I,J)
           IF((MXCOST.GT.ICIJ).OR.(ICIJ.GT.9000)) GO TO 14
           MXCOST=ICIJ
        14 IF(MINC.LE.ICIJ)GO TO 15
           MINC=ICIJ
           IMC=I
        15 CONTINUE
           IF(MINC.LT.9000)GO TO 17
     C*****   FIXUP FOR TASK WHERE ALL AGENTS ARE FLAGGED INFEAS
           IMC=M*RANDU(ISEED)+1
           MINC=1000*RANDU(ISEED)+1
           C(IMC,J)=MINC
           DO 16 K=1,P
        16 A(IMC,J,K)=1000*RANDU(ISEED)+1
           WRITE(6,165) J,IMC,MINC,(A(IMC,J,K),K=1,P)
       165 FORMAT('0TASK',I4,' INFEAS FOR ALL AGENTS -- NEW C AND A VALUES',
          *' FOR AGENT',I4,':',/,' C = ',I5,' A''S =',4I5)
        17 MNCSUM=MNCSUM+MINC
           IOPT(J)=IMC
           DO 19 K=1,P
           B(IMC,K)=B(IMC,K)+A(IMC,J,K)
           IF(B(IMC,K).GT.IBSTOP) IBSTOP=B(IMC,K)
        19 CONTINUE
        20 CONTINUE
           WRITE(6,22) MNCSUM,(IOPT(J),J=1,N)
        22 FORMAT('0UNCONSTR OPT COST = ',I7,/,
          *'0ASGMT VECTOR: ',22I5,/,(15X,22I5))
           WRITE(6,2222)
```

```
 2222 FORMAT('0RESOURCE REQTS OF UNCONSTR OPT:')
      DO 200 K=1,P
      WRITE(6,222) K,(B(I,K),I=1,M)
  222 FORMAT(5X,'RESOURCE',I2,': ',15I6,/,(17X,15I6))
  200 CONTINUE
C*****   SKIP EVERYTHING ELSE IF MPS DECK ONLY DESIRED
      IF(LPFLAG.EQ.2)RETURN
      INFCST=MXCOST*N
      NM=N/M+1
      IF(MTEST.NE.0)GO TO 210
C*****   MTEST = 0:  CALCULATE THE SET OF RHS'S TO BE USED
      IBSTRT=50*P*NM
      IBSTEP=(IBSTOP-IBSTRT)/10
      IBSTEP=IBSTEP+10-MOD(IBSTEP,10)
      IBSTOP=IBSTOP+IBSTEP
      IB=IBSTRT-IBSTEP
      GO TO 50
C*****   MTEST > OR < 0:  READ IN THE SET OF RHS'S TO BE USED
  210 MSTOPR=MTEST
      IF(MTEST.LT.0)MSTOPR=-MTEST
      MSTEPR=0
   50 IF(MTEST.NE.0)GO TO 55
      IB=IB+IBSTEP
      IF(IB.GT.IBSTOP)RETURN
      GO TO 70
   55 MSTEPR=MSTEPR+1
      IF(MSTEPR.GT.MSTOPR)RETURN
      READ(5,56)IB
   56 FORMAT(8I10)
      IF(MTEST.GT.0)GO TO 70
C*****   MTEST < 0:  READ CONTIN SOLN FOR RHS JUST READ. LPMAX WILL BE CALLED.
C*****   ONLY NONZERO CONTIN VBLS ARE READ, SO ZERO OUT MATRIX FIRST
      DO 60 I=1,M
      DO 60 J=1,N
   60 PS(I,J)=0
   63 READ(5,56) I,J,LPX
      IF(I.GT.M)GO TO 70
      PS(I,J)=LPX
      GO TO 63
   70 WRITE(6,25) IB
   25 FORMAT(///,' SOLUTION FOR ALL B(I,K) =',I7)
C*****   FIND UNCONSTRAINED OPTIMUM FOR THIS RESOURCE LEVEL
C*****    (IF POSSIBLY DIFFERENT FROM ABOVE UNCONSTR OPT)
      IF(IB.GT.1000)GO TO 48
      MNCSUM=0
      DO 45 J=1,N
      MINC=INFCST
      IMC=-1
      DO 40 I=1,M
      DO 30 K=1,P
      IF(A(I,J,K).GT.IB)GO TO 40
   30 CONTINUE
      ICIJ=C(I,J)
      IF((ICIJ.GE.MINC).OR.(ICIJ.GT.9000))GO TO 40
      MINC=ICIJ
      IMC=I
   40 CONTINUE
      IOPT(J)=IMC
```

```
0101            MNCSUM=MNCSUM+MINC
0102         45 CONTINUE
0103            WRITE(6,46)MNCSUM,(IOPT(J),J=1,N)
0104         46 FORMAT('0UNCONSTR OPT COST FOR THIS RHS IS ',I7,': ASGT VECTOR:',
               */,(30I4))
0105         48 CALL SOLVER(C,A,B,IB,MM,NN,PP,MXCOST,PS,S)
0106            GO TO 50
0107            END
```

```
*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = SOLOOP   , LINECNT =       60
*STATISTICS*    SOURCE STATEMENTS =       107,PROGRAM SIZE =      3696
*STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
            SUBROUTINE MPSGEN(C,A,B,MM,NN,PP)
C*****
C*****   GENERATES DATA DECK FOR MPS
C*****
            INTEGER C,A,B,PP,P
            DIMENSION C(MM,NN),A(MM,NN,PP),B(MM,PP)
            M=MM
            N=NN
            P=PP
            WRITE(7,777)
        777 FORMAT('NAME           INP-DATA')
            WRITE(7,1)
          1 FORMAT('ROWS',/,' N  R00000')
            M10=10*M
            DO 10 I=10,M10,10
            DO 10 K=1,P
            IR=10000+I+K
            WRITE(7,2)IR
          2 FORMAT(' L  R',I5)
         10 CONTINUE
            DO 20 J=1,N
            IR=2000+J
            WRITE(7,3) IR
          3 FORMAT(' E  R',I4)
         20 CONTINUE
            WRITE(7,4)
          4 FORMAT('COLUMNS')
            IZEROS=100
            IF(N.GT.99)IZEROS=1000
            DO 100 I=1,M
            I10=10*I
            ISHIFT=I*IZEROS
            IJJ=100000+ISHIFT
            DO 100 J=1,N
            IJX=IJJ+J
            DO 40 K=1,P
            IF(A(I,J,K).LE.B(I,K))GO TO 40
            COST=9999999.
            GO TO 45
         40 CONTINUE
            COST=C(I,J)
            IF(COST.GT.9000.)COST=9999999.
         45 WRITE(7,5)IJX,COST
          5 FORMAT('    X',I6,'   R00000',F16.6)
            DO 50 K=1,P
            IR=10000+I10+K
            IF(COST.LT.9998.)GO TO 56
            AIJK=0.
            GO TO 65
         56 AIJK=A(I,J,K)
         65 WRITE(7,6) IJX,IR,AIJK
          6 FORMAT('    X',I6,'   R',I5,F16.6)
         50 CONTINUE
            IR=2000+J
            WRITE(7,7)IJX,IR
          7 FORMAT('    X',I6,'   R',I4,9X,'1.0')
        100 CONTINUE
            WRITE(7,8)
```

```
    8 FORMAT('RHS')
      DO 200 I=1,M
      I10=10*I
      DO 200 K=1,P
      IR=10000+I10+K
      BIK=B(I,K)
      WRITE(7,9)IR,BIK
    9 FORMAT('     RHS1',6X,'R',I5,F16.6)
  200 CONTINUE
      DO 300 J=1,N
      IR=2000+J
      WRITE(7,11) IR
   11 FORMAT('     RHS1',6X,'R',I4,9X,'1.')
  300 CONTINUE
      WRITE(7,12)
   12 FORMAT('BOUNDS')
      DO 400 I=1,M
      ISHIFT=I*IZEROS
      IJJ=100000+ISHIFT
      DO 400 J=1,N
      IJX=IJJ+J
      WRITE(7,13) IJX
   13 FORMAT(' UP BVBLS      X',I6,7X,'1.')
  400 CONTINUE
      WRITE(7,7999)
 7999 FORMAT('ENDATA')
      RETURN
      END
```

```
*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = MPSGEN  , LINECNT =       60
*STATISTICS*    SOURCE STATEMENTS =       83,PROGRAM SIZE =      2284
*STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
0001          SUBROUTINE SOLVER(C,A,B,IB,MM,NN,PP,MXCOST,PS,S)
        C********************************************************************************
        C                                                                               *
        C                              ***  VAMI  ***                                   *
        C                                                                               *
        C********************************************************************************
0002          INTEGER Z,U,H(750),W(750),XB(750)
0003          COMMON /XBCOM/IOPT(750)
0004          EQUIVALENCE(XB(1),IOPT(1))
0005          DIMENSION IP(750)
0006          DIMENSION IXBSAV(750)
0007          COMMON /SWAPC/ IGREED
0008          COMMON/RNVMC/ NBIGQS
0009          COMMON /PRNTC/ IPRINT
0010          INTEGER C,A,B,P,PP
0011          INTEGER PS,S
0012          DIMENSION S(MM,NN),PS(MM,NN)
0013          DIMENSION C(MM,NN),A(MM,NN,PP),B(MM,PP)
0014          INTEGER CBIG
0015          M=MM
0016          N=NN
0017          P=PP
        C*****   COST OF UNASGD TASK IS NO. TASKS TIMES MAX FEAS COST.
        C*****   THIS GUARANTEES THAT A SOLN COVERING N+1 TASKS IS CHEAPER THAN
        C*****   A SOLN COVERING N OR FEWER TASKS.
0018          INFCST=MXCOST*N
0019          WRITE(6,3)
0020        3 FORMAT('0***  VAMI  ***')
        C*****   MAKE NBIGQS ODD IF IT IS EVEN
0021          IF((MOD(NBIGQS,2)).NE.1)NBIGQS=NBIGQS+1
        C*****   CALCULATE INCREMENTS FOR Q < .25 AND Q >= .25
0022          DQLT25=1./(2*(NBIGQS-1))
0023          DQGE25=3.*DQLT25
        C*****   CALCULATE ITERATION NO. WHERE BIGGER INCREMENT STARTS BEING USED
0024          ISTEPN=NBIGQS/2
        C*****   CALCULATE AND SUM INEFFICIENCIES(S); SUM COSTS (C). (FEAS CELLS ONLY)
0025          SUMS=0.
0026          SUMC=0.
0027          DO 1 I=1,M
0028          DO 1 J=1,N
0029          ICIJ=C(I,J)
0030          IF(ICIJ.GT.9000)GO TO 1
0031          SMAX=FLOAT(A(I,J,1))/IB
0032          IF(P.EQ.1)GO TO 4
0033          DO 2 K=2,P
0034          STEMP=FLOAT(A(I,J,K))/IB
0035          IF(STEMP.GT.SMAX)SMAX=STEMP
0036        2 CONTINUE
0037        4 SUMC=SUMC+ICIJ
0038          SMAX=SMAX*1000.
0039          IF(SMAX.GT.1000.)SMAX=1000.
0040          SUMS=SUMS+SMAX
0041          S(I,J)=SMAX
0042        1 CONTINUE
        C*****   CALAULATE FACTOR TO BALANCE AVG S & AVG C; CLEAR Q
0043            F=SUMC/SUMS
0044          Q=0.
        C*****   BIG DO-LOOP GENERATES THE NO. OF SOLNS SPECD
```

```
      DO 1000 NSOLN=1,NBIGQS
C*****   (RE)INITIALIZE NO. UNCOVERED TASKS & OBJ FUN VALUE
      U=0
      Z=0
C*****   (RE)INITIALIZE RHS'S
      DO 5 I=1,M
      DO 5 K=1,P
    5 B(I,K)=IB
C*****   CALCULATE (1-Q) OUTSIDE DO-LOOP
      Q1=1.-Q
C*****   CALCULATE MATRIX (PS) OF RESOURCE-BIASED COSTS
      WRITE(6,124) Q
  124 FORMAT('0Q=',F7.5)
      DO 10 I=1,M
      DO 10 J=1,N
      ICIJ=C(I,J)
      PS(I,J)=ICIJ
      IF(ICIJ.GT.9000)GO TO 10
C*****   WHEN Q=0, PS IS THE SAME AS C AND VAMI IS EQUIV TO VAMC
      IF(NSOLN.NE.1) GO TO 7
      GO TO 10
    7 PS(I,J)=Q1*ICIJ+Q*F*S(I,J)
   10 CONTINUE
C*****   UPDATE Q FOR NEXT TIME
      IF(NSOLN.LE.ISTEPN)Q=Q+DQLT25
      IF(NSOLN.GT.ISTEPN)Q=Q+DQGE25
C*****   CALCULATE PENALTIES
      DO 15 J=1,N
C*****   CLEAR XB FOR LATER CHECKING IF PENALTIES NEED RECALCULATING
      XB(J)=0
      CALL PENCOL(PS(1,J),MM,NN,PP,A,B,J,IH,I1,I2)
      H(J)=IH
      IP(J)=I1*10000+I2
   15 CONTINUE
C*****   CLEAR FLAG TO TEST FOR NO MORE ASGTS FEAS
      NOMO=0
C*****   OPTIMIZE TASKS IN ORDER OF MAX PENALTY
      DO 100 J=1,N
      IF(NOMO.GT.0)GO TO 100
      MAXPEN=-5
      JBIG=0
      IF(J.EQ.1)GO TO 16
C*****   CHECK TO SEE IF ANY PENALTIES NEED TO BE RECALCULATED
      DO 170 JJ=1,N
      IF(XB(JJ).NE.0)GO TO 170
      I1=IP(JJ)/10000
      I2=MOD(IP(JJ),10000)
      IF((I1.NE.IBIG).AND.(I2.NE.IBIG))GO TO 170
      CALL PENCOL(PS(1,JJ),MM,NN,PP,A,B,JJ,IH,I1,I2)
      IP(JJ)=I1*10000+I2
      H(JJ)=IH
  170 CONTINUE
C*****   FIND NEW MAX PENALTY
   16 DO 17 JJ=1,N
      IF((MAXPEN .GT. H(JJ)) .OR. (XB(JJ) .GT. 0)) GO TO 17
      MAXPEN=H(JJ)
      JBIG=JJ
   17 CONTINUE
```

```
              C*****   IF MAX PENALTY < 0 NO MORE ASGTS ARE FEAS.  CALCULATE FINAL INCREMENTS
              C*****   IN COST & NO. UNASGD TASKS. THEN FLAG CORRES ELEMENTS OF ASGT VECTOR.
 0091               IF(MAXPEN.GE.0)GO TO 18
 0092               IUADD=N-J+1
 0093               CBIG=INFCST*IUADD
 0094               U=U+IUADD
 0095               DO 177 JJJ=1,N
 0096               IF(XB(JJJ).EQ.0)XB(JJJ)=-1
 0097           177 CONTINUE
              C*****   SET FLAG FOR NO MORE ASGTS FEAS
 0098               NOMO=1
 0099               IBIG=-1
 0100               GO TO 80
              C*****   MAKE SOME OTHER PENALTY MAX NEXT TIME
 0101            18 H(JBIG)=-1
              C*****   TRY AGENTS IN AN ORDER DETERMINED BY WHO DOES THIS TASK AT LOWEST COST
 0102               DO 22 I=1,M
 0103            22 W(I)=C(I,JBIG)
 0104               DO 75 I=1,M
 0105            23 WMIN=W(1)
 0106               IBIG=1
 0107               DO 25 II=1,M
 0108               IF(W(II).GE.WMIN) GO TO 25
 0109               WMIN=W(II)
 0110               IBIG=II
 0111            25 CONTINUE
 0112               IF(WMIN.GT.9000)GO TO 45
              C*****   CHECK IF RESOURCES OK
 0113               DO 30 K=1,P
 0114               IF(A(IBIG,JBIG,K).GT.B(IBIG,K))GO TO 40
 0115            30 CONTINUE
              C*****   RESOURCES OK -- ASSIGN
 0116               DO 35 K=1,P
 0117               B(IBIG,K)=B(IBIG,K)-A(IBIG,JBIG,K)
 0118            35 CONTINUE
 0119               CBIG=C(IBIG,JBIG)
 0120               GO TO 80
              C*****   RESOURCES NOT OK -- TRY NEXT CHEAPEST AGENT
 0121            40 W(IBIG)=9999
 0122               GO TO 23
              C*****   NO FEAS AGENT FOR THIS TASK -- COST IS BIG & FLAG IS -1
 0123            45 CBIG=INFCST
 0124               IBIG=-1
 0125               U=U+1
 0126               GO TO 80
 0127            75 CONTINUE
              C*****   ADD TO COST AND UPDATE ASGMT VECTOR
 0128            80 Z=Z+CBIG
 0129               XB(JBIG)=IBIG
 0130           100  CONTINUE
              C*****   SOLUTION COMPLETE
              C*****   INITIALIZE FLAG TO CHECK FOR NEW BEST SOLN
 0131               NUBEST=0
              C*****   PRINT OUT ALL NEW BEST SOLNS
 0132               IF(NSOLN.GT.1)GO TO 120
 0133           105 WRITE(6,110)
 0134           110 FORMAT(' ***  NEW BEST SOLN  ***')
 0135               NUBEST=1
```

```
      SUBROUTINE PENCOL(PS,MM,NN,PP,A,B,JJ,IH,I1,I2)
C*****  THIS SUBROUTINE CALCULATES A PENALTY "IH" AS THE DIFFERENCE BETWEEN
C*****  THE TWO SMALLEST FEASIBLE ELEMENTS (MIN2-MINP) OF THE JTH COLUMN OF P.
C*****  THE INDEXES OF MINP AND MIN2 ARE RETURNED IN I1 AND I2 SO CHECKING
C*****  FOR THE NECESSITY OF PENALTY RECALCULATION CAN BE SPEEDED UP.
      INTEGER PS,PP,P,A,B
      DIMENSION PS(MM),A(MM,NN,PP),B(MM,PP)
      M=MM
      N=NN
      P=PP
      J=JJ
      MINP=99999
      I1=0
      DO 10 I=1,M
      IPS=PS(I)
      IF((IPS.LT.0).OR.(IPS.GT.9000))GO TO 10
      DO 5 K=1,P
      IF(A(I,J,K).LE.B(I,K))GO TO 5
      PS(I)=-IPS
      GO TO 10
    5 CONTINUE
      IF(IPS.GE.MINP)GO TO 10
      MINP=IPS
      I1=I
   10 CONTINUE
      MIN2=99998
      I2=0
      DO 20 I=1,M
      IPS=PS(I)
      IF((IPS.LT.0).OR.(I.EQ.I1).OR.(IPS.GT.9000))GO TO 20
      DO 15 K=1,P
      IF(A(I,J,K).LE.B(I,K))GO TO 15
      PS(I)=-IPS
      GO TO 20
   15 CONTINUE
      IF(IPS.GT.MIN2)GO TO 20
      MIN2=IPS
      I2=I
   20 CONTINUE
      IH=MIN2-MINP
      RETURN
      END
```

```
*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = PENCOL  , LINECNT =      60
*STATISTICS*    SOURCE STATEMENTS =       38,PROGRAM SIZE =      1378
*STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
      SUBROUTINE SOLVER(C,A,B,IB,MM,NN,PP,MXCOST,PS,S)
C***************************************************************************
C                                                                          *
C                           ***  RANDR  ***                                *
C                                                                          *
C***************************************************************************
      INTEGER Z,U,T(750),AB(750),XB(750)
      COMMON /XBCOM/ IOPT(750)
      DIMENSION IXBSAV(750)
      EQUIVALENCE (XB(1),IOPT(1))
      INTEGER PS,S
      DIMENSION S(MM,NN),PS(MM,NN)
      COMMON /SWAPC/ IGREED
      COMMON/RNVMC/ NBIGGS
      COMMON /SEEDC/ ISEED
      COMMON /PRNTC/ IPRINT
      INTEGER C,A,B,P,PP
      DIMENSION C(MM,NN),A(MM,NN,PP),B(MM,PP)
      INTEGER CBIG
      M=MM
      N=NN
      P=PP
C*****   COST OF UNASGD TASK IS NO. TASKS TIMES MAX FEAS COST.
C*****   THIS GUARANTEES THAT A SOLN COVERING N+1 TASKS IS CHEAPER THAN
C*****   A SOLN COVERING N OR FEWER TASKS.
      INFCST=MXCOST*N
      WRITE(6,3)
    3 FORMAT('0***  RANDR  ***')
C*****   INITIALIZE VECTORS OF TASK AND AGENT INDEXES TO BE SHUFFLED
      DO 1 J=1,N
      T(J)=J
    1 CONTINUE
      DO 2 I=1,M
    2 AB(I)=I
C*****   DEFAULT SAMPLE SIZE IS 300
      IF(NBIGGS.LT.1) NBIGGS=300
C*****   BIG DO-LOOP GENERATES THE NO. OF SOLNS SPECD IN SAMPLE SIZE
      DO 1000 NSOLN=1,NBIGGS
C*****   (RE)INITIALIZE NO. UNCOVERED TASKS, OBJ FUN VALUE, FLAG FOR LAST AGENT
      U=0
      Z=0
      IEND=0
C*****   (RE)INITIALIZE RHS'S
      DO 5 I=1,M
      DO 5 K=1,P
    5 B(I,K)=IB
C*****   SHUFFLE TASK INDEXES
      NM=N-1
      DO 20 J=1,NM
      JSHUF=(N-J)*RANDU(ISEED)+J+1
      JSWAP=T(J)
      T(J)=T(JSHUF)
      T(JSHUF)=JSWAP
   20 CONTINUE
C*****   ASSIGN TASKS IN RANDOM ORDER
      DO 100 J=1,N
C*****   PICK A TASK AT RANDOM
      JBIG=T(J)
```

```
      C*****   NOTE: THE FOLLOWING LOGIC COMBINES STEPS II.D.2. AND 3. OF RANDR
      C*****   TO SAVE TIME.
      C
      C*****   TRY AGENTS IN RANDOM ORDER DURING (NOT AFTER) SHUFFLE PROCESS
            MM1=MM-1
            DO 75 I=1,MM1
      C*****   PICK RANDOM AGENT
            ISHUF=(M-I)*RANDU(ISEED)+I+1
            IBIG=AB(ISHUF)
            AB(ISHUF)=AB(I)
            AB(I)=IBIG
      C*****   CHECK IF FLAGGED INFEASIBLE
         25 IF(C(IBIG,JBIG).GT.9000)GO TO 40
      C*****   CHECK IF RESOURCES OK
            DO 30 K=1,P
            IF(A(IBIG,JBIG,K).GT.B(IBIG,K))GO TO 40
         30 CONTINUE
      C*****   RESOURCES OK -- ASSIGN
            DO 35 K=1,P
            B(IBIG,K)=B(IBIG,K)-A(IBIG,JBIG,K)
         35 CONTINUE
            CBIG=C(IBIG,JBIG)
            GO TO 80
      C*****   RESOURCES NOT OK -- TRY ANOTHER AGENT AT RANDOM, . . .
         40 IF(I.LT.MM1)GO TO 75
      C*****   UNLESS NOBODY IS LEFT TO TRY, . . .
            IF(IEND.EQ.1)GO TO 45
      C*****   FLAG LAST AGENT AS TRIED, THEN GO TRY HIM
            IEND=1
            IBIG=AB(M)
            GO TO 25
      C*****   NO FEAS AGENT FOR THIS TASK -- COST IS BIG & FLAG IS -1
         45 CBIG=INFCST
            IBIG=-1
            U=U+1
            GO TO 80
         75 CONTINUE
      C*****   ADD TO COST AND UPDATE ASGMT VECTOR
         80 Z=Z+CBIG
            XB(JBIG)=IBIG
        100  CONTINUE
      C*****   SOLUTION COMPLETE
      C*****   INITIALIZE FLAG TO CHECK FOR NEW BEST SOLN
            NUBEST=0
      C*****   PRINT OUT ALL NEW BEST SOLNS
            IF(NSOLN.GT.1)GO TO 120
        105 WRITE(6,110)
        110 FORMAT(' ***  NEW BEST SOLN  ***')
            NUBEST=1
            MINZ=Z
            DO 115 IXBS=1,N
        115 IXBSAV(IXBS)=XB(IXBS)
            GO TO 125
        120 IF(Z.LT.MINZ)GO TO 105
            IF(IPRINT.LT.1)GO TO 500
            IF(NSOLN.LE.5)GO TO 125
            GO TO 500
        125 IRCOST=Z-U*INFCST
```

```
0085              WRITE(6,127) Z,U,IRCOST,NSOLN
0086          127 FORMAT(' COST:',I7,', NO UNASGD TASKS:',I5,
                 *', COST OF ASGD TASKS:',I7,', TRIAL NO.:',I6)
0087              WRITE (6,130) (XB(J),J=1,N)
0088          130 FORMAT(' ASSIGNMENT VECTOR: ',20I5,/,(20X,20I5))
0089              IF(IPRINT.LT.0)GO TO 500
0090              DO 200 K=1,P
0091              WRITE(6,150)K,(B(I,K),I=1,M)
0092          150 FORMAT(' SLACKS FOR RESOURCE',I2,':  ',16I6,/,(25X,16I6))
0093          200 CONTINUE
0094          500 IF(NUBEST.EQ.0)GO TO 1000
            C*****    NEW BEST SOLN -- TRY TO IMPROVE IT
0095              NUBEST=0
0096              CALL SWAPPR(C,A,B,M,N,P,INFCST,Z)
0097         1000 CONTINUE
            C*****    LET GREEDY TRY TO IMPROVE BEST SOLN FOUND FOR THIS RHS
0098              IGSAV=IGREED
0099              IGREED=1
0100              DO 1115 IXBS=1,N
0101         1115 XB(IXBS)=IXBSAV(IXBS)
0102              CALL SWAPPR(C,A,B,M,N,P,INFCST,MINZ)
0103              IGREED=IGSAV
0104              RETURN
0105              END

 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = SOLVER   , LINECNT =       60
 *STATISTICS*    SOURCE STATEMENTS =       105,PROGRAM SIZE =     12334
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
0001          SUBROUTINE SOLVER(C,A,B,IB,MM,NN,PP,MXCOST,PS,S)
        C**************************************************************************
        C                                                                         *
        C                              ***  RANDC  ***                            *
        C                                                                         *
        C**************************************************************************
0002          INTEGER Z,U,T(750),W(750),XB(750)
0003          COMMON /XBCOM/ IOPT(750)
0004          DIMENSION IXBSAV(750)
0005          INTEGER PS,S
0006          DIMENSION S(MM,NN),PS(MM,NN)
0007          EQUIVALENCE (XB(1),IOPT(1))
0008          COMMON /SWAPC/ IGREED
0009          COMMON/RNVMC/ NBIGQS
0010          COMMON /SEEDC/ ISEED
0011          COMMON /PRNTC/ IPRINT
0012          INTEGER C,A,B,P,PP
0013          DIMENSION C(MM,NN),A(MM,NN,PP),B(MM,PP)
0014          INTEGER CBIG
0015          M=MM
0016          N=NN
0017          P=PP
        C*****   COST OF UNASGD TASK IS NO. TASKS TIMES MAX FEAS COST.
        C*****   THIS GUARANTEES THAT A SOLN COVERING N+1 TASKS IS CHEAPER THAN
        C*****   A SOLN COVERING N OR FEWER TASKS.
0018          INFCST=MXCOST*N
0019          WRITE(6,3)
0020        3 FORMAT('0***  RANDC  ***')
        C*****   INITIALIZE VECTOR OF TASK INDEXES TO BE SHUFFLED
0021          DO 1 J=1,N
0022          T(J)=J
0023        1 CONTINUE
        C*****   DEFAULT SAMPLE SIZE IS 300
0024          IF(NBIGQS.LT.1) NBIGQS=300
        C*****   BIG DO-LOOP GENERATES THE NO. OF SOLNS SPECD IN SAMPLE SIZE
0025          DO 1000 NSOLN=1,NBIGQS
        C*****   (RE)INITIALIZE NO. UNCOVERED TASKS & OBJ FUN VALUE
0026          U=0
0027          Z=0
        C*****   (RE)INITIALIZE RHS'S
0028          DO 5 I=1,M
0029          DO 5 K=1,P
0030        5 B(I,K)=IB
        C*****   SHUFFLE TASK INDEXES
0031          NM=N-1
0032          DO 20 J=1,NM
0033          JSHUF=(N-J)*RANDU(ISEED)+J+1
0034          JSWAP=T(J)
0035          T(J)=T(JSHUF)
0036          T(JSHUF)=JSWAP
0037       20 CONTINUE
        C*****   ASSIGN TASKS IN RANDOM ORDER
0038          DO 100 J=1,N
        C*****   PICK A TASK AT RANDOM
0039          JBIG=T(J)
        C*****   TRY AGENTS IN AN ORDER DETERMINED BY WHO DOES THIS TASK AT LOWEST COST
0040          DO 22 I=1,M
0041       22 W(I)=C(I,JBIG)
```

```
             DO 75 I=1,M
          23 WMIN=W(1)
             IBIG=1
             DO 25 II=1,M
             IF(W(II).GE.WMIN) GO TO 25
             WMIN=W(II)
             IBIG=II
          25 CONTINUE
             IF(WMIN.GT.9000)GO TO 45
      C*****    CHECK IF RESOURCES OK
             DO 30 K=1,P
             IF(A(IBIG,JBIG,K).GT.B(IBIG,K))GO TO 40
          30 CONTINUE
      C*****    RESOURCES OK -- ASSIGN
             DO 35 K=1,P
             B(IBIG,K)=B(IBIG,K)-A(IBIG,JBIG,K)
          35 CONTINUE
             CBIG=C(IBIG,JBIG)
             GO TO 80
      C*****    RESOURCES NOT OK -- TRY NEXT CHEAPEST AGENT
          40 W(IBIG)=9999
             GO TO 23
      C*****    NO FEAS AGENT FOR THIS TASK -- COST IS BIG & FLAG IS -1
          45 CBIG=INFCST
             IBIG=-1
             U=U+1
             GO TO 80
          75 CONTINUE
      C*****    ADD TO COST AND UPDATE ASGMT VECTOR
          80 Z=Z+CBIG
             XB(JBIG)=IBIG
         100  CONTINUE
      C*****    SOLUTION COMPLETE
      C*****    INITIALIZE FLAG TO CHECK FOR NEW BEST SOLN
             NUBEST=0
      C*****    PRINT OUT ALL NEW BEST SOLNS
             IF(NSOLN.GT.1)GO TO 120
         105 WRITE(6,110)
         110 FORMAT(' ***  NEW BEST SOLN  ***')
             NUBEST=1
             MINZ=Z
             DO 115 IXBS=1,N
         115 IXBSAV(IXBS)=XB(IXBS)
             GO TO 125
         120 IF(Z.LT.MINZ)GO TO 105
             IF(IPRINT.LT.1)GO TO 500
             IF(NSOLN.LE.5)GO TO 125
             GO TO 500
         125 IRCOST=Z-U*INFCST
             WRITE(6,127) Z,U,IRCOST,NSOLN
         127 FORMAT(' COST:',I7,', NO UNASGD TASKS:',I5,
            *', COST OF ASGD TASKS:',I7,', TRIAL NO.:',I6)
             WRITE (6,130) (XB(J),J=1,N)
         130 FORMAT(' ASSIGNMENT VECTOR: ',20I5,/,(20X,20I5))
             IF(IPRINT.LT.0)GO TO 500
             DO 200 K=1,P
             WRITE(6,150)K,(B(I,K),I=1,M)
         150 FORMAT(' SLACKS FOR RESOURCE',I2,':  ',16I6,/,(25X,16I6))
```

```
  200 CONTINUE
  500 IF(NUBEST.EQ.0)GO TO 1000
C*****    NEW BEST SOLN -- TRY TO IMPROVE IT
      NUBEST=0
      CALL SWAPPR(C,A,B,M,N,P,INFCST,Z)
 1000 CONTINUE
C*****    LET GREEDY TRY TO IMPROVE BEST SOLN FOUND FOR THIS RHS
      IGSAV=IGREED
      IGREED=1
      DO 1115 IXBS=1,N
 1115 XB(IXBS)=IXBSAV(IXBS)
      CALL SWAPPR(C,A,B,M,N,P,INFCST,MINZ)
      IGREED=IGSAV
      RETURN
      END
```

```
*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = SOLVER  , LINECNT =       60
*STATISTICS*     SOURCE STATEMENTS =       103,PROGRAM SIZE =     12274
*STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
      SUBROUTINE SOLVER(C,A,B,IB,MM,NN,PP,MXCOST,XIJ,S)
C***********************************************************************
C                                                                      *
C                              ***  LPMAX  ***                         *
C                                                                      *
C***********************************************************************
      INTEGER Z,U,T(750),W(750),XB(750)
      COMMON /XBCOM/ IOPT(750)
      DIMENSION IXBSAV(750)
      INTEGER XIJ,S
      DIMENSION S(MM,NN),XIJ(MM,NN)
      EQUIVALENCE (XB(1),IOPT(1))
      COMMON /TESTC/ MTEST
      COMMON /SWAPC/ IGREED
      COMMON/RNVMC/ NBIGGS
      COMMON /SEEDC/ ISEED
      COMMON /PRNTC/ IPRINT
      INTEGER C,A,B,P,PP
      DIMENSION C(MM,NN),A(MM,NN,PP),B(MM,PP)
      INTEGER CBIG
      M=MM
      N=NN
      P=PP
C*****   COST OF UNASGD TASK IS NO. TASKS TIMES MAX FEAS COST.
C*****   THIS GUARANTEES THAT A SOLN COVERING N+1 TASKS IS CHEAPER THAN
C*****   A SOLN COVERING N OR FEWER TASKS.
      INFCST=MXCOST*N
      WRITE(6,3)
    3 FORMAT('0***  LPMAX  ***')
C*****   INITIALIZE VECTOR OF TASK INDEXES
      JFRONT=1
      JBACK=N
      DO 1 J=1,N
      XB(J)=0
      DO 2 I=1,M
      IF(XIJ(I,J).LT.1000)GO TO 2
      T(JBACK)=J
      JBACK=JBACK-1
      XB(J)=I
      GO TO 1
    2 CONTINUE
      T(JFRONT)=J
      JFRONT=JFRONT+1
    1 CONTINUE
      NM=JFRONT-2
      NJ=NM+1
      NP1=N+1
C*****   DEFAULT SAMPLE SIZE IS 10
      IF(NBIGGS.LT.1) NBIGGS=10
C*****   BIG DO-LOOP GENERATES THE NO. OF SOLNS SPECD IN SAMPLE SIZE
      DO 1000 NSOLN=1,NBIGGS
C*****   (RE)INITIALIZE NO. UNCOVERED TASKS & OBJ FUN VALUE
      U=0
      Z=0
C*****   (RE)INITIALIZE RHS'S
      DO 5 I=1,M
      DO 5 K=1,P
    5 B(I,K)=IB
```

```
C*****   SHUFFLE INDICES OF TASKS HAVING NO XIJ = 1
         DO 20 J=1,NM
         JSHUF=(NJ-J)*RANDU(ISEED)+J+1
         JSWAP=T(J)
         T(J)=T(JSHUF)
         T(JSHUF)=JSWAP
      20 CONTINUE
C*****   ASSIGN TASKS:  FIRST WHERE SOME XIJ = 1; OTHERS IN RANDOM ORDER
         DO 100 J=1,N
         JBX=NP1-J
C*****     GET INDEX FOR NEXT TASK
         JBIG=T(JBX)
C*****   IF XIJ = 1, MAKE CORRES ASGT. OTHERWISE, GO BY DESCENDING XIJ VALUE.
C*****   IF NO FEAS XIJ > 0, ASSIGN BY ASCENDING COST.
         IBFLAG=0
         IF(JBX.LE.NJ)GO TO 2021
         IBIG=XB(JBIG)
         IBFLAG=1
         GO TO 33
    2021 DO 2202 I=1,M
         IWI=1000-XIJ(I,JBIG)
         IF(IWI.LT.0)IWI=9999
         IF(IWI.EQ.1000)IWI=1000+C(I,JBIG)
         W(I)=IWI
    2202 CONTINUE
      33 DO 75 I=1,M
         IF(IBFLAG.EQ.1)GO TO 3533
      23 WMIN=W(1)
         IBIG=1
         DO 25 II=1,M
         IF(W(II).GE.WMIN) GO TO 25
         WMIN=W(II)
         IBIG=II
      25 CONTINUE
         IF(WMIN.GT.9000)GO TO 45
C*****   CHECK IF RESOURCES OK
         DO 30 K=1,P
         IF(A(IBIG,JBIG,K).GT.B(IBIG,K))GO TO 40
      30 CONTINUE
C*****   RESOURCES OK -- ASSIGN
    3533 DO 35 K=1,P
         B(IBIG,K)=B(IBIG,K)-A(IBIG,JBIG,K)
      35 CONTINUE
         CBIG=C(IBIG,JBIG)
         GO TO 80
C*****   RESOURCES NOT OK -- TRY NEXT CHEAPEST AGENT
      40 W(IBIG)=9999
         GO TO 23
C*****   NO FEAS AGENT FOR THIS TASK -- COST IS BIG & FLAG IS -1
      45 CBIG=INFCST
         IBIG=-1
         U=U+1
         GO TO 80
      75 CONTINUE
C*****   ADD TO COST AND UPDATE ASGMT VECTOR
      80 Z=Z+CBIG
         XB(JBIG)=IBIG
     100 CONTINUE
```

```
            C*****   SOLUTION COMPLETE
            C*****   INITIALIZE FLAG TO CHECK FOR NEW BEST SOLN
0094              NUBEST=0
            C*****   PRINT OUT ALL NEW BEST SOLNS
0095              IF(NSOLN.GT.1)GO TO 120
0096          105 WRITE(6,110)
0097          110 FORMAT(' ***  NEW BEST SOLN  ***')
0098              NUBEST=1
0099              MINZ=Z
0100              DO 115 IXBS=1,N
0101          115 IXBSAV(IXBS)=XB(IXBS)
0102              GO TO 125
0103          120 IF(Z.LT.MINZ)GO TO 105
0104              IF(IPRINT.LT.1)GO TO 500
0105              IF(NSOLN.LE.5)GO TO 125
0106              GO TO 500
0107          125 IRCOST=Z-U*INFCST
0108              WRITE(6,127) Z,U,IRCOST,NSOLN
0109          127 FORMAT(' COST:',I7,', NO UNASGD TASKS:',I5,
                 *', COST OF ASGD TASKS:',I7,', TRIAL NO.:',I6)
0110              WRITE (6,130) (XB(J),J=1,N)
0111          130 FORMAT(' ASSIGNMENT VECTOR: ',20I5,/,(20X,20I5))
0112              IF(IPRINT.LT.0)GO TO 500
0113              DO 200 K=1,P
0114              WRITE(6,150)K,(B(I,K),I=1,M)
0115          150 FORMAT(' SLACKS FOR RESOURCE',I2,':  ',16I6,/,(25X,16I6))
0116          200 CONTINUE
0117          500 IF(NUBEST.EQ.0)GO TO 1000
            C*****   NEW BEST SOLN -- TRY TO IMPROVE IT
0118              NUBEST=0
0119              CALL SWAPPR(C,A,B,M,N,P,INFCST,Z)
0120         1000 CONTINUE
            C*****   LET GREEDY TRY TO IMPROVE BEST SOLN FOUND FOR THIS RHS
0121              IGSAV=IGREED
0122              IGREED=1
0123              DO 1115 IXBS=1,N
0124         1115 XB(IXBS)=IXBSAV(IXBS)
0125              CALL SWAPPR(C,A,B,M,N,P,INFCST,MINZ)
0126              IGREED=IGSAV
0127              RETURN
0128              END
```

```
*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = SOLVER   , LINECNT =       60
*STATISTICS*    SOURCE STATEMENTS =      128,PROGRAM SIZE =    12758
*STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
      SUBROUTINE SWAPPR(C,A,B,MM,NN,PP,INFCST,Z)
C*****   THIS SUBROUTINE IS "CRAFTY," "GREEDY," OR DOES NOTHING AT ALL,
C*****   DEPENDING ON WHETHER "IGREED" IS 2, 1, OR ZERO.
      INTEGER XB,C,A,B,PP,P,Z
      DIMENSION XB(750),C(MM,NN),A(MM,NN,PP),B(MM,PP)
      COMMON /XBCOM/IOPT(750)
      EQUIVALENCE(XB(1),IOPT(1))
      COMMON /SWAPC/ IGREED
      COMMON /PRNTC/ IPRINT
      DIMENSION IHEAD(2,2)
      DATA IHEAD /'GREE','CRAF','DY  ','TY  '/
      IF(IGREED.LT.1)RETURN
C*****   PRINT HEADING FOR WHATEVER METHOD IS TO BE USED.
      WRITE(6,16)(IHEAD(IGREED,J),J=1,2)
   16 FORMAT('0***  ',2A4,'***')
      M=MM
      N=NN
      P=PP
      MP=M+1
C
C*****   RETURN HERE AFTER MAKING SWAP
C
C*****   (RE)SET INDICATOR FOR CHECKING IF THIS TRY FOUND A GOOD SWAP
    1 ISWAP=0
C*****   (RE)SET COST IMPROVEMENT OF BEST SWAP YET FOUND BY "CRAFTY"
      ICHANG=0
      JL=1
C*****   JL IS INDEX OF LEFT-HAND TRIAL TASK, IL IS CURRENT AGENT FOR TASK JL
    2 IL=XB(JL)
      CILJL=INFCST
      IF(IL.GT.0)CILJL=C(IL,JL)
      IF(CILJL.GT.9000)CILJL=INFCST
      IL2=1
C*****   IL2 IS INDEX OF TRIAL NEW AGENT FOR TASK JL
    3 IF(IL2.EQ.IL)GO TO 8
      CIL2JL=INFCST
      IF(IL2.LT.MP)CIL2JL=C(IL2,JL)
      IF(CIL2JL.GT.9000)CIL2JL=INFCST
      JR=JL+1
C*****   JR IS INDEX OF RIGHT-HAND TRIAL TASK; IR IS INDEX OF CURRENT AGENT
    4 IR=XB(JR)
      CIRJR=INFCST
      IF(IR.GT.0)CIRJR=C(IR,JR)
      IF(CIRJR.GT.9000)CIRJR=INFCST
      IR2=1
C*****   IR2 IS INDEX OF TRIAL NEW AGENT FOR TASK JR
    5 IF(IR2.EQ.IR)GO TO 7
      CIR2JR=INFCST
      IF(IR2.LT.MP)CIR2JR=C(IR2,JR)
      IF(CIR2JR.GT.9000)CIR2JR=INFCST
C*****   CALCULATE CHANGE IN TOTAL COST IF SWAP WERE MADE.
      IDELTZ=CILJL+CIRJR-CIL2JL-CIR2JR
C*****   CHECK IF SWAP IS POTENTIALLY PROFITABLE
      IF(IDELTZ.LE.0)GO TO 7
C*****   IF SO, CHECK IF RESOURCES OK WITH FOL DO-LOOP.
C
C*****   APPARENT CLUMSINESS OF LOGIC IS DUE TO POSSIBILITY OF EQUALITY OF ROW.
C*****   INDICES, AND POSSIBILITY FOR ROW INDICES TO INDICATE THAT TASK UNASGD.
```

```
C
C*****   CLEAR FLAG FOR RESOURCE INFEASIBILITY
         INFSA=0
         DO 100 K=1,P
C*****   FOR EACH TASK AND EACH AGENT:  IF AGENT INDICATES REAL ASGMT, SAVE
C*****   CURRENT RESOURCE SUPPLY AND ADD BACK CURRENT USAGE OR SUBTRACT OFF
C*****   POTENTIAL USAGE.
         IF(IL.LE.0)GO TO 20
         ILBSAV=B(IL,K)
         B(IL,K)=ILBSAV+A(IL,JL,K)
      20 IF(IR.LE.0)GO TO 25
         IRBSAV=B(IR,K)
         B(IR,K)=IRBSAV+A(IR,JR,K)
      25 IF(IL2.GT.M)GO TO 30
         I2LSAV=B(IL2,K)
         IBL2K=I2LSAV-A(IL2,JL,K)
         IF(IBL2K.LT.0)INFSA=1
         B(IL2,K)=IBL2K
      30 IF(IR2.GT.M)GO TO 35
         I2RSAV=B(IR2,K)
         IBR2K=I2RSAV-A(IR2,JR,K)
         IF(IBR2K.LT.0)INFSA=1
         B(IR2,K)=IBR2K
C*****   RESTORE RESOURCE SUPPLIES AFTER FEASIBILITY CHECK
      35 IF(IR2.LE.M)B(IR2,K)=I2RSAV
         IF(IL2.LE.M)B(IL2,K)=I2LSAV
         IF(IR .GT.0)B(IR,K)=IRBSAV
         IF(IL .GT.0)B(IL,K)=ILBSAV
C*****   IF SWAP NOT RESOURCE-FEASIBLE, GO TRY ANOTHER
         IF(INFSA.GT.0)GO TO 7
     100 CONTINUE
C*****   FLAG ASGTS SWITCHED TO INFEAS AGENTS AS UNASGD
C*****   BUT FIRST SAVE NEW AGENT INDICES IN CASE THIS IS "CRAFTY"
         IR2SAV=IR2
         IL2SAV=IL2
         IF(CIL2JL.GT.9000)IL2=-1
         IF(CIR2JR.GT.9000)IR2=-1
C*****   TURN ON FLAG THAT OK SWAP HAS BEEN FOUND; IF "GREEDY," MAKE SWAP NOW
         ISWAP=1
         IF(IGREED.EQ.1)GO TO 6
C*****   RESTORE IR2 AND IL2 SO "CRAFTY" WILL BE ABLE TO CONTINUE SEARCH
         IR2=IR2SAV
         IL2=IL2SAV
C*****   IF "CRAFTY," CHECK FOR NEW BEST POSS SWAP.
         IF(ICHANG.GE.IDELTZ)GO TO 7
C*****   SAVE COST IMPROVEMENT, ROWS, AND COLS OF BEST SWAP YET FOUND
         ICHANG=IDELTZ
         IRSAV=IR
         ILSAV=IL
         JRSAV=JR
         JLSAV=JL
C*****   IF BOTTOM OF RIGHT-HAND COL, GO TRY NEW RH COL, ELSE TRY NEXT ROW IN RH
       7 IF(IR2.GT.M)GO TO 78
         IR2=IR2+1
         GO TO 5
C*****   MAKE SWAP:  DECREMENT COST, RESET AGENT ASSIGNMENTS, UPDATE RESOURCES.
       6 Z=Z-IDELTZ
         IF(IL2.GT.M)IL2=-1
```

```
              IF(IR2.GT.M)IR2=-1
              XB(JL)=IL2
              XB(JR)=IR2
              DO 66 K=1,P
              IF(IL.GT.0)B(IL,K)=B(IL,K)+A(IL,JL,K)
              IF(IL2.GT.0)B(IL2,K)=B(IL2,K)-A(IL2,JL,K)
              IF(IR.GT.0)B(IR,K)=B(IR,K)+A(IR,JR,K)
              IF(IR2.GT.0) B(IR2,K)=B(IR2,K)-A(IR2,JR,K)
           66 CONTINUE
              GO TO 500
        C*****   TRY NEW RH COL
           78 JR=JR+1
              IF(JR.LE.N)GO TO 4
        C*****   NO MORE RH COLS -- TRY TO UPDATE ROW INDEX IN LH COL
            8 IF(IL2.GE.MP)GO TO 88
              IL2=IL2+1
              GO TO 3
        C*****   LH COL ALL USED UP -- TRY TO GO TO NEXT LH COL.
           88 JL=JL+1
              IF(JL.LT.N)GO TO 2
        C*****   ALL LH COLS TRIED -- CHECK IF GOOD SWAP FOUND. IF NOT, RETURN.
              IF(ISWAP.EQ.0) GO TO 600
        C*****   KNOWN TO BE "CRAFTY" WITH GOOD SWAP STORED. SET UP SWAP AND GO MAKE IT.
              IDELTZ=ICHANG
              IR2=IR2SAV
              IR=IRSAV
              JR=JRSAV
              IL=ILSAV
              JL=JLSAV
              IL2=IL2SAV
              IF(IL2.LE.0.OR.IL2.GE.M)GO TO 888
              IF(C(IL2,JL).GT.9000)IL2=-1
          888 IF(IR2.LE.0.OR.IR2.GT.M)GO TO 6
              IF(C(IR2,JR).GT.9000)IR2=-1
              GO TO 6
        C*****   COME HERE AFTER SWAP & PRINT RESULTS OF SWAP.
          500 IF(IPRINT.LT.1)GO TO 1
              WRITE(6,501)Z,JL,IL,IL2,JR,IR,IR2
          501 FORMAT('0SUCCESSFUL SWAP -- NEW COST IS: ',I7,/,
             *2('   TASK',I4,':  OLD AGENT WAS',I4,', NEW AGENT IS',I4))
              GO TO 1
          600 WRITE(6,601) Z,(XB(J),J=1,N)
          601 FORMAT('0FINAL SWAP RESULT:  COST=',I7,
             *': ASSIGNMENT VECTOR WAS:',/,(30I4))
              RETURN
              END
```

```
*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = SWAPPR  , LINECNT =       60
*STATISTICS*    SOURCE STATEMENTS =       121,PROGRAM SIZE =     4376
*STATISTICS*  NO DIAGNOSTICS GENERATED

*STATISTICS*  NO DIAGNOSTICS THIS STEP
```

```
FORTRAN IV G LEVEL  21                RANDU              DATE = 78295         16/18/22              PAGE 0001

 0001               FUNCTION RANDU(ISEEDX)
 0002               ISEED=ISEEDX
 0003               ISEED=ISEED*16807
 0004               IF(ISEED)1,1,2
 0005             1 ISEED=ISEED+2147483647+1
 0006             2 RANDU=ISEED*4.656613E-10
 0007               ISEEDX=ISEED
 0008               RETURN
 0009               END

  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = RANDU    , LINECNT =         60
  *STATISTICS*    SOURCE STATEMENTS =         9,PROGRAM SIZE =       412
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
               CONTROL PROGRAM COMPILER - MPS/360 V2-M11                                           PAGE     1  -  78/295

         0001                     PROGRAM
         0002                     INITIALZ
         0065                     MOVE(XOBJ,'R00000')
         0066                     MOVE(XDATA,'INP-DATA')
         0067                     MOVE(XPBNAME,'GENASGT')
         0068                     TITLE('LP SOLN OF MULTI-RESOURCE GENERALIZED ASGT PROBS')
         0069                     MOVE(XRHS,'RHS1')
         0070                     CONVERT('SUMMARY')
         0071                     SETUP('BOUND','BVBLS')
         0072                     BCDOUT
         0073                     PRIMAL
         0074                     SOLUTION
         0075                     EXIT
         0076                     PEND
```

APPENDIX B

OUTPUT SAMPLES FROM PROGRAMS IN APPENDIX A

SEED = 1122334455

MM,NN,PP = 5, 10, 1,

MATRIX OF C(I,J) COEFFICIENTS (9999=INFEASIBLE)

TASK NUMBERS:

	1	2	3	4	5	6	7	8	9	10
AGENT NO. 1:	804	79	570	447	895	645	525	887	795	847
AGENT NO. 2:	498	774	238	93	290	520	963	671	457	169
AGENT NO. 3:	221	9999	815	898	268	401	937	515	9999	46
AGENT NO. 4:	654	964	794	13	555	819	9999	504	486	789
AGENT NO. 5:	753	607	51	213	9999	791	114	566	478	9999

MATRIX OF A(I,J,K) COEFFICIENTS FOR K= 1 (I. E., RESOURCE NO. 1)

TASK NUMBERS:

	1	2	3	4	5	6	7	8	9	10
AGENT NO. 1:	699	199	762	546	765	508	860	451	565	922
AGENT NO. 2:	274	918	276	648	270	803	379	319	668	644
AGENT NO. 3:	158	0	343	158	204	188	726	741	0	304
AGENT NO. 4:	300	970	980	202	106	100	0	166	236	568
AGENT NO. 5:	520	908	340	488	0	72	120	156	513	0

```
*** INFO ABOUT UNCONSTRAINED OPTIMUM ***

UNCONSTR OPT COST =    2154

ASGMT VECTOR:    3    1    5    4    3    3    5    4    2    3

RESOURCE REQTS OF UNCONSTR OPT:
   RESOURCE 1:    199   668   854   368   460


SOLUTION FOR ALL B(I,K) =    310

UNCONSTR OPT COST FOR THIS RHS IS    2370; ASGT VECTOR:
  3   1   2   4   3   3   5   4   4   3

***  RANDR  ***
***  NEW BEST SOLN  ***
COST:  40541, NO UNASGD TASKS:    4, COST OF ASGD TASKS:   1981, TRIAL NO.:     1
ASSIGNMENT VECTOR:     3    1   -1   -1    2    5    5   -1    4   -1
SLACKS FOR RESOURCE 1:     111    40   152    74   118

***  GREEDY  ***

SUCCESSFUL SWAP -- NEW COST IS:   40366
  TASK   1:  OLD AGENT WAS   3, NEW AGENT IS  -1   TASK  10:  OLD AGENT WAS  -1, NEW AGENT IS   3

SUCCESSFUL SWAP -- NEW COST IS:   40314
  TASK   3:  OLD AGENT WAS  -1, NEW AGENT IS   2   TASK   5:  OLD AGENT WAS   2, NEW AGENT IS  -1

SUCCESSFUL SWAP -- NEW COST IS:   39841
  TASK   4:  OLD AGENT WAS  -1, NEW AGENT IS   4   TASK   9:  OLD AGENT WAS   4, NEW AGENT IS  -1

SUCCESSFUL SWAP -- NEW COST IS:   30756
  TASK   1:  OLD AGENT WAS  -1, NEW AGENT IS  -1   TASK   5:  OLD AGENT WAS  -1, NEW AGENT IS   4

SUCCESSFUL SWAP -- NEW COST IS:   30531
  TASK   6:  OLD AGENT WAS   5, NEW AGENT IS  -1   TASK   8:  OLD AGENT WAS  -1, NEW AGENT IS   5

FINAL SWAP RESULT:  COST=  30531; ASSIGNMENT VECTOR WAS:
 -1   1   2   4   4  -1   5   5  -1   3
***  NEW BEST SOLN  ***
COST:  31146, NO UNASGD TASKS:    3, COST OF ASGD TASKS:   2226, TRIAL NO.:     2
ASSIGNMENT VECTOR:     2    1   -1    4    4    3    5    5   -1   -1
SLACKS FOR RESOURCE 1:     111    36   122     2    34

***  GREEDY  ***

SUCCESSFUL SWAP -- NEW COST IS:   30886
  TASK   1:  OLD AGENT WAS   2, NEW AGENT IS  -1   TASK   3:  OLD AGENT WAS  -1, NEW AGENT IS   2

SUCCESSFUL SWAP -- NEW COST IS:   30706
  TASK   1:  OLD AGENT WAS  -1, NEW AGENT IS   3   TASK   6:  OLD AGENT WAS   3, NEW AGENT IS  -1

SUCCESSFUL SWAP -- NEW COST IS:   30531
  TASK   1:  OLD AGENT WAS   3, NEW AGENT IS  -1   TASK  10:  OLD AGENT WAS  -1, NEW AGENT IS   3

FINAL SWAP RESULT:  COST=  30531; ASSIGNMENT VECTOR WAS:
 -1   1   2   4   4  -1   5   5  -1   3
COST:  32449, NO UNASGD TASKS:    3, COST OF ASGD TASKS:   3529, TRIAL NO.:     3
ASSIGNMENT VECTOR:     2    1   -1    3    4    4    5    5   -1   -1
SLACKS FOR RESOURCE 1:     111    36   152   104    34

COST:  40934, NO UNASGD TASKS:    4, COST OF ASGD TASKS:   2374, TRIAL NO.:     4
ASSIGNMENT VECTOR:     4    1    2   -1   -1    5   -1    5   -1    3
SLACKS FOR RESOURCE 1:     111    34     6    10    82
COST:  32449, NO UNASGD TASKS:    3, COST OF ASGD TASKS:   3529, TRIAL NO.:     5
ASSIGNMENT VECTOR:     2    1   -1    3    4    4    5    5   -1   -1
SLACKS FOR RESOURCE 1:     111    36   152   104    34
***  NEW BEST SOLN  ***
COST:  30531, NO UNASGD TASKS:    3, COST OF ASGD TASKS:   1611, TRIAL NO.:    15
ASSIGNMENT VECTOR:    -1    1    2    4    4   -1    5    5   -1    3
SLACKS FOR RESOURCE 1:     111    34     6     2    34

***  GREEDY  ***

FINAL SWAP RESULT:  COST=  30531; ASSIGNMENT VECTOR WAS:
 -1   1   2   4   4  -1   5   5  -1   3
```

```
*** INFO ABOUT UNCONSTRAINED OPTIMUM ***

UNCONSTR OPT CUST =    2154

ASGMT VECTOR:     3    1    5    4    3    3    5    4    2    3

RESOURCE REQTS OF UNCONSTR OPT:
    RESOURCE 1:    199   668   854   368   460



SOLUTION FOR ALL B(I,K) =     630

UNCONSTR OPT COST FOR THIS RHS IS     2175: ASGT VECTOR:
  3   1   5   4   3   3   5   4   5   3

*** RANDC ***
*** NEW BEST SOLN ***

COST:  21482, NO UNASGD TASKS:     2, COST OF ASGD TASKS:   2202, TRIAL NO.:     1
ASSIGNMENT VECTOR:     3    1    2    4    3    3   -1    4    5   -1

*** GREEDY ***

FINAL SWAP RESULT:  COST=   2482: ASSIGNMENT VECTOR WAS:
  2   1   5   4   2   3   5   4   4   3
*** NEW BEST SOLN ***
COST:   2482, NO UNASGD TASKS:     0, COST OF ASGD TASKS:   2482, TRIAL NO.:     2
ASSIGNMENT VECTOR:     2    1    5    4    2    3    5    4    4    3

*** GREEDY ***

FINAL SWAP RESULT:  COST=   2482: ASSIGNMENT VECTOR WAS:
  2   1   5   4   2   3   5   4   4   3

*** GREEDY ***

FINAL SWAP RESULT:  COST=   2482: ASSIGNMENT VECTOR WAS:
  2   1   5   4   2   3   5   4   4   3
```

```
*** INFO ABOUT UNCONSTRAINED OPTIMUM ***

UNCONSTR OPT COST =     2154

ASGMT VECTOR:      3    1    5    4    3    3    5    4    2    3

RESOURCE REQTS OF UNCONSTR OPT:
    RESOURCE 1:   199   668   854   368   460

SOLUTION FOR ALL B(I,K) =    470

UNCONSTR OPT COST FOR THIS RHS IS     2183; ASGT VECTOR:
  3   1   5   4   3   3   5   4   4   3

*** VAMI ***

Q=0.0
*** NEW BEST SOLN ***
COST:  20580, NO UNASGD TASKS:    2, COST OF ASGD TASKS:   1300, TRIAL NO.:     1
ASSIGNMENT VECTOR:     3    1    5    4    2   -1    5   -1    4    3

*** GREEDY ***

FINAL SWAP RESULT:  COST=  20580; ASSIGNMENT VECTOR WAS:
  3   1   5   4   2  -1   5  -1   4   3

Q=0.05000

Q=0.10000

Q=0.15000

Q=0.20000

Q=0.25000

Q=0.40000

Q=0.55000

Q=0.70000
*** NEW BEST SOLN ***
COST:  12194, NO UNASGD TASKS:    1, COST OF ASGD TASKS:   2554, TRIAL NO.:     9
ASSIGNMENT VECTOR:     3    1    2    4   -1    5    5    5    4    3

*** GREEDY ***

FINAL SWAP RESULT:  COST=  12194; ASSIGNMENT VECTOR WAS:
  3   1   2   4  -1   5   5   5   4   3

Q=0.85000

Q=1.00000

*** GREEDY ***

FINAL SWAP RESULT:  COST=  12194; ASSIGNMENT VECTOR WAS:
  3   1   2   4  -1   5   5   5   4   3
```

MATRIX OF C(I,J) COEFFICIENTS (9999=INFEASIBLE)

TASK NUMBERS:

	1	2	3	4	5
AGENT NO. 1:	804	79	9999	447	895
AGENT NO. 2:	645	525	857	9999	847
AGENT NO. 3:	498	774	238	93	9999

MATRIX OF A(I,J,K) COEFFICIENTS FOR K= 1 (I. E., RESOURCE NO. 1)

TASK NUMBERS:

	1	2	3	4	5
AGENT NO. 1:	699	199	0	546	765
AGENT NO. 2:	508	860	451	0	922
AGENT NO. 3:	274	918	276	648	0

```
NAME          INP-DATA
ROWS
 N  R00000
 L  R10011
 L  R10021
 L  R10031
 E  R2001
 E  R2002
 E  R2003
 E  R2004
 E  R2005
COLUMNS
    X100101   R00000        804.00000   R10011        699.00000
    X100101   R2001           1.00000
    X100102   R00000         79.00000   R10011        199.00000
    X100102   R2002           1.00000
    X100103   R00000      99999.00000   R2003           1.00000
    X100104   R00000        447.00000   R10011        546.00000
    X100104   R2004           1.00000
    X100105   R00000      99999.00000   R2005           1.00000
    X100201   R00000        645.00000   R10021        508.00000
    X100201   R2001           1.00000
    X100202   R00000      99999.00000   R2002           1.00000
    X100203   R00000        887.00000   R10021        451.00000
    X100203   R2003           1.00000
    X100204   R00000      99999.00000   R2004           1.00000
    X100205   R00000      99999.00000   R2005           1.00000
    X100301   R00000        498.00000   R10031        274.00000
    X100301   R2001           1.00000
    X100302   R00000      99999.00000   R2002           1.00000
    X100303   R00000        238.00000   R10031        276.00000
    X100303   R2003           1.00000
    X100304   R00000         93.00000   R10031        648.00000
    X100304   R2004           1.00000
    X100305   R00000      99999.00000   R2005           1.00000
RHS
    RHS1      R10011        740.00000   R10021        740.00000
    RHS1      R10031        740.00000   R2001           1.00000
    RHS1      R2002           1.00000   R2003           1.00000
    RHS1      R2004           1.00000   R2005           1.00000
BOUNDS
 UP BVBLS     X100101         1.00000
 UP BVBLS     X100102         1.00000
 UP BVBLS     X100103         1.00000
 UP BVBLS     X100104         1.00000
 UP BVBLS     X100105         1.00000
 UP BVBLS     X100201         1.00000
 UP BVBLS     X100202         1.00000
 UP BVBLS     X100203         1.00000
 UP BVBLS     X100204         1.00000
 UP BVBLS     X100205         1.00000
 UP BVBLS     X100301         1.00000
 UP BVBLS     X100302         1.00000
 UP BVBLS     X100303         1.00000
```

VARIFORM, OBJ = R00000 , RHS = RHS1

TIME = 0.069 MINS. PRICING = 7 XCYCLESW = 15

OLD ETA NON-ZEROS 0	OLD ETA VECTORS 0	OLD ETA RECORDS 1	START INVERT 0.069	ITERATION NO. 0
NEW ETA NON-ZEROS 0	NEW ETA VECTORS 0	NEW ETA RECORDS 1	TIME TAKEN 0.017	ALT. PIVOTS 0
BASIS NON-ZEROS 9	BASIC LOGICALS 9	BASIC STRUCTURALS 0	NO. OF ROWS 9	NO. OF BUMPS 0
DENSITY INCREASE 1.00-	TRIANGLE COLS 0	BUMP COLUMNS 0	RESIDUE COLS 0	NO. OF SPIKES 0

VARIFORM, OBJ = R00000 , RHS = RHS1

TIME = 0.087 MINS. PRICING = 7 XCYCLESW = 15
XSCALE = . . XFUNCT = 703684.0000 . XSIF = 1391.00000 . XNIF = 8

	ITER NUMBER	NUMBER INFEAS	VECTOR OUT	VECTOR IN	REDUCED COST	SUM INFEAS
7/7	1	8	10	10 U	700.000-	691.00000
	2	7	23	23 U	649.000-	232.00000
	3	5	13	13 U	547.000-	226.00000
	4	3	15	15 U	509.000-	6.00000
	5	3	17	17 U	1.00000-	5.00000
	6	2	22	22 U	1.00000-	4.00000
7/6	7	2	11	11 U	1.00000-	3.00000
	8	2	14	14 U	1.00000-	2.00000
	9	1	16	16 U	1.00000-	1.00000
	10	0	19	19 U	1.00000-	.

FEASIBLE SOLUTION

VARIFORM, OBJ = R00000 , RHS = RHS1

TIME = 0.087 MINS. PRICING = 7 XCYCLESW = 15
XSCALE = . . XFUNCT = 400494.0000 . XSIF = . . XNIF = 0
SCALE RESET TO 1.00000

	ITER NUMBER	NUMBER NONOPT	VECTOR OUT	VECTOR IN	REDUCED COST	FUNCTIONAL VALUE
7/5	11	5	6	21 U	99999.0-	400494.0000
	12		7	12 U	99999.0-	400494.0000
	13		8	18 U	99999.0-	400494.0000
	14		9	24 U	99999.0-	400494.0000
	15		5	20 U	498.000-	400494.0000
7/5	16	5	11 U	11	99920.0-	300574.0000
	17		22 U	22	99761.0-	200813.0000
	18		2	13	99552.0-	102172.6250
	19		18	23	99906.0-	101257.7500
	20		12	17	99112.0-	101257.7500
7/2	21	2	4	2	.64835-	101157.1875
7/1	22	1	15 U	15	2.68518-	101154.5000

OPTIMAL SOLUTION

SECTION 1 - ROWS

NUMBER	...ROW..	AT	...ACTIVITY...	SLACK ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	.DUAL ACTIVITY
1	R00000	BS	101154.51852	101154.51852-	NONE	NONE	1.00000
2	R10011	BS	354.03704	385.96296	NONE	740.00000	.
3	R10021	BS	508.00000	232.00000	NONE	740.00000.	.
4	R10031	UL	740.00000	.	NONE	740.00000	.54630
5	R2001	EQ	1.00000	.	1.00000	1.00000	647.68519-
6	R2002	EQ	1.00000	.	1.00000	1.00000	99999.00000-
7	R2003	EQ	1.00000	.	1.00000	1.00000	887.00000-
8	R2004	EQ	1.00000	.	1.00000	1.00000	447.00000-
9	R2005	EQ	1.00000	.	1.00000	1.00000	99999.00000-

SECTION 2 - COLUMNS

	NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
	10	X100101	LL	.	804.00000	.	1.00000	156.31481
	11	X100102	UL	1.00000	79.00000	.	1.00000	99920.00000-
	12	X100103	LL	.	99999.00000	.	1.00000	99112.00000
	13	X100104	BS	.28395	447.00000	.	1.00000	.
A	14	X100105	LL	.	99999.00000	.	1.00000	.
	15	X100201	UL	1.00000	645.00000	.	1.00000	2.68519-
A	16	X100202	LL	.	99999.00000	.	1.00000	.
	17	X100203	BS	.	887.00000	.	1.00000	.
	18	X100204	LL	.	99999.00000	.	1.00000	99552.00000
A	19	X100205	LL	.	99999.00000	.	1.00000	.
	20	X100301	BS	.	498.00000	.	1.00000	.
	21	X100302	BS	.	99999.00000	.	1.00000	.
	22	X100303	UL	1.00000	238.00000	.	1.00000	498.22222-
	23	X100304	BS	.71605	93.00000	.	1.00000	.
	24	X100305	BS	1.00000	99999.00000	.	1.00000	.

APPENDIX C

PROGRAMS FOR LIMITED COMPUTER RESOURCES

```
1990 REM      *** VAMI ***
2000 Q = .05
2010 FOR J2=1 TO 5
2020 IF J2 = 1 THEN 2060
2030 REM TAKE STEPS IN Q
2040 Q = Q+Q
2050 REM (RE)SET RESOURCE SUPPLIES
2060 FOR I = 1 TO M7
2070 U(I) = B(I)
2080 V(I) = T(I)
2090 NEXT I
2100 REM LOOP CALC'S PENALTIES
2110 FOR J=1 TO N
2120 IF J2>1 GO TO 2170
2130 REM PENALTY FOR Q = 0 IS EASIER
2140 P(J)=ABS(INT(C(2,J)-C(1,J)+20)/100)
2150 GO TO 2460
2160 REM UNPACK RESOURCE REQTS
2170 E1=R(1,J)
2180 R1=INT(E1/1000)
2190 E1=E1-R1*1000
2200 E2=R(2,J)
2210 R2=INT(E2/1000)
2220 E2=E2-R2*1000
2230 REM UNPACK COSTS & ROW INDEXES
2240 P1=C(1,J)
2250 C1=INT(P1/100)
2260 P1=P1-C1*100
2270 P2=C(2,J)
2280 C2=INT(P2/100)
2290 P2=P2-C2*100
2300 REM CALC INEF'CY FOR EACH
2310 REM RES"CE ON EACH MACH
2320 R1=R1/B(P1)
2330 E1=E1/T(P1)
2340 R2=R2/B(P2)
2350 E2=E2/T(P2)
2360 REM FIND MAX INEF'CY ON EACH MACH
2370 IF E1≥R1 THEN 2390
2380 E1=R1
2390 IF E2≥R2 THEN 2420
2400 E2=R2
2410 REM CALCULATE PENALTY
2420 Q1=1-Q
2430 C5=(C1+C2)/2
2440 E5=(E1+E2)/2
2450 P(J)=ABS(Q1*(C2-C1)+(Q*C5/E5)*(E2-E1)
2460 NEXT J
2470 C5=0
2480 REM UNTIL ALL JOBS ARE ASSIGNED
2490 FOR J = 1 TO N
2500 REM FIND L = NO. OF JOB W/MAX PEN
2510 M6=P(1)
2520 L=1
2530 FOR J6=2 TO N
2540 IF P(J6)≤M6 THEN 2570
2550 M6=P(J6)
2560 L=J6
2570 NEXT J6
2580 REM KEEP PEN FROM BEING MAX AGAIN
2590 P(L)=-999999
2600 REM OPTIMIZE JOB W/MAX PEN
2610 FOR I = 1 TO M
2620 C2 = C(I,L)
2630 IF C2>999000 THEN 2740
2640 C1 = INT(C2/100)
2650 C2=C2-C1*100
2660 R2=R(I,L)
2670 R1 = INT(R2/1000)
2680 IF R1 > U(C2) THEN 2730
2690 R2 = R2-R1*1000
2700 IF R2 > V(C2) THEN 2730
2710 I6=I
2720 GO TO 2780
2730 NEXT I
2740 A(L)=-1
2750 U6 = U6 + 1
2760 C1 = 500000
2770 GO TO 2810
2780 U(C2) = U(C2) - R1
2790 V(C2) = V(C2) - R2
2800 A(L)=C2
2810 C5 = C5 + C1
2820 NEXT J
2830 PRINT
2840 PRINT 'SOLUTION';J2
2850 IF U6 = 0 THEN 2900
2860 C5 = C5-U6*500000
2870 FOR I = 1 TO U6
2880 PRINT '*';
2890 NEXT I
2900 PRINT 'COST = ';C5
2910 FOR I = 1 TO M
2920 Y1=Y(I)
2930 PRINT 'MACH NO. ';Y1;' ASGD TO :'
2940 N1 = 0
2950 FOR J = 1 TO N
2960 IF A(J) ≠ Y1 THEN 2990
2970 N1 = 1
2980 PRINT '   JOB ';Z(J)
2990 NEXT J
3000 IF N1 ≠ 0 THEN 3020
3010 PRINT '*** NOTHING ***'
3020 PRINT 'UNUSED MATL:';U(Y1)
3030 PRINT 'UNUSED TIME:';V(Y1)
3040 PRINT
3050 NEXT I
3060 IF U6=0 THEN 3130
3070 PRINT 'UNASSIGNED JOBS:'
3080 U6=0
3090 FOR J = 1 TO N
3100 IF A(J) > 0 THEN 3120
3110 PRINT '   JOB ';Z(J)
3120 NEXT J
3130 NEXT J2
3140 PRINT 'NEW RUN?'
3150 INPUT Y$
3160 IF Y$ = 'YES' THEN 3190
3170 IF Y$ ≠ 'RHS' THEN 3210
3180 GO TO 1640
3190 RESTORE
3200 GO TO 1400
3210 PRINT '   *** END   ***'
3220 STOP
3230 END
```

```
1990 REM       *** RANDC ***
2000 PRINT 'RANDOM NO. =?'
2010 INPUT R9
2020 G1=N7*550000
2030 PRINT 'NO. TRIALS = ?'
2040 INPUT Z
2050 K1 = 1
2060 REM K1 & Z ARE RESET IF
2070 REM MORE TRIALS ARE WANTED
2080 FOR K = K1 TO Z
2090 REM (RE) SET FLAG FOR INCOMPL. SOLN
2100 U6 = 0
2110 REM (RE) SET RESOURCE SUPPLIES
2120 FOR I = 1 TO M7
2130 V(I) = B(I)
2140 U(I) = T(I)
2150 NEXT I
2160 REM SHUFFLE INDEXES TO JOBS
2170 FOR J = N8 TO 1 STEP -1
2180 S1 = J+1
2190 R9=RND(R9)
2200 C2=INT(R9*J)+1
2210 C4=P(S1)
2220 P(S1)=P(C2)
2230 P(C2) = C4
2240 NEXT J
2250 C5 = 0
2260 REM OPTIMIZE JOBS PER SHUFFLED INDEXES
2270 FOR J = 1 TO N
2280 L = P(J)
2290 FOR I = 1 TO M
2300 I6=I
2310 C2 = C(I,L)
2320 IF C2>999000 THEN 2450
2330 REM GET COST & ROW INDEX
2340 C1 = INT(C2/100)
2350 C2=C2-C1*100
2360 REM UNPACK & CHECK RESOURCES
2370 R2=R(I,L)
2380 R1 = INT(R2/1000)
2390 IF R1 > V(C2) THEN 2430
2400 R2 = R2-R1*1000
2410 REM IF RESOURCE OK, GO ASSIGN
2420 IF R2 ≤ U(C2) THEN 2500
2430 NEXT I
2440 REM UNASSIGNED JOB
2450 A(L)=-1
2460 U6 = U6+1
2470 C1 = 500000
2480 GO TO 2540
2490 REM DECREMENT RESOURCES & ASSIGN
2500 V(C2) = V(C2) - R1
2510 U(C2) = U(C2) - R2
2520 A(L)=C2
2530 REM ADD COST TO TOTAL
2540 C5 = C5 + C1
2550 NEXT J
2560 REM PRINT SOLN IF NEW BEST
2570 REM OR ONE OF FIRST FIVE
2580 IF K ≤ 5 THEN 2600
2590 IF C5 ≥ G1 THEN 2930
2600 PRINT
2610 IF C5 ≥ G1 THEN 2640
2620 PRINT '** NEW BEST SOLUTION'
2630 G1 = C5
2640 PRINT 'TRIAL NO. ';K
2650 C5 = C5-U6*500000
2660 IF U6 = 0 THEN 2700
2670 FOR I = 1 TO U6
2680 PRINT '*';
2690 NEXT I
2700 PRINT 'COST = ';C5
2710 FOR I = 1 TO M
2720 Y1=Y(I)
2730 PRINT 'MACH NO. ';Y1;' ASGD TO :'
2740 N1 = 0
2750 FOR J = 1 TO N
2760 IF A(J) ≠ Y1 THEN 2790
2770 N1 = 1
2780 PRINT '    JOB ';Z(J)
2790 NEXT J
2800 IF N1 ≠ 0 THEN 2820
2810 PRINT '*** NOTHING ***'
2820 PRINT 'UNUSED MATL:';V(Y1)
2830 PRINT 'UNUSED TIME:';U(Y1)
2840 PRINT
2850 NEXT I
2860 IF U6=0 THEN 2930
2870 PRINT 'UNASSIGNED JOBS:'
2880 U6=0
2890 FOR J = 1 TO N
2900 IF A(J) > 0 THEN 2920
2910 PRINT '    JOB ';Z(J)
2920 NEXT J
2930 NEXT K
2940 REM CHECK FOR MORE TRIALS
2950 PRINT 'MORE TRIALS?'
2960 INPUT Y$
2970 IF Y$ ≠ 'YES' THEN 3030
2980 PRINT 'HOW MANY?'
2990 K1 = Z + 1
3000 INPUT Z
3010 Z = Z + K1 - 1
3020 GO TO 3149
3030 PRINT
3040 PRINT 'NEW RUN'
3050 INPUT Y$
3060 REM CHECK FOR RERUN OF WHOLE PROB
3070 REM OR NEW RESOURCE SUPPLIES
3080 IF Y$ = 'YES' THEN 3110
3090 IF Y$ ≠ 'RHS' THEN 3130
3100 GO TO 3149
3110 RESTORE
3120 GO TO 3149
3130 PRINT '   ***  END   ***'
3140 STOP
3150 END
```

```
1000 REM *** METHODS ARE IDENTICAL THRU STATEMENT 1980 ***
1010 DIM C(7,10),R(7,10),B(7),T(7),V(7)
1020 DIM A(10),P(10),Y(7),Z(10),U(7)
1030 REM *** PREDEFINED DATA ***
1040 REM SORTED INDEXED COSTS AND
1050 REM PACKED RESOURCE REQUIREMENTS
1060 REM JOB 1
1070 DATA 4001,6307,7306,8305,8903,9002,9704
1080 DATA 61019,13015,48036,62033,69039,18057,58025
1090 REM JOB 2
1100 DATA 4804,5606,6303,6805,8701,9507,999902
1110 DATA 16016,0,28062,38089,12069,95019,50033
1120 REM JOB 3
1130 DATA 2405,4802,5803,6001,6706,7107,9604
1140 DATA 72046,59063,49051,87086,34025,27059,82034
1150 REM JOB 4
1160 DATA 1202,3306,3805,5701,7403,8304,8907
1170 DATA 87082,12067,43015,34034,66011,92048,54019
1180 REM JOB 5
1190 DATA1003,2506,4304,5307,6505,7901,9802
1200 DATA 43050,81039,67081,89072,78049,85061,79062
1210 REM JOB 6
1220 DATA 2501,3606,6405,6703,8004,8907,999902
1230 DATA 74024,0,33025,60061,68089,42011,11014
1240 REM JOB 7
1250 DATA 3702,3904,4405,4503,6806,6907,9601
1260 DATA 53012,89024,46023,92076,31044,84059,32019
1270 REM JOB 8
1280 DATA 1102,2903,4107,7101,7705,7906,9704
1290 DATA 74071,84069,52085,96046,74095,50025,55016
1300 REM JOB 9
1310 DATA 2001,3505,4404,4602,4806,5803,6107
1320 DATA 75072,48091,91055,86059,46089,63059,62049
1330 REM JOB 10
1340 DATA 1002,1106,1601,2105,7107,7304,999903
1350 DATA 53056,71075,0,62056,90047,32048,86075
1360 REM ARRAY DIMENSIONS
1370 M7 = 7
1380 N7 = 10
1390 REM READ PREDEFINED DATA
1400 FOR J = 1 TO N7
1410 FOR I = 1 TO M7
1420 READ C(I,J)
1430 NEXT I
1440 FOR I = 1 TO M7
1450 READ R(I,J)
1460 NEXT I
1470 NEXT J
1480 REM INPUT ADDITIONAL DATA
1490 PRINT 'NO. MACHINES ?'
1500 INPUT M
1510 M9 = M + 1
1520 M8 = M - 1
1530 PRINT 'NO. JOBS ?'
1540 INPUT N
1550 N8 = N-1
1560 PRINT 'ENTER';M;'MACHINE NOS. IN ORDER'
1570 FOR I = 1 TO M
1580 INPUT Y(I)
1590 NEXT I
1600 PRINT 'ENTER';N;'JOB NOS. IN ORDER'
1610 FOR J = 1 TO N
1620 INPUT Z(J)
1630 NEXT J
1640 PRINT 'ENTER: MATL THEN TIME FOR'
1650 FOR I = 1 TO M
1660 Y1= Y(I)
1670 PRINT 'MACHINE #';Y(I)
1680 INPUT B(Y1),T(Y1)
1690 NEXT I
1700 REM CHECK IF RERUN WITH
1710 REM NEW RESOURCE SUPPLIES
1720 IF Y$ = 'RHS'THEN 2000
1730 REM INITIALIZE JOB INDEXES
1740 REM FOR SHUFFLING AND
1750 REM COMPRESS COSTS & RESOURCES
1760 REM INTO UPPER LEFT CORNER
1770 REM OF DATA MATRICES
1780 FOR J = 1 TO N
1790 P(J) = J
1800 L= Z(J)
1810 I2 = 1
1820 FOR I = 1 TO M7
1830 C2 = C(I,L)
1840 K = C2-100*(INT(C2/100))
1850 FOR I1 = 1 TO M
1860 K1= Y(I1)
1870 IF K ≠ K1 THEN 1920
1880 U(I2)=R(K,L)
1890 V(I2)=C2
1900 I2=I2+1
1910 IF I2 > M THEN 1940
1920 NEXT I1
1930 NEXT I
1940 FOR I=1 TO M
1950 R(I,J)=U(I)
1960 C(I,J)=V(I)
1970 NEXT I
1980 NEXT J
```

APPENDIX D

OUTPUT SAMPLES FROM PROGRAMS IN APPENDIX C

```
NO. MACHINES ?
? 2
NO. JOBS ?
? 4
ENTER 2    MACHINE NOS. IN ORDER
? 1
? 3
ENTER 4    JOB NOS. IN ORDER
? 1
? 2
? 3
? 5
ENTER: MATL THEN TIME FOR
MACHINE # 1
? 140,150
MACHINE # 3
? 150,130

 SOLUTION 1
*COST =   137
MACH NO.   1      ASGD TO :
   JOB  1
   JOB  2
UNUSED MATL: 63
UNUSED TIME: 115

MACH NO.   3      ASGD TO :
   JOB  5
UNUSED MATL: 83
UNUSED TIME: 49

UNASSIGNED JOBS:
   JOB  3

 SOLUTION 2
*COST =   137
MACH NO.   1      ASGD TO :
   JOB  1
   JOB  2
UNUSED MATL: 63
UNUSED TIME: 115

MACH NO.   3      ASGD TO :
   JOB  5
UNUSED MATL: 83
UNUSED TIME: 49

UNASSIGNED JOBS:
   JOB  3

 SOLUTION 3
*COST =   110
MACH NO.   1      ASGD TO :
   JOB  1
   JOB  3
UNUSED MATL: 7
UNUSED TIME: 85

MACH NO.   3      ASGD TO :
   JOB  5
UNUSED MATL: 83
UNUSED TIME: 49

UNASSIGNED JOBS:
   JOB  2

 SOLUTION 4
*COST =   137
MACH NO.   1      ASGD TO :
   JOB  1
   JOB  2
UNUSED MATL: 63
UNUSED TIME: 115

MACH NO.   3      ASGD TO :
   JOB  5
UNUSED MATL: 83
UNUSED TIME: 49

UNASSIGNED JOBS:
   JOB  3

 SOLUTION 5
 COST =   240
MACH NO.   1      ASGD TO :
   JOB  1
   JOB  5
UNUSED MATL: 36
UNUSED TIME: 81

MACH NO.   3      ASGD TO :
   JOB  2
   JOB  3
UNUSED MATL: 73
UNUSED TIME: 17

NEW RUN
? NO
  ***  END   ***

TIME 0.3 SECS.
```

```
NO. MACHINES ?
? 2
NO. JOBS ?
? 4
ENTER 2    MACHINE NOS. IN ORDER
? 1
? 3
ENTER 4    JOB NOS. IN ORDER
? 1
? 2
? 3
? 5
ENTER: MATL THEN TIME FOR
MACHINE # 1
? 140,150
MACHINE # 3
? 150,130
RANDOM NO. =?
? 5217847
NO. TRIALS = ?
? 10

** NEW BEST SOLUTION
TRIAL NO.  1
*COST =  137
MACH NO.  1     ASGD TO :
   JOB  1
   JOB  2
UNUSED MATL: 63
UNUSED TIME: 115

MACH NO.  3     ASGD TO :
   JOB  5
UNUSED MATL: 83
UNUSED TIME: 49

UNASSIGNED JOBS:
   JOB  3

** NEW BEST SOLUTION
TRIAL NO.  2
COST =  240
MACH NO.  1     ASGD TO :
   JOB  1
   JOB  5
UNUSED MATL: 36
UNUSED TIME: 81

MACH NO.  3     ASGD TO :
   JOB  2
   JOB  3
UNUSED MATL: 73
UNUSED TIME: 17

TRIAL NO.  3
COST =  240
MACH NO.  1     ASGD TO :
   JOB  1
   JOB  5
UNUSED MATL: 36
UNUSED TIME: 81

MACH NO.  3     ASGD TO :
   JOB  2
   JOB  3
UNUSED MATL: 73
UNUSED TIME: 17


TRIAL NO.  4
COST =  246
MACH NO.  1     ASGD TO :
   JOB  2
   JOB  3
UNUSED MATL: 52
UNUSED TIME: 88

MACH NO.  3     ASGD TO :
   JOB  1
   JOB  5
UNUSED MATL: 35
UNUSED TIME: 13


TRIAL NO.  5
COST =  240
MACH NO.  1     ASGD TO :
   JOB  1
   JOB  5
UNUSED MATL: 36
UNUSED TIME: 81

MACH NO.  3     ASGD TO :
   JOB  2
   JOB  3
UNUSED MATL: 73
UNUSED TIME: 17

MORE TRIALS?
? NO

NEW RUN
? NO
   *** END  ***

TIME 0.3 SECS.
```

APPENDIX E

PROGRAM FOR ARTILLERY PROBLEM

```
      C************************************************************************
      C************************************************************************
      C************************************************************************
      C****                                                              *****
      C****                                                              *****
      C****       A FAST HEURISTIC FOR ASSIGNING WEAPONS TO TARGETS      *****
      C****                                                              *****
      C****                                                              *****
      C****                            BY                                *****
      C****                                                              *****
      C****                     HENRY C. THIBAULT                        *****
      C****                            AND                               *****
      C****                      KENNETH E. CASE                         *****
      C****                                                              *****
      C****                                                              *****
      C****                  OKLAHOMA STATE UNIVERSITY                   *****
      C****                                                              *****
      C****                     SEPTEMBER 1,1977                         *****
      C****                                                              *****
      C****                                                              *****
      C************************************************************************
      C************************************************************************
      C************************************************************************
      C
      C
      C
      C
      C
      C     ***************************************************************
      C     *                                                             *
      C     *                  ***  MAIN PROGRAM  ***                     *
      C     *                                                             *
      C     *                                                             *
      C     *     MAIN FUNCTIONS:                                         *
      C     *                                                             *
      C     *          1. READ AND PRINT INPUT DATA                       *
      C     *          2. MAKE PRELIMINARY CALCULATIONS                   *
      C     *          3. FIND UNCONSTRAINED OPTIMUM                      *
      C     *          4. ADD PHANTOM UNIT                                *
      C     *          5. CALL SOLUTION AND OUTPUT SUBROUTINES            *
      C     *                                                             *
      C     *                                                             *
      C     *     NOTE:  THIS MAIN PROGRAM AND THE OUTPUT AND CHART       *
      C     *     SUBROUTINES EXIST PRIMARILY TO DEMONSTRATE HOW TO       *
      C     *     GENERATE AND PRINT THE DATA NEEDED BY AND PRODUCED BY   *
      C     *     THE SOLUTION SUBROUTINES.                               *
      C     *                                                             *
      C     *                                                             *
      C     ***************************************************************
      C
      C
0001            COMMON /ACOM2/ NALPHA
0002            COMMON /PRCOM/ IPX(20)
0003            COMMON /UCOM/ CVEC(1200),RVEC(1200),EVEC(1200),SVEC(1200),
               *A(40),SU(40),T1(40),TU(40),TIME,NT,NU,NN,P,ISAME,PVEC(1200)
0004            DIMENSION C(40,30),R(40,30),E(40,30),S(40,30),P(40,30)
0005            EQUIVALENCE (C(1),CVEC(1)),(R(1),RVEC(1)),(E(1),EVEC(1))
0006            EQUIVALENCE (S(1),SVEC(1)),(P(1),PVEC(1))
```

```
0007           COMMON /SCHED/ SPSTRT(30),SPSTOP(30),NSTART(40),STARTS(1200),
              *STOPS(1200),SHELLS(1200),NTARG(1200),NSRANK(1200),NFIRST(40),
              *INFEAS(30)
0008           DIMENSION SCSTA(40,30),SCSTP(40,30),SCROS(40,30),NSCTRG(40,30),
              *NRANKS(40,30)
0009           EQUIVALENCE (STARTS(1),SCSTA(1)),(STOPS(1),SCSTP(1)),
              * (SHELLS(1),SCROS(1)),(NTARG(1),NSCTRG(1)),(NSRANK(1),NRANKS(1))
0010           COMMON /COMG/ IGMAX,IGX(40),IGG(40)
0011           COMMON /COM2/ IP(30),IG(40),MSIJ(1200),MIIJ(90),MAXPRI
0012           DIMENSION MS(40,30),MI(30,3)
0013           EQUIVALENCE (MS(1),MSIJ(1)),(MI(1),MIIJ(1))
0014           COMMON /COMX/ LGRNG(40)
0015           COMMON /OCOM/ LINE(101),ASTARS(101)
0016           COMMON /ACOM/ ALPHA,TOPINE
0017           COMMON /DCOM/ MAXROW,MAXCOL
0018           DATA INDCMS,INDCSS /0,0/
0019           DIMENSION IZONK(40)
         C
         C
         C
         C
         C                      *****************
         C                      *               *
         C                      * ************* *
         C                      * *           * *
         C                      * *  WARNING  * *
         C                      * *           * *
         C                      * ************* *
         C                      *               *
         C                      *****************
         C
         C         THE FOLLOWING TWO STATEMENTS MUST ALWAYS SET "MAXROW" AND "MAXCOL"
         C         EQUAL TO THE DIMENSIONS DEFINED ABOVE FOR THE MAIN ARRAYS.
         C
0020           MAXROW=40
0021           MAXCOL=30
         C
         C*****                    *****************************
         C*****                    * READ & DISPLAY INPUT DATA *
         C*****                    *****************************
         C*****
         C*****           ************************************************
         C*****           * NOTE THAT ALL "READ", "PRINT", AND "FORMAT"     *
         C*****           * STATEMENTS DEALING WITH BASIC PROBLEM DATA HAVE *
         C*****           * BEEN LOCATED HERE TO FACILITATE PROGRAM         *
         C*****           * MAINTENANCE                                     *
         C*****           ************************************************
         C
         C*****    READ MISSION TIME, NO. TGTS, NO. ROWS(=NO. FUZE/ROUND COMBOS),
         C*****    MAX INEFFICIENCY ALLOWED. CONTROL FOR PRINTING INPUT DATA,
         C*****    NUMBER OF ALPHAS TO BE TRIED  (SHOULD BE ODD, 1 WILL BE ADDED IF EVEN),
         C*****    FRACTION OF MISSION TIME WHICH DEFINES "PLENTY OF TIME."
         C
0022           READ 1,TIME,NT,NU,TOPINE,IPRINT,NALPHA,PLENTY
0023         1 FORMAT(F10.0,2I5,F5.0,2I5,F5.0)
0024           IF(TOPINE.LT. .01)TOPINE=.5
         C
         C*****    READ COST MATRIX ("NSCORE")
```

```
                C
0025                  DO 10 I=1,NU
0026                  READ 3,(C(I,J) ,J=1,NT)
                C
                C*****   THE FOLLOWING FORMAT STATEMENTS ARE USED FOR READING ALL DATA
                C*****   EXCEPT MISSION TIME, NO. TGTS, NO. UNITS, AND MAX. INEFFICIENCY
                C*****   (ALL OF WHICH ARE IN THE FIRST DATA CARD), AND SCHEDULING INFO
                C*****   (WHICH IS IN THE VERY LAST SET OF DATA CARDS). SEE DOCUMENTATION FOR
                C*****   INSTRUCTIONS FOR CHANGING METHODS OF DATA INPUT.
                C
0027                3 FORMAT(20F4.0)
0028               33 FORMAT(20I4)
0029                  DO 399 J=1,NT
0030                  IF(C(I,J).GT.9900)C(I,J)=1000000.
0031              399 CONTINUE
0032               10 CONTINUE
                C
                C*****   READ MATRIX OF ROUND REQUIREMENTS ("NROUNDS")
                C
0033                  DO 20 I=1,NU
0034                  READ 3, (R(I,J),J=1,NT)
0035               20 CONTINUE
                C
                C*****   READ AMMUNITION SUPPLIES
                C
0036                  READ 3,(A(I),I=1,NU)
                C
                C*****   READ TIMES FOR SETUP AND FIRST ROUND
                C
0037                  READ 3,(SU(I),I=1,NU)
                C
                C*****   READ TIMES PER ROUND (SUSTAINED FIRE)
                C
0038                  READ 3,(T1 (I),I=1,NU)
                C
                C*****   READ NO. TUBES FOR EACH ROW
                C
0039                  READ 3,(TU(I),I=1,NU)
                C
                C*****   READ NO'S. OF UNITS CORRESPONDING TO ROWS (MUST BE IN ASCENDING ORDER)
                C
0040                  READ 33,(IG(I),I=1,NU)
                C
                C*****   READ TARGET PRECEDENCES (MUST BE IN ASCENDING ORDER)
                C
0041                  READ 33,(IP(J),J=1,NT)
                C
                C*****   CLEAR MASSING AND SCHEDULING ARRAYS
                C
0042                  NN=NU+1
0043                  DO 1020 J=1,NT
0044                  MI(J,1)=1
0045                  MI(J,2)=0
0046                  MI(J,3)=0
0047                  SPSTRT(J)=9999.
0048                  SPSTOP(J)=9999.
0049                  DO 1020 I=1,NN
0050                  SCSTA(I,J)=0.
```

```
            SCSTP(I,J)=0.
            SCRDS(I,J)=0.
            NSCTRG(I,J)=0.
            MS(I,J)=0
       1020 CONTINUE
            DO 1021 I=1,NN
            NSTART(I)=0
            NFIRST(I)=0
            IGG(I)=0
       1021 CONTINUE
      C
      C*****   READ AND STORE MASSING INFO:
      C*****   A.  READ TGT NO., NO. UNITS TO BE MASSED, AND NO. ROWS FOR
      C*****       PRIMARY (MC ) AND SECONDARY (MCC) CONSIDERATION
      C*****       FOR MASSING.  (TGT NO. .GT. "NT" INDICATES END OF MASSING INFO.)
      C
       1022 READ 33,MX,MN,MC,MCC
            IF(MX.GT.NT)GO TO 1030
            INDCMS=1
      C
      C*****   B.  READ INDEXES OF ROWS FOR PRIMARY, THEN SECONDARY, CONSIDERATION
      C*****       FOR MASSING
      C
            READ 33,(MS(I,MX),I=1,MC)
            IF(MCC.LT.1) GO TO 1025
            MCP1=MC+1
            MCC=MCC+MC
            READ 33,(MS(I,MX),I=MCP1,MCC)
      C
      C*****   C.  STORE MASSING CONTPOL INFO FOR TGT NO. "MX"
      C
            MI(MX,3)=MCC-MC
       1025 MI(MX,1)=MN
            MI(MX,2)=MC
      C*****   ADJUST C AND R FOR MASSING WHEN R-VALUE < 1 VOLLEY
            IF(MC.LT.2)GO TO 1022
            DO 1027 I=1,NU
            CIMX=C(I,MX)
            IF(CIMX.GT.9900.)GO TO 1027
            TUIX=TU(I)
            RIMX=R(I,MX)
            IF(TUIX.LE.RIMX)GO TO 1027
            C(I,MX)=CIMX*TUIX/RIMX
       1027  CONTINUE
            GO TO 1022
      C
      C*****   READ SCHEDULING INFO:
      C*****   A.  READ TGT NO., START & STOP TIMES (TGT NO. .GT. "NT" INDICATES
      C*****       END OF SCHEDULING INFO).  NOTE USE OF SPECIAL FORMAT STATEMENT.
      C
       1030 READ 1033,J,SSJ1,SSJ2
       1033 FORMAT(I4,2F8.0)
            IF(J.GT.NT)GO TO 1666
            INDCSS=1
            IF((SSJ2.LE.SSJ1).AND.(SSJ1.LT.TIME))SSJ2=9999.
            SPSTRT(J)=SSJ1
            SPSTOP(J)=SSJ2
            GO TO 1030
```

```
             C
             C*****   PRINT INPUT DATA
             C
0090          1666 IF(IPRINT.EQ.0)GO TO 1667
0091          1066 PRINT 6
0092             6 FORMAT(1H1,/,13(1H0,/),51X,29(1H*),/,51X,1H*,27X,1H*,/,
                  *51X,29H*   SUMMARY OF INPUT DATA   *,/,51X,1H*,27X,1H*,/,51X,
                  *29(1H*))
0093               PRINT 2,TIME,NT,NU,TOPINE,NALPHA
0094             2 FORMAT(18H1MISSION DURATION:,F8.2,/,13H0NO. TARGETS:,I4,/,
                  *10H0NO. ROWS:,I3,/,27H0MAX. INEFFICIENCY ALLOWED:,F7.4,/,
                  *24H0NO. ALPHAS TO BE TRIED:,I3,/,
                  *13H1COST MATRIX:,/,5H0ROW:)
0095               DO 43 I=1,NU
0096               PRINT 4,I,(C(I,J),J=1,NT)
0097             4 FORMAT(1H0,I3,1H:,15F8.0,/,(5X,15F8.0))
0098            43 CONTINUE
0099               PRINT 5
0100             5 FORMAT(42H1NO. ROUNDS NEEDED OF FUZE I FOR TARGET J:,/,5H0ROW:)
0101               DO 200 I=1,NU
0102               PRINT 4,I,(R(I,J),J=1,NT)
0103           200 CONTINUE
0104               DO 1060 I=1,NU
0105          1060 LINE(I)=I
0106               PRINT 1070,(LINE(I),I=1,NU)
0107          1070 FORMAT(1H1,51X,27H***   UNIT PARAMETERS   ***,/////,
                  *9H ROW NO.:,/,(15I8))
0108               PRINT 7,(A(I),I=1,NU)
0109             7 FORMAT(20H0AMMO SUPPLY VECTOR:,/,(15F8.0))
0110               PRINT 37,(SU(I),I=1,NU)
0111            37 FORMAT(///,47H VECTOR OF TIMES (MIN) FOR SETUP & FIRST ROUND:,/,
                  *(15F8.3))
0112               PRINT 47,(T1(I),I=1,NU)
0113            47 FORMAT(///,50H VECTOR OF TIMES (MIN) PER ROUND (SUSTAINED FIRE):,
                  */,(15F8.3))
0114               PRINT 57,(TU(I),I=1,NU)
0115            57 FORMAT(///,29H VECTOR OF NO. TUBES PER ROW:,/,(15F8.0))
0116               PRINT 1103,(IG(I),I=1,NU)
0117          1103 FORMAT(///,30H VECTOR OF UNIT GROUP NUMBERS:,/,(15I8))
0118               PRINT 1101
0119          1101 FORMAT(1H1,50X,29H***   TARGET PARAMETERS   ***,/////)
0120          1667 DO 1105 I=1,20
0121               NL=0
0122               DO 1106 J=1,NT
0123               IF(IP(J).NE.I)GO TO 1106
0124               NL=NL+1
0125               LINE(NL)=J
0126          1106 CONTINUE
0127               IF(NL.EQ.0)GO TO 1105
0128               MAXPRI=I
0129               IF(IPRINT.EQ.0)GO TO 1105
0130               PRINT 1107,I,(LINE(J),J=1,NL)
0131          1107 FORMAT(11H0PRECEDENCE,I3,9H TARGETS:,25I4,(23X,25I4))
0132          1105 IPX(I)=NL
0133               IF(IPRINT.EQ.0)GO TO 1668
0134               IF(INDCMS.EQ.0)GO TO 1115
0135               PRINT 1102
0136          1102 FORMAT(21H1MASSING INFORMATION:,//,
```

```
     *39H TGT  NO. UNITS  ROWS TO BE CONSIDERED:,/,
     *54H NO.   TO MASS   (OTHERS HAVE BEEN FLAGGED INFEASIBLE))
      DO 1110 J=1,NT
      K=MI(J,1)
      IF(K.EQ.1)GO TO 1110
      L=MI(J,2)
      PRINT 1111,J,K,(MS(I,J),I=1,L)
 1111 FORMAT(1H0,I3,I7,5X,9H PRIMARY:,I6,24I4,(27X,25I4))
      MIJ3=MI(J,3)
      IF(MIJ3.LT.1)GO TO 1110
      LP1=L+1
      MCC=L+MIJ3
      PRINT 1112,(MS(I,J),I=LP1,MCC)
 1112 FORMAT(17X,10HSECONDARY:,25I4,(27X,25I4))
 1110 CONTINUE
 1115 IF(INDCSS.EQ.0)GO TO 1147
      PRINT 1120
 1120 FORMAT(46H1START-STOP INFORMATION (9999 = NOT SPECIFIED))
      PRINT 1122
 1122 FORMAT(///,32H TARGET   START TIME   STOP TIME,//)
      DO 1125 J=1,NT
      IF((SPSTRT(J).GT.9990.).AND.(SPSTOP(J).GT.9990.))GO TO 1125
      PRINT 1126,J,SPSTRT(J),SPSTOP(J)
 1126 FORMAT(I5,F14.3,F12.3)
 1125 CONTINUE
C
C*****                 ****************************
C*****                 * PRELIMINARY CALCULATIONS *
C*****                 ****************************
C
C
C*****   PRINT HEADING
C
 1147 PRINT 1006
 1006 FORMAT(1H1,/,13(1H0,/),43X,43(1H*),/,43X,1H*,41X,1H*,/,
     *43X,43H*   RESULTS OF PRELIMINARY CALCULATIONS   *,/,
     *43X,1H*,41X,1H*,/,43X,43(1H*))
C
C*****   ADJUST MISSION TIME AND AMMO SUPPLIES TO ALLOW FOR ROUNDOFF
C*****   ERROR WHEN SUBTRACTING USAGES DURING EXECUTION OF SOLUTION
C
 1668 TIME=TIME*1.00001
      DO 269 I=1,NU
  269 A(I)=A(I)*1.00001
C
C*****   ESTIMATE (FOR USE IN DO-LOOPS BELOW) AN UPPER LIMIT TO DEFINE WHEN A
C*****   UNIT HAS "PLENTY OF TIME" TO COVER ALL TARGETS
C
      TIME75=TIME*PLENTY
C
C*****   CLEAR SUMS FOR ESTIMATING ALPHA AND CALCULATING FACTOR FOR
C*****   BALANCING INEFFICIENCIES TO COSTS
C
      SUMC=0.
      SUMS=0.
      SSS=0.
C
      DO 77 I=1,NU
```

```
      SUMEI=0.
      SUMAMO=0.
      DO 76 J=1,NT
C
C*****   FLAG INFEASIBILITIES IN AMMO, TIME, AND INEFFICIENCY MATRICES
C
      IF (C (I,J).LT.990000.)GO TO 762
 2762 R(I,J)=0.
      E(I,J)=1000000.
      S(I,J)=1000000.
      GO TO 76
C
C*****   CALCULATE ENGAGEMENT TIMES
C
  762 EX=R(I,J)/TU(I)-1.
      IF((EX-IFIX(EX)).GT.0.001)EX=EX+1.
      EX=IFIX(EX)*T1(I)
      IF(EX.LT.0.)EX=0.
      E(I,J)=EX+SU(I)
C
C*****   ADD INFEASIBILITIES FOR NON-MASS TGTS DUE TO TIME OR AMMO
C
      IF((MI(J,1).GT.1).OR.((E(I,J).LT.TIME).AND.(R(I,J).LT.A(I))))
     *   GO TO 761
      C(I,J)=1000000.
      GO TO 2762
C
C*****   CALCULATE TIME AND AMMO INEFFICIENCIES
C
  761 R1=E(I,J)/TIME
      R2=R(I,J)
C*****   ADJUST FOR ONE-VOLLEY MINIMUM ON MASS TGT IF NEEDED
      IF((MI(J,1).GT.1).AND.(TU(I).GT.R2))R2=TU(I)
      R2=R2/A(I)
C
C*****   BUILD INEFFICIENCY MATRIX (APPLYING MAX ALLOWABLE INEFFICIENCY)
C
      IF(R1.GE.R2) S(I,J)=R1
      IF(R2.GT.R1)S(I,J)=R2
      IF(S(I,J).GT.TOPINE)S(I,J)=TOPINE
C
C*****   SUM AND COUNT FEASIBLE COSTS AND INEFFICIENCIES FOR LATER
C*****   CALCULATIONS OF FACTOR FOR BALANCING COSTS TO INEFFICIENCIES
C
      SUMC=SUMC+C(I,J)
      SUMS=SUMS+S(I,J)
      SSS=SSS+1.
      SUMAMO=SUMAMO+R(I,J)
      SUMEI=SUMEI+E(I,J)
   76 CONTINUE
C
C*****   SET SWITCH TO TURN OFF CONSIDERATION OF INEFFICIENCIES FOR UNITS
C*****   WITH PLENTY OF TIME AND AMMO TO COVER ALL POSSIBLE TARGETS
C
      LGRNG(I)=0
      IF(SUMEI.GT.(TIME75) .AND. SUMAMO.GT.A(I)) LGRNG(I)=1
   77 CONTINUE
C
```

```
C*****   "B" IS FACTOR FOR BALANCING COSTS & INEFFICIENCIES
C
         B=SUMC/SUMS
C
C*****   COMPLETION OF MASSING SUBSCRIPT ("MS") ARRAY:
C
C        A.  COLUMNS OF "MS" CORRESPONDING TO MASSED TGTS WILL CONTAIN INDEXES
C            TO ROWS GROUPED AS FOLLOWS FROM TOP TO BOTTOM:
C            1.  INDEXES TO ROWS FOR PRIMARY MASSING CONSIDERATION.
C            2.  INDEXES TO ROWS FOR SECONDARY MASSING CONSIDERATION.
C            3.  INDEX TO "PHANTOM" UNIT
C            4.  INDEXES TO INFEASIBLE ROWS
C        B.  OTHER TGTS:
C            1.  INDEXES TO FEASIBLE ROWS
C            2.  INDEX TO "PHANTOM" UNIT
C            3.  INDEXES TO INFEASIBLE ROWS
C        C.  FOR THE JTH TARGET:
C            1.  MI(J,1) = NO. UNITS TO MASS (= 1 FOR B. ABOVE)
C            2.  MI(J,2) = NO. PRIMARY ROWS FOR A. ABOVE;
C                        = NO. FEASIBLE ROWS FOR B. ABOVE
C            3.  MI(J,3) = NO. SECONDARY ROWS (= 0 FOR B. ABOVE)
C        NOTE:  A.1. AND A.2. WERE DONE WHEN MASSING INFO WAS READ FROM CARDS.
C               C. HAS BEEN PARTIALLY DONE.
C
         INXXA=NU+1
         DO 2000 J=1,NT
         MIJ1=MI(J,1)
         MIJ2=MI(J,2)
         MIJ3=MI(J,3)
         MI23=MIJ2+MIJ3
         IF(MIJ1.EQ.1)GO TO 2004
         DO 2001 I=1,NU
    2001 IZONK(I)=1
         DO 2002 I=1,MI23
         NOZONK=MS(I,J)
    2002 IZONK(NOZONK)=0
         INXI=INXXA
         DO 2003 I=1,NU
         IF(IZONK(I).EQ.0) GO TO 2003
         C(I,J)=1000000.
         R(I,J)=0.
         E(I,J)=1000000.
         S(I,J)=1000000.
         MS(INXI,J)=I
         INXI=INXI-1
    2003 CONTINUE
         GO TO 2007
    2004 MC=0
         INXI=INXXA
         DO 2006 I=1,NU
         IF(C(I,J).LT.990000.)GO TO 2005
         MS(INXI,J)=I
         INXI=INXI-1
         GO TO 2006
    2005 MC=MC+1
         MS(MC,J)=I
    2006 CONTINUE
         MI(J,2)=MC
```

```
0237        2007 MS(INXI,J)=INXXA
0238        2000 CONTINUE
            C
            C*****              ******************************
            C*****              * FIND UNCONSTRAINED OPTIMUM *
            C*****              ******************************
            C
0239              SUMCOL=0.
0240              DO 65 I=1,NT
0241              COLMIN=1000000
0242              DO 64 J=1,NU
0243              IF(C(J,I).LT.COLMIN)COLMIN=C(J,I)
0244          64 CONTINUE
0245              IF(COLMIN.GT.990000)COLMIN=0.
0246              SUMCOL=SUMCOL+COLMIN
0247          65 CONTINUE
0248              IF (IPRINT.EQ.0)GO TO 1669
0249              PRINT 66,SUMCOL
0250          66 FORMAT(1H1,45X,27H SUM OF COLUMN COST MINIMA:,F9.2,/////)
            C
            C*****   PRINT ENGAGEMENT TIMES CALCULATED ABOVE
            C
0251              PRINT 67
0252          67 FORMAT (28H MATRIX OF ENGAGEMENT TIMES:,/,
                *57H (NOTE NEW INFEASIBILITIES DUE TO MASSING, TIME, OR AMMO),/,
                *5H0ROW:)
0253        1669 CONTINUE
0254              DO 8 I=1,NU
0255              IF(IPRINT.EQ.0)GO TO 4069
0256              PRINT 88,I,(E(I,J),J=1,NT)
0257          88 FORMAT (1H0,I3,1H:,15(1X,F7.3),/,(5X,15(1X,F7.3)))
            C
            C
            C*****             ***************************************************
            C*****             * SET UP FUZING INDEXES TO SPEED UP ASGMT LOGIC *
            C*****             ***************************************************
            C
            C
            C*****   ZERO OUT COUNTS OF ROWS IN UNIT GROUPS
            C
0258        4069 DO 4000 K=1,NU
0259        4000 IGX(K)=0
0260              IGMAX=0
0261              DO 5000 K=1,NU
            C
            C*****   FIND OUT WHAT UNIT GROUP EACH ROW BELONGS TO AND ADD 1
            C*****   TO COUNT OF ROWS IN THAT UNIT GROUP
            C
0262              IGXX=IG(K)
0263              IGX(IGXX)=IGX(IGXX)+1
            C
            C*****   FIND MAX OF ALL UNIT GROUPS FOR LATER USE WITH DUMMY UNIT
            C
0264              IF(IGXX.GT.IGMAX)IGMAX=IGXX
0265        5000 CONTINUE
            C-
            C*****   DETERMINE AND SAVE THE ROW EACH UNIT GROUP STARTS IN
            C
```

```
          KK=1
          DO 5050 K=1,IGMAX
          IGG(K)=KK
          KK=KK+IGX(K)
     5050 CONTINUE
C*****    APPLY BALANCING FACTOR TO INEFFICIENCIES AND PRINT THE
C*****    BALANCED INEFFICIENCIES
C
          DO 8 J=1,NT
          S(I,J)=B*S(I,J)
        8 CONTINUE
          IF(IPRINT.EQ.0)GO TO 1670
          PRINT 82,B
       82 FORMAT(4H1B(=,F8.3,31H)-WEIGHTED MAX(R/A,E/T)-MATRIX:,/,5H0ROW:)
          DO 89 I=1,NU
          PRINT 88,I,(S(I,J),J=1,NT)
       89 CONTINUE
C
C*****                       ********************
C*****                       * ADD PHANTOM UNIT *
C*****                       ********************
C
     1670 NN=NU+1
          DO 9 I=1,NT
          S(NN,I)=500000.
          C(NN,I)=500000.
          R(NN,I)=1.
          E(NN,I)=TIME/(NT+100)
          P(NN,I)=500000.
        9 CONTINUE
          A(NN)=2000000000.
          LGRNG(NN)=0
          SU(NN)=.001
          TI(NN)=.001
          TU(NN)=1
          IG(NN)=0
C
C*****           **********************************************
C*****           * CALL SUBROUTINES FOR SOLUTIONS AND OUTPUT *
C*****           **********************************************
C
C
C*****    CALL MAIN CONTROL SUBROUTINE (WHICH CALLS ALL OTHERS, INCLU-
C*****    DING OUTPUT)
C
          CALL FRBIAS
          PRINT 9999
     9999 FORMAT(1H1,50X,27H***  NORMAL END OF JOB  ***)
          STOP
          END
```

```
*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = MAIN    , LINECNT =      60
*STATISTICS*    SOURCE STATEMENTS =      298,PROGRAM SIZE =     9234
*STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
0001              SUBROUTINE FRBIAS
            C
            C                  ************************************
            C                  *                                  *
            C                  *   ***************************    *
            C                  *   *                         *    *
            C                  *   *  MAIN CONTROL SUBROUTINE *   *
            C                  *   *                         *    *
            C                  *   ***************************    *
            C                  *                                  *
            C                  ************************************
            C
0002              COMMON /AICD/ ICODE
0003              COMMON /ACOM2/ NALPHA
0004              COMMON /FRCOM/ IPX(20)
0005              COMMON /UCOM/ CVEC(1200),RVEC(1200),EVEC(1200),SVEC(1200),
                 *A(40),SU(40),T1(40),TU(40),TIME,NT,NU,NN,B,ISAME,PVEC(1200)
0006              COMMON /SCHED/ SPSTRT(30),SPSTOP(30),NSTART(40),STARTS(1200),
                 *STOPS(1200),SHELLS(1200),NTARG(1200),NSRANK(1200),NFIRST(40),
                 *INFEAS(30)
0007              COMMON /COMG/ IGMAX,IGX(40),IGG(40)
0008              COMMON /COM2/ IP(30),IG(40),MSIJ(1200),MIIJ(90),MAXPRI
0009              COMMON /COMX/ LGRNG(40)
0010              COMMON /ACOM/ ALPHA,TOPINE
0011              COMMON /DCOM/ MAXROW,MAXCOL
            C
0012              DAL1=0.
0013              DAL2=0.
            C*****   NALPHA MUST BE ODD FOR THIS ROUTINE TO WORK
0014              IF(MOD(NALPHA,2).NE.1)NALPHA=NALPHA+1
            C*****   PUT PAGE EJECT IN FRONT OF RESULTS FOR EACH ALPHA
0015              NEJ=1
            C*****   IF ONLY ALPHA=0 IS WANTED, GO DO IT
0016              IF (NALPHA.EQ.1)GO TO 10
            C*****   CALCULATE INCREMENTS FOR ALPHA
0017              NALM1=NALPHA-1
0018              DAL1=0.5/FLOAT(NALM1)
0019              DAL2=3.*DAL1
            C*****   CALCULATE INDEX FOR SWITCHING FROM SMALL TO LARGE INCREMENT IN ALPHA
0020              NSWICH=NALM1/2
            C
0021           10 ALPHA=0.
            C*****   DO-LOOP TRIES "NALPHA" DIFFERENT ALPHA VALUES FROM 0 TO 1 INCLUSIVE
0022              DO 100 NB=1,NALPHA
0023              IF(NB.EQ.NALPHA) ALPHA=1.
0024              IF(NALPHA.EQ.1)ALPHA=0.
            C
            C*****   INNER LOOP COMBINES COST AND INEFFICIENCY MATRICES ACCORDING TO ALPHA.
            C*****   VECTOR ADDRESSING IS USED TO SAVE TIME.
            C
0025              ALCOMP=1.-ALPHA
0026              J=MAXROW*NT
0027              IADD=MAXROW-NN-1
0028              I=0
0029           20 I=I+1
0030              LI=MOD(I,MAXROW)
0031              IF((LI.LE.NN).AND.(LI.NE.0))GO TO 12
0032              IF(LI.GT.NN)I=I+IADD
```

```
0033            GO TO 23
0034         12 CVI=CVEC(I)
0035            IF(CVI.GT.900000.)GO TO 14
0036            PVI=CVI
0037            IF(LGRNG(LI).EQ.0)GO TO 15
0038            IF(NB.EQ.1)GO TO 15
0039            PVI=CVI*ALCOMP+SVEC(I)*ALPHA
0040            GO TO 15
0041         14 PVI=1000000.
0042         15 PVEC(I)=PVI
0043            IF(NB.EQ.1)GO TO 23
          C*****   CLEAR ANY INTERMEDIATE INFEASIBILITY FLAGS FROM PREV A LPHA
0044            MSIJI=MSIJ(I)
0045            IF(MSIJI.LT.0)MSIJ(I)=(-MSIJI)
0046         23 IF(I.LT.J)GO TO 20
          C
          C*****   CALL SOLUTION ROUTINE FOR EACH ALPHA.
          C
          C*****   (RE)INITIALIZE OUTPUT ROUTINE
0047            ICODE=0
0048            CALL OUTPUT
          C*****   CALL SOLUTION ROUTINE
0049            CALL VOEGLN
          C*****   SIGNAL COMPLETE SOLUTION TO OUTPUT ROUTINE
0050            ICODE=4
0051            CALL OUTPUT
          C
          C
          C*****   CALCULATE NEXT VALUE OF ALPHA
          C
0052            IF(NALPHA.EQ.1)GO TO 100
0053            IF(NB.LE.NSWICH)ALPHA=ALPHA+DAL1
0054            IF(NB.GT.NSWICH)ALPHA=ALPHA+DAL2
0055        100 CONTINUE
0056            PRINT 69
0057         69 FORMAT('1',10X,'***  END  ***')
0058            RETURN
0059            END

 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = FRBIAS  , LINECNT =       60
 *STATISTICS*    SOURCE STATEMENTS =       59,PROGRAM SIZE =     1432
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
0001          SUBROUTINE VOEGLN
        C*****************************************************************
        C*****************************************************************
        C*****************************************************************
        C****                                                        *****
        C****                                                        *****
        C****            MODIFIED VOGEL APPROXIMATION METHOD         *****
        C****                                                        *****
        C****                                                        *****
        C*****************************************************************
        C*****************************************************************
        C*****************************************************************
        C
        C
        C
0002          COMMON /SCOM/ K,KSS
0003          COMMON /PRCOM/ IPX(20)
0004          COMMON /COMG/ IGMAX,IGX(40),IGG(40)
0005          COMMON /COM2/ IP(30),IG(40),MSIJ(1200),MIIJ(90),MAXPRI
0006          COMMON /SCHED/ SPSTRT(30),SPSTOP(30),NSTART(40),STARTS(1200),
             *STOPS(1200),SHELLS(1200),NTARG(1200),NSRANK(1200),NFIRST(40),
             *INFEAS(30)
0007          COMMON /UCOM/ CVEC(1200),RVEC(1200),EVEC(1200),SVEC(1200),
             *A(40),SU(40),TI(40),TU(40),TIME,NT,NU,NN,B,ISAME,PVEC(1200)
0008          COMMON /COMX/ LGRNG(40)
0009          COMMON /ACOM/ ALPHA,TOPINE
0010          COMMON /DCOM/ MAXROW,MAXCOL
0011          COMMON /AICD/ ICODE
0012          COMMON /AMCM/ MIJC,MI2C,MPUVEC(40),IIAM
0013          COMMON /ASCM/ IGAM,MSAM,AFRACT,AMMFIX,BEGINS,ENDS,IIAMX,TCOST,SUAM
0014          DIMENSION PENLTY(200)
0015          DIMENSION COVERM(40),IUFGMS(40),IUFGME(40),IUEFLG(40)
0016          DIMENSION IUMFLG(40),IUSFLG(40),IUFFLG(40),TIMMAS(4)
0017          DIMENSION CHEAP(40),IXCHEP(40),IAUNIT(40),INDPEN(200),COVRAG(40)
0018          COMMON /ASTS/ AS(40),TS(40)
0019          DIMENSION IUNITF(40)
0020          DIMENSION IUNF2(40)
0021          DATA MCTMCT/1./
0022          KK=1-MAXROW
0023          KKMR=KK
0024          JX=0
0025          MAXCO2=MAXCOL+MAXCOL
0026          JPS=0
        C*****   REINITIALIZE AMMO AND TIME SUPPLIES,ALSO SOME SCHED & PENALTY INDEXES
0027          DO 3 I=1,NN
0028          TS(I)=TIME
0029          AS(I)=A(I)
0030          NSTART(I)=0
0031          IXCHEP(I)=0
0032          IAUNIT(I)=0
0033          CHEAP (I)=0.
0034        3 CONTINUE
        C*****   INITIALIZE POINTER TO FIRST ENGAGEMENT IN EACH UNIT'S SCHEDULE
0035          DO 10 I=1,IGMAX
0036       10 NFIRST(I)=1
        C*****   REINITIALIZE SCHEDULING ARRAYS
0037          J=MAXROW*NT
0038          IADD=MAXROW-NN-1
```

```
      I=0
   20 I=I+1
      LI=MOD(I,MAXROW)
      IF((LI.LE.NN).AND.(LI.NE.0))GO TO 12
      IF(LI.GT.NN)I=I+IADD
      GO TO 23
   12 STARTS(I)=0.
      STOPS (I)=0.
      SHELLS(I)=0.
      NTARG(I)=0
      NSRANK(I)=0
   23 IF(I.LT.J)GO TO 20
      DO 2000 IPRTY=1,MAXPRI
      IPN=IPX(IPRTY)
      IF(IPN.EQ.0)GO TO 2000
      JP=JPS+1
      JPS=JPS+IPN
C
C*****   OBTAIN PENALTIES FOR THIS PRECEDENCE CLASS (WHOSE COLS GO JP TO JPS)
C
      DO 200 J=JP,JPS
      JX=JX+1
      MIJ1=MIIJ(JX)
      MIJ2=MIIJ(JX+MAXCOL)
      MIJ3=MIIJ(JX+MAXCO2)
      KK=KK+MAXROW
      KMIJ21=KK+MIJ2-1
      IF(MIJ2.LT.2)GO TO 31
      K=KK
      KSS=KMIJ21
      CALL SORTER
   31 IF(MIJ3.LT.2)GO TO 35
      K=KSS+1
      KSS=KSS+MIJ3
      CALL SORTER
   35 PENMAX=-1000000.
      NCMXPN=0
C
C*****   SAVE P-VALUES, ROW NO. & UNIT NO. OF AMMO/TIME-FEASIBLE ROWS.
C*****   COUNT DISTINCT UNITS.
C
      ICHEAP=0
      NOUNIT=0
      DO 50 I=KK,KMIJ21
      II=MSIJ(I)
      IPV=KK+II-1
      PVIPV=PVEC(IPV)
      IF(CVEC(IPV).GT.450000.) GO TO 48
      IF(MIJ1.NE.1)GO TO 40
      IF((RVEC(IPV).GT.AS(II)).OR.(EVEC(IPV).GT.TS(II)))GO TO 48
      GO TO 42
   40 IF((SU(II).GT.TS(II)).OR.(TU(II).GT.AS(II)))GO TO 48
   42 IGII=IG(II)
      IF(ICHEAP.GT.0)GO TO 45
      NOUNIT=1
      GO TO 44
   45 INOFLG=1
      DO 46 LL=1,ICHEAP
```

```
0090            IF(IAUNIT(LL).NE.IGII)GO TO 46
0091            INOFLG=0
0092            IF(IGII.GE.0)IGII=(-IGII)
0093         46 CONTINUE
0094            IF(INOFLG.EQ.1)NOUNIT=NOUNIT+1
0095         44 ICHEAP=ICHEAP+1
0096            CHEAP(ICHEAP)=PVIPV
0097            IXCHEP(ICHEAP)=II
0098            IAUNIT(ICHEAP)=IGII
0099            GO TO 50
          C
          C*****   FLAG AMMO OR TIME INFEASIBILITY BY MAKING CORRESPONDING VALUES
          C*****   IN P AND MS NEGATIVE.
          C
0100         48 PVEC(IPV)=(-PVIPV)
0101            MSIJ(I)=(-II)
0102         50 CONTINUE
          C
          C*********************************************************************
          C                                                                    *
          C                    ***  PENALTY CALCULATIONS  ***                  *
          C                                                                    *
          C        METHOD OF CALCULATING PENALTY DEPENDS ON NO. OF UNITS AVAILABLE *
          C        VS. NO. REQUIRED.                                           *
          C                                                                    *
          C*********************************************************************
          C
0103            IF(NOUNIT.GT.0)GO TO 60
          C
          C*****   A.  IF NO UNITS ARE AVAILABLE, PENALTY IS -1 (LOWER THAN ANY OTHER
          C*****       PENALTY) SINCE THERE IS NOTHING TO BE GAINED BY MAKING AN
          C*****       EARLY ASSIGNMENT.
          C
0104            PENLTJ=-1.
0105            GO TO 100
0106         60 IF(NOUNIT.GT.MIJ1) GO TO 65
          C
          C*****   B.  IF TOO FEW OR EXACTLY ENOUGH UNITS ARE AVAILABLE, PENALTY IS
          C*****       500000+100000*(NO. UNITS NEEDED - NO UNITS AVAILABLE)
          C
0107            PENLTJ=500000.+100000.*FLOAT(MIJ1-NOUNIT)
0108            GO TO 100
0109         65 MIJP=MIJ1+1
0110            IF(NOUNIT.GT.MIJP)GO TO 80
          C
          C*****   C.  IF ONLY ONE UNIT MORE THAN REQUIRED IS AVAILABLE, PENALTY IS
          C*****       100000 + (LARGEST DIFF BETWEEN 2 SUCCEEDING P-VALUES)
          C
0111            BIGDIF=CHEAP(2)-CHEAP(1)
0112            IF(ICHEAP.EQ.2)GO TO 75
0113            DO 70 I=3,ICHEAP
0114            PDIFF=CHEAP(I)-CHEAP(I-1)
0115            IF(PDIFF.GT.BIGDIF)BIGDIF=PDIFF
0116         70 CONTINUE
0117         75 PENLTJ=100000.+BIGDIF
0118            GO TO 100
          C
          C*****   D.  IF THE NO. UNITS AVAILABLE EXCEEDS THE NO. UNITS REQUIRED BY
```

```
C*****       MORE THAN 1, PENALTY IS LARGEST DIFF BETWEEN UNIT "CHAMPS."
C*****   ("CHAMP" IS DEFINED AS THE ROW OF A UNIT HAVING THE SMALLEST
C*****   NONNEGATIVE P-VALUE, AND IS FLAGGED BY A POSITIVE ENTRY IN IAUNIT.)
   80 IPSWCH=0
      DO 90 I=1,ICHEAP
      IF(IAUNIT(I).LT.0)GO TO 90
      CHEAPI=CHEAP(I)
      IF(IPSWCH.GT.0)GO TO 83
      IPSWCH=1
      CHLAST=CHEAPI
      GO TO 90
   83 IF(IPSWCH.GT.1)GO TO 85
      IPSWCH=2
      PENLTJ=CHEAPI-CHLAST
   84 CHLAST=CHEAPI
      GO TO 90
   85 PDIFF=CHEAPI-CHLAST
      IF(PDIFF.GT.PENLTJ)PENLTJ=PDIFF
      GO TO 84
   90 CONTINUE
C
C*****   INSERT PENALTY AND ITS COLUMN INTO SORTED ARRAYS TO DETERMINE ORDER
C*****   IN WHICH THIS PRECEDENCE GROUP IS TO BE OPTIMIZED.
C
  100 IF(J.GT.JP) GO TO 110
      PENLTY(J)=PENLTJ
      INDPEN(J)=J
      GO TO 200
  110 JJ=J
      DO 120 I=JP,J
      IF(I.LT.J)GO TO 115
      PENLTY(J)=PENLTJ
      INDPEN(J)=JJ
      GO TO 120
  115 PENLTI=PENLTY(I)
      IF(PENLTJ.LE.PENLTI)GO TO 120
      PSWAP=PENLTI
      PENLTY(I)=PENLTJ
      PENLTJ=PSWAP
      ISWAP=JJ
      JJ=INDPEN(I)
      INDPEN(I)=ISWAP
  120 CONTINUE
  200 CONTINUE
C
C****************************************************************************
C                                                                           *
C     ***  OPTIMIZE COLUMNS IN ORDER OF HIGHEST-TO-LOWEST PENALTIES  ***    *
C                                                                           *
C****************************************************************************
      DO 500 J=JP,JPS
C*****   GET INDEX OF COL W/J'TH LOWEST PEN AND CLEAR ITS INFEAS FLAG
      II=INDPEN(J)
      INFEAS(II)=0
C*****   CALCULATE VECTOR INDEX OF 1ST ELEMENT IN COL
      III=KKMR+MAXROW*II
      IIM=III-1
      IIIM=IIM
```

```
          C*****   USE MASSING SPECS FOR THIS COL TO GET INDEXES FOR CHECKING FEASIBILITY
0162            JX=II
0163            MIJ1=MIIJ(JX)
0164            MIJ2=MIIJ(JX+MAXCOL)
0165            MIJ3=MIIJ(JX+MAXCO2)
0166            MIJ23=MIJ2+MIJ3
0167            MI23=MIJ23+IIIM
0168            MII2=IIIM+MIJ2
          C****    SET UP OUTPUT ROUTINE FOR NEW TGT
0169            IIAM=II
0170            MIJC=MIJ1
0171            MI2C=MIJ2
0172            I2=1
0173            DO 150 I=III,MII2
0174            MPUVEC(I2)=IABS(MSIJ(I))
0175            I2=I2+1
0176        150 CONTINUE
0177            ICODE=1
0178            CALL OUTPUT
          C*****   CLEAR MAX COVERAGE FOUND SO FAR FOR MASS TGTS
0179            COVMAX=0.
          C*****   TURN ON EXECUTION OF CODE PASSAGES STARTING "DO 202..." AND "202 CON.."
0180            ITMSXN=-10
          C
          C*************************************************************************
          C                                                                        *
          C    CONTROL IS RETURNED TO STATEMENT 2011 WITH MIJ1=1 IF ATTEMPTING TO  *
          C    TRY ANOTHER PERIOD LENGTH FOR MASSED TARGET WITH ONLY START OR STOP SPEC *
          C                                                                        *
          C*************************************************************************
          C
          C*****   GET START & STOP TIMES (IF ANY) FOR THIS TGT, CLEAR COUNTERS AND
          C*****   FLAGS IF MASSING WANTED (OTHERWISE GO DO SINGLE UNIT ASSIGNMENT).
0181       2011 SPSTT=SPSTRT(II)
0182            SPSTP=SPSTOP(II)
0183            IF(MIJ1.EQ.1)GO TO 203
0184            IUCHKD=0
0185            MASSER=0
0186            NAVPM =0
0187            MASFLG=0
0188            SUMCOV=0.
0189            AVAMIN=9999.
0190            AVAMAX=0.
          C*****   POINTERS CLEARED IN FOL DO-LOOP ARE USED TO KEEP UP WITH WHERE A GAP
          C*****   STARTS AND ENDS IN A UNIT'S SCHEDULE (IUSFLG,IUEFLG,IUFGMS,IUFGME).
          C*****   INDEXES TO CORRESPONDING ROWS (FUZES) ARE KEPT IN IUFFLG AND IUMFLG.
          C*****   IUSFLG,IUEFLG, AND IUFFLG ARE PERMANENT WITHIN DO-LOOP STARTING AT
          C*****   STATEMENT 203; OTHERS ARE TEMPORARY.
          C*****   ALSO, IUNITF IS USED TO KEEP 2 ROWS FROM SAME UNIT FROM BEING MASSED
          C*****   TOGETHER ON THE SAME TGT.
0191            DO 202 I=1,IGMAX
0192            IF(ITMSXN.GT.(-5)) GO TO 2022
0193            IUSFLG(I)=0
0194            IUEFLG(I)=0
0195            IUFFLG(I)=0
0196            COVRAG(I)=0.
0197       2022 IUFGMS(I)=0
0198            IUFGME(I)=0
```

```
            IUMFLG(I)=0
            COVERM(I)=0.
            IUNITF(I)=0
        202 CONTINUE
C*****   FOR MASSING WITHOUT SPECIFIED START AND/OR STOP:
C*****   FIND TIME EACH PRIMARY UNIT WOULD REQUIRE FOR ITS "SHARE" OF A
C*****   "PERFECT MASS:" SAVE MAX AND MIN OF THESE ALONG WITH CORRESPONDING
C*****   UNIT NUMBER.
            IF(ITMSXN.GT.(-5))GO TO 203
            IF((SPSTRT(II).GT.9990.).OR.(SPSTOP(II).GT.9990.))GO TO 2023
            TMASMX=SPSTOP(II)-SPSTRT(II)
            GO TO 203
       2023 TMASMX=0.
            TMASMN=TIME
            MAXUNT=-1
            MINUNT=-1
            DO 201 I=III,MII2
            MSIJI=MSIJ(I)
            IF(MSIJI.LT.0)GO TO 201
            MAI=MSIJI+IIM
            EIJ=EVEC(MAI)
            VOLLYS=RVEC(MAI)/(FLOAT(MIJ1)*TU(MSIJI))
            VOLFIX=IFIX(VOLLYS)
            IF((VOLLYS-VOLFIX).LE.0.01)VOLFIX=VOLFIX-1.
            TMAS=T1(MSIJI)*VOLFIX
            IF(TMAS.LE.TMASMX)GO TO 204
            TMASMX=TMAS
            MAXUNT=MSIJI
        204 IF(TMAS.GE.TMASMN)GO TO 201
            TMASMN=TMAS
            MINUNT=MSIJI
        201 CONTINUE
            TMDIFF=TMASMX-TMASMN
            TMAS14=TMDIFF/4
            ITMSXN=4
            ITMINC=1
            IF(TMDIFF.LT.2.)ITMINC=2
            IF(TMDIFF.LT.1)ITMINC=4
C**************************************************************************
C                                                                         *
C    SEARCH COLUMN FROM BEST-TO-WORST FOR ROWS TO ASSIGN THIS TGT TO.     *
C    VALUES ARE FIRST OBTAINED THAT ARE NEEDED FOR ALL TGTS.  THEN WE BRANCH *
C    TO ONE OF THE OPTIMIZING ROUTINES ACCORDING TO TYPE OF TARGET:       *
C                                                                         *
C                     I.  START AND/OR STOP SPECIFIED: NON-MASS           *
C                    II.  NO START OR STOP: NON-MASS                      *
C                   III.  START AND/OR STOP: MASSING SPECIFIED            *
C                    IV.  NO START AND/OR STOP: MASSING SPECIFIED         *
C                                                                         *
C**************************************************************************
C
        203 DO 450 I=III,MI23
            SPSTT=SPSTRT(II)
            SPSTP=SPSTOP(II)
            IUCHKD=I-III+1
C*****   GET INDEXES TO MAIN ARRAYS (MSIJI FOR UNIT PARAMETERS, MAI FOR AMMO
C*****   AND TIME).
            MSIJI=MSIJ(I)
```

```
0238          MAI=MSIJI+IIM
        C*****   FEASIBILITY CHECK
0239          IF(MSIJI.LT.0)GO TO 450
        C***  UNIT PARAMETERS, UNIT NO., AMMO & TIME, FIRE TIME, TOTAL UNIT SCHED LENGTH
0240          TRI=T1(MSIJI)
0241          TUI=TU(MSIJI)
0242          SUI=SU(MSIJI)
0243          IGXX=IG(MSIJI)
0244          EIJ=EVEC(MAI)
0245          RVMA=RVEC(MAI)
0246          TCOST=CVEC(MAI)
0247          FIRTIM=EIJ-SUI
0248          TULONG=TIME+SUI
        C*****   CHECK IF II. OR IV.
0249          IF((SPSTT.GT.9990.).AND.(SPSTP.GT.9990.))GO TO 300
        C*****   CHECK IF III.
0250          IF(MIJ1.GT.1)GO TO 270
        C
        C************************************************************************
        C                                                                       *
        C*****             I.  START/STOP AND NON-MASS                          *
        C                                                                       *
        C        APPROACH IS TO CHECK THIS UNIT'S SCHED.  IF OK, ASSIGN.        *
        C                                                                       *
        C************************************************************************
        C
        C*****   IF ONE END OF PERIOD UNSPEC'D, GO CALCULATE IT FROM OTHER END
0251          IF((SPSTT.GT.9990.).OR.(SPSTP.GT.9990.))GO TO 205
        C*****   CHECK IF UNIT IS FAST ENOUGH; IF SO, ALLOW FOR SET-UP TIME
0252          IF((SPSTP-SPSTT).LT.FIRTIM)GO TO 450
0253      210 SPSTT=SPSTT-SUI
0254          GO TO 220
0255      205 IF(SPSTT.GT.9990.)GO TO 215
0256          SPSTP=SPSTT+FIRTIM
0257          GO TO 210
0258      215 SPSTT=SPSTP-EIJ
        C*****   FIND OUT HOW MANY TGTS HAVE BEEN ASGD TO THIS UNIT.  IF > 0 GO CHECK
        C*****   FOR SCHED INTERFERENCE.  OTHERWISE MAKE THE ASSIGNMENT.
0259      220 NSTRTI=NSTART(IGXX)
0260          NSTPO=NSTRTI+1
0261          IF(NSTRTI.GT.0)GO TO 230
        C*****   MAKE INITIAL ASGMT FOR UNIT IGXX
0262      222 NSTART(IGXX)=1
0263          NFIRST(IGXX)=1
0264          STARTS(IGXX)=SPSTT
0265          STOPS(IGXX)=SPSTP
0266          SHELLS(IGXX)=RVMA
0267          NTARG(IGXX)=1000*II+MSIJI
0268          NSRANK(IGXX)=0
        C*****   PASS ASGMT TO OUTPUT SUBROUTINE
0269     2223 IGAM=IGXX
0270          MSAM=MSIJI
0271          AFRACT=1.
0272          AMMFIX=RVMA
0273          BEGINS=SPSTT
0274          ENDS=SPSTP
0275          IIAMX=II
0276          SUAM=SUI
```

```
            ICODE=2
            CALL OUTPUT
        223 AS(MSIJI)=AS(MSIJI)-RVMA
      C*****   SUBTRACT TIME FOR ALL ROWS IN UNIT
            IGGIX=IGG(IGXX)
            IGGIXS=IGGIX+IGX(IGXX)-1
            SCTIME=SPSTP-SPSTT
            DO 225 IGGG=IGGIX,IGGIXS
        225 TS(IGGG)=TS(IGGG)-SCTIME
            GO TO 500
      C*****   CHECK IF UNIT'S SCHED FITS WITH START/STOP
        230 IFRST=NFIRST(IGXX)
        232 IVSA=(IFRST-1)*MAXROW+IGXX
            STARTC=STARTS(IVSA)
            NRIVSA=NSRANK(IVSA)
            IF(STARTC.GE.SPSTP)GO TO 240
            IFROLD=IFRST
            IFRST=MOD(NRIVSA,1000)
            IF(IFRST.EQ.0)GO TO 250
            GO TO 232
        240 IPREV=NRIVSA/1000
            I260=0
            IF(IPREV.EQ.0)GO TO 260
            IVSB=(IPREV-1)*MAXROW+IGXX
            STOPC=STOPS(IVSB)
            IF(STOPC.GT.SPSTT)GO TO 450
      C*****   ASGN TO UNIT IGXX IN TIME SLOT SPSTT TO SPSTP
        243 NSTXX=NSTRTI*MAXROW+IGXX
            IPTHOU=IPREV*1000
            NSRANK(NSTXX)=IPTHOU+IFRST
            NSRANK(IVSA)=NRIVSA-IPTHOU+NSTPO*1000
            IF(I260.EQ.0)NSRANK(IVSB)=NSRANK(IVSB)-IFRST+NSTPO
        245 STARTS(NSTXX)=SPSTT
            STOPS(NSTXX)=SPSTP
            SHELLS(NSTXX)=RVMA
            NTARG(NSTXX)=1000*II+MSIJI
            NSTART(IGXX)=NSTPO
      C*****   COMPLETE ASGMT BY GOING TO WHERE TIME AND AMMO SUPPLIES ARE ADJUSTED
            GO TO 2223
        250 STOPC=STOPS(IVSA)
            IF(STOPC.GT.SPSTT)GO TO 450
            NSTART(IGXX)=NSTPO
            NSTXX=NSTRTI*MAXROW+IGXX
            NSRANK(IVSA)=NRIVSA+NSTPO
            NSRANK(NSTXX)=IFROLD*1000
            GO TO 245
        260 NFIRST(IGXX)=NSTPO
            I260=1
            GO TO 243
      C***********************************************************************
      C                                                                      *
      C                    III.  START/STOP AND MASSING SPECIFIED            *
      C                                                                      *
      C
      C    APPROACH DEPENDS UPON WHETHER OR NOT BOTH START AND STOP ARE SPECIFIED:  *
      C    A.  IF BOTH ARE SPECIFIED, WE SIMPLY DETERMINE IF TGT CAN BE COVERED IN  *
      C        SPEC'D PERIOD AND BRANCH OFF TO ASSIGN IT AS SOON AS ENOUGH UNITS ARE *
      C        MASSED OR NO MORE AVAILABLE, PROVIDED COVERAGE IS SUFFICIENT.         *
```

```
      C    B.  IF ONE END OF PERIOD IS UNSPEC'D, A RECORD IS KEPT OF ALL UNITS      *
      C        HAVING TIME TO GET OFF AT LEAST ONE VOLLEY.  THIS IS DONE BY SETTING *
      C        UP A DUMMY PERIOD EQUAL TO SET-UP TIME AND PROCEEDING AS IN A. ABOVE, *
      C        FLAGGING SUCH UNITS WITH A 2 OR 3 IN INFSII FOR FURTHER PROCESSING.  *
      C                                                                             *
      C******************************************************************************
      C
      C*****   UNLESS A ROW FROM THIS UNIT WAS ALREADY TRIED, SET POINTER TO THIS GAP.
        270 IF(IUNITF(IGXX).NE.0)GO TO 450
            IFRST=NFIRST(IGXX)
            IUEFLG(IGXX)=IFRST
            IF(SPSTT.GT.9990.)GO TO 285
            IF(SPSTP.GT.9990.)GO TO 286
      C*****   BOTH START AND STOP SPEC'D -- ALLOW FOR SETUP TIME AND SET FLAG.
            SPSTX=SPSTT-SUI
            INFSII=0
            GO TO 287
        285 INFSII=2
            STOPC=-SUI
        284 SPSTX=SPSTP-SUI
            GO TO 287
        286 INFSII=3
            STARTC=TIME
            SPSTP=SPSTT
            GO TO 284
        287 NSTRTI=NSTART(IGXX)
            NSTPC=NSTRTI+1
            IF(NSTRTI.NE.0)GO TO 272
            IFRST=0
            GO TO 273
        272 IVSA=(IFRST-1)*MAXROW+IGXX
            STARTC=STARTS(IVSA)
            NRIVSA=NSRANK(IVSA)
            IF(STARTC.GT.SPSTP)GO TO 274
            IFROLD=IFRST
            IFRST=MOD(NRIVSA,1000)
            IF(IFRST.EQ.0)GO TO 271
            GO TO 272
        274 IPREV=NRIVSA/1000
            IF(IPREV.EQ.0)GO TO 273
            IVSB=(IPREV-1)*MAXROW+IGXX
            STOPC=STOPS(IVSB)
            IF(STOPC.GT.SPSTX)GO TO 450
            IUSFLG(IGXX)=IPREV
            GO TO 273
        271 STOPC=STOPS(IVSA)
            IF(STOPC.GT.SPSTX)GO TO 450
            IUSFLG(IGXX)=IFROLD
        273 MASSER=MASSER+1
            IUNITF(IGXX)=IGXX
            IUFFLG(IGXX)=MSIJI
            IUEFLG(IGXX)=IFRST
            IF(INFSII.EQ.0)GO TO 275
      C*****   FOR ONE END UNSPEC'D, SAVE MAX & MIN AVAIL TIME (DISREGARD SET-UP) AND
      C*****   INDEXES TO CORRESP UNITS, COUNT UNITS WHOSE AVAIL TIME EXCEEDS THASMX.
            IF(INFSII.EQ.3)AVAILT=STARTC-SPSTP
            IF(INFSII.EQ.2)AVAILT=SPSTT-STOPC
            IF(AVAILT.GE.AVAMIN)GO TO 276
```

```
              AVAMIN=AVAILT
              IAVMIN=IGXX
          276 IF(AVAILT.LE.AVAMAX)GO TO 277
              AVAMAX=AVAILT
              IAVMAX=IGXX
          277 IF(AVAILT.LT.TMASMX)GO TO 450
              NAVPM=NAVPM+1
              GO TO 450
C*****   FOR BOTH ENDS SPEC'D, ASSIGN IF FULL COVER POSS. IF NOT FLAG INFEAS.
          275 ASMS=AS(MSIJT)
              REQTIM=SPSTP-SPSTT
              VOLLYS=IFIX(REQTIM/TRI+1.)
              SHLPOS=VOLLYS*TUI
              IF(ASMS.LT.SHLPOS)SHLPOS=ASMS
              COVADD=SHLPOS/RVMA
              COVRAG(IGXX)=COVADD
              SUMCOV=SUMCOV+COVADD
              IF((MASSER.GE.MIJ1).AND.(SUMCOV.GE.1.))GO TO 278
              IF(IUCHKD.LT.MIJ23)GO TO 450
              IF(SUMCOV.GE.1.)GO TO 278
              INFEAS(II)=1
              GO TO 500
          278 TMASMX=REQTIM
              GO TO 451
C
C***********************************************************************
C                                                                      *
C                  II. & IV.:  NO START OR STOP SPECIFIED              *
C                                                                      *
C   UNLESS UNIT WAS ALREADY TRIED, GET PARAMS COMMON TO BOTH MASS AND NON-  *
C   MASS. THEN SPLIT LOGIC.                                            *
C                                                                      *
C***********************************************************************
C
          300 IFRST=IGXX
              IUEFLG(IGXX)=IFRST
              NSTRTI=NSTART(IGXX)
              NSTPO=NSTRTI+1
              IF(MIJ1.GT.1)GO TO 370
C
C***********************************************************************
C                                                                      *
C                  II.  NO START/STOP AND NON-MASS                     *
C                                                                      *
C       APPROACH IS TO FIND SMALLEST GAP IN UNIT'S SCHED THAT CAN COVER  *
C       THIS TARGET.                                                   *
C                                                                      *
C***********************************************************************
C
              IF(NSTRTI.GT.0)GO TO 305
C*****   NONE ASGD YET; START AS EARLY AS POSS
              SPSTTX=0.
              SPSTT=-SUI
              SPSTP=FIRTIM
              GO TO 222
C*****   SECOND & SUBSEQUENT ASGMTS: SEEK SMALLEST GAP THAT FITS
          305 IFRST=NFIRST(IGXX)
              SLOTMN=TULONG
```

```
      OLDSTP=-SUI
  310 IVSA=(IFRST-1)*MAXROW+IGXX
      STARTC=STARTS(IVSA)
      NRIVSA=NSRANK(IVSA)
      SLOT=STARTC-OLDSTP
      IF((SLOT.LT.SLOTMN).AND.(SLOT.GE.EIJ))GO TO 320
  315 OLDSTP=STOPS(IVSA)
      IFROLD=IFRST
      IFRST=MOD(NRIVSA,1000)
      IF(IFRST.EQ.0)GO TO 330
      GO TO 310
C*****   NEW MIN FITTING GAP:  SAVE NEW MIN AND SAVE POINTERS TO ENGAGEMENTS
C*****   ALREADY SCHEDULED ON ENDS OF GAP
  320 SLOTMN=SLOT
      IAFTER=IFRST
      IBEFOR=NRIVSA/1000
      SPSTTX=OLDSTP
      GO TO 315
C*****   END OF UNIT'S SCHED:  CHECK IF ADEQUATE GAP WAS FOUND; CHECK IF THIS
C*****   END GAP WAS IT
  330 SLOT=TIME-OLDSTP
      IF((SLOT.LT.EIJ).AND.(TULONG.EQ.SLOTMN))GO TO 450
      IF((SLOT.GE.SLOTMN).OR.(SLOT.LT.EIJ))GO TO 335
      IBEFOR=IFROLD
      IAFTER=IFRST
      SLOTMN=SLOT
      IAFTER=0
      IBEFOR=IFROLD
      SPSTTX=OLDSTP
  335 SPSTT=SPSTTX
      SPSTP=SPSTT+EIJ
      NSTART(IGXX)=NSTPO
      NSTXX=NSTRTI*MAXROW+IGXX
      NSRANK(NSTXX)=1000*IBEFOR+IAFTER
      IF (IBEFOR.NE.0)GO TO 340
      NFIRST(IGXX)=NSTPO
      IVSA=(IAFTER-1)*MAXROW+IGXX
      NSRANK(IVSA)=NSRANK(IVSA)+1000*NSTPO
      GO TO 245
  340 IF(IAFTER.NE.0)GO TO 345
      IVSA=(IFROLD-1)*MAXROW+IGXX
      NSRANK(IVSA)=NSRANK(IVSA)+NSTPO
      GO TO 245
  345 IVSA=(IAFTER-1)*MAXROW+IGXX
      NRIVSA=MOD(NSRANK(IVSA),1000)
      NSRANK(IVSA)=NRIVSA+1000*NSTPO
      IVSA=(IBEFOR-1)*MAXROW+IGXX
      NRIVSA=NSRANK(IVSA)/1000
      NSRANK(IVSA)=NRIVSA*1000+NSTPO
      GO TO 245
C
C****************************************************************************
C                                                                           *
C                   IV.  MASSING SPEC'D BUT NO START OR STOP                *
C                                                                           *
C****************************************************************************
C
C*****   APPROACH IS TO TRY FITTING A "PERFECT MASS" INTO GAPS IN THIS ROW'S
```

```
C*****   SCHED BY CHECKING OTHER ROW'S SCHEDS FOR COMPATIBILITY WITH THE
C*****   START/STOP TIMES THAT WOULD RESULT IF THE ENGAGEMENT BEGAN AS EARLY
C*****   IN THE GAP AS POSSIBLE AND LASTED AS LONG AS THE SLOWEST PRIMARY UNIT
C*****   WOULD REQUIRE TO ENGAGE EXACTLY ITS "SHARE" OF THE TARGET.  THIS DURA-
C*****   TION HAS ALREADY BEEN CALCULATED IN THE DO-LOOP ENDING IN STATEMENT 201
C*****   AND IS CALLED TMASMX.  ASGMT IS MADE IN THE FIRST GAP WHERE A "PERFECT
C*****   MASS" WILL FIT.  IF NO SUCH GAP IS FOUND, ASGMT IS MADE IN THE GAP
C*****   WHERE THE MOST PRIMARY UNITS ARE AVAILABLE, PROVIDED ENOUGH SECONDARY
C*****   UNITS ARE AVAILABLE, OR PRIMARY UNITS CAN BE "STRETCHED" OR "SPEEDED
C*****   UP" TO COVER THE TARGET ADEQUATELY.  IF THAT DOESN'T WORK, A NEW VALUE
C*****   IS PUT IN TMASMX WHICH IS CLOSER TO TMASMN, WITH SUCCESSIVELY SHORTER
C*****   LENGTHS BEING TRIED UNTIL ONE WORKS OR TMASMN IS REACHED.
C*****   (SEE COMMENTS BETWEEN STATEMENTS 475 AND 476.)
C
  370 IF(IUNITF(IGXX).NE.0)GO TO 450
      INFSII=5
      PERIOD=SUI+TMASMX
      IF(NSTRTI.GT.0)GO TO 375
C*****   NONE ASSIGNED: START TRIAL PERIOD AS EARLY AS POSSIBLE
      SPSTTX=0.
      SPSTX=-SUI
      SPSTPX=TMASMX
      GO TO 390
  375 IFRST=NFIRST(IGXX)
      OLDSTP=-SUI
  380 IVSA=(IFRST-1)*MAXROW+IGXX
      STARTC=STARTS(IVSA)
      NRIVSA=NSRANK(IVSA)
      SLOT=STARTC-OLDSTP
      IF(SLOT.LT.PERIOD)GO TO 385
  383 SPSTX=OLDSTP
      SPSTPX=SPSTX+PERIOD
      SPSTTX=OLDSTP+SUI
      GO TO 390
  385 OLDSTP=STOPS(IVSA)
      IFROLD=IFRST
  386 IFRST=MOD(NRIVSA,1000)
      IF(IFRST.NE.0)GO TO 380
C*****   END OF UNIT'S SCHED
      SLOT=TIME-OLDSTP
      IF(SLOT.LT.PERIOD)GO TO 450
      GO TO 383
C*****   CHECK IF GAP WILL FIT INTO ENOUGH OTHER UNITS' SCHEDS.  LOGIC IS
C*****   SIMILAR TO SECTIONS STARTING AT STATEMENTS 203 AND 270.
  390 DO 3391 LL=1,IGMAX
      IUFGMS(LL)=0
      IUFGME(LL)=0
      IUMFLG(LL)=0
      COVERM(LL)=0
      IUNF2(LL)=0
 3391 CONTINUE
      SUMCOV=0.
      MASSER=0
      NPRIM =0
      NPRMAX=-1
      DO 410 LL=III,MI23
      MSIJXX=MSIJ(LL)
      IF(MSIJXX.LT.0)GO TO 410
```

```
      IGXXX=IG(MSIJXX)
      IF((IGXX.EQ.IGXXX).AND.(MSIJI.NE.MSIJXX))GO TO 410
      MAIXX=MSIJXX+IIIM
      IF(IUNF2(IGXXX).NE.0)GO TO 410
      IFIRS=NFIRST(IGXXX)
      IUFGME(IGXXX)=IFIRS
      TRIX=T1(MSIJXX)
      SUIX=SU(MSIJXX)
      RVMX=RVEC(MAIXX)
      FIJX=EVEC(MAIXX)
      FIRXX=EIJX-SUIX
      TUIX=TU(MSIJXX)
      TULXX=TIME+SUIX
      SPSTXX=SPSTTX-SUIX
      NSTRIX=NSTART(IGXXX)
      NSTPOX=NSTRIX+1
      IF(NSTPIX.NE.0)GO TO 392
      IFIRS=0
      GO TO 393
C*****   FOLLOWING LOGIC IS SIMILAR TO THAT STARTING AT STATEMENT 272
  392 IVSX=(IFIRS-1)*MAXROW+IGXXX
      STARTC=STARTS(IVSX)
      NRIVSX=NSRANK(IVSX)
      IF(STARTC.GT.SPSTPX) GO TO 394
      IFRSOL=IFIRS
      IFIRS=MOD(NRIVSX,1000)
      IF(IFIRS.EQ.0)GO TO 391
      GO TO 392
  394 IPREVX=NRIVSX/1000
      IF(IPREVX.EQ.0)GO TO 393
      IVSBX=(IPREVX-1)*MAXROW+IGXXX
      STOPC=STOPS(IVSBX)
      IF(STOPC.GT.SPSTXX)GO TO 410
      IUFGMS(IGXXX)=IPREVX
      GO TO 393
  391 STOPC=STOPS(IVSX)
      IF(STOPC.GT.SPSTXX)GO TO 410
      IUMFLG(IGXXX)=MSIJXX
      IUFGMS(IGXXX)=IFRSOL
  393 MASSER=MASSER+1
      IUMFLG(IGXXX)=MSIJXX
      IUFGME(IGXXX)=IFIRS
      IUNF2(IGXXX)=IGXXX
      IF(LL.GT.MII2)GO TO 395
      NPRIM=MASSER
  395 VOLLYS=IFIX(TMASMX/TRIX + 1.)
      SHLPOS=VOLLYS*TUIX
      ASMX=AS(MSIJXX)
      IF(ASMX.LT.SHLPOS)SHLPOS=ASMX
      COVADD=SHLPOS/RVMX
      IF(COVADD.GT.1)COVADD=1.
      SUMCOV=SUMCOV+COVADD
      COVERM(IGXXX)=COVADD
      IF((SUMCOV.LT.1).OR.(NPRIM.LT.MIJ1).OR.(LL.GT.MII2))GO TO 399
C*****   SUITABLE SLOT FOR "PERFECT MASS."  SET INDICATORS AND GO ASSIGN.
C*****   ALSO SHORTEN LENGTH IF POSSIBLE TO FULLY COVER TGT IN LESS TIME
      COVMAX=SUMCOV
      SPSTP =SPSTPX
```

```
      STOPPR=SPSTPX
      COVRMN=9999.
      RATIOM=1./FLOAT(MASSER)
      DO 396 LLL=1,IGMAX
      IUSFLG(LLL)=IUFGMS(LLL)
      IUEFLG(LLL)=IUFGME(LLL)
      IUFFLG(LLL)=IUMFLG(LLL)
      COVRML=COVERM(LLL)
      COVPAG(LLL)=COVRML
      IF(IUMFLG(LLL).LT.1)GO TO 396
      IF(COVRMN.GT.COVRML)COVRMN=COVRML
  396 CONTINUE
      INFSII=4
C*****   IF LENGTH CANNOT BE REDUCED, GO ASSIGN.  ELSE SHORTEN LENGTH.
      IF(COVRMN.LE.RATIOM)GO TO 451
C*****   FIND MIN LENGTH REQ'D BY A PARTICIPATING UNIT FOR ITS SHARE
C*****   AND ADJUST PERIOD ACCORDINGLY (COVERAGE FRACTION BECOMES RATIOM
C*****   FOR ALL UNITS BECAUSE THIS IS AN EQUALLY-SHARED TGT.)
      TLNGTH=9999.
      DO 397 LLL=1,IGMAX
      IXROW=IUFFLG(LLL)
      IF(IXROW.LT.1)GO TO 397
      MAIX=IIM + IXROW
      VOLLYS=RVEC(MAIX)*RATIOM/TU(IXROW)
      VOLFIX=IFIX(VOLLYS)
      IF((VOLLYS-VOLFIX).LE..01)VOLFIX=VOLFIX-1.
      TRYLEN=T1(IXROW)*VOLFIX
      IF(TRYLEN.LT.TLNGTH)TLNGTH=TRYLEN
      COVPAG(LLL)=RATIOM
  397 CONTINUE
      TMASMX=TLNGTH
      STOPPR=SPSTTX+TLNGTH
      SPSTP=STOPPR
      GO TO 451
C*****   "PERFECT MASS" NOT SATISFIED YET.  CHECK IF FULL COVERAGE YET ACHIEVED.
C*****   CHECK ALSO IF ALL PRIMARY UNITS HAVE BEEN TRIED.
  399 IF((SUMCOV.GE.1.).AND.(LL.GE.MII2)) GO TO 411
  410 CONTINUE
C*****   CHECK IF THIS IS BEST TRIAL GAP YET TESTED.  IF SO, SAVE ITS PARAMETERS
  411 IF(SUMCOV.LT.1)GO TO 420
      IF(NPRIM.LE.NPRMAX)GO TO 425
      NPRMAX=NPRIM
  413 COVMAX=SUMCOV
      IUNITF(IGXX)=IGXX
      STOPPR=SPSTPX
      DO 415 LLL=1,IGMAX
      COVRML=COVERM(LLL)
      COVPAG(LLL)=COVRML
      IUSFLG(LLL)=IUFGMS(LLL)
      IUEFLG(LLL)=IUFGME(LLL)
      IUFFLG(LLL)=IUMFLG(LLL)
  415 CONTINUE
      GO TO 425
  420 IF(SUMCOV.LE.COVMAX)GO TO 425
      GO TO 413
  425 IF(IFRST.NE.0)GO TO 386
  450 CONTINUE
C
```

```
      C**********************************************************************
      C                                                                     *
      C              ***  END OF DO-LOOP THAT CYCLES THRU ROWS  ***         *
      C                                                                     *
      C**********************************************************************
      C
      C*****   IF NON-MASS, NO FEAS ASGMT COULD BE FOUND. FLAG TGT AS UNASGD.
            IF(MIJ1.GT.1)GO TO 451
            INFEAS(II)=1
            GO TO 500
      C
      C**********************************************************************
      C                                                                     *
      C              ***   CONTINUATION OF MASSING LOGIC   ***              *
      C                                                                     *
      C    (AT THIS POINT, ALL UNITS' RESOURCES HAVE BEEN CHECKED, AND THE BEST *
      C     SET OF UNIT RESOURCES THAT IS AVAILABLE SUBJECT TO ALL OTHER RESTRICTIONS *
      C     IS KNOWN.  FOR SOME UNITS (INFSII=0 OR INFSII=4, MEANING UNSPEC SCHED *
      C     WHERE PERFECT MASS WAS FOUND, OR BOTH STOP AND START SPECIFIED WITH *
      C     SATISFACTORY MASS POSSIBLE), ASGMT CAN BE IMMEDIATELY MADE.  FOR  *
      C     INFSII=2 OR 3, MEANING ONLY ONE END OF ENGAGEMENT PERIOD SPEC'D, WE *
      C     KNOW ONLY WHICH UNITS CAN GET OFF AT LEAST ONE VOLLEY.   INFSII=5  *
      C     MEANS NO PERFECT MASS FOUND FOR MASS TGT WITH START/STOP UNSPEC, ALTHOUGH *
      C     THE PARAMETERS OF THE MOST PROMISING GAP HAVE BEEN SAVED STARTING AT *
      C     STATEMENT 411 ABOVE.   FOR INFSII=2,3, OR 5, FURTHER WORK IS NECESSARY.) *
      C                                                                     *
      C**********************************************************************
      C
        451 INFEAS(II)=INFSII
            IF((INFSII.NE.0).AND.(INFSII.NE.4))GO TO 475
      C*****   AT THIS POINT IT IS KNOWN THAT A MASS IS POSSIBLE USING UNITS WHOSE
      C*****   INDEXES POINT TO NONZERO VALUES IN IUFFLG.  THESE NONZERO VALUES
      C*****   ARE THE ROW NUMBERS.  UNIT INDEXES ALSO POINT TO CELLS OF IUSFLG AND
      C*****   IUEFLG THAT CONTAIN POINTERS TO THE SCHEDULED ENGAGEMENTS FOR THAT UNIT
      C*****   THAT PRECEDE AND FOLLOW THE GAP WHERE THE MASS WILL FIT.  IGNORING
      C*****   SET-UP TIME, THE LENGTH OF THE MASS IS TMASMX AND IT ENDS AT TIME
      C*****   SPSTP.  APPROACH TO MASSING HERE IS TO TRY TO ACHIEVE A PERFECT MASS.
      C*****   IF NOT POSSIBLE, SLOWEST UNIT(S) IS (ARE) GIVEN AS MUCH AS POSSIBLE,
      C*****   AND OTHER UNIT'S SHARES ARE REVISED UPWARDS, WITH THE PROCEDURE
      C*****   BEING REPEATED UNTIL COVERAGE IS COMPLETE (WHICH WE KNOW WILL BE POSS
      C*****   BECAUSE OF EARLIER CHECKING INVOLVING COVRAG(.) AND SUMCOV).
        452 SPSTT=SPSTP-TMASMX
      C*****   CALCULATE IDEAL MASS COVERAGE & COMPARE IT TO LOWEST PREV CALC'D COVERG
        453 MASSER=0
            COVLOW=9999.
            DO 4531 I=1,IGMAX
            IUFLGI=IUFFLG(I)
            IF(IUFLGI.LT.1)GO TO 4531
            COVRML=COVRAG(I)
            IF(COVRML.LE.0.005) GO TO 4532
            MASSER=MASSER+1
            IF(COVLOW.LE.COVRML) GO TO 4531
            COVLOW=COVRML
            LLSAV=I
            GO TO 4531
       4532 IUFFLG(I)=0
       4531 CONTINUE
            FNMASS=MASSER
```

```
            FNUMER=1.
        454 RATIOM=FNUMER/FNMASS
      C*****   ALLOW FOR ROUNDOFF ERROR
            COVLOW=COVLOW+.001
        455 IF(COVLOW.GT.RATIOM)GO TO 465
      C*****   UNIT CAN'T COVER ITS SHARE -- ASSIGN AS MUCH AS IT CAN COVER.
            AFRACT=COVLOW
            IGOTO=456
            GO TO 470
        456 FNUMER=FNUMER-AFRACT
      C*****   FIND NEXT SLOWEST UNIT AND WHAT IT CAN COVER
            COVRAG(LLSAV)=9999.
            COVLOW=9999.
            DO 457 I=1,IGMAX
            COVRML=COVRAG(I)
            IF(COVRML.GE.COVLOW)GO TO 457
            IF(IUFFLG(I).EQ.0)GO TO 457
            COVLOW=COVRML
            LLSAV=I
        457 CONTINUE
      C*****   IF MORE UNITS ARE NOT ASGD YET, GO RECALCULATE SHARES FOR THEM.
            FNMASS=FNMASS-1.
            IF(FNMASS.GT..50)GO TO 454
            GO TO 500
      C*****   PERFECT MASS -- ASGN EACH UNIT AN EQUAL SHARE
        465 LLSAV=0
        466 LLSAV=LLSAV+1
            AFRACT=RATIOM
            IF(LLSAV.GT.IGMAX)GO TO 500
            IF((COVRAG(LLSAV).GT.9000.).OR.(IUFFLG(LLSAV).EQ.0))GO TO 466
            IGOTO=466
      C
      C***************************************************************************
      C                                                                          *
      C        ****   ASSIGNMENT OF ONE UNIT IN A MASSED ENGAGEMENT   ****       *
      C                                                                          *
      C***************************************************************************
      C
        470 IU=IUFFLG(LLSAV)
            SPSTTX=SPSTT-SU(IU)
            NSTRTI=NSTART(LLSAV)
            NSTPO=NSTRTI+1
            NSTXX=NSTRTI*MAXROW+LLSAV
            IUSF=IUSFLG(LLSAV)
            IUEF=IUEFLG(LLSAV)
            STARTS(NSTXX)=SPSTTX
            STOPS(NSTXX)=SPSTP
            IUII=(II-1)*MAXROW+IU
            RVMA=RVEC(IUII)
            AMMO=RVMA*AFRACT
            AMMFIX=IFIX(AMMO)
            IF((AMMO-AMMFIX).GT.0.01)AMMFIX=AMMFIX+1
            TUBES=TU(IU)
            IF(AMMFIX.LT.TUBES)AMMFIX=TUBES
            AFRACT=AMMFIX/RVMA
            ASIU=AS(IU)
            IF(AMMFIX.GT.ASIU)AMMFIX=ASIU
            TCOST=CVEC(IUII)*AFRACT
```

```
            SHELLS(NSTXX)=AMMFIX
            NTARG(NSTXX)=1000*II+IU
            AS(IU)=AS(IU)-AMMFIX
            IGGIX=IGG(LLSAV)
            IGGIXS=IGGIX+IGX(LLSAV)-1
            SCTIME=SPSTP-SPSTTX
C*****   PASS ASGT TO OUTPUT SUBROUTINE
            IGAM=LLSAV
            MSAM=IU
            BEGINS=SPSTTX
            ENDS=SPSTP
            SUAM=SU(IU)
            IIAMX=II
            ICODE=3
            CALL OUTPUT
            DO 471 IGGG=IGGIX,IGGIXS
        471 TS(IGGG)=TS(IGGG)-SCTIME
            NSTART(LLSAV)=NSTPO
            IF(NSTRTI.NE.0)GO TO 472
            NSRANK(NSTXX)=0
            GO TO 474
        472 IUSF=IUSFLG(LLSAV)
            IF(IUSF.EQ.0)NFIRST(LLSAV)=NSTPO
            IUEF=IUEFLG(LLSAV)
            IUSFTH=IUSF*1000
            NSRANK(NSTXX)=IUSFTH+IUEF
            IF(IUEF.EQ.0)GO TO 4724
            IUEFX=(IUEF-1)*MAXROW+LLSAV
            NSRANK(IUEFX)=NSRANK(IUEFX)-IUSFTH+NSTPO*1000
       4724 IF(IUSF.EQ.0)GO TO 474
            IUSFX=(IUSF-1)*MAXROW+LLSAV
            NSRANK(IUSFX)=NSRANK(IUSFX)-IUEF+NSTPO
        474 IF(IGOTO.EQ.466)GO TO 466
            IF(IGOTO.EQ.456)GO TO 456
C
C***************************************************************************
C                                                                         *
C                                                                         *
C        ***  ALL MASSES WITH INFSII=2,3, OR 5 COME HERE  ***             *
C                                                                         *
C***************************************************************************
C
        475 IF(INFSII.NE.5)GO TO 477
            IF(COVMAX.LT.1)GO TO 4752
            SPSTP=STOPPR
            GO TO 452
C*****   FULL COVERAGE NOT YET FOUND FOR A NON-START-STOP MASSED TGT.  SHORTEN
C*****   TRIAL PERIOD LENGTH BY THE %-AGE OF (TMASMX - TMASMN) INDICATED:
C*****              A.  25% IF TMASMX - TMASMN > 2
C*****              B.  50% IF 2 > (TMASMX - TMASMN) > 1
C*****              C. 100% IF (TMASMX - TMASMN) < 1
C*****   AND TRY AGAIN, UNLESS TRIAL PERIOD IS NOW HOPELESSLY SHORT (DEFINED
C*****   BY TIME REQUIRED BY FASTEST UNIT TO COVER ITS "SHARE" OF TGT.)  IF NO
C*****   MASS IS POSSIBLE AT ALL, FLAG TGT AS INFEASIBLE.
       4752 ITMSXN=ITMSXN-ITMINC
            IF(ITMSXN.GE.0)GO TO 476
            INFEAS(II)=1
            GO TO 500
```

```
0703          476 THASMX=TMASMN+FLOAT(ITMSXN)*TMAS14
0704              GO TO 2011
            C
            C**************************************************************************
            C                                                                        *
            C        ***  MASS WITH ONLY ONE END OF ENGAGEMENT PERIOD SPECIFIED  ***   *
            C                              (INFSII=2 OR 3)                           *
            C                                                                        *
            C**************************************************************************
            C
            C*****   WHEN WE GET HERE WE KNOW WHICH UNITS HAVE TIME TO GET OFF AT LEAST
            C*****   ONE VOLLEY (IUFFLG IS NONZERO AND POINTS AT ROW) AND WHERE IN THEIR
            C*****   SCHEDS (POINTERS IUSFLG AND IUEFLG) THE VOLLEY OR WHATEVER WILL FIT.
            C*****   WE ALSO KNOW (AVAMIN,AVAMAX) THE MAX AND MIN GAP LENGTHS AMONG THESE
            C*****   UNITS, AND (IAVMIN,IAVMAX) WHICH UNITS THEY ARE ASSOCIATED WITH.
            C*****   FINALLY, WE HAVE A COUNT OF UNITS (NAVPM) WHOSE GAP LENGTHS CAN HOLD
            C*****   A PERFECT MATCH.  INFSII=2 MEANS START UNSPEC'D; =3 FOR UNSPEC'D STOP.
            C*****   AMMO HAS NOT BEEN CHECKED IN DETERMINING THESE AVAILABILITIES BEYOND
            C*****   CAPACITY FOR A SINGLE VOLLEY.  IT WILL BE ATTEMPTED TO FIT MASSES IN
            C*****   THE FOLLOWING ORDER OF LENGTH:  TMASMX,TMASMN,AVAMIN,AVAMAX.  CASES
            C*****   KNOWN IN ADVANCE NOT TO FIT (LIKE TMASMX>AVAMAX) WILL BE EXCLUDED.
            C*****   NOTE THAT THE MINIMUM NUMBER OF UNITS TRIED FOR A MASS IS 1, WHICH
            C*****   MEANS THAT A SINGLE UNIT COULD BE ASSIGNED IF NO MASS WORKS.
0705          477 ITMNDX=0
0706              IF(TMASMX.GT.AVAMAX)ITMNDX=1
0707              IF(TMASMN.GT.AVAMAX)ITMNDX=2
0708              TIMMAS(1)=TMASMX
0709              TIMMAS(2)=TMASMN
0710              TIMMAS(3)=AVAMIN
0711              TIMMAS(4)=AVAMAX
0712          478 ITMNDX=ITMNDX+1
0713              IF(ITMNDX.LE.4)GO TO 479
0714              INFEAS(II)=1
0715              GO TO 500
            C*****   INITIALIZE AND TRY NEXT LENGTH.
0716          479 TLNGTH=TIMMAS(ITMNDX)
0717              IF(INFSII.EQ.3)SPSTP=SPSTT+TLNGTH
0718              IF(INFSII.EQ.2)SPSTT=SPSTP-TLNGTH
            C*****   CLEAR FLAGS, POINTERS, AND COVERAGES
0719              DO 483 I=1,IGMAX
0720              IUMFLG(I)=0
0721              IUFGMS(I)=0
0722              IUFGME(I)=0
0723              COVRAG(I)=0
0724          483 CONTINUE
0725              MASSER=0
0726              SUMCOV=0.
0727              DO 490 I=III,MI23
            C*****   GET ROW NO. & UNIT NO.
0728              MSIJI=MSIJ(I)
0729              IF(MSIJI.LT.1)GO TO 490
0730              IGXX=IG(MSIJI)
            C*****   IF THIS UNIT CAN GET OFF A VOLLEY, FIND OUT IF ITS GAP FITS TRIAL
            C*****   LENGTH AND, IF SO, HOW MUCH OF TGT IT CAN COVER IN THAT PERIOD.
0731              IF(IUFFLG(IGXX).NE.MSIJI)GO TO 490
0732              MAI=IIM+MSIJI
0733              SUI=SU(MSIJI)
0734              SPSTTX=SPSTT-SUI
```

```
            IUSF=IUSFLG(IGXX)
            IUEF=IUEFLG(IGXX)
            GAPBEG=-SUI
            GAPEND=TIME
            IF(IUSF.EQ.0)GO TO 4904
            IGIU=(IUSF-1)*MAXROW+IGXX
            GAPBEG=STOPS(IGIU)
      4904  IF(IUEF.EQ.0)GO TO 4907
            IGIU=(IUEF-1)*MAXROW+IGXX
            GAPEND=STARTS(IGIU)
      4907  IF((INFSII.EQ.2).AND.(GAPEND.GT.SPSTP))GAPEND=SPSTP
            IF((INFSII.EQ.3).AND.(GAPBEG.LT.SPSTTX))GAPBEG=SPSTTX
            GAPLNG=GAPEND-GAPBEG
            IF(GAPLNG.LT.(TLNGTH+SUI))GO TO 490
            ASMS=AS(MSIJI)
            RVMA=RVEC(MAI)
            VOLLYS=IFIX(TLNGTH/T1(MSIJI))+1.
            SHLPOS=VOLLYS*TU(MSIJI)
            IF(ASMS.LT.SHLPOS)SHLPOS=ASMS
            COVADD=SHLPOS/RVMA
            SUMCOV=SUMCOV+COVADD
            MASSER=MASSER+1
            IUMFLG(IGXX)=MSIJI
            IUFGMS(IGXX)=IUSF
            IUFGME(IGXX)=IUEF
            COVRAG(IGXX)=COVADD
            IF(SUMCOV.LT.1)GO TO 490
            IF((MASSER.GE.MIJ1).OR.(I.GE.MI23))GO TO 491
       490  CONTINUE
            GO TO 478
C*****      A SUITABLE MASS HAS BEEN FOUND -- INITIALIZE AND GO ASSIGN.
C*****      ALSO TRY TO SHORTEN LENGTH IF POSSIBLE
       491  COVRMN=9999.
            RATIOM=1./FLOAT(MASSER)
            DO 493 I=1,IGMAX
            IUEFLG(I)=IUFGME(I)
            IUSFLG(I)=IUFGMS(I)
            IUFFLG(I)=IUMFLG(I)
            COVADD=COVRAG(I)
            IF(IUMFLG(I).LT.1)GO TO 493
            IF(COVRMN.GT.COVADD)COVRMN=COVADD
       493  CONTINUE
C*****      IF LENGTH CAN BE SHORTENED,GO DO IT
            IF(COVRMN.GT.RATIOM)GO TO 494
            TMASMX=TLNGTH
            GO TO 452
C*****      FIND MIN LENGTH REQ'D BY A PARTICIPATING UNIT FOR ITS SHARE AND
C*****      ADJUST PERIOD ACCORDINGLY
       494  TLNGTH=9999.
            DO 495 I=1,IGMAX
            IXROW=IUFFLG(I)
            IF(IXROW.LT.1)GO TO 495
            MAIX=IIM + IXROW
            VOLLYS=RVEC(MAIX)*RATIOM/TU(IXROW)
            VOLFIX=IFIX(VOLLYS)
            IF((VOLLYS-VOLFIX).LE..01)VOLFIX=VOLFIX-1.
            TRYLEN=T1(IXROW)*VOLFIX
            IF(TRYLEN.LT.TLNGTH)TLNGTH=TRYLEN
```

```
          COVRAG( I )=RATIOM
      495 CONTINUE
          IF(INFSII.EQ.2)SPSTT=SPSTP-TLNGTH
          IF(INFSII.EQ.3)SPSTP=SPSTP+TLNGTH
          TMASMX=TLNGTH
          GO TO 452
      500 CONTINUE
     2000 CONTINUE
          RETURN
          END
```

```
*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = VOEGLN  , LINECNT =      60
*STATISTICS*    SOURCE STATEMENTS =      797,PROGRAM SIZE =    19778
*STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
      SUBROUTINE SORTER
C
C        ******************************************************************
C        *  THIS SUBROUTINE SORTS THE K'TH THRU KS'TH ELEMENTS OF MSIJ ACCOR- *
C        *  DING TO DESCENDING P-VALUES THEY POINT AT.  REDEFINITION OF MAT-  *
C        *  RICES MS AND P TO VECTORS MSIJ AND PVEC SPEEDS UP EXECUTION       *
C        ******************************************************************
C
      COMMON /SCOM/K,KSS
      COMMON /COM2/ IP(30),IG(40),MSIJ(1200),MIIJ(90),MAXPRI
      COMMON /UCOM/ CVEC(1200),RVEC(1200),EVEC(1200),SVEC(1200),
     *A(40),SU(40),TI(40),TU(40),TIME,NT,NU,NN,B,ISAME,PVEC(1200)
      KSX=KSS
      KS=KSS-1
      KP=K+1
      KM=K-1
      DO 31 L=K,KS
      ISWAP=0
      KAD=MSIJ(K)
      PVI=1000000.
      IF(KAD.GT.0) PVI=PVEC(KM+KAD)
      DO 32 M=KP,KSX
      KADD=KAD
      KAD=MSIJ(M)
      PVII=PVI
      PVI=1000000.
      IF(KAD.GT.0) PVI=PVEC(KM+KAD)
      IF(PVI.GE.PVII) GO TO 32
      MM=M-1
      MSWAP=MSIJ(MM)
      MSIJ(MM)=MSIJ(M)
      MSIJ(M)=MSWAP
      ISWAP=1
      KAD=KADD
      PVI=PVII
   32 CONTINUE
      IF(ISWAP.NE.1)GO TO 33
      KSX=KSX-1
   31 CONTINUE
   33 RETURN
      END
```

OPTIONS IN EFFECT ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
OPTIONS IN EFFECT NAME = SORTER , LINECNT =   60
STATISTICS   SOURCE STATEMENTS =   33,PROGRAM SIZE =   776
STATISTICS NO DIAGNOSTICS GENERATED

```
      SUBROUTINE OUTPUT
      COMMON /ASCM/ IGAM,MSAM,AFRACT,AMMFIX,BEGINS,ENDS,IIAMX,TCOST,SUAM
      COMMON /AMCM/ MIJC,MI2C,MPUVEC(40),IIAM
      COMMON /AICD/ ICODE
      COMMON /PRCOM/ IPX(20)
      COMMON /UCOM/ CVEC(1200),RVEC(1200),EVEC(1200),SVEC(1200),
     *A(40),SU(40),T1(40),TU(40),TIME,NT,NU,NN,R,ISAME,PVEC(1200)
      COMMON /SCHED/ SPSTRT(30),SPSTOP(30),NSTART(40),STARTS(1200),
     *STOPS(1200),SHELLS(1200),NTARG(1200),NSRANK(1200),NFIRST(40),
     *INFEAS(30)
      COMMON /COMG/ IGMAX,IGX(40),IGG(40)
      COMMON /COMX/ LGRNG(40)
      COMMON /COM2/ IP(30),IG(40),MSIJ(1200),MIIJ(90),MAXPRI
      COMMON /ACOM/ ALPHA,TOPINE
      COMMON /DCOM/ MAXROW,MAXCOL
      COMMON /ASTS/ AS(40),TS(40)
      DIMENSION ICOVRD(200)
C*****
C*****   ICOVRD(J) WILL HAVE A BIT MASK BUILT UP IN IT TO FLAG PROBLEMS WITH
C*****   COVERAGE OF TARGET J.  BITS MEAN FROM RIGHT TO LEFT (LO TO HI):
C*****
C*****          1.  0=NOT COVERED
C*****          2.  1=SECONDARY UNIT(S) INVOLVED
CQ****          3.  1=UNEVEN COVERAGE
C*****          4.. 1=WRONG NO. UNITS MASSED
C*****
      DATA ICOVRD,IFIRST/200*0,1/
       DATA IMOLD,ITGOLD/0,0/
      IF((ICODE.EQ.1).OR.(ICODE.EQ.4))GO TO 500
      IF(ICODE.EQ.0)GO TO 1000
      IF(ICODE.EQ.2)GO TO 25
C*****   UPDATE COUNT OF UNITS MASSED & CHECK FOR PRIMARY UNIT
      MASSCT=MASSCT+1
      IMOKK=1
      DO 10 I=1,MI2C
      IF(MPUVEC(I).EQ.MSAM)IMOKK=0
   10 CONTINUE
      IMOK=IMOK+IMOKK
C*****   IF >1 UNIT MASSED, CHECK FOR UNEVEN COVERAGE (UNLESS FOUND EARLIER)
      IF(MASSCT.GT.1)GO TO 20
      IPOK=0
      AFCHEK=AFRACT
      GO TO 25
   20 IF(IPOK.NE.0)GO TO 25
      IF(ABS(FMIJC*(AFCHEK-AFRACT)).GT.0.1)IPOK=1
      AFCHEK=AFRACT
C*****   PRINT ASGT, ADD TO COST
   25 PRINT 601,IGAM,MSAM,AFRACT,AMMFIX,BEGINS,ENDS,SUAM,TCOST
  601 FORMAT(2I5,F10.6,F8.0,4F7.2)
      TTCOST=TTCOST+TCOST
      IF(ICODE.EQ.3)RETURN
C*****   FLAG COVERAGE OK
      ICOVRD(IIAMX)=1
      RETURN
  500 IF(IFIRST.EQ.1)GO TO 550
      IF(IMOLD.LT.2)GO TO 540
C*****   CHECK FOR PROPER NO. UNITS MASSED IF COVERED AT ALL
      IF(MASSCT.EQ.0) GO TO 540
```

```
                ICOVRG=1
                IF(MASSCT.EQ.IMOLD) GO TO 520
         C*****    FLAG WRONG NO. UNITS MASSED
                ICOVRG=ICOVRG+8
                PRINT 505,IMOLD,MASSCT
            505 FORMAT(' ***  NO. UNITS DESIRED FOR MASSING WAS',I3,
               *'; NO. ACTUALLY USED WAS',I3,'  ***')
            520 IF(IMOK.EQ.0)GO TO 530
         C*****    FLAG SECONDARY INVOLVEMENT
                ICOVRG=ICOVRG+2
                PRINT 525,IMOK
            525 FORMAT(' ***',I5,' SECONDARY UNIT(S) NEEDED  ***')
            530 IF (IPOK.EQ.0)GO TO 537
         C*****    FLAG UNEVEN COVERAGE
                ICOVRG=ICOVRG+4
                PRINT 535
            535 FORMAT(' ***  COVERAGE UNEVEN  ***')
            537 ICOVRD(ITGOLD)=ICOVRG
            540 IF(ICOVRD(ITGOLD).NE.0)GO TO 550
                PRINT 545
            545 FORMAT(' ***  UNASSIGNED  ***')
            550 IF(ICODE.EQ.4) GO TO 1100
                IFIRST=0
                PRINT 600,IIAM,MIJC,(MPUVEC(I),I=1,MI2C)
            600 FORMAT('0ASGMT FOR TGT',I4,'; NO. UNITS DESIRED:',I3,
               *'; PRIMARY ROWS:',20I3,/,(56X,20I3))
                PRINT 660
            660 FORMAT(' UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST')
                ITGOLD=IIAM
                IMOLD=MIJC
                FMIJC=FLOAT(MIJC)
                MASSCT=0
                IMOK=0
                IPOK=0
                RETURN
           1000 PRINT 1025,ALPHA
           1025 FORMAT(20H1RESULTS FOR ALPHA =,F7.5,1H:)
                IFIRST=1
                TTCOST=0.
                RETURN
           1100 PRINT 1101,TTCOST
           1101 FORMAT(50X,'--------',/,47X,F9.2)
         C*****    SUMMARY: TARGET ASSIGNMENTS: VARIATIONS FROM SPECS
                TTCOST=0.
                NUASGD=0
                NSECDY=0
                NBADMS=0
                NUNEVN=0
                NOK=0
         C*****    FOL DO-LOOP CHECKS BIT MASK (DECOMPOSED BY MOD FUNCTION) FOR FLAGS
         C*****    INDICATING PROBLEMS IN ASSIGNMENTS
                DO 1200 I=1,NT
                ICOVRG=ICOVRD(I)
                IF(MOD(ICOVRG,2).EQ.0)GO TO 1110
                NOK=NOK+1
                GO TO 1115
           1110 NUASGD=NUASGD+1
                GO TO 1200
```

```
      1115 ICOVRG=ICOVRG/2
           IF(MOD(ICOVRG,2).EQ.0)GO TO 1120
           NSECDY=NSECDY+1
      1120 ICOVRG=ICOVRG/2
           IF(MOD(ICOVRG,2).EQ.0)GO TO 1125
           NUNEVN=NUNEVN+1
      1125 IF(ICOVRG.LE.1)GO TO 1200
           NBADMS=NBADMS+1
      1200 CONTINUE
           PRINT 1250,NT,NOK,NUASGD,NSECDY,NUNEVN,NBADMS
      1250 FORMAT('0ASSIGNMENT SUMMARY ',/,' NO. TGTS PRESENT WAS',I4,
          *', NO. TGTS ASGD WAS',I4,', NO. TGTS UNASGD WAS',I4,/,
          *' NO. TGTS ASGD TO SECONDARY ROWS WAS',I4,/,
          *' NO. TGTS WITH UNEVEN COVERAGE WAS',I4,/,
          *' NO. TGTS WITH WRONG NO. UNITS MASSED WAS',I4,//)
C
C*****   THIS SECTION DEMONSTRATES TWO WAYS OF DISPLAYING THE SCHEDULE OF
C*****   ASSIGNMENTS.
C
           PRINT 2010
      2010 FORMAT('1SCHEDULE OF FIRING ASSIGNMENTS:',//,' UNIT')
           DO 2500 I=1,IGMAX
           JS=NSTART(I)
           IF(JS.GT.0)GO TO 2030
           PRINT 2020,I
      2020 FORMAT('0',I3,5X,'***  UNASSIGNED  ***')
           GO TO 2500
      2030 PRINT 2031,I
      2031 FORMAT('0',I3,6X,'START   STOP   TGT   ROW   SHELLS')
           JX=NFIRST(I)
      2040 IVSA=(JX-1)*MAXROW+I
           IROW=NTARG(IVSA)
           ITGT=IROW/1000
           IROW=IROW-ITGT*1000
           ISHELS=IFIX(SHELLS(IVSA))
           PRINT 2050,STARTS(IVSA),STOPS(IVSA),ITGT,IROW,ISHELS
      2050 FORMAT(F15.2,F7.2,2I6,I7)
           JX=MOD(NSPANK(IVSA),1000)
           IF(JX.NE.0)GO TO 2040
           IGXI=IGX(I)
           IROW=IGG(I)
           IRSTOP=IROW+IGXI-1
           PRINT 2060,TS(IROW),(J,AS(J),J=IROW,IRSTOP)
      2060 FORMAT(10X,'SLACK TIME: ',G12.5,/,(10X,'SLACK AMMO IN ROW',I4,
          *' IS',F6.0,' SHELLS'))
      2500 CONTINUE
C*****   PRINT SCHEDULE IN BAR CHART FORM.
           CALL CHART
           RETURN
           END
```

```
*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = OUTPUT  , LINECNT =       60
*STATISTICS*    SOURCE STATEMENTS =      131,PROGRAM SIZE =      4362
*STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
      SUBROUTINE CHART
      COMMON /UCOM/ CVEC(1200),RVEC(1200),EVEC(1200),SVEC(1200),
     *A(40),SU(40),T1(40),TU(40),TIME,NT,NU,NN,B,ISAME,PVEC(1200)
      COMMON /SCHED/ SPSTRT(30),SPSTOP(30),NSTART(40),STARTS(1200),
     *STOPS(1200),SHELLS(1200),NTARG(1200),NSRANK(1200),NFIRST(40),
     *INFEAS(30)
      COMMON /COMG/ IGMAX,IGX(40),IGG(40)
      COMMON /COMP/ IP(30),IG(40),MSIJ(1200),MIIJ(90),MAXPRI
      DIMENSION ICHART(40,100),ICVECT(4000)
      EQUIVALENCE (ICHART(1),ICVECT(1))
      DIMENSION SCSTA(40,30),SCSTP(40,30),SCRDS(40,30),NSCTRG(40,30),
     *NRANKS(40,30)
      EQUIVALENCE (STARTS(1),SCSTA(1)),(STOPS(1),SCSTP(1)),
     * (SHELLS(1),SCRDS(1)),(NTARG(1),NSCTRG(1)),(NSRANK(1),NRANKS(1))
      DATA ROWB/'ROW '/
C
C*****   TINC IS TIME INCREMENT FOR CHART
C
      DATA TINC/1./
C
      RT=1./TINC
C CLEAR CHARTING ARRAY
      DO 1 I=1,4000
      ICVECT(I)=0
    1 CONTINUE
      ISMIN=100
      ISMAX=1
      DO 10 I=1,IGMAX
C FOR EACH UNIT FIND NO OF TGTS ASGD & INDEXES OF 1ST & LAST ROWS IN UNIT
      NSTRTI=NSTART(I)
      IF(NSTRTI.LT.1)GO TO 10
      IGGIX=IGG(I)
      IGGIXS=IGGIX+IGX(I)-1
      DO 11 K=1,NSTRTI
C FOR EACH TGT ASGD TO UNIT, GET TGT & ROW INDEXES, USE THEM TO GET START/STOP
C TIMES FOR CALC'G CHARTING INDEXES ON BASIS OF TINC MIN PER CELL.
      NTRG=NSCTRG(I,K)
      IROW=MOD(NTRG,1000)
      ITGT=(NTRG-IROW)/1000
      START=SCSTA(I,K)
      STOP=SCSTP(I,K)
      START=(START+5.)*RT+.5
      STOP =(STOP +5.)*RT+.5
      ISTT=START
      ISTP=STOP
C STAY INSIDE ARRAY FOR TIMES < -5 OR >(95*TINC)
      IF(ISTT.LT.1)ISTT=1
      IF(ISTP.GT.100)ISTP=100
C DETERMINE MIN & MAX CELLS USED (EARLIEST START, LATEST STOP) FOR "DO 60 ..."
      IF(ISTT.LT.ISMIN)ISMIN=ISTT
      IF(ISTP.GT.ISMAX)ISMAX=ISTP
      DO 12 J=IGGIX,IGGIXS
      MARK=9999
C FOR EACH ROW IN UNIT GENERATE A CHART SYMBOL '****' EXCEPT FOR ASGD ROW WHERE
C TGT NO BECOMES SYMBOL
      IF(J.EQ.IROW)MARK=ITGT
      DO 13 L=ISTT,ISTP
C FOR EACH HALF-MIN BETWEEN START & STOP, STORE CHART SYMBOL
```

```
              ICHART(J,L)=MARK
           13 CONTINUE
           12 CONTINUE
           11 CONTINUE
           10 CONTINUE
              PRINT 20,(IG(J),J=1,NU)
           20 FORMAT('1',10X,'***  UNITS  ***',/,(8X,41I3))
              PRINT 30
           30 FORMAT('0',10X,'***  ROWS  ***')
              PRINT 40,(J,J=1,NU)
           40 FORMAT(8X,41I3)
              PRINT 45
           45 FORMAT(' TIME')
              CTIME=FLOAT(ISMIN)*TINC-5.
        C PRINT CHART LINES ONLY AFTER FIRST ENGAGEMENT BEGINS & BEFORE LAST ONE ENDS.
              DO 60 I=ISMIN,ISMAX
              PRINT 50,CTIME,(ICHART(J,I),J=1,NU)
           50 FORMAT(F5.1,3X,41(' ',I2))
              CTIME=CTIME+TINC
           60 CONTINUE
              RETURN
              END

*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = CHART    , LINECNT =       60
*STATISTICS*    SOURCE STATEMENTS =        61,PROGRAM SIZE =     17728
*STATISTICS*  NO DIAGNOSTICS GENERATED

*STATISTICS* 002 DIAGNOSTICS THIS STEP
```

APPENDIX F

OUTPUT SAMPLES FROM PROGRAM IN APPENDIX E

****
*   *
*   SUMMARY OF INPUT DATA   *
*   *
****

MISSION DURATION: 30.00

NO. TARGETS: 10

NO. ROWS: 5

MAX. INEFFICIENCY ALLOWED: 0.5000

NO. ALPHAS TO BE TRIED: 3

COST MATRIX:

ROW:										
1:	50.	45.	40.	50.	60.	30.	20.	1000000.	70.	10.
2:	91.	36.	58.	19.	39.	62.	34.	61.	74.	65.
3:	522.	1000000.	42.	70.	95.	77.	1000000.	38.	86.	47.
4:	47.	1000000.	1000000.	20.	32.	36.	19.	39.	79.	48.
5:	1000000.	22.	42.	37.	60.	86.	14.	1000000.	39.	62.

NO. ROUNDS NEEDED OF FUZE I FOR TARGET J:

ROW:										
1:	36.	60.	21.	24.	57.	35.	48.	0.	98.	34.
2:	53.	94.	52.	28.	29.	69.	38.	14.	11.	7.
3:	1.	0.	63.	17.	19.	57.	0.	55.	40.	87.
4:	55.	0.	0.	32.	83.	45.	90.	94.	43.	81.
5:	0.	43.	62.	43.	49.	91.	21.	21.	72.	12.

*** UNIT PARAMETERS ***

```
ROW NO.:
     1        2        3        4        5

AMMO SUPPLY VECTOR:
   114.     594.     638.     246.     478.

VECTOR OF TIMES (MIN) FOR SETUP & FIRST ROUND:
  2.000    1.000    1.000    1.000    2.000

VECTOR OF TIMES (MIN) PER ROUND (SUSTAINED FIRE):
  0.400    0.050    0.100    0.100    0.500

VECTOR OF NO. TUBES PER ROW:
     6.       1.       6.       6.       1.

VECTOR OF UNIT GROUP NUMBERS:
     1        2        3        3        4
```

*** TARGET PARAMETERS ***

```
PRECEDENCE  1 TARGETS:   1   2
PRECEDENCE  2 TARGETS:   3
PRECEDENCE  3 TARGETS:   4
PRECEDENCE  4 TARGETS:   5
PRECEDENCE  5 TARGETS:   6   7   8
PRECEDENCE  6 TARGETS:   9
PRECEDENCE  7 TARGETS:  10
```

```
MASSING INFORMATION:

TGT  NO. UNITS  ROWS TO BE CONSIDERED:
NO.   TO MASS   (OTHERS HAVE BEEN FLAGGED INFEASIBLE)

 1       3      PRIMARY:     1   2   3   4   5

 2       2      PRIMARY:     1   2
                SECONDARY:   5

 3       2      PRIMARY:     1   3   4   5
                SECONDARY:   2
```

START-STOP INFORMATION (9999 = NOT SPECIFIED)

TARGET	START TIME	STOP TIME
1	4.000	16.000
2	9999.000	20.000
4	8.000	9999.000

RESULTS OF PRELIMINARY CALCULATIONS

SUM OF COLUMN COST MINIMA: 322.00

MATRIX OF ENGAGEMENT TIMES:
(NOTE NEW INFEASIBILITIES DUE TO MASSING, TIME, OR AMMO)

ROW:

1:	4.000	5.600	3.200	3.200	5.600	4.000	4.800	*******	8.400	4.000
2:	3.600	5.650	3.550	2.350	2.400	4.400	2.850	1.650	1.500	1.300
3:	1.000	*******	2.000	1.200	1.300	1.900	*******	1.900	1.600	2.400
4:	1.900	*******	*******	1.500	2.300	1.700	2.400	2.500	1.700	2.300
5:	*******	23.000	32.500	23.000	26.000	*******	12.000	*******	*******	7.500

H(= 255.591)-WEIGHTED MAX(R/A,E/T)-MATRIX:

ROW:

1:	80.712	127.796	47.082	53.808	127.794	78.470	107.616	*******	127.796	76.228
2:	30.671	48.136	30.245	20.021	20.447	37.486	24.281	14.057	12.779	11.076
3:	8.520	*******	25.238	10.224	11.076	22.835	*******	22.033	16.024	34.853
4:	57.144	*******	*******	33.247	86.235	46.754	93.508	97.664	44.676	84.157
5:	*******	127.796	127.796	127.796	127.796	*******	102.235	*******	*******	63.897

```
RESULTS FOR ALPHA =0.00000:

ASGMT FOR TGT   1: NO. UNITS DESIRED:  3: PRIMARY ROWS:  4  1  2
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
   1    1  0.333333     12.   2.00  16.00   2.00  16.67
   2    2  0.339623     18.   3.00  16.00   1.00  30.91
   3    4  0.345455     19.   3.00  16.00   1.00  16.24

ASGMT FOR TGT   2: NO. UNITS DESIRED:  2: PRIMARY ROWS:  2  1
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
***  UNASSIGNED  ***

ASGMT FOR TGT   3: NO. UNITS DESIRED:  2: PRIMARY ROWS:  1  3  5
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
   1    1  0.523809     11.  16.00  18.40   2.00  20.95
   3    3  0.507936     32.  17.00  18.40   1.00  21.33

ASGMT FOR TGT   4: NO. UNITS DESIRED:  1: PRIMARY ROWS:  2  4  5
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
   4    5  1.000000     43.   6.00  29.00   2.00  37.00

ASGMT FOR TGT   5: NO. UNITS DESIRED:  1: PRIMARY ROWS:  4  2  1
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
   3    4  1.000000     83.  -1.00   1.30   1.00  32.00

ASGMT FOR TGT   8: NO. UNITS DESIRED:  1: PRIMARY ROWS:  3  4  2
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
   3    3  1.000000     55.  18.40  20.30   1.00  38.00

ASGMT FOR TGT   6: NO. UNITS DESIRED:  1: PRIMARY ROWS:  1  4  2
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
   1    1  1.000000     35.  -2.00   2.00   2.00  30.00

ASGMT FOR TGT   7: NO. UNITS DESIRED:  1: PRIMARY ROWS:  5  4  1
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
   3    4  1.000000     90.  20.30  22.70   1.00  19.00

ASGMT FOR TGT   9: NO. UNITS DESIRED:  1: PRIMARY ROWS:  1  2  3
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
   2    2  1.000000     11.  -1.00   0.50   1.00  74.00

ASGMT FOR TGT  10: NO. UNITS DESIRED:  1: PRIMARY ROWS:  1  3  4
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
   1    1  1.000000     34.  18.40  22.40   2.00  10.00
                                                -------
                                                 346.09

ASSIGNMENT SUMMARY
NO. TGTS PRESENT WAS  10, NO. TGTS ASGD WAS   9, NO. TGTS UNASGI
NO. TGTS ASGD TO SECONDARY ROWS WAS   0
NO. TGTS WITH UNEVEN COVERAGE WAS   0
NO. TGTS WITH WRONG NO. UNITS MASSED WAS   0
```

```
SCHEDULE OF FIRING ASSIGNMENTS:

UNIT

  1       START   STOP    TGT    ROW    SHELLS
          -2.00   2.00     6      1      35
           2.00  16.00     1      1      12
          16.00  18.40     3      1      11
          18.40  22.40    10      1      34
          SLACK TIME:   5.6003
          SLACK AMMO IN ROW   1 IS   22. SHELLS

  2       START   STOP    TGT    ROW    SHELLS
          -1.00   0.50     9      2      11
           3.00  16.00     1      2      18
          SLACK TIME:   15.500
          SLACK AMMO IN ROW   2 IS  565. SHELLS

  3       START   STOP    TGT    ROW    SHELLS
          -1.00   1.30     5      4      83
           3.00  16.00     1      4      19
          17.00  18.40     3      3      32
          18.40  20.30     8      3      55
          20.30  22.70     7      4      90
          SLACK TIME:   9.0003
          SLACK AMMO IN ROW   3 IS  551. SHELLS
          SLACK AMMO IN ROW   4 IS   54. SHELLS

  4       START   STOP    TGT    ROW    SHELLS
           6.00  29.00     4      5      43
          SLACK TIME:   7.0003
          SLACK AMMO IN ROW   5 IS  435. SHELLS
```

*** UNITS ***

1	2	3	3	4

*** ROWS ***

TIME	1	2	3	4	5
-2.0	6	0	0	0	0
-1.0	6	9	**	5	0
0.0	6	9	**	5	0
1.0	6	9	**	5	0
2.0	6	0	0	0	0
3.0	1	1	**	1	0
4.0	1	1	**	1	0
5.0	1	1	**	1	0
6.0	1	1	**	1	4
7.0	1	1	**	1	4
8.0	1	1	**	1	4
9.0	1	1	**	1	4
10.0	1	1	**	1	4
11.0	1	1	**	1	4
12.0	1	1	**	1	4
13.0	1	1	**	1	4
14.0	1	1	**	1	4
15.0	1	1	**	1	4
16.0	3	1	**	1	4
17.0	3	0	3	**	4
18.0	10	0	8	**	4
19.0	10	0	8	**	4
20.0	10	0	**	7	4
21.0	10	0	**	7	4
22.0	10	0	**	7	4
23.0	0	0	**	7	4
24.0	0	0	0	0	4
25.0	0	0	0	0	4
26.0	0	0	0	0	4
27.0	0	0	0	0	4
28.0	0	0	0	0	4
29.0	0	0	0	0	4

RESULTS FOR ALPHA =0.25000:

ASGMT FOR TGT 1: NO. UNITS DESIRED: 3: PRIMARY ROWS: 4 1 2 3 5

UNIT	ROW	FRACTION	SHELLS	START	STOP	SETUP	COST
1	1	0.333333	12.	2.00	16.00	2.00	16.67
2	2	0.339623	18.	3.00	16.00	1.00	30.91
3	4	0.345455	19.	3.00	16.00	1.00	16.24

ASGMT FOR TGT 2: NO. UNITS DESIRED: 2: PRIMARY ROWS: 2 1

UNIT	ROW	FRACTION	SHELLS	START	STOP	SETUP	COST

*** UNASSIGNED ***

ASGMT FOR TGT 3: NO. UNITS DESIRED: 2: PRIMARY ROWS: 1 3 5 4

UNIT	ROW	FRACTION	SHELLS	START	STOP	SETUP	COST
1	1	0.523809	11.	16.00	18.40	2.00	20.95
3	3	0.507936	32.	17.00	18.40	1.00	21.33

ASGMT FOR TGT 4: NO. UNITS DESIRED: 1: PRIMARY ROWS: 2 4 5 1 3

UNIT	ROW	FRACTION	SHELLS	START	STOP	SETUP	COST
4	5	1.000000	43.	6.00	29.00	2.00	37.00

ASGMT FOR TGT 5: NO. UNITS DESIRED: 1: PRIMARY ROWS: 4 2 5 1 3

UNIT	ROW	FRACTION	SHELLS	START	STOP	SETUP	COST
3	4	1.000000	83.	-1.00	1.30	1.00	32.00

ASGMT FOR TGT 8: NO. UNITS DESIRED: 1: PRIMARY ROWS: 3 4 2

UNIT	ROW	FRACTION	SHELLS	START	STOP	SETUP	COST
3	3	1.000000	55.	18.40	20.30	1.00	38.00

ASGMT FOR TGT 6: NO. UNITS DESIRED: 1: PRIMARY ROWS: 4 1 2 3

UNIT	ROW	FRACTION	SHELLS	START	STOP	SETUP	COST
3	4	1.000000	45.	1.30	3.00	1.00	36.00

ASGMT FOR TGT 7: NO. UNITS DESIRED: 1: PRIMARY ROWS: 5 4 2 1

UNIT	ROW	FRACTION	SHELLS	START	STOP	SETUP	COST
3	4	1.000000	90.	20.30	22.70	1.00	19.00

ASGMT FOR TGT 9: NO. UNITS DESIRED: 1: PRIMARY ROWS: 2 1 3 4

UNIT	ROW	FRACTION	SHELLS	START	STOP	SETUP	COST
2	2	1.000000	11.	-1.00	0.50	1.00	74.00

ASGMT FOR TGT 10: NO. UNITS DESIRED: 1: PRIMARY ROWS: 1 3 4 5 2

UNIT	ROW	FRACTION	SHELLS	START	STOP	SETUP	COST
1	1	1.000000	34.	-2.00	2.00	2.00	10.00

							352.09

ASSIGNMENT SUMMARY
NO. TGTS PRESENT WAS 10, NO. TGTS ASGD WAS 9, NO. TGTS UNASGD WAS 1
NO. TGTS ASGD TO SECONDARY ROWS WAS 0
NO. TGTS WITH UNEVEN COVERAGE WAS 0
NO. TGTS WITH WRONG NO. UNITS MASSED WAS 0

SCHEDULE OF FIRING ASSIGNMENTS:

UNIT

UNIT	START	STOP	TGT	ROW	SHELLS
1	-2.00	2.00	10	1	34
	2.00	16.00	1	1	12
	16.00	18.40	3	1	11

SLACK TIME: 9.6003
SLACK AMMO IN ROW 1 IS 57. SHELLS

UNIT	START	STOP	TGT	ROW	SHELLS
2	-1.00	0.50	9	2	11
	3.00	16.00	1	2	18

SLACK TIME: 15.500
SLACK AMMO IN ROW 2 IS 565. SHELLS

UNIT	START	STOP	TGT	ROW	SHELLS
3	-1.00	1.30	5	4	83
	1.30	3.00	6	4	45
	3.00	16.00	1	4	19
	17.00	18.40	3	3	32
	18.40	20.30	8	3	55
	20.30	22.70	7	4	90

SLACK TIME: 7.3003
SLACK AMMO IN ROW 3 IS 551. SHELLS
SLACK AMMO IN ROW 4 IS 9. SHELLS

UNIT	START	STOP	TGT	ROW	SHELLS
4	6.00	29.00	4	5	43

SLACK TIME: 7.0003
SLACK AMMO IN ROW 5 IS 435. SHELLS

	*** UNITS ***				
	1	2	3	3	4
	*** ROWS ***				
	1	2	3	4	5
TIME					
-2.0	10	0	0	0	0
-1.0	10	9	**	5	0
0.0	10	9	**	5	0
1.0	10	9	**	6	0
2.0	10	0	**	6	0
3.0	1	1	**	6	0
4.0	1	1	**	1	0
5.0	1	1	**	1	0
6.0	1	1	**	1	4
7.0	1	1	**	1	4
8.0	1	1	**	1	4
9.0	1	1	**	1	4
10.0	1	1	**	1	4
11.0	1	1	**	1	4
12.0	1	1	**	1	4
13.0	1	1	**	1	4
14.0	1	1	**	1	4
15.0	1	1	**	1	4
16.0	3	1	**	1	4
17.0	3	0	3	**	4
18.0	3	0	8	**	4
19.0	0	0	8	**	4
20.0	0	0	**	7	4
21.0	0	0	**	7	4
22.0	0	0	**	7	4
23.0	0	0	**	7	4
24.0	0	0	0	0	4
25.0	0	0	0	0	4
26.0	0	0	0	0	4
27.0	0	0	0	0	4
28.0	0	0	0	0	4
29.0	0	0	0	0	4

```
RESULTS FOR ALPHA =1.00000:

ASGMT FOR TGT    1: NO. UNITS DESIRED:  3: PRIMARY ROWS:  4  1  2  3  5
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
  1    1  0.333333     12.    2.00  16.00   2.00  16.67
  2    2  0.339623     18.    3.00  16.00   1.00  30.91
  3    4  0.345455     19.    3.00  16.00   1.00  16.24

ASGMT FOR TGT    2: NO. UNITS DESIRED:  2: PRIMARY ROWS:  2  1
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
***  UNASSIGNED  ***

ASGMT FOR TGT    3: NO. UNITS DESIRED:  2: PRIMARY ROWS:  3  5  1  4
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
  4    5  0.387097     24.   15.00  28.35   2.00  16.26
  3    3  0.619048     39.   16.00  28.35   1.00  26.00
***  COVERAGE UNEVEN  ***

ASGMT FOR TGT    4: NO. UNITS DESIRED:  1: PRIMARY ROWS:  2  4  5  1  3
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST

ASGMT FOR TGT    5: NO. UNITS DESIRED:  1: PRIMARY ROWS:  4  2  5  3  1
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
  3    4  1.000000     83.   -1.00   1.30   1.00  32.00

ASGMT FOR TGT    8: NO. UNITS DESIRED:  1: PRIMARY ROWS:  3  4  2
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
  2    2  1.000000     14.   -1.00   0.65   1.00  61.00

ASGMT FOR TGT    7: NO. UNITS DESIRED:  1: PRIMARY ROWS:  5  4  2  1
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
  4    5  1.000000     21.   -2.00  10.00   2.00  14.00

ASGMT FOR TGT    6: NO. UNITS DESIRED:  1: PRIMARY ROWS:  4  2  3  1
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
  3    4  1.000000     45.    1.30   3.00   1.00  36.00

ASGMT FOR TGT    9: NO. UNITS DESIRED:  1: PRIMARY ROWS:  2  4  3  1
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
  2    2  1.000000     11.    0.65   2.15   1.00  74.00

ASGMT FOR TGT   10: NO. UNITS DESIRED:  1: PRIMARY ROWS:  3  4  5  2  1
UNIT  ROW  FRACTION  SHELLS  START   STOP  SETUP   COST
  2    2  1.000000      7.   16.00  17.30   1.00  65.00
                                                -------
                                                 388.07

ASSIGNMENT SUMMARY
NO. TGTS PRESENT WAS  10, NO. TGTS ASGD WAS    9, NO. TGTS UNASGD WAS   1
NO. TGTS ASGD TO SECONDARY ROWS WAS    0
NO. TGTS WITH UNEVEN COVERAGE WAS    1
NO. TGTS WITH WRONG NO. UNITS MASSED WAS    0
```

```
SCHEDULE OF FIRING ASSIGNMENTS:

UNIT

 1        START   STOP    TGT    ROW    SHELLS
           2.00  16.00      1      1      12
          SLACK TIME:   16.000
          SLACK AMMO IN ROW    1 IS   102. SHELLS

 2        START   STOP    TGT    ROW    SHELLS
          -1.00   0.65      8      2      14
           0.65   2.15      9      2      11
           3.00  16.00      1      2      18
          16.00  17.30     10      2       7
          SLACK TIME:   12.550
          SLACK AMMO IN ROW    2 IS   544. SHELLS

 3        START   STOP    TGT    ROW    SHELLS
          -1.00   1.30      5      4      83
           1.30   3.00      6      4      45
           3.00  16.00      1      4      19
          16.00  28.35      3      3      39
          SLACK TIME:  0.65027
          SLACK AMMO IN ROW    3 IS   599. SHELLS
          SLACK AMMO IN ROW    4 IS    99. SHELLS

 4        START   STOP    TGT    ROW    SHELLS
          -2.00  10.00      7      5      21
          15.00  28.35      3      5      24
          SLACK TIME:   4.6503
          SLACK AMMO IN ROW    5 IS   433. SHELLS
```

	*** UNITS ***				
	1	2	3	3	4
	*** ROWS ***				
	1	2	3	4	5
TIME					
-2.0	0	0	0	0	7
-1.0	0	8	**	5	7
0.0	0	8	**	5	7
1.0	0	9	**	6	7
2.0	1	9	**	6	7
3.0	1	1	**	6	7
4.0	1	1	**	1	7
5.0	1	1	**	1	7
6.0	1	1	**	1	7
7.0	1	1	**	1	7
8.0	1	1	**	1	7
9.0	1	1	**	1	7
10.0	1	1	**	1	7
11.0	1	1	**	1	0
12.0	1	1	**	1	0
13.0	1	1	**	1	0
14.0	1	1	**	1	0
15.0	1	1	**	1	3
16.0	1	10	3	**	3
17.0	0	10	3	**	3
18.0	0	0	3	**	3
19.0	0	0	3	**	3
20.0	0	0	3	**	3
21.0	0	0	3	**	3
22.0	0	0	3	**	3
23.0	0	0	3	**	3
24.0	0	0	3	**	3
25.0	0	0	3	**	3
26.0	0	0	3	**	3
27.0	0	0	3	**	3
28.0	0	0	3	**	3

*** END ***

VITA 2

Henry Crawford Thibault, II

Candidate for the Degree of

Doctor of Philosophy

Thesis: HEURISTIC SOLUTION METHODS FOR MULTI-RESOURCE GENERALIZED ASSIGNMENT PROBLEMS

Major Field: Industrial Engineering

Biographical:

Personal Data: Born in El Dorado, Arkansas, December 13, 1941, the son of Mr. and Mrs. Henry C. Thibault.

Education: Graduated from Norphlet High School, Norphlet, Arkansas, in May, 1959; enrolled in pre-medical coursework at the University of Arkansas, 1959-1963; received Bachelor of Science in Business Administration degree (major field: Data Processing and Quantitative Analysis) from the University of Arkansas in 1975; received Master of Science in Operations Research degree from the University of Arkansas in 1976; completed requirements for the Doctor of Philosophy degree at Oklahoma State University in December, 1978.

Professional Experience: Data Processing Specialist, U. S. Army Security Agency, 1963-67; Instructor of Computer Programming, Control Data Institut der Control Data GmbH (Germany), 1967-68; programmer/analyst and systems designer, RCA International Service Corporation (Germany), 1968-72; consultant, 1972-75; graduate teaching and research assistant, University of Arkansas, 1974-75; graduate teaching and research assistant, Oklahoma State University, 1975-77; Assistant Professor of Data Processing and Quantitative Analysis, University of Arkansas, 1977 to present.