DESIGN AND IMPLEMENTATION

OF A

COMPUTER PACKAGE OF OPERATIONAL RESEARCH ALGORITHMS

by

Daniel Sing-Ip Lai

A thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
Master of Arts
in
Department of Actuarial and Management Science

Winnipeg, Manitoba, 1984

DESIGN AND IMPLEMENTATION

OF A

COMPUTER PACKAGE OF OPERATIONAL RESEARCH ALGORITHMS

by

Daniel Sing-Ip Lai

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba in partial fulfillment of the requirements
of the degree of

MASTER OF ARTS

ABSTRACT

This thesis details the development and design of a computer software package called FORUM. FORUM stands for a ***FORTRAN-77 package for Operational Research developed at University of Manitoba.*** It is a general, multi-purpose, easy-to-use and well-documented portable software package for users in operational research. This package contains a collection of algorithms to deal with Transportation and Network-Flow problems. They are further divided into five major topics, which are (1) Path algorithms, (2) Transportation algorithms, (3) Assignment algorithms, (4) Sequencing algorithms and (5) Critical Path Method. Gerrit A. Blaauw's methodology is used to design and implement the above algorithms into a computer package.

ACKNOWLEDGEMENTS

I wish to thank Dr. Amir Bukhari for consenting to be my advisor for this thesis and his sound judgement in pointing out the general direction this thesis should take. I am greatly indebted to Dr. Roger Hall for his numerous insightful comments and general encouragement. I also thank Dr. Earl Rosenbloom for his help and patience. Finally, I wish to express my appreciation to Irvin Boschmann for his editing and proofreading skills and the time he took for discussing this thesis with me.

CONTENTS

Chapter I

INTRODUCTION

This thesis describes the development of an Operational Research applications package called FORUM which stands for ***FORTRAN-77 package for Operational Research developed at University of Manitoba.*** This package will contain programs which implement four classes of algorithms: allocation algorithms, sequencing algorithms, co-ordination algorithms, and path algorithms. These programs are not available in any present OR application packages to the author's knowledge. The basic structure of FORUM is shown in Figure 1:

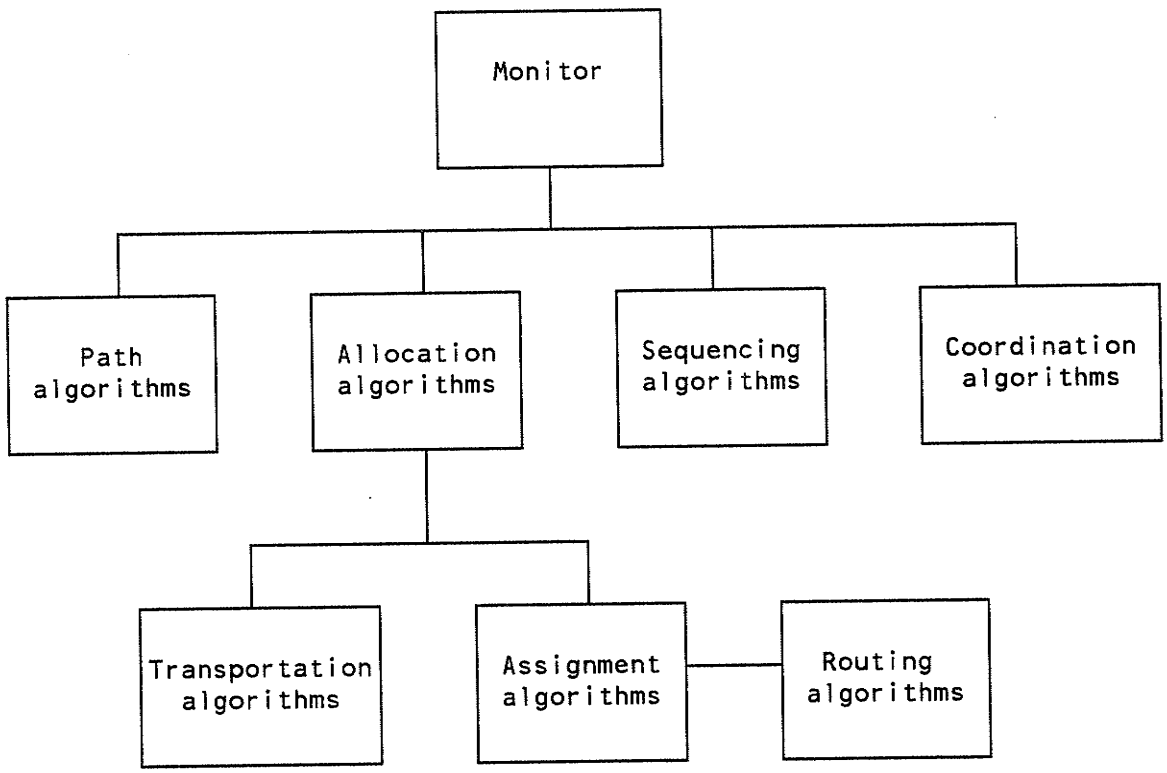


Figure 1: Block Structure of Forum

The monitor is the master program in FORUM which invokes the appropriate subroutines. It is programmed to recognize the first six characters of the name of each subroutine or function. These subroutines and functions contain the code for the algorithms in FORUM.

Path algorithms can be used to solve a number of optimization problems. The solutions to the problems of finding the shortest path in a network and all shortest paths in a network are discussed in chapter 3.

An allocation problem is one that involves determining how resources should be used so as to perform a set of jobs in the best way. In chapters 4 and 5, two relatively simple types of allocation problem are considered: transportation and assignment.

The transportation (or distribution) problem involves the allocation of resources from one or more sources to jobs requiring them (destination), where the jobs may be performed by combining resources from several sources. The algorithms presented stress that it is more important to send a larger shipment by a cheap route than it is to send a small shipment. Most problems involving the redistribution of empty freight cars, trucks, and planes are of this type, as are problems of supplying demand at many locations from several supply points.

In the assignment problem, a unique one-to-one pairing of resources and jobs is made so as to minimize the sum of the measures of performance of each pairing. If there are more jobs to do than can be done, we can decide either which job to leave undone or what resources

to add. The classical Hungarian method [Lee, 1981] is used to illustrate this point.

Both the assignment and the transportation problems are special cases of the more general allocation problem. The Simplex method is the computational procedure used by most of the textbooks to solve the general linear allocation problem.

Routing problems involve finding a path through a network which minimizes (or maximizes) some function of a property of the links in the path selected. We consider only one branch and bound method solution for the traveling salesman problem, namely, Eastman's algorithm [Eastman, 1958]. It should be clear that further improvements in currently available algorithms, particularly the Little et al's [Little, 1963] method, are desirable.

In the last two chapters, two types of problems are considered that involve the order of tasks: the sequencing problem and the coordination problem. The objective of the former is to find an order in which to perform independent tasks that use common facilities, so that some measure of performance is optimized that covers the entire set of tasks. The latter provides the order of interdependent tasks. Its objectives are to (1) find the critical tasks that control the time required to complete the set and (2) how much to spend on getting the critical tasks done so as to meet a desired completion date at least cost.

As Ackoff mentions, only the simplest of sequencing problems can be solved exactly. Others generally require simulation, which does not assure an optimal solution, but, yields better solutions in most cases than can be obtained by intuition and experience. As he puts it:

> In sequencing problems that involve large numbers of servicing facilities, particularly when there is some uncertainty concerning task times, it has been found preferable to apply a rule for ordering jobs waiting at each work center rather than attempt a complete a priori ordering. [Ackoff, 1968, p.300]

Coordination problems, on the other hand, can be treated by PERT (Program Evaluation and Review Technique) and the CPM (Critical Path Method). The former can take account of uncertainties in task times under special conditions, but does not concern itself with direct control of task times by allocation of resources to tasks. The CPM does this, but only in a deterministic context. Despite the limitations, these techniques are widely used and are reported as being very helpful in a wide variety of problems.

1.1 LANGUAGE AND ALGORITHMS

Efficiency with respect to languages can be measured several ways. The expressiveness of the language itself may make it possible to write programs that require fewer execution steps. An example of this is the implicit ability of APL to handle rows and columns of arrays compared with the need for explicit DO loops in FORTRAN for the same procedures. APL might not only result in a shorter source program but the resulting object program may run faster.

For many applications involving small to moderate data arrays, APL is an ideal language and the functions are sufficiently concise to be entered into the computer by hand. FORTRAN is more efficient than APL for repeated analysis on large arrays.

The solution process for all the applications in FORUM involves the manipulation of arrays. Depending on the nature of the problem (user

specifications), these arrays could be very large and many passes through them could be required.

Another consideration in the choice of computer language is the availability of easily usable algorithms. Communications of the ACM[1] publish algorithms submitted by its readers because of their general interest to other readers. Many of these are mathematical procedures, such as matrix operations, internal sorts, and statistical routines written in the form of ALGOL programs (or FORTRAN or PL/1). Thus, if a problem can be expressed in algorithmic form, the design process will be shortened. Most algorithms can be represented by flowcharts or algebraic equations. This makes it fairly easy to trace the flow and calculate the execution time of the program.

These two factors, plus the fact that FORTRAN is more universally available, favor FORTRAN as the language to use in FORUM rather than APL, even though the amount of code would be considerably less in APL.

1.2 TOP-DOWN DESIGN

Top-down is the natural way to design a program unit. As long as the unit is conceptually manageable, most designers will approach it in this way. By completely defining the inputs, the data sets, the outputs, and the transforms, the designer has structured a solution to the problem. The importance of top-down design is that it establishes the logical structure of the solution before it decides on the detailed elements of the solution. The approach leads itself to

---

[1] Communications of the Association for Computing Machinery, 1960-1971. The Algorithms department of Comm. ACM has collected algorithms to make them more useful to a larger group of users.

controlled, verifiable, economical development of a design that satisfies the tradeoff criteria.

Top-down design simultaneously provides complete design control and logical modularity. These advantages can be carried over into implementation by top-down programming; that is, writing the code for each level of the design as the design is developed. The alternative method of implementation delays coding until most or all of the lowest level modules have been designed. The separate modules are then integrated from bottom up.

This thesis will attempt to construct FORUM by using a particular top-down design method that was developed by Blaauw for the design and implementation of digital system.

1.3 BOTTOM-UP PROGRAMMING

The only difference between top-down and bottom-up programming at the program unit level is in the amount of detail in the control logic at the time the functional blocks are coded. The bottom-up approach typically starts with a flowchart outlining the completed design of the program unit. Each block in the flowchart is programmed without much regard to the other blocks. By taking this approach, the programmer does not have all the block dependencies and structures defined at the start.

According to Aron [Aron, 1974], a temporary "driver" is necessary which defines the data and the control interface the block needs for execution. The definition of the "driver" is built up from the local needs of the block. It has no relation to the rest of the program

environment. Gradually, as several blocks are coded, it will become clear that some of the support used in the "driver" is common to several blocks. Data structures and improved interfaces can be constructed with this knowledge.

Top-down design and bottom-up programming each theoritically have their advantages. FORUM uses essentially a top down design approach but some bottom up programming is done as well to test the relative effectiveness of the techniques.

Thus, many of the intervening variables in FORUM are shared and stored in the COMMON blocks to make them available to different algorithms. For example, the COST matrix is required by most of the transportation algorithms. In FORUM, it is stored in COMMON, and subroutines DIJKST, NORTHW, CHEAP, VAM, BECTOR, ASSIGN can all access it in order to compute and/or iterate to the optimal solution. It becomes a global element in FORUM.

1.4 STRUCTURED PROGRAMMING

Advances in programming technique now allow the use of structured programming. A structured program is written in execution sequence so that there is a visible relationship between the program listing and the dynamic execution of the program [Linger, 1972] instead of a spaghetti of pointers which results when unconditional transfer of control instructions are used to implement decisions in a program in an uncontrolled manner. A decisionless structured program would be a simple string of declarations and executable statements. Ideally, decisions are incorporated, using the constructions:

```
IF  S  THEN DO
    A
ELSE
    B
END IF
```

If the logical expression S is true, execution continues with the first statements in the *A* block; otherwise, the program control is passed directly to the first statemen in block *B*.

```
WHILE  S  DO
    A
END WHILE
```

Statements in block *A* are executed and performed as a loop control procedure until the value of the logical expression S becomes false.

```
DO CASE S
  CASE 1
  CASE 2
      .
      .
  CASE n
END CASE
```

The integer expression, denoted above by S, is evaluated at the point of reference, program control is transferred to the statement identifier by the CASE i-th statement in the list following the keyword DO CASE.

In this approach, no unconditional branches are allowed; therefore, some people refer to programs structured in this manner as GOTO-free programs. Algorithms in FORUM are designed using a similar structured methodology. FORUM is implemented in FORTRAN-77 which has only the IF-THEN-ELSE structure. Other control structures (DO-WHILE, DO-CASE)

are simulated in FORTRAN-77 using GOTOs. In some cases, there is also a limited, controlled use of GOTOs. The use of a GOTO statement is not necessarily bad, in fact, in certain cases it provides more efficient code. It is the uncontrolled use of GOTOs which results in programs which are difficult to follow, test and debug. Thus, unconditional branches are excluded from the design, but not the implementation.

1.5 DECISION TABLES

If a program is very simple, with only a few conditions, intuition will often be sufficient to do a proper job of analysis, design and implementation. Modularity is often not a concern, since the entire program can be rewritten quickly. However, when the number of inputs begins to grow, and when the interrelationships between the inputs become more complex, an intuitive approach is not sufficient.

A better way of designing a program is to construct a ***Decision Table.*** A decision table can be built to show all combinations of input and output. It can easily be rearranged so that all combinations of input conditions leading to certain action are grouped together as a decision rule. As Yourdon states it:

> They [the decision rules] list all possible combinations of input conditions, even though many combinations may be impossible, erroneous, or redundant or meaningless. This has the advantage of leading to a very simple programming implementation: the indexed-branch approach. [Yourdon, 1975, p.104]

The monitor in FORUM uses the COMPUTED GOTO to implement this technique.

1.6 DYNAMIC STORAGE ALLOCATION

The size of tables has been defined symbolically in the FORTRAN subroutines in the appendices (see subroutine EASTMN). This approach becomes particularly valuable when there are several tables, whose sizes are all related to one another. If there is a static allocation in the subroutines, a user with a larger than usual problem would have to change all the subroutines to get sufficient storage. As it stands, one need only change a few parameters in the mainline program. Conversely, a static allocation large enough to handle all the ordinary problems and most of the "large" ones would generally lead to a far greater storage allocation than necessary.

Chapter II

METHODOLOGY

In the design and implemenation of systems: first, the architecture must be defined, ie. WHAT the system is supposed to do. Then, the implementation process is detailed, ie. HOW the WHAT will be accomplished. Finally, the actual components and their interconnections are developed. Blaauw calls this the realization step. This can be more simply described as the WHERE of the system.

Blaauw [1976] gives a number of examples of how his methodology can be used in the construction of combinational circuits. A simple example taken from Blaauw, (that of the bit adder) is discussed in the appendix. Blaauw has used a programming language for describing the architecture, and for describing the details of a proposed implementation. The rationale behind this is that a language is required to describe things. If a language is chosen which has the qualities of precise and unambiguous expression, then it can serve as a good descriptive tool. Furthermore, if the language chosen happens to be a real programming language then it gives the added benefit that anything described in it can be executed on a computer and the results verified. For example, if the behaviour of an electronic circuit is described in a programming language algorithmically, then executing this program will result in an explicit display of the behaviour of the circuit. By providing the program with different inputs the

behaviour of the circuit can be logically tested. Upon completion of such testing, if the circuit was built physically, then a great deal of confidence could be placed in the physical circuit as a logically designed entity.

Blaauw has chosen the APL language. APL is particularly suited to this purpose, since it allows expression at the highest architectural level, at the lowest implementation level, and at all levels between. It is a powerful interactive language in which complicated computational tasks can be specified very succinctly. The idea of using a formal language to specify the behaviour of a system is not entirely new. It is now common to use pseudo code to specify algorithms before they are expressed in a programming language. However, the methodology to use an actual programming language, say A, to express the intended behaviour of a program and then using another programming language, say B, to actually implement the program for full production use has not been used. The advantages of such an approach will be realized if it is possible to express large computing task by very short, perhaps inefficient programs in programming language A and very efficient but large programs in language B. It then becomes possible to write complex software in programming language A, and to test and verify the method. Once testing is complete then efficient software can be developed in programming language B using the program developed in programming language A as a guide. For our purposes, APL will qualify as programming language A and FORTRAN-77 as programming language B.

In FORUM this methodology was applied to implement several algorithms. In some cases, certified algorithms were available in ALGOL 60 programming language. These were used as the blue print to write FORTRAN programs. Since ALGOL 60 version is certified to be true, this fact gives us a way to test out the FORTRAN version. If, for some data, the same results are produced by the two programs, the FORTRAN program can be taken to be correct. In the case of Dijkstra's algorithm, APL was used to first write a program implementing it. This programs consists of 7 lines excluding comments (the same program could be further condensed into 4 lines with some loss of clarity). Then a FORTRAN program was written using the APL program as a blue print.

If the larger program in FORTRAN were written independently it would require considerably more debugging and testing than writing a 4-line program because of its larger size. Assuming that it is worthwhile to write these blue prints for writing larger programs in other languages, then results of larger program can be checked against the results of the shorter program, and in this way, the behaviour of the larger program can be verified. This method is useful if the number of lines to be written in the shorter program is considerably less than those in the larger program (say, in a ratio 1:10).

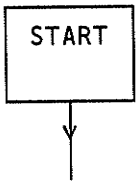
In summary, the suggested approach is as follows:

1. Implement an algorithm in a programming language A, say, in m lines of code
2. Verify and test it to be correct
3. Use the program in 1 as a blue print to write a program in programming language B, say, in n lines
4. Run both programs with the same data. If the results are the same then accept the second program as written.

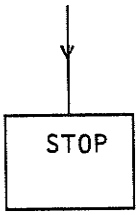
The approach will become more advantageous as m/n becomes smaller. The intuitive reason for adopting this two step approach is that to write large programs from first principle is a difficult task. A blue print of the work will always be an aid.

There are several structured methodologies available for writing software, such as Structured Flowcharts, Decision Tables [IBM, 1962], HIPO [IBM, 1976], Warnier/Orr diagrams [Warnier, 1974] [Orr, 1977], Boehm's method [Boehm, 1972]. One example is the methodology which proceeds in two steps.

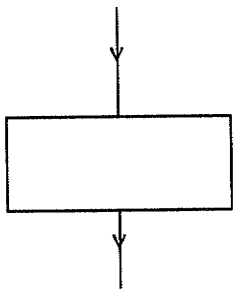
STEP 1: Construct a flowchart of the algorithm to be written in a programming language using only the following components:



Signifies "Start" of the program.

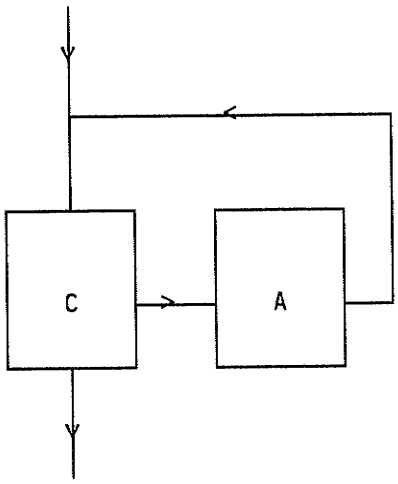


Signifies "End" of the program.



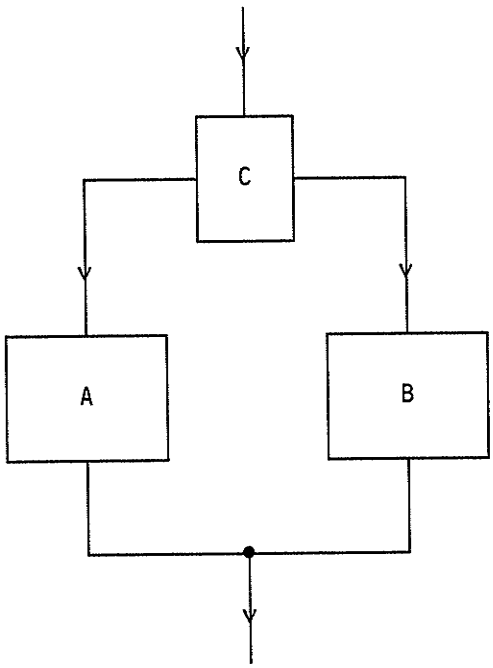
COMPUTATION

Computations done are shown in the rectangular box.



ITERATION

Upon entry condition C is tested, while it is true computation A is performed. When C becomes false, exit is made from the loop.



ALTERNATION

Condition C is tested. If it is true, A is performed; otherwise B is performed and the exit is the same.

The rules are:

START and STOP occur only once in the flowchart. All other components use a single input and emit a single output.

STEP 2: Each part of the flow chart is then translated into the programming language in a one-to-one manner.

2.1 BLAAUW'S METHODOLOGY AND FORUM

Blaauw's methodology is more structured than that described above. If applied to software design and implementation, it becomes a three-step procedure:

STEP 1: Define the architecture (WHAT) of the problem. This involves specifying the inputs and outputs required in a problem. The solution process is treated as a 'black box' at this stage.

STEP 2: Formulate the implementation (HOW) of the problem. This involves opening the 'black box' and specifying the steps necessary to move from the given inputs to the required outputs.

STEP 3: Develop the realization (WHERE) of the problem. This involves the actual coding of the programs necessary to use the solution algorithms on the computer.

The design and implementation of FORUM will follow Blaauw's approach.

By doing this, the present study will deal with the issue of whether a highly structured approach, which has proven effective in the design and implementation of hardware systems, can be used in the same form to design and implement a software system.

2.2 TECHNICAL CONSIDERATIONS

A number of technical decisions and tradeoffs need to be made while composing the actual coding. One decision is whether to simplify equations in order to speed up computation. This can be done by replacing simple divisions by multiplications, and simple multiplications by additions. For example,

$$A = B/2. + C^2 + 2*D$$

can be written as:

$$A = B*0.5 + C * C + D + D$$

The improvement would become obvious when hundreds of thousands of iterations on this formula are performed.

Furthermore, the efficiencies will become even better when the number of multiplications and additions are reduced by mathematical simplification; ie., the following equation

$$W = 6X^3 + 3X^2 + 5X + 24$$

which in FORTRAN would be written

```
W = 6*X**3 + 3*X**2 + 5*X + 24
```

involves

multiplication:	1	2	1	1	1	total 6
addition:		1		1	1	total 3

The same equation could be written in a form like the following:

```
W = ( ( 2*X + 1 )*3*X + 5 )*X + 24
```

which involved

multiplication:	1	1	1	1	total 4
addition:		1	1	1	total 3

Efficiency is improved by the elimination of two multiplications.

However, developing such simplifications is very time consuming. Also, any future modifications become more difficult as the equations are no longer in 'standard' form. Thus, the code developed in FORUM does not contain these simplifications.

Another decision is the number of check-points to put into the programs. Each set of algorithms includes a number of 'special' cases and in order for an applications package to be complete, it must be able to handle all of these. This implies that there is some special-purpose code which is not required for the solution of 'regular' problems. One can either accept these redundancies or use a series of IF statements to make certain that a problem is not subjected to any unnecessary solution processes. If the latter approach is taken, one gains efficiency. The cost, however, is an increase in the complexity and a decrease in the clarity of the coding. In the development of FORUM, efficiency is sacrificed for simplicity and clarity. These two examples show that conceptual clarity and simplicity are often in conflict with efficiency and decisions and tradeoffs must be made.

The next 5 chapters (chapter 3 to chapter 7) each discuss one set of algorithms. The structure of each of these chapters will be the same. First, the WHAT of the set of algorithms will be developed, followed by the HOW and the WHERE.

Chapter III

PATH ALGORITHMS

3.1 SHORTEST PATH ALGORITHM

Shortest-path problems are among the most fundamental of all combinatorial optimization problems since a large number of such problems can be reduced to finding a shortest path in a network. For instance, a linear programming model of the Shell Oil Company's petroleum distribution network is presented by Zierer, Mitchell and White [Zierer, 1974]. Products from 3 Shell refineries can be shipped by three different modes of transportation --- pipelines, barges and ocean tankers to various terminals. Thus, the nodes (vertices) of the network represent the refineries and terminals, while the arcs represent the costs/distances between each pair of nodes. For such a large network, there will obviously be many different paths between any two nodes. Using the shortest path between the refineries and terminals would be the simplest way to cut down the cost.

Further, there are several types of shortest-path problems: (1) between two specified vertices; (2) between one specified vertex and all other vertices in a network; (3) between every pair of vertices in a network; (4) between two specified vertices which must pass through one or more specified vertices; and (5) the first, second or third shortest paths in a network.

One of the best shortest-path algorithms (virtually identical to the minimum-weight spanning tree algorithm) is Dijkstra's algorithm [Bondy, 1980, pp.15-20], which determines the shortest distance from a specified vertex to a final vertex.

The network is represented by an adjacency matrix A, where

A(i,j) = the length of the edge between vertices i and j

A(i,j) = infinity, if no edge exists between i and j

A(i,i) = 0 for all i=1,2,..,M (M is the number of vertices)

Dijkstra's algorithm repeatedly performs the following steps:

1. It starts by determining the direct distances; that is the distance along an edge, from the specified vertex Vo to all other vertices
2. It then picks the shortest of these distances, say Vi, as a "temporary" shortest distance
3. The algorithm them adds this shortest distance to the length of the edge from the new vertex Vi to each other vertex
4. It compares this sum with the previous distance from Vo to each vertex, and it updates the old distance if the new distance is shorter
5. The new vertex Vi is then removed from the list of those vertices, the shortest distances to which have not yet been determined
6. Repeat steps 2 to 5 until the list is empty

3.1.1 *Architecture of Dijkstra's Algorithm*

The requirements of the architecture of the above algorithm are:

The input parameters are:

1. A(i,j) - the length of the edge between i and j (adjacency matrix)
2. Vo - the initial vertex
3. Vt - the final vertex
4. M - total number of vertices in the network

The output parameters are:

1. EDGES - number of edges in the shortest path tree
2. Information about each edge in the shortest path:
   a) FROM(i) - the beginning vertex of edge i
   b) TO(i) - the ending vertex of edge i
   c) LENGTH(i) - the length of edge i
3. LENGTH(EDGES) - Total length of Shortest Path

3.1.2 *Implementation of Dijkstra's Algorithm*

The problem is to determine the shortest distance DIST(Vt) from a specified initial vertex Vo to a final vertex Vt in a positively weighted (positive cost) network, with M vertices represented by an adjacency matrix A. The steps for achieving this are:[2]

1. [Label all vertices]

   Label Vo "determined", and label all other vertices "undetermined"

2. [Initialize variables]

   Set DIST(U)<-A(Vo,U) for all vertices U in matrix A,

   set DIST(Vo)<-0

   and set NEXT<-Vo

3. [Iterate]

   While NEXT not equals Vt, repeat step 4 and 5. Otherwise, STOP.

4. [Update shortest distance to each undetermined vertex]

   For each undetermined vertex U, set

   DIST(U)<-Min(DIST(U),DIST(NEXT)+A(NEXT,U)) (1)

5. [Pick a shortest distance to an undetermined vertex]

   Let U be an undetermined vertex having the smallest distance DIST(U) of all undetermined vertices; label U "determined" and set NEXT<-U.

---

[2] the symbol <- stands for assignment, ie. A <- B means the quantity B is assigned to A.

3.1.3 *Realization of Dijkstra's Algorithm*

This set of instructions for implementation is realized here by programming a subroutine DIJKST. In subroutine DIJKST, the inputs are the adjacency matrix A, the number of vertices M, and the initial and final vertices Vo and Vt. The output of the subroutine consists of a set of 3 vectors --- FROM(i), TO(i) and LENGTH(i) --- which are the edges (FROM(i),TO(i)) of the shortest-path tree selected by the algorithm in finding the shortest distance to the vertex Vt; LENGTH(i) records the shortest distance from Vo to the vertex TO(i). The program in FORTRAN is as follows:

```
      SUBROUTINE DIJKST(M,VO,W,A,FROM,TO,LENGTH,EDGES)
      INTEGER M,A(100,M),VO,W,FROM(M),TO(M),LENGTH(M),EDGES
      INTEGER DIST(100),NEXT,NUMUN,UNDET(100),VERTEX(100)
C
C     INITIALIZE CONDITIONS
C
      EDGES=0
      NEXT=VO
      NUMUN=M-1
C
C     LABEL ALL VERTICES AS "UNDETERMINED"
C     SET ALL VERTICES AS UNCOLORED VERTICES
C
C     INITIALIZE VARIABLE DIST
C
      DO 100 I=1,M
           UNDET(I)=I
           DIST(I)=A(VO,I)
           VERTEX(I)=VO
  100 CONTINUE
      UNDET(VO)=M
      DIST(VO)=DIST(M)
      GOTO 350
C
C     ITERATE M-1 TIMES
C
  200 DO 300 I=1,NUMUN
           J=UNDET(I)
           JK=L+A(NEXT,J)
C
C     UPDATE SHORTEST DISTANCE TO EACH UNDETERMINED VERTEX
C
           IF(DIST(I).LE.JK)GOTO 300
```

```
                VERTEX(I)=NEXT
                DIST(I)=JK
  300 CONTINUE
C
C     PICK A SHORTEST DISTANCE TO AN UNDETERMINED VERTEX
C
  350 K=1
      L=DIST(1)
      DO 400 I=1,NUMUN
           IF(DIST(I).LT.L)THEN
                L=DIST(I)
                K=I
           END IF
  400 CONTINUE
C
C     ADD EDGE TO SHORTEST PATH TREE
C
      EDGES=EDGES+1
      FROM(EDGES)=VERTEX(K)
      TO(EDGES)=UNDET(K)
      LENGTH(EDGES)=L
      NEXT=UNDET(K)
C
C     HAVE WE REACHED W ?
C
      IF(NEXT.EQ.W)GOTO 500
C
C     DELETE NEWLY DETERMINED VERTEX FROM UNDETERMINED LIST
C
      DIST(K)=DIST(NUMUN)
      UNDET(K)=UNDET(NUMUN)
      VERTEX(K)=VERTEX(NUMUN)
C
C     ANY UNDETERMINED VERTICES LEFT
C
      NUMUN=NUMUN-1
      GOTO 200
C
  500 WRITE(6,600)V0,W,LENGTH(EDGES)
  600 FORMAT('-','THE SHORTEST PATH FROM',I3,' TO',I3,' =',I3)
      RETURN
      END
```

The same program in APL (given in Appendix C) consists of 7 lines only

(not including the comments).

3.2 NICHOLSON'S ALGORITHM

The preceding section considered the problem of finding a shortest path between two specific points in a connected network. T.A.J. Nicholson describes a fast method for finding the shortest route between a specified vertex and every other vertex in the graph. One of the algorithms he presented may be modified to obtain the list of the nodes on the shortest path in an efficient manner. As Nicholson put it [Nicholson, 1966, pp.275-280]:

> The algorithm consists of two steps: the first selects a route for extension and extends the selected route, while the second step provides the check to see if the shortest route has been found. The steps are repeated cyclically until the condition in the second step is satisfied. The necessary initialization is as follows:
>
> S(i) = A(s,i)
> T(i) = A(i,t)
> P(i) = s
> Q(i) = t for all i

3.2.1 Architecture of Nicholson's algorithm

The input parameters:

1. s - the starting point
2. t - the terminating point
3. A(i,j) - the adjacency matrix

The output parameters are:

1. P(i) - the point preceding i in the current optimal route from s to i
2. Q(i) - the point preceding i in the current optimal route from i to t

3.2.2 *Implementation of Nicholson's algorithm*

1. [Examine all routes] For each route starting at s of next shortest length, examine all connections from its end point m.
2. [Shorter path found?] If there is a connection from s to the point i, it will be of length S(m) + A(m,i).
3. [Replacement] Replaces the existing value of S(i) when it is shorter. Set P(i) to m.
4. [Complete route?] The least value of S(i)+T(i) is at least as small as the sum of the lengths of the shortest route out of s and the shortest route out of t, then a shortest complete route has been found. Otherwise, repeat step 2 to step 4.
5. [Final route] The shortest route can be traced back through the values of P(i) and Q(i).

The subroutine NETPTH implements the above algorithm, while SHORTP is a modified version which produces a chain of the nodes on the shortest route from Vo to Vt.

3.2.3 *Realization of Nicholson's Algorithm*

Subroutine NETPTH is used to find the shortest path between start node and all other nodes of a network. It yields MINCOS(i), the value of the shortest path from node Vo to all other nodes i, i=1,2,..,M, in a connected M-node network having up to M(M-1) direct links. The array FROM(M) is a chained list of node numbers such that FROM(i) contains the node number preceding node i on the shortest route.

```
      SUBROUTINE NETPTH(A,M,VO,PRECED,MINCOS)
C
C     MINPTH  -  SHORTEST PATH BETWEEN START NODE AND ALL OTHER NODES
C                OF A NETWORK
C
      IMPLICIT INTEGER(A-Z)
```

```
      INTEGER A(100,M),PRECED(M),MINCOS(M),SCAN(100),
     C        BIG M/2147483647/
      DO 1 I=1,M
           SCAN(I)=0
           PRECED(I)=0
           MINCOS(I)=BIG M
    1 CONTINUE
      MINCOS(VO)=0
      SCAN(VO)=1
C
C     ITERATE BEGINS
C
  100 DO 50 I=1,M
           IF(SCAN(I).EQ.0)GOTO 50
                MINI=MINCOS(I)
                DO 40 J=1,M
                     JCOST=A(I,J)+MINI
                     IF(JCOST.LT.MINCOS(J))THEN
                          MINCOS(J)=JCOST
                          SCAN(J)=1
                          PRECED(J)=I
                     END IF
   40           CONTINUE
                SCAN(I)=0
                GOTO 100
   50 CONTINUE
      WRITE(6,60)VO,(MINCOS(I),I=1,M)
   60 FORMAT(1H-,'MINIMUM COST FROM NODE',I3,' TO ALL OTHER NODES:',/,
     C       5(4013,/) )
      RETURN
      END
```

Subroutine SHORTP evaluates in the first n positions of FROM(M) the list of nodes on the shortest path from start node Vo to end node in an M-node connected network. The remaining elements of the route are set to zero. Information necessary for determining the path may be supplied in FROM(i) in the form obtained by the previous use of procedure NETPTH.

```
      SUBROUTINE SHORTP(M,VO,W,FROM,TO,EDGES)
C
C     SHORTP  -  THE LIST OF NODES ON THE SHORTEST PATH
C                FROM START NODE TO END NODE OF A NETWORK
C
      INTEGER FROM(100),TO(100),VO,W,M,EDGES
      J=W
      TO(M)=W
      K=M-1
C
```

```
C      REPEAT UNTIL VO IS NOT FOUND IN FROM(I)
C
  100 I=FROM(J)
      IF (I.NE.VO) THEN
            J=I
            TO(K)=I
            K=K-1
            GOTO 100
      END IF
C
      TO(1)=VO
      J=K+1
      K=2
      DO 50 L=J,M
            TO(K)=TO(L)
            K=K+1
   50 CONTINUE
C
C      EMPTY THE REMAINING LIST
C
      DO 20 J=K,M
   20 TO(J)=0
      EDGES=K-1
      WRITE(6,30) VO,W,(TO(I),I=1,EDGES)
   30 FORMAT(1H-,'SHORTEST PATH FROM NODE',I3,' TO',I3,/,
     C       5(4013,/) )
      RETURN
      END
```

NETPTH and SHORTP are associated procedures which together may be used to find the shortest path between any specified node and all others. The array PRECED is a chained list of node numbers such that PRECED(i) contains the node number preceding node i on the shortest path. This array may subsequently be used by SHORTP to evaluate the list of nodes on the shortest route from Vo to any specified Vt node.

3.3 ALL SHORTEST PATH ALGORITHM

After the shortest path between a start node to all other nodes is found, the next problem of possible interest to a user is to find the shortest paths between all pairs of nodes in the network. Instead of computing shortest paths from an origin to each of the other M-1 nodes, it may be of interest to compute shortest paths from each of the M modes to each of the other M-1 nodes: M(M-1) shortest paths in all.

In this section, the objective is to consider the problem of finding a shortest path between every pair of vertices in the graph. Of course, this second and more general problem could be solved by repeating the Dijkstra shortest path algorithm once for each vertex in the graph taken as the initial vertex Vo. However, this would require a great many computations, and fortunately algorithms exist that are more efficient than repeating the Dijkstra shortest path algorithm once for each vertex in the graph. This algorithm allows arc lengths to be negative (one gains by traversing them rather than having a cost assessed) so long as the network does not have a negative value in total.

Floyd [Lawler, 1976, pp.86-90] has developed a computational method for solving this problem:

3.3.1 *Architecture of Floyd's Method*

Input parameters:

1. M - number of vertices
2. A(i,j) - the adjacency matrix

Output parameter:

A(i,j) - the adjacency matrix (after M times iterations)

3.3.2 *Implementation of Floyd's Method*

1. [Initialization] A(i,i)=0 for all i, for every A(i,j) whose i,j-th element equals the length of the shortest arc from vertex i to vertex j. Let A(i,j) = infinity if no such arc exists. [Iterations] Determine the elements of A(i,j) from the following recursive formula:

$$d_{ij}^{n} = \min\{ d_{in}^{n-1} + d_{nj}^{n-1} , d_{ij}^{n-1} \} \qquad (2)$$

2. [Termination] Repeat previous step M times, n=1,2,..,M. As each element is determined, record the path that it represents.

Upon termination, the i,j-th element of matrix A(i,j) represents the length of a shortest path from vertex i to vertex j.

3.3.3 *Realization of Floyd's Method*

The algorithm is so simple that it can be implemented directly into FORTRAN code. The following is the FLOYD subroutine:

```
      SUBROUTINE FLOYD(A,M)
C
C     FLOYD-WARSHALL METHOD
C
C     A  -  ADJACENCY MATRIX
C     M  -  SIZE OF THE MATRIX A
C
      INTEGER M,A(100,M),S,BIG M/999999999/
      DO 1 I=1,M
      DO 1 J=1,M
C
C     BIG M   REPRESENTS THE NON-EXISTING ARC
C
      IF(A(J,I).GE.BIG M)GOTO 1
           DO 2 K=1,M
           IF(A(I,K).GE.BIG M)GOTO 2
C
C          A(J,K) = MINIMUM(  A(J,K),  A(J,I)+A(I,K)  )
C
           S=A(J,I)+A(I,K)
           IF(S.LT.A(J,K))A(J,K)=S
    2      CONTINUE
    1 CONTINUE
      WRITE(6,5)
      DO 3 I=1,M
    3 WRITE(6,4)(A(I,J),J=1,M)
    4 FORMAT(1X,2013)
    5 FORMAT(1H-,'ALL SHORTEST PATH MATRIX:',//)
      RETURN
      END
```

3.4 *COMPUTATIONAL COMPLEXITY*

All shortest path algorithms consist of essentially two arithmetic operations, addition and minimization. To analyze the computational complexity of one of these algorithms, some way of comparing addition operations with minimization operations is needed. Of course, this comparison varies between computers(human and mechanical), but for

expediency, these two operations may be assumed to have equivalent amounts of computational time.

The Floyd algorithm must compute N (number of vertices) matrices D^1, D^2,.. , N times. Each of these matrices consists of N^2 elements. Hence, a total of N^3 elements must be computed by the Floyd algorithm. Each of these computations requires (by equation (2) in Floyd's algorithm) one addition and one minimization. Hence, the Floyd algorithm requires roughly N^3 additions and N^3 minimizations. The total amount of computation required by the Floyd algorithm is proportional to $2N^3$. Or, in more technical terminology, the Floyd algorithm requires $O(2N^3)$ running time.[3]

Now, consider the computational complexity of the Dijkstra algorithm. At the first iteration, N-1 uncolored vertices must be examined. From equation (1) in Dijkstra's algorithm, this requires N-1 additions, N-1 minimizations, and the selection of the smallest of N-1 numbers, ie. another N-1 minimizations. Thus, $3(N-1)$ operations are required by the first iteration. Similarly, $3(N-2)$ operations are required by the seond iteration. In total, $3N(N-1)/2$ operations are required. Thus the Dijkstra algorithm requires $O(1.5N^2)$ running time.

This suggests that, the Floyd algorithm could be replaced by repeating the Dijkstra algorithm N times; once for each vertex as the initial vertex. This would require $O(1.5N^3)$ running time which is superior to the $O(2N^3)$ running time of the Floyd algorithm. If,

---

[3] The O-notation is used to describe the behavior of an algorithm as a function of parameters. For example, $O(n)$ means the order of magnitude is proportional to n. If an algorithm takes time $O(\log n)$ it is faster, for sufficiently large n, than if it had taken $O(n)$. Similarly, $O(n \log n)$ is better than $O(n^2)$ but not as good as $O(n)$.

however, some arc lengths are negative, then the Floyd algorithm must be used in place of the Dijkstra algorithm.

Therefore, subroutine DIJKST will be used to find the cost of the shortest path between starting node Vo and ending node Vt; and subroutine FLOYD is invoked for the computation of the all shortest path algorithm. However, the monitor of FORUM can be designed to also invoke subroutine DIJKST when the adjacency matrix contains no negative values (arcs).

Chapter IV

TRANSPORTATION ALGORITHMS

There is a class of *LP* models that are of particular interest because of their special structures as well as their applications. The transportation model deals with the transportation of a commodity from *m* sources to destinations. The supply available at source i is a(i) units and the demand required at destination j is b(j). If the sum of a(i) is A and the sum of b(j) is B, the formulation requires that A = B, which can always be satisfied by adding a fictitious source or destination to absorb the difference | A - B |. A direct extension of the transportation model is to allow transshipments so that (part of) a shipment may pass transiently through other sources and destinations before reaching its ultimate destination. The new information can be accounted for by formulating a transportation model with m+n sources and m+n destinations.

The transportation problem is also a form of minimum cost flow problem, but, instead of using graph theory to to find the optimal solution, FORUM uses the TP (Transportation Problem) approach to solve the transportation problems.

The transportation algorithm is iterative, and similar in scope as the Simplex algorithm. It changes only one route at a time, similar to the Simplex method. The closer to the optimal solution the initial solution, technically known as the BFS (Basic Feasible Solution) is,

the fewer iterations the algorithm has to do. An algorithm will save time if it does less work in finding a better initial solution than it would in doing an iteration. There are many initialization procedures, including the three "classical" transportation algorithms, North-West Corner Method, Cheapest Route Method and Vogel's Approximation Method (VAM), and six methods based on Bector's Weighted Penalty Methods and two versions of the VAM's Weighted Penalty Method. These methods will be built into FORUM.

4.1 CLASSICAL METHODS

4.1.1 Northwest Corner Method

The northwest corner rule simply states that the first allocation is arbitrarily made to the northwest (upper left) corner cell of the cost matrix. Compare the supply to the demand of the column, let C(i,j) equal the lesser of the two quantities. If the supply is exhausted without satisfying the demand, move one cell down the column and repeat the allocation process. On the other had, if the column demand is satisfied, move one cell to the right in that row and again repeat the allocation process. This allocation process produces a stairstep pattern of occupied cells versus empty cells.

4.1.2 Cheapest Route Method

Allocation is made first to the lowest cost cell, then move to the next lowest cost cell. In view of the remaining capacity and requirements of its row and column, make an allocation. This process is continued until all demands are satisfied.

4.1.3 *Vogel's Approximation Method*

VAM is a procedure that enables a fairly good initial solution to be determined. This VAM technique aims to find the initial basic feasible solution as close to the optimal as possible.

1. [Compute penalties] Compute for each row and each column the difference between the lowest and next lowest cost cell in the row or column.
2. [Selection] Among those rows and columns at hand, select the one with maximum difference. Allocate as much as possible to that cell.
3. [Decrement] Decrease the corresponding supply and demand. Drop the row or column whose supply or demand is zero.
4. [Termination] If no rows or columns remain, stop. Otherwise, return to step 1 with the reduced problem.

This method is inefficient for very small problems because the initial amount of computation required is large. However, in larger problems computation saving may be realized.

4.2 *WEIGHTED PENALTY METHOD*

The method developed by Bector [Bector, 1982] recognizes that the number of items being shipped also affects the assigment order. It is more important to send a larger shipment by a cheap route than it is to send a small shipment. The method considers only the destinations (the columns). For each column, calculate the difference between the cheapest and the most expensive route. Then multiply each range by the number of units required at that destination to get the Bector number

B. Assign first to the cell with the lowest cost in the column with the largest B, then the second largest B and so on.

The Bector number B is not restricted to the Range only. The Mean or the Coefficient of Variation can be used to achieve the goal. The Weighted Penalty Method (WPM) can be varied in 6 different ways and they are numbered as the following in FORUM accordingly,

WPM 1 - Range of columns

WPM 2 - Range of rows

WPM 3 - Mean of columns

WPM 4 - Mean of rows

WPM 5 - Coefficient of Variation of columns

WPM 6 - Coefficient of Variation of rows

Different simulations [Lai, 1983] on these 11 transportation algorithms were tested in fall, 1983. Including a comparison of the algorithms' efficiencies on CPU time, comparisons, data movement, storage basic feasible solution and number of iterations. The results of the simulations indicated Bector's Weighted Penalty Methods have a better performance on general cases, that is, without restricted routes and no degeneracy.

4.3 VAM'S WEIGHTED PENALTY METHOD

The VAM's Weighted Penalty Method developed by Bector is a modification of the VAM and the Weighted Penalty Methods, or more precisely, a combination of VAM and the Weighted Penalty Methods. Instead of taking the range or mean or coefficient of variation of the rows/columns as in Weighted Penalty Method, it takes the difference

between the lowest cost and the second lowest cost from the rows/colums (VAM), then multiplies the difference by the demand (WPM) requirements to get the weighted penalty score. From these scores of the rows or columns, priority is given to the score with highest weight.

The subroutines VAM1 and VAM2 used in FORUM stands for:

VAM1 - VAM's Weighted Penalty Method on columns

VAM2 - VAM's Weighted Penalty Method on rows

4.4 TEST OF OPTIMALITY

After obtaining an initial solution, the solution is tested. If it is not optimal, a new allocation is made and again tested for optimality. This process continues until an optimal solution is determined. Two commonly used methods exist for performing these iterations: (1) the stepping stone method, and (2) the modified distribution method (MODI). FORUM uses MODI to test of optimality of the cost matrix. It consists of the following steps:

1. [Initialization] Any of the methods above can be used to obtain the initial feasible solution.
2. [Evaluate empty cells] Determining the net cost effect of shifting 1 unit from one of the occupied cells to that empty cell, and then shifting units between other occupied cells so that the rim requirements are not violated.
3. [Improvement] If it is not optimal, the cell with the "highest absolute cell evaluator" of the wrong sign is made an occupied cell, and one or more of the currently occupied cells become "empty" cells.

4. [Optimal ?] Repeat steps 2 and 3 until an optimal solution is obtained.

If the new matrix has a total of less than m+n-1 occupied cells, the new solution is a degenerate solution. Application of the MODI method will (1) verify that an optimal solution has been reached or (2) enable the VAM initial solution to be improved through successive iterations until optimality has been achieved.

4.5 CHARACTERISTICS OF TRANSPORTATION ALGORITHMS

In some practical cases, one or more of the theoretical routes can have additional constraints (including their prohibition from use). Such cases arise especially where the transportation model is used for solving production scheduling problems. Prohibition constraints on source-destination combinations can be accomplished by assigning a very large cost M, to a given route.

WPMs are mostly affected if any prohibition routes are found in the cost matrix; for instance, the Bector's number B would be distorted by M. In the cases like this, VAM and Cheapest Route Method both show forth their advantage by ignoring those huge numbers and just looking at the lowest and second lowest cost values. FORUM will have designed this rule into the monitor to aid in using algorithms for the input cost matrices.

Chapter V

ASSIGNMENT ALGORITHMS

A special case of the transportation model is the assignment problem,[4] which classically seeks the assignment of m people to n jobs with each person exhibiting varying degrees of competence on the various jobs. The transportation model specializes to the assignment model by letting a(i) = b(j) = 1 for all i and j. The cost of assigning resource i to task j represents c(i,j). The assignment model requires optimization of an objective function with one-to-one matching. In other words, the assignment in such a way is restricted to assign one source to one destination exclusively. As a matter of fact, all assignment problems are problems of "matching" two factors. The assignment problem is of particular importance because it can be used directly in the development of solution algorithms for the well-known *Traveling Salesman problem.*

Two algorithms will be presented in this chapter: the Hungarian Method and the Eastman Algorithm. The regular simplex method can be used to solve the problem, but this is an inefficient method for solving large assignment problems. Since the assignment problem is a special case of the transportation problem it can be handled by transportation methods.

---

[4] The transportation model can also be viewed as a special case of the assignment problem

Loomba [Loomba, 1974, p.218] also mentions the "Near-Optimal" Methods by Votaw and Orden who have developed rapid computational methods for solving the assignment problem. These methods are not optimal, but they give a near-optimal solution. Duality theory, which is the basis of the Hungarian method, has also been employed in obtaining near-optimal solutions to the assignment problem. Several other methods can be applied to the solution of the assignment problem; for example, the branch-and-bound technique has been applied successfully to the problem.

5.1 CHARACTERISTICS OF ASSIGNMENT MODEL

First, in the assignment model m=n; that is, the assignment matrix is a square matrix. If, in the original problem, m≠n, it can be changed to m=n by adding dummy origins and destinations. Second, since the assignment model requires a one-to-one assignment, the number of solution variables in a nxn problem is exactly n. Third, the total number of possible solutions for an nxn assignment problem is always n! This means that even for a small problem if, say 10 origins and 10 destinations, the number of alternate solutions is 10!=3,628,800.

The objective is to determine an optimal assignment. An optimal assignment typically minimizes cost or maximizes some desirable objective (profit, welfare, satisfaction, productivity). Although an optimal assignment can be determined by (1) enumerating the various combinations of assignment possible, (2) computing the cost of each alternative assignment, and (3) selecting the lowest-cost assignment, the amount of computational effort required is prohibitive.

5.2 HUNGARIAN METHOD

The Hungarian mehtod is credited to the Hungarian mathematician D. Konig who proved an essential theorem for the development of the method. Basically, the method successively modifies the rows and columns of the cost matrix until there is at least one zero component in each row and column such that a complete assignment corresponding to these zeros can be made. This complete assignment will be an optimal assignment and the resulting total cost will be a minimum. The method will always converge to an optimal assignment in a finite number of steps. The basis of the method is the fact that a constant can be added to or subtracted from any row or column without changing the set of optimal assignments.

5.3 ALGORITHM - HUNGAR

The following is a brief algorithm that uses the Hungarian method to solve the general m-resource, m-activity assignment problem for the minimum total cost.

1. [Initialization] For each row of the cost matrix, subtract the minimum element in the row from each element in the row. For each column of the resulting matrix, subtract the minimum element in the column form each element in the column. The result is a reduced matrix.
2. [Reduction] Draw the minimum number of lines through the rows and columns to cover all zeros in the reduced matrix.
3. [Optimal ?] If the minimum number of lines is m, then an optimal solution is available. Stop. Otherwise, go to step 4.

4. [Select again] Select the minimum uncovered element. Subtract this element from each uncovered element and add it to each twice-covered element. Return to step 2.

If the feasible solution from among the zero cells of the matrix is not found, then the process of drawing lines and adjusting the matrix has to be repeated. There is only a finite number of iterations before an optimal solution can be obtained. Clearly, an optimal soluton can be found if all reduced costs become zero.

Since the entries in the reduced cost matrix are always nonnegative by construction, and since the sum of the entries is reduced by a positive integer at each iteration, the algorithm stops in a finite number of steps. At termination we have an optimal solution since the Kuhn-Tucker conditions hold.

Program HUNGAR solves the m-resource, m-activity assignment problem for the assignment that will optimize the total cost. The elements in the cost matrix must be integers. If the original matrix is not square (either more resources than activities or vice versa), users must supply the necessary rows or columns or zeros before using the program. The source coding of subroutine HUNGAR can be found in the appendix.

5.4 THE OPTIMALITY TEST

Since all entries in our cost matrix are non-negative, the minimum value of the objective function cannot be negative, no matter what alternative assignments are made. Therefore, if an assignment is selected such that its total cost equals zero, this will be the lowest

possible cost (an optimal assignment). This can be done only if all the opportunity costs are equal to zero. Thus, the obvious optimality test is to see if x(i,j)=1 in the cells with zero values in the cost matrix. Of course, the one-to-one requirement must also be satisfied. Otherwise, the cost matrix must be revised and searched for the optimal solution.

5.5 *TRAVELING SALESMAN PROBLEM*

The purpose of this section is to explore a branch and bound algorithm that minimize the total distance traveled by a salesman located in a given city, and he wishes to visit each of N-1 cities once and only once and then return to the city he started from. Although a great deal of research effort has been put forth in the past 10 years to solve the general traveling salesman problem, algorithms that find the exact optimal solution are still limited to fewer than 100 cities.

Wagner [Wagner, 1970, pp.300-309] exhibited three different versions of the branch and bound approach towards the traveling salesman problem, namely, (1) Method of Excluded Subtours, (2) Method of Designated Routes, and (3) Method of Partial Tours. The method of Excluded Subtours begins with the upper bound on the optimal value of the objective function, then succeeds in limiting the number of branches to be explored by calculating an effective lower bound for any tour emanating from each problem. The Method of Designated Routes pays the price for this simplification by having to explore more branches in the associated tree for this method. The third method ---

the Method of Partial Tours --- explores the branches dependent on the selection of problems from the master list, but, redundant calculations can occur when there are several partial tours on the master list composed of the same m cities, for example, city 1 to city 2 to city 3, and city 1 to city 3 to city 2. In this event, the bounding computations are duplicated.

Most of the branch and bound algorithms that find the exact solution for small-to-moderate-size traveling salesman problems are based on the algorithm by Eastman's algorithm [Eastman, 1958]. However, Little et al [Little, 1963, pp.979-989] modified both branching and bounding procedures used by Eastman to eliminate two-city subtours. Since the Eastman and Little algorithms form the basis for all existing traveling salesman branch-and-bound algorithms, one of them, namely, Eastman's algorithm, will be presented in this chapter.

5.6 ALGORITHM - EASTMN

Let D(i,j) be the distance from city i to city j and N be the number of cities. A tour is a complete route or cycle through N cities where no city is visited more than once. If the salesman visits a certain city more than once, the cities involved form a subtour. Eastman's algorithm solves the easier assignment problem that allows subtours and then systematically forbids subtours until finally a tour is obtained that is optimal.

1. [Inital solution] Solve the associated assignment problem, where the distances D(i,j) are elements of the cost matrix. The

solution provides a lower bound on the optimal solution of the traveling salesman problem.

2. [Optimal ?] If at least one subtour exists in the solution, go to step 3, otherwise the optimal solution of the assignment problem is also an optimal solution of the traveling salesman problem. Stop.
3. [Subtour exist ?] Select a subtour, and let k be the number of arcs in the selected subtour. EASTM selects the subtour with the smallest k.
4. [Subtour is a Cycle ?] Branch into k sublevels. If the subtour is a cycle, set sublevel k's D(k,1) = M where M is BIG M (huge number).
5. [New lower bound] Solve the k new assignment problems. Each solution distance is a lower bound for the corresponding subproblem.
6. [Selection] From the set of all unexplored nonfeasible sublevels, select the sublevels with the smallest lower bound for further branching. Go to step 2.

5.7 BRANCH-AND-BOUND TECHNIQUE

The branch-and-bound technique involves a well-structured systematic search of the space of all feasible solutions of constrained optimization problems. Usually the space of all feasible solutions is repeatedly partitioned into smaller and smaller subsets (branching) and a lower bound (for a minimization problem) is calculated for the cost of solutions with each subset (bounding).

Thus, large subsets of solutions may be excluded from consideration without examining each solution in these subsets.

At any point in a branch-and-bound algorithm where a branching decision must be made, any subset with a bound that is less than the least upper bound for all feasible solutions discovered is a possible candidate for branching (minimization problem). There are basically two branching decision rules, namely,

Branch from Lowest Bound

In general, this policy has the advantage that it examines fewer sublevels than other rule, but has the disadvantage of requiring more computer storage because more intermediate data must be stored.

Branch from Newest Active Bound

This rule chooses from the subsets most recently generated the subset with the lowest bound to branch from next. This rule generally requires many more branching operations than the other decision rule but requires much less storage.

5.8 OTHER METHODS

A number of methods have been devised for the solution of the traveling salesman problem, the most effective of which is the method of Little et al. It is possible to formulate the traveling salesman problem as an integer programming problem, but in most cases the resulting problems are too large to be solved. The problem also may be attacked by means of dynamic programming, but the resulting method is computationally less efficient than that of Little et al. Little et al have programmed their algorithm for the IBM 7090 and they cite computational experience (pp.986):

> Problems up to 20 cities usually require only a few seconds. The time grows exponentially, however, and by 40 cities is beginning to be appreciable, averaging a little over 8 minutes. As a rule of thumb, adding 10 cities to the problem multiplies the time by a factor of 10.

This exponential growth of the computational task as the number of cities increases is characteristic of every algorithm for the traveling salesman problem that has been offered to date. In fact, Little's algorithm retards this growth more effectively than Eastman's algorithm, and is therefore capable of solving substantially larger problems.

Another interesting procedure for the traveling salesman problem is to always to move to the next closest city which has not yet been visited. The *Closest Unvisited City* algorithm for the traveling salesman problem is: given a present location corresponding to a specification of a row in the matrix, one chooses for the next city the one associated with the minimum element in that row. Not only is this the obvious commonsense heuristic for the problem, but its analogy to the shortest-processing-time sequencing commends it.

At this point, FORUM has not provided any facilities for this algorithm but it should not be too complicated to implement. The computation of this algorithm is only increased by a factor of n and is still modest in comparison with other procedures that guarantee an optimum solution. In particular, the computation increases only as n^2 for this procedure, which means that it can be applied to very large problems. Only if there were frequent ties in the selection of the closest city, and only if one were to try to enumerate all possible closest city solutions, would the computation become burdensome.

Chapter VI

SEQUENCING

Sequencing problems are concerned with the order and timing which items (jobs) in the process of being serviced or constructed are acted upon by units of men and equipment (stations). The order (sequence) sought is one which will optimize some variable factor involving time. The objective is to minimize total elapsed time, minimize the total time required to complete all jobs; or alternatively, the same criterion may be used to minimize or totally eliminate the amount of time that any job must wait idly between stations. Other objectives might be: minimizing total tardiness, minimizing maximum tardiness, minimizing in-process inventory costs, and minimizing the cost of being late.

A job shop scheduling problem is concerned with scheduling jobs in a manufacturing plant. In the job shop scheduling problem, the order in which each job will be processed by each machine in the shop must be determined. Usually, the processing time which each job must spend on each machine is known in advance. Also, precedence relations, which specify the order in which each job must be worked on by the various machines, are specified.

Different types of job shop scheduling models will be presented in this chapter.

6.1 ONE-MACHINE / N-JOB

When a list of jobs is waiting on one machine, the simplest way of scheduling them is to employ the FIFO (First In First Out) schedule. But, by referring to Johnson and Montgomery [Johnson L.A., 1974], the schedule that results in minimum average time in the shop is given as:

Process the jobs according to their processing time in ascending order

That is, the job to be processed first is the one with the smallest processing time. A proof has shown the schedule described above is optimal. There is a considerable improvement in the average waiting time in this schedule than in FIFO schedule.

6.2 TWO-MACHINE / N-JOB

This is a simple type of sequencing problem in which a given number of jobs must pass through two stations and the total time is minimized when all jobs go through both stations in the same order. One very simple method of solving such problems is by means of a Gantt chart, which can be used to determine the total elapsed time between the time the first job enters the first station and the time the last job leaves the second station. It also determines the total amount of idle time for each station. By comparing the Gantt charts for every possible sequence, it is possible to solve the sequencing problem for five-job, two-station case.

This method of solving the problem is tedious, however, since it will involve comparing n! possible sequences when there are n jobs to be serviced by two stations. Fortunately, this is not the only method available. An alternative four-step method has been developed by Johnson [Johnson S.M., 1954, pp.61-68]

6.2.1 *Algorithm - NJOB2M*

The procedure consists of the following four steps:

1. [Find minimum] Find the minimum service time required for any job in either station.
2. [Ordering] If this minimum service time is associated with a job in station 1, this job is placed first in the sequence. However, if this minimum service time involves a job in station 2, this job is placed last in the sequence.
3. [Unique minimum?] If there is more than one minimum service time for either station,
   a) [Station 1] If the minimum service times occur at station 1, discriminate in favor of the job requiring the least service time in station 2.
   b) [Station 2] If the minimum service times occur at station 2, select the job requiring the least service time at station 1.
4. [List empty?] Repeat steps 1 to 3 until the entire sequence of all jobs has been determined.

This method requires considerably less effort to determine the optimal solution. The advantages include:

1. It does not require identifying and comparing n! sequences.
2. It is a simple, systematic procedure.
3. An optimal solution always results.

Johnson gives a proof that this algorithm will produce an optimal sequence. This method is probably one of the most referenced papers

in production scheduling. It makes the usual assumptions about inventory being present when needed, no passing of jobs, set-up and tear-down time included, no breakdowns, and no strikes.

6.3 THREE-MACHINE / N-JOB

An optimal solution to this problem may be derived by analytical means if either or both of the following conditions hold. The solution is straightforward (see the reference in the previous section)

1. The minimum time required by any job at the first station must be at least as great as the maximum time required by any job at the second station.
2. The minimum time required by any job at the last station must be at least as reat as the maximum time required by any job at the second station.

An additional requirement for the solution of this type of problem is the assumption that all jobs must pass through each station in the same order. The remaining procedure is now identical to the 2-Machine, N-Job case.

6.3.1 Algorithm - NJOB3M

If either or both of the above conditions hold, then a procedure that can be used to solve the N-job, three-machine sequencing problem is:

1. [Sum M1, M2] Sum the processing times on machines 1 and 2 for each job.

2. [Sum M2, M3] Sum the processing times on machines 2 and 3 for each job.
3. [Call NJOB2M] Use the sums calculated in steps 1 and 2 for each job as the processing times on the two machies i a N-job, two-machine sequencing problem and solve this new problem using NJOB2M.
4. [Optimal solution] The optimal sequence for the two-machine sequencing problem in step 3 is also optimal for the original three-machine sequencing problem.

This method, also due to Johnson, is the natural extension of the above problem to the three-machine case. In this extension, Johnson shows that if the minimum processing time over all jobs on the first process is greater than or equal to the maximum processing time on the second process, then an equivalent two-machine job shop problem can be constructed.

Further, another equivalent two-machine problem can be constructed if the minimum processing time on the third process is greater than or equal to the maximum element in the second process. In short, if the second process is dominated by either the first or the last, the problem can be solved as an equivalent two-machine problem.

6.4 M-MACHINE / N-JOB

Scheduling problems involving only two or three machines are very simple in nature, but it is quite common to have m machines and n jobs in the real life job shop problem. The general case is an integer programming problem which can be solved by repeated application of the simplex method. However, very large problems (N or M≥25) may be too difficult to solve optimally. In this case, heuristics are necessary.

Heuristic algorithms can obtain solutions to large problems with limited computational effort, and their computational requirements are predictable for problems of a given size. The drawback of heuristic approaches is that they do not guarantee optimality.

Baker [Baker, 1974] mentioned 3 heuristic methods in his book and described them as quick, suboptimal solution techniques for the makespan problem. In the formulation of the problem, let

$$t(i,j) \qquad \text{where} \qquad i = 1\ (1)\ N$$
$$j = 1\ (1)\ M$$

represent the time for processing the ith job on the jth machine in an N-job, M-machine makespan sequencing problem.

6.4.1 *Palmer's Method*

There are many ways to assign priority to jobs having the strongest tendency to progress from short times to long times in the sequence of operations. Palmer [Palmer, 1965] proposed the calculation of a slope index, S(j), for each job.

$$S(j) = (m-1)\,t[j,m] + (m-3)\,t[j,m-1] + .. + (m-3)\,t[j,2] + (m-1)\,t[j,1]$$

Then a permutation schedule is constructed using the job ordering:

$$S(1) \geq S(2) \geq \ldots \geq S(n)$$

This method is slightly different than Johnson's rule and it will not guarantee an optimal solution.

6.4.2 *Gupta's Method*

Gupta [Gupta, 1971] sought a transitive job ordering in the previous form that would produce good schedules. He noted that when Johnson's rule is optimal in the form:

Two-Machine case

$$S(j) = \mathbb{S} \,/\, \min\{\, t[i,1], t[i,2] \,\}$$

$$\text{where } \mathbb{S} = \begin{cases} 1 & \text{if } t[i,1] \geq t[i,2] \\ -1 & \text{otherwise} \end{cases}$$

Three-Machine case

$$S(j) = \mathbb{S} \,/\, \min\{\, t[i,1]+t[i,2] \;,\; t[i,2]+t[i,3] \,\}$$

$$\text{where } \mathbb{S} = \begin{cases} 1 & \text{if } t[i,1] \geq t[i,3] \\ -1 & \text{otherwise} \end{cases}$$

Generalizing from this structure, Gupta proposed that for $m > 2$,

$$S(j) = \mathbb{S} \,/\, \min\{\, t[i,k] + t[i,k+1] \,\} \qquad k = 1 \;(1)\; m-1$$

$$\text{where } \mathbb{S} = \begin{cases} 1 & \text{if } t[i,1] \geq t[i,m] \\ -1 & \text{otherwise} \end{cases}$$

Thereafter the jobs are sequenced according to Palmer's order. Gupta compared this heuristic to Palmer's extensively and found it to generate better schedules than Palmer's in a substantial majority of cases (problems with known optimal solutions, problems with unknown optimal solutions, CPU time). The simplicity and ease of computation of Gupta's algorithm makes it possible to solve reasonably large-sized problems manually without the aid of any automatic computing devices.

6.4.3 *Campbell's Method*

Baker presents the method of CDS [Campbell, 1970] as the most significant heuristic method for makespan problems. Its strength lies in two properties:

1. It uses Johnson's rule in a heuristic fashion, and
2. it generally creates several schedules from which a "best" schedule can be chosen.

The CDS algorithm corresponds to a multistage use of Johnson's rule applied to a new problem, at stage 1:

The processing time for the i-th job on the first machine = Sum of t[i,j] where j = 1 (1) k

at stage 2:

The processing time for the i-th job on the second machine = Sum of t[i,j] where j = m+1-k (1) m

For each stage i, the job order obtained is used to calculate a makespan for the original problem. For each auxiliary problem, Johnson's N-job, Two-machine method is used to determine the optimal sequences, thus generating S(1), S(2), .. ,S(i) for i≤M-1 auxiliary problems. The best sequence among these is chosen on the basis of minimum total processing time.

Campbell tested his algorithm extensively and examined its performance as compared to Palmer's heuristic in several problems. He

found that the CDS algorithm was generally more effective, for both small and large problems.

6.4.4 *Algorithm - CAMPBL*

The algorithm presented here provides for the generation of up to m-1 sequences.

1. [Initialization] Establish an n x m matrix of processing time, where t[i,j] is the processing time of the jth job on the ith machine.
2. [Auxiliary problem] Establish a number of auxiliary n-job, 2-machine problems, where, $p \leq m-1$.
3. [Stage 1] Establish a column vector of processing times on Machine 1 (M1) for each job for kth auxiliary problem.
4. [Stage 2] Establish a column vector of processing times on Machine 2 (M2) for each job for kth auxiliary problem.
5. [Call NJOB2M] Apply Johnson's n-job, 2-machine algorithm to Stage 1 & 2.
6. [Loop] If the number of iterations is less than p, repeat step 3 to step 5.
7. [Time vector] Using the original n x m matrix of processing times, compute total processing time for each of the p sequence generated.
8. [Selection] Select the minimum total processing time sequence as the best sequence.

6.4.5 *Economic Consideration*

At the end of Campbell's paper, he predicted the times for the larger problems which were not solved by hand:

20-job, 20-machine -- 130 minutes
30-job, 20-machine -- 130 minutes
60-job, 30-machine -- 380 minutes
60-job, 60-machine -- 740 minutes

If the problems are of a size permitting a choice between approximate and exact procedures, several factors must be considered. Approximate methods will give results quickly and with less computation costs than with an exact algorithm. Conversely, the approximate algorithm results will not be as accurate as results from an exact algorithm.

> The choice between exact sequencing procedures and approximate procedures exists only for limited problem sizes. Therefore, until improved exact procedures are developed, approximate methods must be used [Campbell, 1970, p.636].

The two factors, computational costs and costs of nonoptimal solutions, appear to be of primary importance when choosing between approximate and exact sequencing procedures. Since each situation will be unique, an economic analysis of the costs involved should be made before a procedure is selected.

Chapter VII

PERT / CPM

An OR tool used on large-scale projects to aid management in expediting and controlling the utilization of personnel, materials, facilities, and time is the Program Evaluation and Review Technique *(PERT)*. This technique is used to pinpoint critical areas in a project so necessary adjustments can be made in order to meet the scheduled completion date of the project.

Closely related to PERT, but developed independently, is the technique known as the Critical Path Method, or *CPM*. It is basically concerned with obtaining the trade-off between applying more men or resources to shorten the duration of a project, and obtain time and cost estimates of the various phases of the project.

CPM is not concerned with uncertain job times as is PERT; rather it deals with time-cost trade-offs. Because of these differences, PERT is used more in research and development projects while CPM is used in projects where there has been some experience in handling similar endeavors.

A difference between PERT and CPM may regard as: PERT is event-oriented; while CPM is often know as activity-oriented. Therefore, unlike PERT, CPM does not make use of probabilistic job times; it is a "deterministic" rather than a "probabilistic" model.

7.1 PERT MODEL

PERT takes uncertainties into specific account. It assumes that the activities and their network relationships have been well defined, but allows for uncertainties in activity times. Three time estimates are normally used for each activity. They are:

1. a = optimistic time, the expected time if everything goes better than expected without delays
2. m = most likely time, the most realistic time available
3. b = pessimistic time, the expected time if just about everything goes wrong

7.1.1 Expected Times for Activities

PERT calculates the expected value of an activity duration as a weighted average of the three time estimates. Specifically, it makes the assumption that the optimistic and pessimistic activity times, a and b, are about equally likely to occur. It also assumes that the most probable activity time m, is four times more likely to occur than either of the other two. Therefore,

$$t = (a + 4m + b) / 6$$

likewise, the variance of the completion time of each activity is estimated to be:

$$Vt = (b - a)^2 / 36$$

The weights are based on an approximation of a statistical frequency known as Beta. The Beta distribution is unimodal (has a single peak), has finite and non-negative end points, and is not necessary symmetrical. All of these are desirable properties for the

distribution of activity times. The normal distribution, more widely used and commonly known, fails to satisfy the last two criteria mentioned. Since most activities in development projects occur just once, frequency distributions of such activity times cannot be developed from past data.

7.1.1.1 Earliest Expected Completion Time of Events

ET(j) represents the earliest time the event can be accomplished if each activity j on every path leading to the given event is completed.

ET(j) = earliest expected completion time of event j

= max{ ET(i) + t(i,j) } where i ranges over all activities from START to event j

7.1.1.2 Latest Allowable Event Completion Time

LT(j) is the latest allowable time an event can occur without delaying the scheduled completion date of the project if all succeeding events are completed as anticipated.

LT(j) = latest allowable event completion time for event j

= min{ LT(i) - t(i,j) } where i ranges over all activites from event j to END

7.1.1.3 Slack Times

The slack time SL(j) for each event j is the amount of time the event can be delayed without affecting the scheduled completion time for the project. It is given by:

SL(j) = LT(j) - ET(j)

7.1.2 *Critical Path*

The longest path through a PERT network is referred to as the critical path. Each event on the critical path will have a slack time equal to the slack time for the final event. Thus, if the event is expected to be completed exactly on schedule, each event on the critical path will have a zero slack time. It is possible to have more than one critical path. Any delay of activities on the critical path(s) will cause a corresponding delay in the completion of the project.

One of the main objectives of a PERT analysis is to pinpoint the critical path so resources from outside the system or activities on noncritical paths can be supplied to those on the critical path if needed. Wiest and Levy [Wiest, 1969, p.47] mentioned the effects of a Near-Critical Path in a PERT network:

> The expected length of the project T, obtained from adding the expected times of the critical path activities, might not always be the best estimate of project length. It is possible that, under some combination of activity times and variances, a near-critical path may exist with a higher variance than the "official" critical path. It is quite possible for the former to exceed the latter in length--that is, to become critical itself.

Thus the normal PERT procedure which bases the estimates of $t(i,j)$ and V_t on a single critical path can possibly overstate the probabilities of completing a project by a given date, especially if there are one or more parallel paths through the network which are nearly critical, and/or which have relatively large variances.

7.1.3 *Probability of Completing Events on Schedule*

A scheduled completion time is almost always given for the final event and it can be used to determine the probability of completing events on schedule. The only assumption necessary is that the completion time of each event has a normal distribution with mean ET(j) and $\mathbb{S}^2$, where $\mathbb{S}^2$ is the sum of variances of those activity times on the path that is used to calculate ET(j).

For early events the normal distribution assumption may not be too valid; but it is quite relevant for the late events, since the late events are the sums of several activity completion times, each having a certain mean and variance. By the central limit theorem, the completion time for the event approaches the normal distribution.

Suppose D(j) is the scheduled due date for event j, then

$$Z = (D(j) - ET(j)) / \mathbb{S}$$

The probability is given by:

$$P[\ ET(j) \le D(j)\ ] = \int_{-\infty}^{Z} N(0,1)\ dt$$

After calculating the probability of completing each event on schedule, events with low probabilities are noted and measures are taken to speed up those activities leading to the events in trouble.

7.2 *PROCEDURE - PERT*

Unlike the previous chapters, the PERT model is quite standard to some extent. Thus, instead of an algorithm form, the steps of PERT are listed as follows:

1. [t(i,j), Vt(i,j)] Calcalate the estimated mean completion time and estimated variance of completion time.

2. [ET(j)] Calculate the earliest expected completion time for event j
3. [LT(j)] Calculate the latest allowable completion time for event j.
4. [SL(j)] Calculate slack time for event j.
5. [Critical Path] Find the activitiies on critical path.
6. [Probability on schedule] If the schedule due date is given, calculate the critical value of cumulative distribution to determine probability of completing events before the deadline.

7.3 CPM MODEL

Most jobs, CPM argues, can be reduced in duration if extra resources (men, machines, money) are assigned to them. The cost for getting the job done may increase, but if other advantages outweight this added cost, the job should be expedited, or crashed. On the other hand, if there is no reason to shorten a particular job, then the job should be done at its normal or most efficient pace, with a lesser assigment of resources. Thus, only the critical jobs need be crashed and CPM attempts to determine which jobs to crash and by how much.

Project schedules affect two kinds of costs --- direct costs associated with individual activities (which increase if the activities are expedited) and indirect costs associated with the project (which decrease if the project if shortened). The cost profile of the typical activity could be estimated by:

1. Crash Cost (Cc) - the minimum direct cost that is anticipated in completing an activity within the crash time.

2. Crash Time (Tc) - the minimum time required to complete an activity.
3. Normal Cost (Cn) - the lowest possible direct cost required to complete an activity.
4. Normal Time (Tn) - the minimum time required to complete an activity at normal cost.

The cost slope indicates the additional cost gained per unit of time saved in reducing the duration of an activity. It is a measure of the efficiency incurred in crashing an activity in order to reduce the time. The cost slope is typically computed as follows:

$$\text{Activity cost slope} = (Cc - Cn) / (Tn - Tc)$$

This computation yields a constant cost slope that is a reasonable approximation of the actual cost slope. However, if greater accuracy is desired, the direct cost curve can be divided into a piecewise linear line, or, a curved line is even better.

7.3.1 Lowest Cost Schedule

From an intuitive viewpoint, it seems likely there is some optimum porject length --- a medium between one with excessive direct costs for shorteneding jobs and one with excessive indirect costs for lengthening the project. The CPM model verifies this notion mathematically, and it specifies a method for finding the optimum point representing the least cost schedule.

The mathematics of the Kelley's [Kelley, 1961] method are quite complicated, but the general procedure can be easily explained.

First of all, a preliminary schedule is generated in which all jobs are assigned at their early start times and with normal resources. This is the maximum length schedule; it can be reduced by crashing one or more of the jobs at added cost. If this added cost is less than the savings in indirect costs which result from shortening the project, then a less expensive schedule can be found. Improvements are made in stepwise fashion. The goal is to determine which jobs to crash and how far to carry the schedule shortening procedure.

At each step of the process, only jobs along the critical path are considered for crashing. The cost-time slope of each critical job is examined, and the job with the least slope is determined. The process is repeated until no further shortening of critical jobs is possible or until the costs of such shortening would exceed the savings that result from reducing the project length.

7.3.2 Briggs' Method

Kelley developed an algorithm, suitable for machine computation, for solving this problem. He described a linear programming procedure for finding optimal solutions for networks of jobs with linear cost-time trade-off curves. Because of certain special features of the problem, Kelley's algorithm is more efficient than the simplex method. However, the number of variables, even in a small problem, is quite large and the method is rather tedious for hand computation.

Another procedure is Briggs' [Briggs, 1963] method. The algorithm is a logical, systematic one, beginning with the development of the network diagram and terminating when the desired project completion time has been achieved.

7.4 ALGORITHM - BRIGGS

Briggs' Algorithm is a modified version of Kelley's method in Critical Path Scheduling. The original pseudo-codes were in ALGOL form and it was translated into FORTRAN and tested. The modifications suggested by Muth [Muth, 1967] were included. Results were correct in all tested schedules.

1. [Exit Condition] Every task must proceed from a lower-numbered node to a higher-numbered one, if not, exit.
2. [Initialization] Setup the SOURCE and SINK values to form a network-type description of a project.
3. [CPM] Determine the normal-duration critical path.
4. [Reduction] Reduce the durations of the tasks with the flattest cost slope.
5. [Loop] Adjust the critical path and repeat the previous step until minimum durations are reached.

Chapter VIII

CONCLUSION

The objectives of this thesis were three-fold:

1. To build an easy to use, efficient Operations Research applications computer package.
2. To use a highly structured design methodology to design and implement this package.
3. To test a two language approach to writing complex programs to solve OR problems not explicitly dealt with in available packages: path algorithms, transportation problems, assignment problems, sequencing and PERT/CPM.

A discussion of each of these objectives is given below.

8.1 THE FORUM PACKAGE

The FORUM package is now ready for use and the various modules have been tested using worked examples from the following textbooks:

1. Management Science [Lee, 1981]
2. Introduction to Operations Research Models [Cooper, 1977]
3. Operations Research [Siemens, 1973]
4. Introduction to Operations Research [Gillett, 1976]
5. Fundamental of Operations Research [Ackoff, 1968]
6. Operations Research for Immediate Application [Woolsey, 1975]

Each set of programs was subjected from 6 to 10 tests, using examples with widely different input data. The transportation programs were also tested using randomly generated input data [Lai, 1983].

The data input requirements to the programs are very simple. The examples (which were all problems of moderate size) required less than 10 lines of input. As problem size increases, so will the necessary inputting, but it should never reach the point where it is intimidating.

Also, the programs only require data which is intuitively obvious to the user. For example, the m-machine, n-job sequencing problem requires a total of n+1 rows of input, 1 row to specify m and n, and n additional rows, each specifying the time that particular job requires on each machine. The only exception to this is the PERT routine which requires the user to completely specify dummy activities,[5] including optimistic, most likely and pessimistic times of completion (all zero).

The programs were also tested for storage and CPU time. The following table shows the average requirements for some sample subroutines when run on the University of Manitoba's AMDAHL 5850.

---

[5] A basic rule of CPM/PERT modeling --- two or more network activities cannot simultaneously share the same start and finish nodes. In order to assign a unique name to each activity, a dummy activity is used to correct this problem. A dummy activity does not consume time or resources, but it does preserve the required precedence relationship among project activities. Thus, the time duration associated with dummy activity is zero.

Subroutine	Size	CPU (sec.)
MINSCT	0.93K	0.001092
MINWTS	1.32K	0.001456
MINTRD	0.93K	0.001040
MAXTRD	1.23K	0.001716
MINLAT	1.72K	0.001820
CAMPBL	2.63K	0.002080
JONSON	1.06K	0.003744
GUPTA	2.04K	0.002236
HUNGAR	4.18K	0.005356
EASTMN	5.82K	1.277847
NJOB2M	2.79K	0.003458
NJOB3M	3.63K	0.004472
PERT	3.81K	0.010088
DIJKST	2.87K	0.008892
FLOYD	1.08K	0.039624
NETPTH	1.68K	0.003068
SHORTP	0.91K	0.000416

The total core requirement for FORUM is in the order of 600K. The small size is possible due to the dynamic allocation used in the subroutines. The CPU requirements are also not excessive. If the user is operating in a CPU-scarce environment, these could be decreased by compiling each module and storing it in a LOAD library (from which it can be accessed without further compilation). Also, specifying OPT=2 (Optimization level 2) in the compile step will lead to savings in execution time.

8.2 DESIGN AND IMPLEMENTATION OF FORUM

Blaauw's methodology has some advantages in that the designer is forced to specify the inputs, outputs and key functions of each module before beginning the actual programming. If no ready-to-use algorithms were available, it would be an essential process. As matters stand, algorithms written either in pseudo-code or in a format which lends itself easily to immediate programming are becoming more numerous and easy to find. If an algorithm written in psuedo-code obviously serves one's purpose, going through the HOW, WHAT and WHERE process is more a formality than a necessity. In such instances, the most efficient (and least frustrating) method is simply to program the algorithm and make "fine-tuning" changes later.

Similar comments apply to the two-language approach. The problem must be highly complex or the ratio of lines in language A to language B very small before it becomes useful. In other words, at times it can be a highly advantageous tool, but it is not a method to be followed rigorously.

In summary, the key to successful and efficient software design and implementation appears to be a judicious mix of a number of tools and approaches. No one approach or tool can be designated as the way to design and implement software. The circumstances under which each approach is effective are dependent on the problem and the designer's intuitive feel for it. This means that software design, much more so than hardware design, will remain an art rather than a science.

Appendix A

DIGITAL SYSTEM IMPLEMENATION

A.1 ARCHITECTURE OF THE BIT ADDER

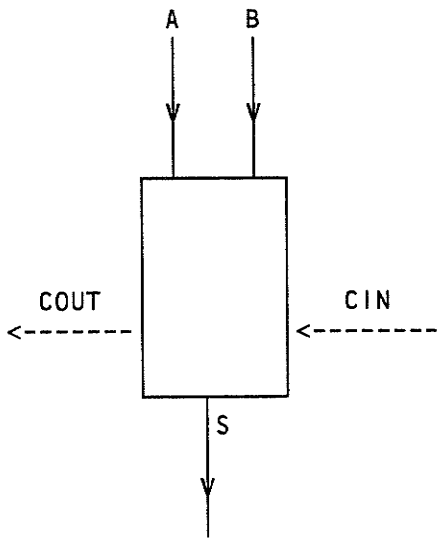


Figure 2: Bit Adder

According to Blaauw:

> "A bit adder can be considered as an adder whose operands and sum are vectors with a single element, hence with dimension 1. Figure 1 shows that besides the operands A and B, the carry-in CIN participates in the addition. The result of the addition is the sum S and the carry-out COUT. The carry-out can be used to determine whether the sum has become too large or too small. Therefore, the adder forms the sum S from the operands A and B and the carry-in CIN, and gives an overflow indication COUT" [Blaauw, 1976, p.24]

The architecture of a bit adder can also be formally specified as the following APL program.

```
      ∇ ARCHBITADD;SUM
[1]   SUM←A+B+CIN
[2]   S←2⊤SUM
[3]   COUT←SUM≥2
      ∇
```

Figure 3: Bit Adder Architecture

A.2 IMPLEMENTATION OF THE BIT ADDER

As shown in the architecture section the bit adder is a circuit with three inputs and two outputs. The next step is to describe the process which changes inputs into output. This is done by identifying and describing the nature of the functional components of the adder as follows:

1. Half Adders For each 'bit adder' the operands A and B belong to the numbers to be added and are available at the start of the operation. In contrast, the carry-in CIN of a bit adder arises as the carry-out of the adjacent bit adder and is normally available later than A and B.

2. Transmission The sum T of bits A and B is again a single bit. Hence, this sum is determined modulo 2 and is equivalent to the exclusive-or of A and B.

   in APL: T <- A≠B

3. Carry Generation The first half-adder also determines if A and B give rise to a carry. This signal is called the generated carry, or G.

   in APL: G <- A∧B

4. Carry Assimilation In the second half-adder the partial sum is added to the carry-in, the result is the ultimate sum S.

   in APL: S <- T≠CIN

5. Carry Propagation The second half-addr also determines if the carry-in, together with the partial sum, gives rise to a carry-out of the bit-adder, the propagated carry P.

   in APL: P <- T↑CIN

6. Carry-out Since T and G cannot be 1 simultaneously, the generated and propogated carry never appear at the same time.

   in APL: COUT <- G∨P

These points can then be summarized as the following APL program.

```
      ∇ IMPBITADD
[1]   ⍝ TRANSMISSION AND GENERATION
[2]    T←A≠B
[3]    G←A∧B
[4]   ⍝ CARRY PROPAGATION
[5]    P←T∧CIN
[6]   ⍝ CARRY ASSIMILATION
[7]    S←T≠CIN
[8]    COUT←G∨P
      ∇
```

Figure 4: Bit Adder Implementation

Every value of A, B and CIN, S is assigned a value by the definition of the previous steps; hence we can write S = m(A,B) where m is a function that can be constructed from A, B, T, G, P and CIN. In our above example, m is the function that, given a value for its argument, gives the OR of the CARRY PROPAGATION and CARRY GENERATION

as an answer. This means that the interconnected system may be represented by a single function block. Systems with more than one output can be constructed and analyzed, simply by putting a number of single-output systems having common inputs side by side(they may share some common blocks).

A.3 REALIZATION OF THE BIT ADDER

Figure 4 shows the expressions in APL for the bit adder IMPBITADD [Blaauw, 1976, p.25], it uses the logical symbol ↑ for 'and' and V for 'or'; for the relation 'unequal' ≠ is used. The following figure shows a circuit for the bit adder, which is constructed from 'and', 'or' and 'not' components using standard Boolean algebra. The two 'exclusive-ORs' are also constructed from these elements, although they are sometimes available as a single switching element.

IMPBITADD has a one-to-one correspondence to the circuit elements of Figure &reall (assuming that the exclusive-OR is available as a unit). When the 'exclusive-OR' is to be constructed from 'and', 'or' and 'inverter elements', lines 2 and 7 (statements [2] and [7] of IMPBITADD) should be changed accordingly. ARCHBITADD and IMPBITADD illustrate how APL can be used both for a high-level specification and for a detailed circuit description.

Once the individual circuit components are identified by an APL function, the circuit can be realized. This realization still involves many decisions, described by Breuer for a hardware design as:

> "...selection of the component type, the location of the component within a module or on a chip, and the routing of the connecting wires" [Breuer, 1972, p.115]

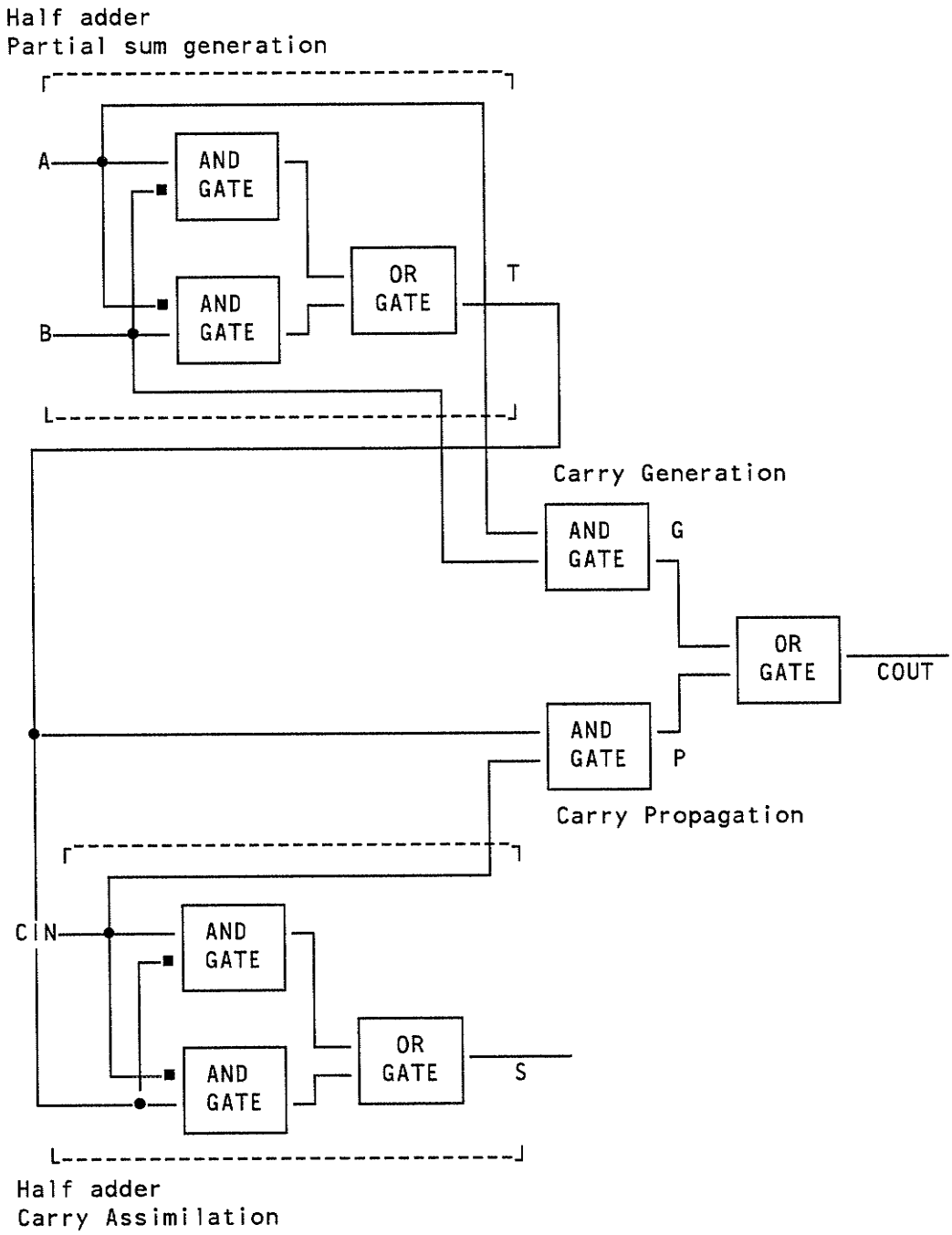


Figure 5: Bit Adder Circuit

These decisions are greatly influenced by electrical and physical considerations for a hardware device. The implementation functions

that will be derived by these considerations constitute the logical specification of the realization design.

While this is conceptually the most simple realization (using only ANDs and ORs), more efficient ones exist that reduce both the number of gates and number of circuits. However, these require the introduction of such ideas as NANDs (NOT ANDs) which are far more difficult to visualize. This is often the case: there is a tradeoff between conceptual clarity and efficiency in the design and implementation of hardware systems.

Appendix B

GLOSSARY

Adjacency matrix	Representing a graph of n vertices by an n x n matrix A, where A(i,j) is the length of an edge connecting vertex v(i) to v(j) and A(i,j)=infinity, if not.
Bit Adder	In binary arithmetic, the digits or bits are either 0 or 1. For each digit position, called bit position, two operand bits and a carry bit should be added. The equipment required for this addition is called a bit adder.
Circuit	A path on a graph whose start and end vertices are the same.
Coefficient of Variation	The ratio of the standard deviation to the mean value.
Crashing	Earlier completion is possible at an increased cost.
Degeneracy	The number of occupied cells is less than m+n-1 in a m x n matrix.
Digital System	The digital technique leads to the use of a multitude of components, a purposeful interconnection of such a multitude of components forms a system. The prime example of a digital system is the computer.
IF-THEN-ELSE	The IF statement provide the conditional facility in FORTRAN-77. IF-THEN-ELSE statement permit blocks of statements to be executed on a conditional basis.
Makespan	In the basic single machine problem, the time to complete all jobs is a constant. The length of time required to complete all jobs is called the makespan and is denoted as M.
Uncolored vertices	Untraversed vertices in a connected graph.

APPENDIX C

APL FUNCTION DIJKSTRA

```
      ∇ MAP;I;J
[1]   ⍝
[2]   ⍝ CONSTRUCT A MAP OF SHORTEST PATHS
[3]   ⍝
[4]    M←DIMENSION A
[5]    ROUTE←(M,M)⍴0
[6]    I←J←1
[7]  LOOP: ROUTE[I;J]←I SHORT∆PATH J
[8]    →(M≥J←J+1)/LOOP
[9]    J←1
[10]   →(M≥I←I+1)/LOOP
       ∇
```

```
      ∇ Z←V0 SHORT∆PATH W;VEC;VI
[1]   ⍝ CHECK IF V0 = W ?
[2]    →(0=LENGTH←V0-W)/END
[3]   ⍝ DIRECT ROUTE EXIST ALREADY ?
[4]    →(0≠LENGTH←[V0;W])/END
[5]   ⍝ INITIALIZATION
[6]    M←DIMENSION A
[7]    ADJ←A+(A=0)×HIGH∆VALUE
[8]    LENGTH←ADJ[V0;]
[9]   ⍝ INVOKE DIJKSTRA'S ALGORITHM
[10]  ⍝
[11]   LENGTH←ADJ[V0;]
[12]   DIJKSTRA
[13]  ⍝
[14]  ⍝ PRINT THE COST OF THE SHORTEST PATH
[15] END: Z←MINIMUM LENGTH
[16]   'SHORTEST PATH FROM ',(⍕V0),' TO ',(⍕W),' = ',⍕Z
       ∇
```

```
          ∇ DIJKSTRA;MIN
 [1]   ⍝
 [2]   ⍝ [ STEP 1 ] RETURN WHEN THE LIST IS EMPTY
 [3]   ⍝
 [4]    →0×⍳W=(MINIMUM LENGTH) IN LENGTH
 [5]   ⍝
 [6]   ⍝ [ STEP 2 ] FIND THE NODE WITH SHORTEST DISTANCE FROM V0
 [7]   ⍝
 [8]    VI←(MIN←MINIMUM LENGTH) IN LENGTH
 [9]   ⍝
[10]   ⍝ [ STEP 3 ] ADDS SHORTEST DISTANCE TO THE EDGES FROM VI
[11]   ⍝
[12]    VEC←MIN+ADJ[VI;]
[13]   ⍝
[14]   ⍝ [ STEP 4 ] COMPARE THIS SUM WITH PREVIOUS DISTANCES
[15]   ⍝             FROM VI, TAKE THE MINIMUM DISTANCE
[16]   ⍝
[17]    LENGTH←LENGTH⌊VEC
[18]   ⍝
[19]   ⍝ [ STEP 5 ] REMOVE EDGE FROM LIST
[20]   ⍝
[21]    LENGTH[MIN IN LENGTH]←HIGH∆VALUE
[22]   ⍝
[23]   ⍝ [ STEP 6 ] REPEAT THE ALGORITHM
[24]   ⍝
[25]    DIJKSTRA
[26]    →0
        ∇


        ∇ Z←DIMENSION A
 [1]    Z←1↑⍴A
        ∇


        ∇ Z←HIGH∆VALUE
 [1]    Z←⌊/⍴0
        ∇


        ∇ Z←A IN B
 [1]    Z←B⍳A
        ∇


        ∇ Z←MINIMUM A
 [1]    Z←⌊/A
        ∇
```

Appendix D

SUBROUTINE BECTR1

```
      SUBROUTINE BECTR1(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)
      IMPLICIT INTEGER(A-Z)
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
      INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),
     C        BECTOR(N),DSCEND(N)
      INTEGER HIGH/2147483647/
      REAL TOTAL
C
C     B E C T O R   M E T H O D     # 1
C
C     CALCULATE THE RANGE R, THE DIFFERENCE BETWEEN THE CHEAPEST &
C     THE MOST EXPENSIVE ROUTE.
C     MULTIPLY EACH R BY THE # OF UNITS REQUIRED AT THAT DESTINATION
C     TO GET B, THE BECTOR'S NUMBER.
C     ASSIGN FIRST IN THE COLUMN WITH THE LARGEST B, THEN SECOND
C     LARGEST B AND SO ON.
C
C
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      FACTRY(I)=SUPPLY(I)
      DO 1 J=1,N
      COSTS(I,J)=COST(I,J)
    1 ALLOC(I,J)=0
      DO 2 J=1,N
    2 WAREHO(J)=DEMAND(J)
C
C     GENERATE THE BECTOR NUMBERS FOR COLUMNS (WAREHOUSES)
C
      DO 3 J=1,N
    3 BECTOR(J)=WAREHO(J)*RANGE(M,N,0,J)
C
C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER
C
      CALL SORT(N,BECTOR,DSCEND)
C
C     FILL UP THE CELL WITH LOWEST COST
C
      DO 6 J=1,N
      COL=DSCEND(J)
C
```

```
    4 LOWEST=COSTS(1,COL)
      ROW=1
      DO 5 I=2,M
      IF(COSTS(I,COL).GE.LOWEST)GOTO 5
      ROW=I
      LOWEST=COSTS(ROW,COL)
    5 CONTINUE
C
      COSTS(ROW,COL)=HIGH
      ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))
      FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)
      WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)
      IF(WAREHO(COL).GT.0)GOTO 4
    6 CONTINUE
C
C
C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD
C
      TOTAL=0.
      DO 30 I=1,M
      DO 30 J=1,N
   30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))
      WRITE(6,31)TOTAL
   31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 1 = ',E16.10)
C
C     OUTPUT THE ALLOCATION MATRIX
C
      DO 28 I=1,M
   28 WRITE(6,29)(ALLOC(I,J),J=1,N)
   29 FORMAT(1X,3314)
      RETURN
      END
```

Appendix E

SUBROUTINE BECTR2

```
      SUBROUTINE BECTR2(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)
      IMPLICIT INTEGER(A-Z)
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
      INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),
     C        BECTOR(M),DSCEND(M)
      INTEGER HIGH/2147483647/
      REAL TOTAL
C
C     B E C T O R   M E T H O D     # 2
C
C     CALCULATE THE RANGE R, THE DIFFERENCE BETWEEN THE CHEAPEST &
C     THE MOST EXPENSIVE ROUTE.
C     MULTIPLY EACH R BY THE # OF UNITS REQUIRED AT THAT SOURCE
C     TO GET B, THE BECTOR'S NUMBER.
C     ASSIGN FIRST IN THE ROW WITH THE LARGEST B, THEN SECOND
C     LARGEST B AND SO ON.
C
C
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      FACTRY(I)=SUPPLY(I)
      DO 1 J=1,N
      COSTS(I,J)=COST(I,J)
    1 ALLOC(I,J)=0
      DO 2 J=1,N
    2 WAREHO(J)=DEMAND(J)
C
C     GENERATE THE BECTOR NUMBERS FOR ROWS (FACTORIES)
C
      DO 3 I=1,M
    3 BECTOR(I)=FACTRY(I)*RANGE(M,N,I,0)
C
C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER
C
      CALL SORT(M,BECTOR,DSCEND)
C
C     FILL UP THE CELL WITH LOWEST COST
C
      DO 6 I=1,M
      ROW=DSCEND(I)
C
```

```
    4 LOWEST=COSTS(ROW,1)
      COL=1
      DO 5 J=2,N
      IF(COSTS(ROW,J).GE.LOWEST)GOTO 5
      COL=J
      LOWEST=COSTS(ROW,COL)
    5 CONTINUE
C
      COSTS(ROW,COL)=HIGH
      ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))
      FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)
      WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)
      IF(FACTRY(ROW).GT.0)GOTO 4
    6 CONTINUE
C
C
C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD
C
      TOTAL=0.
      DO 30 I=1,M
      DO 30 J=1,N
   30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))
      WRITE(6,31)TOTAL
   31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 2 = ',E16.10)
C
C     OUTPUT THE ALLOCATION MATRIX
C
      DO 28 I=1,M
   28 WRITE(6,29)(ALLOC(I,J),J=1,N)
   29 FORMAT(1X,3314)
      RETURN
      END
```

Appendix F

SUBROUTINE BECTR3

```
      SUBROUTINE BECTR3(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)
      IMPLICIT INTEGER(A-Z)
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
      INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),
     C        BECTOR(N),DSCEND(N)
      INTEGER HIGH/2147483647/
      REAL TOTAL,MEAN
C
C     B E C T O R   M E T H O D    # 3
C
C     CALCULATE THE MEAN M, THE AVERAGE AMONG THE ROUTES ON THAT COL.
C     MULTIPLY EACH M BY THE # OF UNITS REQUIRED AT THAT DESTINATION
C     TO GET B, THE BECTOR'S NUMBER.
C     ASSIGN FIRST IN THE COLUMN WITH THE LARGEST B, THEN SECOND
C     LARGEST B AND SO ON.
C
C
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      FACTRY(I)=SUPPLY(I)
      DO 1 J=1,N
      COSTS(I,J)=COST(I,J)
    1 ALLOC(I,J)=0
      DO 2 J=1,N
    2 WAREHO(J)=DEMAND(J)
C
C     GENERATE THE BECTOR NUMBERS FOR COLUMNS (WAREHOUSES)
C
      DO 3 J=1,N
    3 BECTOR(J)=WAREHO(J)*MEAN(M,N,0,J)
C
C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER
C
      CALL SORT(N,BECTOR,DSCEND)
C
C     FILL UP THE CELL WITH LOWEST COST
C
      DO 6 J=1,N
      COL=DSCEND(J)
C
    4 LOWEST=COSTS(1,COL)
```

```
      ROW=1
      DO 5 I=2,M
      IF(COSTS(I,COL).GE.LOWEST)GOTO 5
      ROW=I
      LOWEST=COSTS(ROW,COL)
    5 CONTINUE
C
      COSTS(ROW,COL)=HIGH
      ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))
      FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)
      WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)
      IF(WAREHO(COL).GT.0)GOTO 4
    6 CONTINUE
C
C
C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD
C
      TOTAL=0.
      DO 30 I=1,M
      DO 30 J=1,N
   30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))
      WRITE(6,31)TOTAL
   31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 3 = ',E16.10)
C
C     OUTPUT THE ALLOCATION MATRIX
C
      DO 28 I=1,M
   28 WRITE(6,29)(ALLOC(I,J),J=1,N)
   29 FORMAT(1X,3314)
      RETURN
      END
```

Appendix G

SUBROUTINE BECTR4

```
      SUBROUTINE BECTR4(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)
      IMPLICIT INTEGER(A-Z)
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
      INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),
     C        BECTOR(M),DSCEND(M)
      INTEGER HIGH/2147483647/
      REAL TOTAL,MEAN
C
C     B E C T O R   M E T H O D     # 4
C
C     CALCULATE THE MEAN M, THE AVERAGE AMONG THE ROUTES ON THAT ROW.
C     MULTIPLY EACH M BY THE # OF UNITS REQUIRED AT THAT SOURCE
C     TO GET B, THE BECTOR'S NUMBER.
C     ASSIGN FIRST IN THE ROW WITH THE LARGEST B, THEN SECOND
C     LARGEST B AND SO ON.
C
C
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      FACTRY(I)=SUPPLY(I)
      DO 1 J=1,N
      COSTS(I,J)=COST(I,J)
    1 ALLOC(I,J)=0
      DO 2 J=1,N
    2 WAREHO(J)=DEMAND(J)
C
C     GENERATE THE BECTOR NUMBERS FOR ROWS (FACTORIES)
C
      DO 3 I=1,M
    3 BECTOR(I)=FACTRY(I)*MEAN(M,N,I,0)
C
C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER
C
      CALL SORT(M,BECTOR,DSCEND)
C
C     FILL UP THE CELL WITH LOWEST COST
C
      DO 6 I=1,M
      ROW=DSCEND(I)
C
    4 LOWEST=COSTS(ROW,1)
```

```
      COL=1
      DO 5 J=2,N
      IF (COSTS(ROW,J) .GE.LOWEST) GOTO 5
      COL=J
      LOWEST=COSTS(ROW,COL)
    5 CONTINUE
C
      COSTS(ROW,COL)=HIGH
      ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))
      FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)
      WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)
      IF (FACTRY(ROW) .GT.0) GOTO 4
    6 CONTINUE
C
C
C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD
C
      TOTAL=0.
      DO 30 I=1,M
      DO 30 J=1,N
   30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))
      WRITE(6,31) TOTAL
   31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 4 = ',E16.10)
C
C     OUTPUT THE ALLOCATION MATRIX
C
      DO 28 I=1,M
   28 WRITE(6,29) (ALLOC(I,J),J=1,N)
   29 FORMAT(1X,3314)
      RETURN
      END
```

Appendix H

SUBROUTINE BECTR5

```
      SUBROUTINE BECTR5(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)
      IMPLICIT INTEGER(A-Z)
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
      INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),
     C        BECTOR(N),DSCEND(N)
      INTEGER HIGH/2147483647/
      REAL TOTAL,COEFF
C
C     B E C T O R   M E T H O D     # 5
C
C     CALCULATE THE COEFFICIENT OF VARIATION C, THE QUOTIENT OF THE
C     STD. DEV AND MEAN OF THE COLUMN.
C     MULTIPLY EACH C BY THE # OF UNITS REQUIRED AT THAT DESTINATION
C     TO GET B, THE BECTOR'S NUMBER.
C     ASSIGN FIRST IN THE COLUMN WITH THE LARGEST B, THEN SECOND
C     LARGEST B AND SO ON.
C
C
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      FACTRY(I)=SUPPLY(I)
      DO 1 J=1,N
      COSTS(I,J)=COST(I,J)
    1 ALLOC(I,J)=0
      DO 2 J=1,N
    2 WAREHO(J)=DEMAND(J)
C
C     GENERATE THE BECTOR NUMBERS FOR COLUMNS (WAREHOUSES)
C
      DO 3 J=1,N
    3 BECTOR(J)=WAREHO(J)*COEFF(M,N,0,J)
C
C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER
C
      CALL SORT(N,BECTOR,DSCEND)
C
C     FILL UP THE CELL WITH LOWEST COST
C
      DO 6 J=1,N
      COL=DSCEND(J)
C
```

```
    4 LOWEST=COSTS(1,COL)
      ROW=1
      DO 5 I=2,M
      IF(COSTS(I,COL).GE.LOWEST)GOTO 5
      ROW=I
      LOWEST=COSTS(ROW,COL)
    5 CONTINUE
C
      COSTS(ROW,COL)=HIGH
      ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))
      FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)
      WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)
      IF(WAREHO(COL).GT.0)GOTO 4
    6 CONTINUE
C
C
C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD
C
      TOTAL=0.
      DO 30 I=1,M
      DO 30 J=1,N
   30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))
      WRITE(6,31)TOTAL
   31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 5 = ',E16.10)
C
C     OUTPUT THE ALLOCATION MATRIX
C
      DO 28 I=1,M
   28 WRITE(6,29)(ALLOC(I,J),J=1,N)
   29 FORMAT(1X,3314)
      RETURN
      END
```

Appendix I

SUBROUTINE BECTR6

```
      SUBROUTINE BECTR6(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)
      IMPLICIT INTEGER(A-Z)
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
      INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),
     C        BECTOR(M),DSCEND(M)
      INTEGER HIGH/2147483647/
      REAL TOTAL,COEFF
C
C     B E C T O R   M E T H O D     # 6
C
C     CALCULATE THE COEFFICIENT OF VARIATION C, THE QUOTIENT OF THE
C     THE STD. DEV AND MEAN OF THE ROW.
C     MULTIPLY EACH C BY THE # OF UNITS REQUIRED AT THAT SOURCE
C     TO GET B, THE BECTOR'S NUMBER.
C     ASSIGN FIRST IN THE ROW WITH THE LARGEST B, THEN SECOND
C     LARGEST B AND SO ON.
C
C
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      FACTRY(I)=SUPPLY(I)
      DO 1 J=1,N
      COSTS(I,J)=COST(I,J)
    1 ALLOC(I,J)=0
      DO 2 J=1,N
    2 WAREHO(J)=DEMAND(J)
C
C     GENERATE THE BECTOR NUMBERS FOR ROWS (FACTORIES)
C
      DO 3 I=1,M
    3 BECTOR(I)=FACTRY(I)*COEFF(M,N,I,0)
C
C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER
C
      CALL SORT(M,BECTOR,DSCEND)
C
C     FILL UP THE CELL WITH LOWEST COST
C
      DO 6 I=1,M
      ROW=DSCEND(I)
C
```

```
    4 LOWEST=COSTS(ROW,1)
      COL=1
      DO 5 J=2,N
      IF(COSTS(ROW,J).GE.LOWEST)GOTO 5
      COL=J
      LOWEST=COSTS(ROW,COL)
    5 CONTINUE
C
      COSTS(ROW,COL)=HIGH
      ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))
      FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)
      WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)
      IF(FACTRY(ROW).GT.0)GOTO 4
    6 CONTINUE
C
C
C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD
C
      TOTAL=0.
      DO 30 I=1,M
      DO 30 J=1,N
   30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))
      WRITE(6,31)TOTAL
   31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 6 = ',E16.10)
C
C     OUTPUT THE ALLOCATION MATRIX
C
      DO 28 I=1,M
   28 WRITE(6,29)(ALLOC(I,J),J=1,N)
   29 FORMAT(1X,3314)
      RETURN
      END
```

Appendix J

SUBROUTINE CAMPBL

```
      SUBROUTINE CAMPBL(N,IA,IB,IAB,IBA,JAB,JBA,PAB,PBA,ICHK)
C
C     ASSUME A BATCH OF JOBS TO BE PROCESSED ON 2 MACHINES A AND B
C     SOME JOBS ARE TO BE PROCESSED ONLY ON MACHINE; SOME ONLY ON
C     MACHINE B. SOME ARE TO BE PROCESSED ON BOTH MACHINES A AND B
C     IN THE ORDER A,B
C     SOME ARE TO BE PROCESSED ON BOTH MACHINES A AND B IN THE ORDER
C     B, A. THE GOAL IS TO MINIMIZE TOTAL PROCESSING TIME.
C
      DIMENSION IA(N),IB(N),IAB(N),IBA(N),JAB(N),JBA(N),PAB(2,N),
     C          PBA(2,N),ICHK(N)
      INTEGER CASE
      WRITE(6,8)N
      KA=0
      KB=0
      KAB=0
      KBA=0
      DO 5 I=1,N
      READ(5,*)CASE,A,B
C
C     CASE          ORDER
C      1            1, 2
C      2            2, 1
C      3            1, 0
C      4            0, 2
C
      GOTO(1,2,3,4),CASE
    1 KAB=KAB+1
      IAB(KAB)=I
      PAB(1,KAB)=A
      PAB(2,KAB)=B
      WRITE(6,9)A,B
      GOTO 5
    2 KBA=KBA+1
      IBA(KBA)=I
      PBA(1,KBA)=B
      PBA(2,KBA)=A
      WRITE(6,10)A,B
      GOTO 5
    3 KA=KA+1
      IA(KA)=I
      WRITE(6,11)A,B
      GOTO 5
    4 KB=KB+1
```

```
      IB(KB)=I
      WRITE(6,12)A,B
    5 CONTINUE
C
C
      CALL JONSON(KAB,JAB,IAB,PAB,ICHK)
      CALL JONSON(KBA,JBA,IBA,PBA,ICHK)
      K=1
      WRITE(6,7)K,(JAB(I),I=1,KAB),(IA(I),I=1,KA),(JBA(I),I=1,KBA)
      K=2
      WRITE(6,7)K,(JBA(I),I=1,KBA),(IB(I),I=1,KB),(JAB(I),I=1,KAB)
    7 FORMAT('OOPTIMAL ORDER ON MACHINE ',I1,' IS:',/(10(2X,I2)/))
    8 FORMAT(1H-,'2 MACHINES     ',I3,' JOBS',//,
     C ' MACHINE',6X,'PROCESSING TIME',/'  ORDER',8X,'FOR EACH JOB',/,
     C ' --------',5X,'---------------')
    9 FORMAT(1H ,'ONE TWO',9X,2F5.0)
   10 FORMAT(1H ,'TWO ONE',9X,2F5.0)
   11 FORMAT(1H ,'ONE ONLY',8X,2F5.0)
   12 FORMAT(1H ,'TWO ONLY',8X,2F5.0)
      RETURN
      END
```

Appendix K

SUBROUTINE CHEAP

```
      SUBROUTINE CHEAP(M,N,ALLOC,FACTRY,WAREHO,COSTS)
      IMPLICIT INTEGER(A-Z)
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
      INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N)
      INTEGER HIGH/2147483647/
      REAL TOTAL
C
C     CHEAPEST ROUTE METHOD:
C
C     FILL UP THE CELL WITH THE LOWEST COST FIRST AND SO ON
C
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      FACTRY(I)=SUPPLY(I)
      DO 1 J=1,N
      COSTS(I,J)=COST(I,J)
    1 ALLOC(I,J)=0
      DO 2 J=1,N
    2 WAREHO(J)=DEMAND(J)
C
C     FIND THE LOWEST COST ON COST MAP "COSTS"
C
    3 LOWEST=COSTS(1,1)
      ROW=1
      COL=1
      DO 4 I=1,M
      DO 4 J=1,N
      IF(COSTS(I,J).GE.LOWEST)GOTO 4
      LOWEST=COSTS(I,J)
      ROW=I
      COL=J
    4 CONTINUE
C
C     THE CHEAPEST ROUTE IS ASSIGNED WITH A HIGH VALUE NUMBER
C     AFTER BEING USED
C
      COSTS(ROW,COL)=HIGH
      ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))
      FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)
      WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)
C
```

```
C     FIND THE NEXT CHEAPEST ROUTE IF THE REQUIREMENT OF WAREHOUSES
C     AND FACTORIES ARE NOT MET
C
      DO 5 I=1,M
    5 IF(FACTRY(I).GT.0)GOTO 3
      DO 6 J=1,N
    6 IF(WAREHO(J).GT.0)GOTO 3

C
C
C     CALCULATE THE TOTAL COST FOR THE B.F.S IN CHEAPEST ROUTE
C
      TOTAL=0.
      DO 20 I=1,M
      DO 20 J=1,N
   20 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))
      WRITE(6,21)TOTAL
   21 FORMAT('0','TOTAL COST IN  CHEAPEST ROUTE METHOD = ',E16.10)
C
C     OUTPUT THE ALLOCATION MATRIX
C
      DO 18 I=1,M
   18 WRITE(6,19) (ALLOC(I,J),J=1,N)
   19 FORMAT(1X,3314)
      RETURN
      END
```

Appendix L

SUBROUTINE EASTMN

```
      SUBROUTINE EASTMN(N,IIAS,IDIST,ISDIST,IAS,JCOMP,BFSD,BRANCH,R,C,P,
     C                  TD,NARCS,SUBT)
C
C     E A S T M A N  '  S       A L G O R I T H M
C
C     SOLVE THE TRAVELING SALESMAN PROBLEM BY USING THE HUNGARIAN
C     ALGORITHM TO SOLVE THE ASSOCIATED ASSIGNMENT PROBLEM. SUBROUTINE
C     HUNGRY MUST BE SUPPLIED.
C
C     IDIST     DISTANCE MATRIX
C     IAS       ASSIGNMENT MATRIX
C     MD        MINIMUM DISTANCE OF THE ASSOCIATED ASSIGMENT PROBLEM
C     OPTSOL    DISTANCE OF OPTIMAL FEASIBLE SOLUTION
C
C
C     WHEN N > 15, ALL THE "15" COMPONENTS IN THE DIMENSION STATEMENT
C     HAVE TO CHANGE TO THE VALUE OF N
C     VARIABLE IDIST,KTEMP,ICOL,ICHECK,JCHECK CANNOT BE DYNAMICALLY
C     ALLOCATED (ALL THE VARIABLES WITH DIMENSION > 1 AND APPEAR ON
C     THE READ STATEMENT HAVE A DIFFERENT ADDRESS IN FORTRAN-77)
C
      DIMENSION IAS(N),IDIST(15,15),ISDIST(N,N),JCOMP(N),BFSD(N),
     C          IIAS(N*30,N),BRANCH(N*30),R(N*30),C(N*30),P(N*30),
     C          TD(N*30),NARCS(N*30),SUBT(N*30),
     C          KTEMP(15,15),ICOL(15),ICHECK(15),JCHECK(15)
      INTEGER BIG M/999999999/
      DO 2 I=1,N
    2 READ(5,*) (IDIST(I,J),J=1,N)
      KODE=0
      WRITE(6,5)
      DO 3 I=1,N
    3 WRITE(6,4) (IDIST(I,J),J=1,N)
    4 FORMAT(1018)
    5 FORMAT(1H-,'THE DISTANCE MATRIX:')
C
C     SET MINIMUM TOTAL DISTANCE (OPTSOL) TO LARGE POSITIVE NUMBER
C
      OPTSOL=1E10
C
C     CALL SUBROUTINE HUNGRY TO SOLVE THE ASSOCIATED ASSIGNMENT PROBLEM
C     BY THE HUNGARIAN METHOD
C     IF AT LEAST ONE SUBTOUR EXISTS, SKIP THIS STEP.
C     OTHERWISE, AN OPTIMAL SOLUTION OF THE ASSIGNMENT PROBLEM IS THE
C     OPTIMAL SOLUTION OF THE TRAVELING SALESMAN PROBLEM. STOP.
```

```
C
      CALL HUNGAR(N,IDIST,KTEMP,IAS,ICOL,ICHECK,JCHECK,MD,KODE,1)
      DO 71 I=1,N
   71 JCOMP(I)=IAS(I)
      DO 70 II=2,N-2
      DO 72 I=1,N
      J=JCOMP(I)
      JCOMP(I)=IAS(J)
      IF(JCOMP(I).EQ.I)GOTO 80
   72 CONTINUE
   70 CONTINUE
      GOTO 200
C
C     A SUBTOUR EXISTS OF WHICH WE HAVE THE SHORTEST. DETERMINE THE
C     PARAMETERS OF THE SHORTEST SUBTOUR
C
   80 IARS=II
      INT=I
      DO 701 KK=1,N
  701 IIAS(1,KK)=IAS(KK)
      K=1
C
C     BRANCH INTO K SUBLEVELS
C
      P(1)=1
      IPRED=1
      GOTO 110
  700 LL1=IPRED
  111 IF(P(LL1).EQ.1)GOTO 112
      LL2=R(LL1)
      LL3=C(LL1)
      ISDIST(LL2,LL3)=IDIST(LL2,LL3)
      IDIST(LL2,LL3)=BIG M
      LL1=P(LL1)
      GOTO 111
  112 LL2=R(LL1)
      LL3=C(LL1)
      ISDIST(LL2,LL3)=IDIST(LL2,LL3)
      IDIST(LL2,LL3)=BIG M
      IARS=NARCS(IPRED)
      INT=SUBT(IPRED)
  110 ITERM=INT
      DO 90 IJ=1,IARS
      IBEGIN=ITERM
      ITERM=IIAS(IPRED,IBEGIN)
      K=K+1
      R(K)=IBEGIN
      C(K)=ITERM
      ITEMP=IDIST(IBEGIN,ITERM)
      IDIST(IBEGIN,ITERM)=BIG M
C
C     SOLVE THE K SUBLEVELS AS NEW ASSIGNMENT PROBLEMS, AND LET
C     EACH SOLUTION DISTANCE BE A LOWER BOUND FOR THE CORESPONDING
C     SUBLEVEL
```

```
C
      CALL HUNGAR(N,IDIST,KTEMP,IAS,ICOL,ICHECK,JCHECK,MD,KODE,1)
      TD(K)=MD
      P(K)=IPRED
      IDIST(IBEGIN,ITERM)=ITEMP
      DO 125 MM=1,N
      IIAS(K,MM)=IAS(MM)
  125 CONTINUE
      DO 73 I=1,N
      JCOMP(I)=IAS(I)
   73 CONTINUE
      DO 75 II=2,NN
      DO 74 I=1,N
      J=JCOMP(I)
      JCOMP(I)=IAS(J)
      IF(JCOMP(I).EQ.I)GOTO 131
   74 CONTINUE
   75 CONTINUE
C
C     IF THERE EXISTS 1 OR MORE FEASIBLE SOLUTIONS FROM PREVIOUS STEP
C     AND IF THE SMALLEST TOTAL DISTANCE FOR THAT FEASIBLE SOLUTION
C     IS SMALLER THAN OPTSOL, SET OPTSOL = STD AND SAVE THE
C     CORRESPONDING FEASIBLE SOLUTION
C
      IF(OPTSOL.LT.TD(K))GOTO 130
      OPTSOL=TD(K)
      DO 129 JJ=1,N
  129 BFSD(JJ)=IAS(JJ)
  130 BRANCH(K)=0
      GOTO 90
  131 BRANCH(K)=1
      SUBT(K)=I
      NARCS(K)=II
   90 CONTINUE
      IF(IPRED.EQ.1)GOTO 160
      LL1=IPRED
  150 IF(P(LL1).EQ.1)GOTO 152
      LL2=R(LL1)
      LL3=C(LL1)
      LL1=P(LL1)
      IDIST(LL2,LL3)=ISDIST(LL2,LL3)
      GOTO 150
  152 LL2=R(LL1)
      LL3=C(LL1)
      IDIST(LL2,LL3)=ISDIST(LL2,LL3)
C
C     IF OPTSOL IS LESS THAN THE LOWER BOUNDS ON ALL OF THE UNEXPLORED
C     SUBLEVELS, THE SOLUTION CORESPONDING TO OPTSOL IS AN OPTIMAL
C     SOLUTION, SO STOP. OTHERWISE, GO TO NEXT STEP
C
  160 DO 132 I=2,K
      IF(BRANCH(I).EQ.0)GOTO 132
      IF(OPTSOL.GT.TD(I))GOTO 134
  132 CONTINUE
```

```
      GOTO 135
C
C     SELECT THE UNEXPLORED SUBLEVEL WITH THE SMALLEST VALUE
C     FIND THE MINIMUM DISTANCE FOR THE UNFEASIBLE NODES
C     SET THE INITIAL MINIMAL NODE TO THE VALUE OF THE COUNTER IN THE
C     PREVIOUS LOOP, AND RETURN FOR FURTHER BRANCHING
C
  134 MINND=I
      DO 142 J=1,K
      IF(BRANCH(J).EQ.0)GOTO 142
      IF(TD(J).LT.TD(MINND))MINND=J
  142 CONTINUE
      BRANCH(MINND)=0
      IPRED=MINND
      GOTO 700
C
C     OUTPUT THE FINAL MINIMAL DISTANCE
C     OUTPUT THE ASSIGNMENT FOR THE OPTIMAL SOLUTION
C
  135 WRITE(6,222)OPTSOL
  222 FORMAT(//,' THE OPTIMAL FEASIBLE SOLUTION DISTANCE IS',F12.0)
      DO 136 I=1,N
      WRITE(6,220)I,BFSD(I)
  220 FORMAT(' THE ASSIGNMENT FOR',I4,' IS ',F12.0)
  136 CONTINUE
      GOTO 999
C
C     THE SOLUTION TO THE ASSIGNMENT PROBLEM IS THE OPTIMAL SOLUTION
C     TO THE TRAVELING SALESMAN PROBLEM
C
  200 WRITE(6,201)
  201 FORMAT(////5X,'THE SOLUTION HAS NO SUBTOURS',//5X,
     C     'THE OPTIMAL SOLUTION OF THE ASSIGNMENT PROBLEM IS',/5X,
     C     'ALSO AN OPTIMAL SOLUTION OF THE TRAVELING SALESMAN PROBLEM')
      WRITE(6,202)MD
  202 FORMAT(5X,'THE TOTAL MINIMAL DISTANCE IS',I8,//)
      DO 203 I=1,N
  203 WRITE(6,204)I,IAS(I)
  204 FORMAT(5X,'THE ASSIGNMENT FOR',I4,' IS ',I8)
  999 RETURN
      END
```

Appendix M

SUBROUTINE GUPTA

```
      SUBROUTINE GUPTA(NJ,NP,V,IO,JO,P,PAB,ICHK)
C
C     GUPTA'S   METHOD     ( N JOB / M MACHINE )
C
      DIMENSION P(NJ,NP),PAB(2,NJ),V(NJ),IO(NJ),JO(NJ),ICHK(NJ)
      REAL BIG M/1.E9/
      WRITE(6,1)NP,NJ
    1 FORMAT(1H-,I3,' MACHINES      ',I3,' JOBS',/
     C ' PROCESSING TIME FOR EACH JOB ON PROCESS',/)
      IF(NP.NE.2)THEN
           DO 5 I=1,NJ
           READ(5,*) (P(I,J),J=1,NP)
           WRITE(6,10) I, (P(I,J),J=1,NP)
           V(I)=BIG M
           DO 6 J=1,NP-1
    6      V(I)=AMIN1(V(I), (P(I,J)+P(I,J+1)))
           IF(P(I,1).LT.P(I,NP))V(I)=-V(I)
    5      V(I)=1./V(I)
           CALL SORTX(NJ,V,IO)
      ELSE
           DO 8 I=1,NJ
           JO(I)=I
           READ(5,*)PAB(1,I),PAB(2,I)
    8      WRITE(6,10) I,PAB(1,I),PAB(2,I)
           CALL JONSON(NJ,IO,JO,PAB,ICHK)
      END IF
   10 FORMAT(' JOB ',I3,':',30F4.0)
      WRITE(6,11) (IO(I),I=1,NJ)
   11 FORMAT('0OPTIMUM SEQUENCE TO MINIMIZE TOTAL PROCESSING TIME IS:',/
     C       2014,/)
      RETURN
      END
```

Appendix N

SUBROUTINE HUNGAR

```
      SUBROUTINE HUNGAR(N,IIF,IEFF,IROW,ICOL,ICHECK,JCHECK,TOTAL,KODE,
     C                  ESTMAN)
C
C         H U N G A R I A N     A L G O R I T H M
C
C         A METHOD FOR SOLVING THE GENERAL M-RESOURCE, M-ACTIVITY
C         ASSIGNMENT PROBLEM FOR TOTAL EFFECTIVENESS. THE PROBLEM
C         MAY BE EITHER MINIMIZATION OR MAXIMIZATION. THE PROGRAM
C         USES INTEGER IN THE MATRICES.
C
C         IIF     ASSINGMENT MATRIX
C         TOTAL   OPTIMAL COST OF ASSIGNMENTS
C         KODE    0 - MINIMIZATION PROBLEM
C                 1 - MAXIMIZATION PROBLEM
C         ESTMAN SWITCH INDICATED CALL FROM SUBROUTINE EASTMN
C
      INTEGER IIF(15,15),IEFF(N,N),IROW(N),ICOL(N),ICHECK(N),
     C        JCHECK(N),TOTAL,ESTMAN
      IF(ESTMAN.EQ.0)THEN
      WRITE(6,3)
      DO 2 I=1,N
      READ(5,*) (IIF(I,J),J=1,N)
      WRITE(6,1) (IIF(I,J),J=1,N)
    1 FORMAT(1X,1018)
    2 CONTINUE
    3 FORMAT(1H-,'ORIGINAL MATRIX:')
      END IF
C
C     CONSTRUCT THE EFFECTIVENESS MATRIX. IF TOTAL EFFECTIVENESS IS TO
C     BE MAXIMIZED, LET THE MATRIX BE THE NEGATIVE OF THE ORIGINAL
C     MATRIX; OTHERWISE, THE MATRIX EQUALS THE ORIGINAL MATRIX.
C
      IF(KODE.EQ.0)THEN
           DO 110 I=1,N
           DO 110 J=1,N
  110      IEFF(I,J)=IIF(I,J)
      ELSE
           DO 120 I=1,N
           DO 120 J=1,N
  120      IEFF(I,J)=-IIF(I,J)
      END IF
C
C     FIND THE SMALLEST ELEMENT OF EACH ROW AND SUBTRACT IT FROM THE
C     OTHER ELEMENTS OF THE SAME ROW
```

```
C
      DO 210 I=1,N
      MINROW=IEFF(I,1)
      DO 220 J=1,N
      IF(IEFF(I,J).LE.MINROW)MINROW=IEFF(I,J)
  220 CONTINUE
      DO 210 J1=1,N
      IEFF(I,J1)=IEFF(I,J1)-MINROW
  210 CONTINUE
C
C     FIND THE SMALLEST ELEMENT OF EACH COLUMN AND SUBTRACT IT FROM THE
C     OTHER ELEMENTS IN THE SAME COLUMN
C
      DO 300 J=1,N
      MINCOL=IEFF(1,J)
      DO 310 I=1,N
      IF(IEFF(I,J).LE.MINCOL)MINCOL=IEFF(I,J)
  310 CONTINUE
      DO 300 I1=1,N
      IEFF(I1,J)=IEFF(I1,J)-MINCOL
  300 CONTINUE
C
C     ROW ASSIGNMENTS
C
  400 DO 410 I=1,N
      IROW(I)=0
  410 ICOL(I)=0
      NOMADE=0
  420 LOOP=0
      NZEROS=0
      DO 430 I=1,N
      NOZR=0
      IF(IROW(I).EQ.0)THEN
      DO 440 J=1,N
      IF(ICOL(J).EQ.0)THEN
      IF(IEFF(I,J).EQ.0)THEN
            NOZR=NOZR+1
            NZEROS=NZEROS+1
            NRPT=J
            NREFF=I
      END IF
      END IF
  440 CONTINUE
      IF(NOZR.EQ.1)THEN
            IROW(I)=NRPT
            ICOL(NRPT)=1
            NOMADE=NOMADE+1
            LOOP=LOOP+1
      END IF
      END IF
  430 CONTINUE
C
C     COLUMN ASSIGNMENTS
C
```

```
      DO 500 I=1,N
      NOZC=0
      IF(ICOL(I).EQ.0)THEN
      DO 510 J=1,N
      IF(IROW(J).EQ.0)THEN
      IF(IEFF(J,I).EQ.0)THEN
            NOZC=NOZC+1
            NZEROS=NZEROS+1
            IRPT=J
      END IF
      END IF
  510 CONTINUE
      IF(NOZC.EQ.1)THEN
            ICOL(I)=1
            IROW(IRPT)=I
            NOMADE=NOMADE+1
            LOOP=LOOP+1
      END IF
      END IF
  500 CONTINUE
C
C     IF COMPLETE ASSIGNMENT HAS BEEN MADE, RETURN
C     IF AT LEAST ONE ASIGNMENT HAS BEEN MADE, RETURN TO ROW ASSIGNMENTS
C     TO MAKE ADDITIONAL ASSIGNMENTS.
C
      IF(NOMADE.EQ.N)GOTO 1500
      IF(LOOP.NE.0)GOTO 420
      IF(NZEROS.EQ.0)GOTO 800
C
C     PICK AN ARBITRARY ZERO AND RETURN TO ROW ASSIGNMENTS
C
      IROW(NREFF)=NRPT
      NOMADE=NOMADE+1
      ICOL(NRPT)=1
      GOTO 420
C
C     CHECK FOR UNASSIGNED ROW
C
  800 DO 810 I=1,N
      ICHECK(I)=0
      JCHECK(I)=0
      IF(IROW(I).EQ.0)ICHECK(I)=1
  810 CONTINUE
C
C     CHECK COLUMNS THAT HAVE ZEROS IN CHECKED ROWS
C
  900 NCHECK=0
      DO 910 I=1,N
      IF(ICHECK(I).NE.0)THEN
      DO 920 J=1,N
      IF(JCHECK(J).EQ.0)THEN
      IF(IEFF(I,J).EQ.0)THEN
            JCHECK(J)=J
            NCHECK=NCHECK+1
```

```
      END IF
      END IF
  920 CONTINUE
      END IF
  910 CONTINUE
C
C     CHECK ROWS THAT HAVE ASSIGNMENTS IN CHECKED COLUMNS
C     REPEAT UNTIL CHAIN OF CHECKING IS COMPLETED.
C     ELEIMINATE ALL UNCHECKED ROWS AND ALL CHECKED COLUMNS
C
      IF(NCHECK.NE.0)THEN
      DO 1010 I=1,N
      IF(JCHECK(I).GT.0)THEN
      DO 1020 J=1,N
      IF(JCHECK(I).EQ.IROW(J))THEN
      ICHECK(J)=J
      NCHECK=NCHECK+1
      END IF
 1020 CONTINUE
      END IF
 1010 CONTINUE
      END IF
      IF(NCHECK.NE.0)GOTO 900
C
C    FIND THE MINIMUM UNMARKED ELEMENT
C
      MINELM=99999999
      DO 1310 I=1,N
      IF(ICHECK(I).NE.0)THEN
      DO 1320 J=1,N
      IF(JCHECK(J).EQ.0)THEN
           IF(IEFF(I,J).LT.MINELM)MINELM=IEFF(I,J)
      END IF
 1320 CONTINUE
      END IF
 1310 CONTINUE
C
C     REDUCE MATRIX AND GOTO TO ROW REDUCTIONS
C
      DO 1400 I=1,N
      DO 1400 J=1,N
      IF(ICHECK(I).GE.0)THEN
           IF(ICHECK(I).EQ.0)THEN
                IF(JCHECK(J).GT.0)IEFF(I,J)=IEFF(I,J)+MINELM
           ELSE
                IF(JCHECK(J).EQ.0)IEFF(I,J)=IEFF(I,J)-MINELM
           END IF
      END IF
 1400 CONTINUE
      GOTO 400
C
C     CALCULATE TOTAL AND RETURN TO MAIN PROGRAM
C
 1500 TOTAL=0
```

```
      DO 1510 I=1,N
      K=IROW(I)
 1510 TOTAL=TOTAL+IIF(I,K)
C
C
      IF(ESTMAN.EQ.0)THEN
      WRITE(6,1520)TOTAL
 1520 FORMAT(1HO,5X,'THE OPTIMAL SOLUTION OF THE ASSIGNMENT PROBLEM''S T
     COTAL COST = ',I8/)
      DO 1530 I=1,N
 1530 WRITE(6,1540)I,IROW(I)
 1540 FORMAT(10X,'THE ASSIGNMENT FOR ',I3,' = ',I8)
      END IF
      RETURN
      END
```

Appendix O

SUBROUTINE JONSON

```
      SUBROUTINE JONSON(N,IO,JO,PAB,ICHK)
C
C     JOHNSON'S METHOD
C
      DIMENSION IO(N),PAB(2,N),ICHK(N),JO(N)
      K=N
      J=1
      DO 1 I=1,N
    1 ICHK(I)=0
    2 M=0
      DO 3 I=1,N
      IF(ICHK(I).EQ.1)GOTO 3
      IF(M)4,4,5
    5 IF(PAB(1,I).GT.X.AND.PAB(2,I).GT.X)GOTO 3
    4 M=I
      X=AMIN1(PAB(1,I),PAB(2,I))
      L=1
      IF(X.EQ.PAB(2,I))L=2
    3 CONTINUE
      IF(M.EQ.0)RETURN
      ICHK(M)=1
      IF(L-1)6,6,7
    6 IO(J)=JO(M)
      J=J+1
      GOTO 2
    7 IO(K)=JO(M)
      K=K-1
      GOTO 2
      END
```

Appendix P

SUBROUTINE MAXTRD

```
      SUBROUTINE MAXTRD(N,P,IO,W)
C
C     MAXIMIZE THE MINIMUM TARDINESS ON ONE PROCESS (SLACK RULE)
C
C     ASSUME A BATCH OF JOBS WITH KNOW DUE DATES, COUNTED FROM TODAY,
C     AND KNOW OR ESTIMATED PROCESSING TIMES. OUT OF ALL THE POSSIBLE
C     WAYS TO SEQUENCE THESE JOBS, CHOOSE THE SEQUENCE THAT WILL
C     MAXIMIZE THE MINIMUM TARDINESS
C
C     ORDER THE JOBS FROM LEFT TO RIGHT IN ORDEROF INCREASING
C     (DUE DATE MINUS PROCESSING TIME). STOP WHEN THE OPTIMAL
C     SEQUENCE HAS BEEN FOUND.
C
C
      DIMENSION P(N),IO(N),W(N)
      IF(P(1).EQ.0.)READ(5,*) (P(I),I=1,N)
      IF(W(1).EQ.0.)READ(5,*) (W(I),I=1,N)
      WRITE(6,5)N, (P(I),I=1,N)
      WRITE(6,6) (W(I),I=1,N)
      DO 1 I=1,N
    1 P(I)=W(I)-P(I)
      CALL SORTX(N,P,IO)
      WRITE(6,2)
    2 FORMAT('0OPTIMUM SEQUENCE TO MAXIMIZE MINIMUM LATENESS')
      WRITE(6,3) (IO(I),I=1,N)
    3 FORMAT(2014,/)
      DO 4 I=1,N
    4 P(I)=W(I)-P(I)
    5 FORMAT(1H-,'1 MACHINE      ',13,' JOBS',//,
     C' PROCESSING TIMES FOR EACH JOB:',5(/,2X,20F4.0) )
    6 FORMAT(1HO,'DUE DATES FROM NOW FOR EACH JOB:',5(/,2X,20F4.0) )
      RETURN
      END
```

Appendix Q

SUBROUTINE MINLAT

```
      SUBROUTINE MINLAT(N,P,IO,W)
C
C     MINIMIZE THE NUMBER OF LATE JOBS, N JOBS ON ONE MACHINE
C
C     ASSUME A BATCH OF N JOBS WITH KNOWN DUE DATES D(I) AND
C     KNOWN OR ESTIMATED PROCESSING TIMES P(I)
C
C     ORDER THE JOBS FROM LEFT TO RIGHT IN ORDER OF INCREASING DUE DATES
C     THAT IS, THE JOB WITH THE EARLIEST DUE DATE IS ORDERED FIRST AND
C     THE JOB WITH LATEST DUE DATE IS ORDERED LAST. THIS IS THE CURRENT
C     SEQUENCE
C
C     USING THE CURRENT SEQUENCE, FIND THE FIRST LATE JOB. IF ONE IS
C     FOUND, SKIP THIS STEP. OTHERWISE, STOP - THE SEQUENCE IS OPTIMAL
C
C     LOOK AT THE SUBSEQUENCE UP TO AND INCLUDING THE LATE JOB. FIND
C     THE JOB IN THIS SUBSEQUENCE WITH THE LARGEST PROCESSING TIME AND
C     REJECT IT. REPEAT THE PREVIOUS STEP
C
C
      DIMENSION P(N),IO(N),W(N)
      IF(P(1).EQ.0.)READ(5,*) (P(I),I=1,N)
      IF(W(1).EQ.0.)READ(5,*) (W(I),I=1,N)
      WRITE(6,10)N,(P(I),I=1,N)
      WRITE(6,11) (W(I),I=1,N)
      CALL SORTX(N,W,IO)
C
C     FIND LATE JOBS, IF ANY
C
      NN=N-1
      NJ=N
    1 T=0
      DO 2 I=1,NJ
      J=IO(I)
      T=T+P(J)
    2 IF(T.GT.W(J))GOTO 6
      WRITE(6,3)
    3 FORMAT('0OPTIMUM SEQUENCE TO MINIMIZE NUMBER OF LATE JOBS')
      WRITE(6,4) (IO(I),I=1,N)
    4 FORMAT(2014,/)
      NN=N-NJ
      WRITE(6,5)NJ,NN
    5 FORMAT(1HO,13,' JOBS ON TIME, ',13,' JOBS LATE')
      RETURN
```

```
C
C     REJECT LONGEST JOB
C
    6 LATEST=1
      DO 7 J=1,I
      K=IO(J)
    7 IF(P(LATEST).LT.P(K))LATEST=J
      J=LATEST
      LATEST=IO(J)
      NJ=NJ-1
      WRITE(6,8)LATEST
    8 FORMAT('OLATE JOB IS ',I3)
      DO 9 K=J,NN
    9 IO(K)=IO(K+1)
      IO(N)=LATEST
      GOTO 1
   10 FORMAT(1H-,'1 MACHINE      ',I3,' JOBS',//,
     C' PROCESSING TIMES FOR EACH JOB:',5(/,2X,20F4.0) )
   11 FORMAT(1HO,'DUE DATES FROM NOW FOR EACH JOB:',5(/,2X,20F4.0) )
      END
```

Appendix R

SUBROUTINE MINSCT

```
      SUBROUTINE MINSCT(N,P,IO,W)
C
C     MINIMIZE SUM OF COMPLETION TIMES OR SUM OF WAITING TIMES
C     ON ONE PROCESS
C
C     ASSUME A BATCH OF JOBS WITH KNOWN OR ESTIMATED PROCESSING TIMES.
C     SEQUENCE JOBS IN SUCH A WAY THAT
C     1) THE SUM OF COMPLETION TIMES OR
C     2) THE SUM OF WAITING TIMES IS MINIMIZED.
C
C     ORDER THE JOBS FROM LEFT TO RIGHT IN ORDER OF INCREASING
C     PROCESSING TIME. STOP WHEN THE OPTIMAL SEQUENCE HAS BEEN FOUND.
C
C
      DIMENSION P(N),IO(N),W(N)
      IF(P(1).EQ.0.)READ(5,*) (P(I),I=1,N)
      WRITE(6,1)N,(P(I),I=1,N)
    1 FORMAT(1H-,'1 MACHINE      ',I3,' JOBS',//,
     C' PROCESSING TIMES FOR EACH JOB:',5(/,2X,20F4.0) )
      CALL SORTX(N,P,IO)
      WRITE(6,2)
    2 FORMAT('0OPTIMAL SEQUENCE TO MINIMIZE SUM OF COMPLETION TIMES')
      WRITE(6,3) (IO(I),I=1,N)
    3 FORMAT(2014,/)
      RETURN
      END
      SUBROUTINE SORTX(N,X,I)
      DIMENSION X(N),I(N)
      K=1
      I(K)=1
    2 K=K+1
      IF(K.GT.N)RETURN
      KK=K
      I(K)=K
    3 KK=KK-1
      IF(KK.EQ.0)GOTO 2
      KKK=I(KK)
      IF(X(KKK).LE.X(K))GOTO 2
      I(KK+1)=KKK
      I(KK)=K
      GOTO 3
      END
```

Appendix S

SUBROUTINE MINTRD

```
      SUBROUTINE MINTRD(N,P,IO,W)
C
C     TO MINIMIZE THE MAXIMUM TARDINESS ON ONE PROCESS
C
C     ASSUME A BATCH OF JOBS WITH KNOW DUE DATES, COUNTED FROM TODAY,
C     AND KNOWN OR ESTIMATED PROCESSING TIMES. OUT OF ALL THE
C     POSSIBLE WAYS TO SEQUENCE THESE JOBS WE WISH TO CHOOSE THE
C     SEQUENCE THAT WILL MINIMIZE THE MAXIMUM TARDINESS ON ANY GIVEN JOB
C
C     ORDER THE JOBS FROM LEFT TO RIGHT IN ORDER OF INCREASING DUE DATE
C     STOP WHEN THE OPTIMAL SEQUENCE HAS BEEN FOUND
C
C
      DIMENSION P(N),IO(N),W(N)
      IF(W(1).EQ.0.)READ(5,*) (W(I),I=1,N)
      WRITE(6,3)N,(W(I),I=1,N)
      CALL SORTX(N,W,IO)
      WRITE(6,1)
    1 FORMAT('OOPTIMAL SEQUENCE TO MINIMIZE MAXIMUM TARDINESS')
      WRITE(6,2) (IO(I),I=1,N)
    2 FORMAT(2014,/)
    3 FORMAT(1H-,'1 MACHINE      ',13,' JOBS',//,
     C' DUE DATES FROM NOW FOR EACH JOB:',5(/,2X,20F4.0) )
      RETURN
      END
```

Appendix T

SUBROUTINE MINWTS

```
      SUBROUTINE MINWTS(N,P,IO,W,WT)
C
C     MINIMIZE WEIGHTED SUM OF COMPLETION TIMES OR
C     WEIGHTED SUM OF WAITING TIMES ON ONE PROCESS
C
C     ASSUME THAT A GROUP OF JOBS WITH KNOWN OR ESTIMATED PROCESSING
C     TIMES P(I). SOME JOBS ARE MORE IMPORTANT THAN OTHERS, SO
C     ASSIGN A POSITIVE WEIGHT W(I) TO EACH OF THE JOBS. THE GREATER
C     THE WEIGHT, THE MORE IMPORTANT THE JOB. SEQUENCE THE JOBS IN
C     SUCH A WAY THAT THE WEIGHTED SUM OF COMPLETION TIMES IS
C     MINIMIZED.
C
C     ORDER THE JOBS FROM LEFT TO RIGHT IN ORDER OF THE INCREASING
C     RATIO P(I)/W(I). STOP WHEN THIS IS THE OPTIMAL SEQUENCE.
C
C
      DIMENSION P(N),IO(N),W(N),WT(N)
      IF(P(1).EQ.0.)READ(5,*) (P(I),I=1,N)
      READ(5,*) (WT(I),I=1,N)
      WRITE(6,5)N,(P(I),I=1,N)
      WRITE(6,6) (WT(I),I=1,N)
      DO 1 I=1,N
    1 P(I)=P(I)/WT(I)
      CALL SORTX(N,P,IO)
      WRITE(6,2)
    2 FORMAT('OOPTIMAL SEQUENCE TO MINIMIZE WEIGHTED SUM OF COMPLETI',
     C       'ON TIMES')
      WRITE(6,3) (IO(I),I=1,N)
    3 FORMAT(2014,/)
      DO 4 I=1,N
    4 P(I)=P(I)*WT(I)
    5 FORMAT(1H-,'1 MACHINE      ',13,' JOBS',//,
     C' PROCESSING TIMES FOR EACH JOB:',5(/,2X,20F4.0) )
    6 FORMAT(1HO,'WEIGHTS FOR EACH JOB:',5(/,2X,20F4.0) )
      RETURN
      END
```

Appendix U

SUBROUTINE NJOB2M

```
      SUBROUTINE NJOB2M(N,T,S,TIM1,TIM2,TOM1,TOM2,OSVECT)
C
C      SEQUENCING  N-JOBS THROUGH 2 MACHINES
C
C
C
C      T(I,1)  TIME TO PROCESS JOB I ON MACHINE 1
C      T(I,2)  TIME TO PROCESS JOB I ON MACHINE 2
C
C      OSVECT    OPTIMAL SEQUENCE FOR PROCESSING N JOBS THRU 2 MACHINES
C      TIM1,TOM1 TIME IN AND TIME OUT FOR MACHINE 1 FOR EACH JOB
C      TIM2,TOM2 TIME IN AND TIME OUT FOR MACHINE 2 FOR EACH JOB
C      ITM1      IDLE TIME ON MACHINE 1
C      ITM2      IDLE TIME ON MACHINE 2
C
C
      DIMENSION T(N,2),S(N,2),TIM1(N),TIM2(N),TOM1(N),TOM2(N)
      REAL ITM1,ITM2,BIG M/1.E16/
      INTEGER OSVECT(N)
      DO 11 I=1,N
   11 READ *,(T(I,J),J=1,2)
      WRITE(6,100)N
  100 FORMAT(1X,'SEQUENCING',I4,' JOBS THROUGH 2 MACHINES',///,
     C       ' JOB    PROCESSING TIME PROCESSING TIME',//,
     C       12X,'MACHINE 1',7X,'MACHINE 2',/)
      DO 1 I=1,N
    1 WRITE(6,101)I,(T(I,J),J=1,2)
  101 FORMAT(I3,2F16.2)
C
C     SAVE THE PROCESSING TIME VALUES INTO ARRAY S
C
      DO 2 I=1,N
      DO 2 J=1,2
    2 S(I,J)=T(I,J)
      K=1
      L=1
C
C     SELECT THE SMALLEST T(I,J) OF THOSE NOT-YET-ASSIGNED JOBS
C     CALCULATE THE OPTIMAL SEQUENCE AND PLACE IN OSVECT
C
    8 IF(T(1,1).GT.T(1,2))THEN
           SMALL=T(1,2)
           IZERO=1
           JZERO=2
```

```
      ELSE
           SMALL=T(1,1)
           IZERO=1
           JZERO=1
      END IF
C
      DO 5 I=2,N
      DO 5 J=1,2
      IF(T(I,J).LT.SMALL)THEN
           SMALL=T(I,J)
           IZERO=I
           JZERO=J
      END IF
    5 CONTINUE
C
C     IF JZERO = 1, ASSIGN JOB IZERO AS CLOSE TO FRONT AS POSSIBLE
C     OTHERWISE,    ASSIGN IT AS CLOSE TO END AS POSSIBLE
C
      IF(JZERO.EQ.1)THEN
           OSVECT(K)=IZERO
           K=K+1
      ELSE
           OSVECT(N-L+1)=IZERO
           L=L+1
      END IF
C
C     ELIMINATE JOB IZERO FROM FURTHER CONSIDERATION
C
      T(IZERO,1)=BIG M
      T(IZERO,2)=BIG M
C
C     IF ALL JOBS HAVE BEEN ASSIGNED, GOTO THE FOLLOWING STEP
C     OTHERWISE, REPEAT AL PREVIOUS STEPS
C
      IF(L+K.NE.N+2)GOTO 8
C
C     CALCULATE TIME IN & TIME OUT
C
      TIM1(1)=0
      TOM1(1)=S(OSVECT(1),1)
      TIM2(1)=TOM1(1)
      TOM2(1)=TIM2(1)+S(OSVECT(1),2)
      ITM2=TIM2(1)
C
C     CALCULATE IDLE TIME FOR EACH MACHINE
C
      DO 10 K=2,N
      TIM1(K)=TOM1(K-1)
      TOM1(K)=TIM1(K)+S(OSVECT(K),1)
      IF(TOM2(K-1).GE.TOM1(K))THEN
           TIM2(K)=TOM2(K-1)
      ELSE
           TIM2(K)=TOM1(K)
           ITM2=ITM2+TOM1(K)-TOM2(K-1)
```

```
      END IF
   10 TOM2(K)=TIM2(K)+S(OSVECT(K),2)
      ITM1=TOM2(N)-TOM1(N)
C
C       PRINT SEQUENCE FOR EACH MACHINE
C
      WRITE(6,102)
  102 FORMAT(1H-,' OPTIMAL SEQUENCE FOLLOWS:')
      WRITE(6,103) (OSVECT(I),I=1,N)
  103 FORMAT(1X,2514)
      WRITE(6,104)TOM2(N)
  104 FORMAT(1HO,1X,'TOTAL ELAPSED TIME     =',F10.2)
      WRITE(6,105)ITM1,ITM2
  105 FORMAT(1HO,1X,'IDLE TIME ON MACHINE 1 =',F10.2,//
     C           2X,'IDLE TIME ON MACHINE 2 =',F10.2)
      RETURN
      END
```

Appendix V

SUBROUTINE NJOB3M

```
      SUBROUTINE NJOB3M(N,T,S,TIM1,TIM2,TIM3,TOM1,TOM2,TOM3,OSVECT)
C
C      SEQUENCING  N-JOBS THROUGH 3 MACHINES
C
C      DETERMINE THE OPTIMAL SEQUENCE OF N JOBS THROUGH 3 MACHINES
C      WHEN THE SMALLEST PROCESSING TIME ON MACHINE 1 AND/OR MACHINE 3
C      IS AT LEAST AS GREAT AS THE LARGEST PROCESSING TIME ON MACHINE 2
C
C
C      T(I,1)  TIME TO PROCESS JOB I ON MACHINE 1
C      T(I,2)  TIME TO PROCESS JOB I ON MACHINE 2
C      T(I,3)  TIME TO PROCESS JOB I ON MACHINE 3
C
C      OSVECT    OPTIMAL SEQUENCE FOR PROCESSING N JOBS THRU 2 MACHINES
C      TIM1,TOM1 TIME  IN AND TIME OUT FOR MACHINE 1 FOR EACH JOB
C      TIM2,TOM2 TIME  IN AND TIME OUT FOR MACHINE 2 FOR EACH JOB
C      TIM3,TOM3 TIME  IN AND TIME OUT FOR MACHINE 3 FOR EACH JOB
C      ITM1      IDLE TIME ON MACHINE 1
C      ITM2      IDLE TIME ON MACHINE 2
C      ITM3      IDLE TIME ON MACHINE 3
C
C
      DIMENSION T(N,3),S(N,3),TIM1(N),TIM2(N),
     C          TOM1(N),TOM2(N),TIM3(N),TOM3(N)
      REAL ITM1,ITM2,ITM3,BIG M/1.E16/
      INTEGER OSVECT(N)
      WRITE(6,100)N
  100 FORMAT(1H-,'SEQUENCING',I4,' JOBS THROUGH 3 MACHINES',///,
     C       ' JOB    PROCESSING TIME PROCESSING TIME',//,
     C       12X,'MACHINE 1',7X,'MACHINE 2',7X,'MACHINE 3',/)
      DO 11 I=1,N
   11 READ *,(T(I,J),J=1,3)
      DO 1 I=1,N
    1 WRITE(6,101)I,(T(I,J),J=1,3)
  101 FORMAT(I3,3F16.2)
C
C     SAVE THE ORIGINAL PROCESSING TIME VALUES INTO ARRAY S
C
      DO 2 I=1,N
      DO 2 J=1,3
    2 S(I,J)=T(I,J)
C
C     CONVERT TO A 2-MACHINE PROBLEM
C
```

```
      DO 3 I=1,N
      T(I,1)=T(I,1)+T(I,2)
    3 T(I,2)=T(I,2)+T(I,3)
      K=1
      L=1
C
C     USE NJOB2M ALGORITHM TO FIND THE OPT. SEQ OF
C     THE NEW N-JOB,2-MACHINE PROBLEM
C
C     SELECT THE SMALLEST T(I,J) OF THOSE NOT-YET-ASSIGNED JOBS
C     CALCULATE THE OPTIMAL SEQUENCE AND PLACE IN OSVECT
C
    8 IF(T(1,1).GT.T(1,2))THEN
           SMALL=T(1,2)
           IZERO=1
           JZERO=2
      ELSE
           SMALL=T(1,1)
           IZERO=1
           JZERO=1
      END IF
C
      DO 5 I=2,N
      DO 5 J=1,2
      IF(T(I,J).GE.SMALL)GOTO 5
      SMALL=T(I,J)
      IZERO=I
      JZERO=J
    5 CONTINUE
C
C     IF JZERO = 1, ASSIGN JOB IZERO AS CLOSE TO FRONT AS POSSIBLE
C     OTHERWISE,    ASSIGN IT AS CLOSE TO END AS POSSIBLE
C
      IF(JZERO.EQ.1)THEN
           OSVECT(K)=IZERO
           K=K+1
      ELSE
           OSVECT(N-L+1)=IZERO
           L=L+1
      END IF
C
C     ELIMINATE JOB IZERO FROM FURTHER CONSIDERATION
C
      T(IZERO,1)=BIG M
      T(IZERO,2)=BIG M
C
C     IF ALL JOBS HAVE BEEN ASSIGNED, GOTO THE FOLLOWING STEP
C     OTHERWISE, REPEAT AL PREVIOUS STEPS
C
      IF(L+K.NE.N+2)GOTO 8
C
C     CALCULATE TIME IN & TIME OUT
C
      TIM1(1)=0
```

```
      TOM1(1)=S(OSVECT(1),1)
      TIM2(1)=TOM1(1)
      TOM2(1)=TIM2(1)+S(OSVECT(1),2)
      TIM3(1)=TOM2(1)
      TOM3(1)=TIM3(1)+S(OSVECT(1),3)
      ITM2=TIM2(1)
      ITM3=TIM3(1)
C
C     CALCULATE IDLE TIME FOR EACH MACHINE
C
      DO 12 K=2,N
      TIM1(K)=TOM1(K-1)
      TOM1(K)=TIM1(K)+S(OSVECT(K),1)
      IF(TOM2(K-1).GE.TOM1(K))THEN
           TIM2(K)=TOM2(K-1)
      ELSE
           TIM2(K)=TOM1(K)
           ITM2=ITM2+TOM1(K)-TOM2(K-1)
      END IF
      TOM2(K)=TIM2(K)+S(OSVECT(K),2)
      IF(TOM3(K-1).GE.TOM2(K))THEN
           TIM3(K)=TOM3(K-1)
      ELSE
           TIM3(K)=TOM2(K)
           ITM3=ITM3+TOM2(K)-TOM3(K-1)
      END IF
   12 TOM3(K)=TIM3(K)+S(OSVECT(K),3)
      ITM1=TOM3(N)-TOM1(N)
      ITM2=ITM2+TOM3(N)-TOM2(N)
C
C      PRINT SEQUENCE FOR EACH MACHINE
C
      WRITE(6,102)
  102 FORMAT(1H-,' OPTIMAL SEQUENCE FOLLOWS:')
      WRITE(6,103)(OSVECT(I),I=1,N)
  103 FORMAT(1X,2514)
      WRITE(6,104)TOM3(N)
  104 FORMAT(1HO,1X,'TOTAL ELAPSED TIME     =',F10.2)
      WRITE(6,105)ITM1,ITM2,ITM3
  105 FORMAT(1HO,1X,'IDLE TIME ON MACHINE 1 =',F10.2,//
     C           2X,'IDLE TIME ON MACHINE 2 =',F10.2,//
     C           2X,'IDLE TIME ON MACHINE 3 =',F10.2 )
      WRITE(6,106)
  106 FORMAT(1H-,13X,'MACHINE 1',13X,'MACHINE 2',13X,'MACHINE 3',/
     C       11X,'TIME',5(7X,'TIME'),/1X,'JOB',3(8X,'IN',9X,'OUT'),/)
      DO 14 I=1,N
   14 WRITE(6,120)OSVECT(I),TIM1(I),TOM1(I),TIM2(I),TOM2(I),TIM3(I),
     C            TOM3(I)
  120 FORMAT(I4,6F11.2)
      RETURN
      END
```

Appendix W

SUBROUTINE NORTHW

```
      SUBROUTINE NORTHW(M,N,ALLOC,FACTRY,WAREHO)
      IMPLICIT INTEGER(A-Z)
      INTEGER ALLOC(M,N),FACTRY(M),WAREHO(N)
      REAL TOTAL
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
C
C      NORTH-WEST CORNER METHOD:
C           ALWAYS START FROM POSITION (1,1) AS A CONVENTIONAL START PT
C           (AS A MATTER OF FACT, START FROM SOUTH-EAST IS PERFECTLY
C            ALL RIGHT, JUST AN ARBITARY POSITION)
C
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      FACTRY(I)=SUPPLY(I)
      DO 1 J=1,N
    1 ALLOC(I,J)=0
      DO 2 J=1,N
    2 WAREHO(J)=DEMAND(J)
      COL=1
      ROW=1
C
C     RANK = MINIMUM(ROW,COL)
C
      RANK=MINO(M,N)
      DO 10 IT=1,RANK
C
C     SUMCOL = SUM OF FACTORY(1) TO FACTORY(IT)
C     SUMROW = SUM OF WAREHOUSE(1) TO WAREHOSE(IT)
C
      SUMCOL=0
      DO 3 I=1,ROW
    3 SUMCOL=SUMCOL+ALLOC(I,COL)
      SUMROW=0
      DO 4 J=1,COL
    4 SUMROW=SUMROW+ALLOC(ROW,J)
C
C   TAKE THE MINIMUM OF SUPPLY & DEMAND AT ALLOCATION(I,J)
C
      ALLOC(ROW,COL)=MINO(FACTRY(ROW)-SUMROW,WAREHO(COL)-SUMCOL)
C
C     HIT LAST ROW OR COL?
```

```
C
C       IF(ROW.EQ.RANK.AND.M.LE.N .OR. COL.EQ.RANK.AND.N.LE.M)GOTO 11
        IF(IT.EQ.RANK)GOTO 11
C
C       FILL UP ALLOC(I+1,J) OR ALLOC(I,J+1) DEPENDS ON THE RIM CONDITION
C
        IF(ALLOC(ROW,COL).LT.FACTRY(ROW))GOTO 7
C
C      FULFIL THE WAREHOUSE(I) REQUIREMENT
C
     5  SUMCOL=SUMCOL+ALLOC(ROW,COL)
        IF(SUMCOL.EQ.WAREHO(COL))GOTO 6
        ALLOC(ROW+1,COL)=MINO(WAREHO(COL)-SUMCOL,FACTRY(ROW+1))
        ROW=ROW+1
        GOTO 5
     6  COL=COL+1
        GOTO 10
C
C       SEND OUT FACTORY(I) PRODUCTION
C
     7  SUMROW=SUMROW+ALLOC(ROW,COL)
        IF(SUMROW.EQ.FACTRY(ROW))GOTO 8
        ALLOC(ROW,COL+1)=MINO(FACTRY(ROW)-SUMROW,WAREHO(COL+1))
        COL=COL+1
        GOTO 7
     8  ROW=ROW+1
    10  CONTINUE
C
C       FILL UP THE REMAINING ROW
C
    11  SUM=0
        IF(M.GT.N)GOTO 14
        DO 12 J=1,COL
        RP1=COL+1
    12  SUM=SUM+ALLOC(RANK,J)
        DO 13 J=RP1,N
        ALLOC(RANK,J)=MINO(FACTRY(RANK)-SUM,WAREHO(J))
    13  SUM=SUM+ALLOC(RANK,J)
        GOTO 17
C
C       FILL UP THE REMAINING COL
C
    14  DO 15 I=1,ROW
    15  SUM=SUM+ALLOC(I,RANK)
        RP1=ROW+1
        DO 16 I=RP1,M
        ALLOC(I,RANK)=MINO(WAREHO(RANK)-SUM,FACTRY(I))
    16  SUM=SUM+ALLOC(I,RANK)
C
    17  CONTINUE
C
C       CALCULATE THE TOTAL COST FOR THE B.F.S IN NORTH-WEST METHOD
C
        TOTAL=0.
```

```
      DO 20 I=1,M
      DO 20 J=1,N
   20 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))
      WRITE(6,21)TOTAL
   21 FORMAT('0','TOTAL COST IN  NORTH-WEST CORNER METHOD = ',E16.10)
C
C     OUTPUT THE ALLOCATION MATRIX
C
      DO 18 I=1,M
   18 WRITE(6,19) (ALLOC(I,J),J=1,N)
   19 FORMAT(1X,3314)
      RETURN
      END
```

Appendix X

SUBROUTINE PERT

```
      SUBROUTINE PERT(M,N,C,CPATH,SD,ET,LT,SL,SIGTE,FLAG)
C
C      P  E  R  T
C
C      OUTPUT:
C
C      DETERMINE THE CRITICAL PATH IN A PERT NETWORK AND WILL PROVIDE
C      THE PROBABILITY OF COMPLETING EACH EVENT ON OR BEFORE THE
C      SCHEDULED COMPLETION TIME.
C
C      INPUT:
C
C      C(N,7):    DESCRIPTION OF THE NETORK
C
C        C(I,1) - ACTIVITY FROM EVENT I
C        C(I,2) - ACTIVITY TO EVENT J
C        C(I,3) - OPTIMISTIC COMPLETION TIME FOR ACTIVITY (I,J)
C        C(I,4) - MOST LIKELY COMPLETION TIME FOR ACTIVITY(I,J)
C        C(I,5) - PESSIMISTIC COMPLETION TIME FOR ACTIVITY(I,J)
C        C(I,6) - ESTIMATED MEAN COMPLETION TIME FOR ACTIVITY(I,J)
C        C(I,7) - ESTIMATED VARIANCE OF COMPLETION TIME OF ACTIVITY(I,J)
C
C      M          - MAXIMUM # OF EVENTS ON NETWORK
C      N          - MAXIMUM # OF ACTIVITIES ON NETWORK
C
C      SD(J)      - SCHEDULED COMPLETION TIME FOR EVENT J
C
C      INTERMEDIATE VARIABLES:
C
C      ET(M)      - EARLIEST EXPECTED COMPLETION TIME OF EVENT J
C      LT(M)      - LATEST ALLOWABLE EVENT COMPLETION TIME FOR EVENT I
C      SIGTE(M)   - SUM OF VARIANCES OF ACTIVITY TIME IN PATH USED TO
C                   CALCULATE ET(J)
C      CPATH(N)   - CRITICAL PATH
C
C
      INTEGER M,N,CPATH(N),FLAG(N)
      REAL C(N,7),SD(M),ET(M),LT(M),SL(M),SIGTE(M),
     C     BIG M/1.E12/,ERROR/1.E-4/
      DO 1 I=1,N
      READ(5,*) (C(I,J),J=1,5)
    1 FLAG(I)=0
C
C     CALCULATE ESTIMATED MEAN COMPLETION TIME OF ACTIVITY T(I,J)
```

```
C      CALCULATE ESTIMATED VARIANCE OF COMPLETION TIME FOR EACH
C                EVENT VT(I,J)
C
       WRITE(6,2)
     2 FORMAT(1H-,31X 'P  E  R  T',//,
      C6X,'I',9X,'J',10X,'A',9X,'M',9X,'B',6X,'T(I,J)    VT(I,J)'/)
       DO 3 I=1,N
       C(I,6)=(C(I,3)+4.*C(I,4)+C(I,5))/6.
       C(I,7)=(C(I,5)**2 - 2*C(I,5)*C(I,3) + C(I,3)**2)/36.
     3 WRITE(6,4) (C(I,J),J=1,7)
     4 FORMAT(7F10.3)
C
C      CALCALATE EARLIEST EXPECTED COMPLETION TIME FOR EVENT J - ET(J)
C
       ET(1)=0.
       SIGTE(1)=0.
       DO 6 J=2,M
       TMAX=0.
       SIGMAX=0.
       DO 5 K=1,N
       IF(C(K,2).EQ.J)THEN
            ET(J)=ET(C(K,1))+C(K,6)
            SIGTE(J)=SIGTE(C(K,1))+C(K,7)
            IF(ET(J).GT.TMAX)THEN
                 TMAX=ET(J)
                 SIGMAX=SIGTE(J)
            END IF
       END IF
     5 CONTINUE
       ET(J)=TMAX
       SIGTE(J)=SIGMAX
     6 CONTINUE
C
C      CALCULATE LATEST ALLOWABLE COMPLETION TIME FOR EVENT J - LT(J)
C
       LT(M)=ET(M)
       SD(M)=LT(M)
       DO 8 I=M-1,1,-1
       TMIN=BIG M
       DO 7 K=N,1,-1
       IF(C(K,1).EQ.I)THEN
            LT(I)=LT(C(K,2))-C(K,6)
            IF(LT(I).LT.TMIN)TMIN=LT(I)
       END IF
     7 CONTINUE
       LT(I)=TMIN
     8 CONTINUE
C
C       CALCULATE SLACK TIME FOR EACH EVENT J - SL(J)
C
       WRITE(6,9)
     9 FORMAT(//,19X,'EARLIEST',9X,'LATEST',
      C        /,7X,'EVENT',11X,'TIME',11X,'TIME',10X,'SLACK')
       DO 10 J=1,M
```

```
      SL(J)=LT(J)-ET(J)
   10 WRITE(6,11) J,ET(J),LT(J),SL(J)
   11 FORMAT(/7X,I5,3F15.3)
C
C     FIND ACTIVITIES ON CRITICAL PATH
C
      DO 12 K=1,N
      I=C(K,1)
      J=C(K,2)
      IF(ABS(LT(I)-ET(I)-SL(M)).LE.ERROR)THEN
      IF(ABS(LT(J)-ET(J)-SL(M)).LE.ERROR)THEN
      IF(ABS(ET(J)-ET(I)-LT(J)+LT(I)).LE.ERROR.AND.
     C   ABS(LT(J)-LT(I)-C(K,6)).LE.ERROR)     FLAG(K)=1
      END IF
      END IF
   12 CONTINUE
      CPATH(1)=1
      KP=1
      DO 14 K=1,N
      I=C(K,1)
      J=C(K,2)
      IF(FLAG(K).NE.0)THEN
      IF(I.EQ.CPATH(KP))THEN
            KP=KP+1
            CPATH(KP)=J
      END IF
      END IF
   14 CONTINUE
      WRITE(6,15) (CPATH(I),I=1,KP)
   15 FORMAT(//5X,'CRITICAL PATH',1514)
C
C     CALCULATE CRITICAL VALUE OF CUMULATIVE N(0,1) DISTRIBUTION
C     TO DETERMINE PROBABILITY OF COMPLETING EVENT BEFORE SCHEDULED
C     COMPLETION TIME
C
C     MDNOR - NORMAL PROBABILITY DISTRIBUTION FUNCTION (IMSL LIBRARY)
C
      READ(5,*,END=999) (SD(I),I=1,M)
      WRITE(6,16)
   16 FORMAT(//19X,'EARLIEST',12X,'DUE',9X,'PROBA-',
     C        /7X,'EVENT',11X,'TIME',11X,'DATE',9X,'BILITY')
      DO 17 K=2,M
      X=(SD(K)-ET(K))/SQRT(SIGTE(K))
      CALL MDNOR(X,PROB)
   17 WRITE(6,11) K,ET(K),SD(K),PROB
  999 RETURN
      END
```

Appendix Y

SUBROUTINE VAM1

```
      SUBROUTINE VAM1(M,N,ALLOC,COSTS,FACTRY,WAREHO,VAMROW,VAMCOL)
      IMPLICIT INTEGER(A-Z)
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
      INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),
     C        VAMROW(M),VAMCOL(N)
      INTEGER HIGH/2147483647/
      REAL TOTAL
C
C     MODIFIED V.A.M. METHOD:
C
C     CALCULATE THE VAM NUMBERS BY TAKING THE DIFFERENCE BETWEEN THE
C     CHEAPEST ROUTE AND THE SECOND CHEAPEST ROUTE.
C     PICK UP THE HIGHEST VAM NUMBER AMONG THE ROWS AND COLUMNS
C     FILL UP THAT ROW OR COLUMN FROM THE CHEAPEST COST.
C     IF A TIE IS FOUND ON THE VAM NUMBERS, TEST THOSE ROWS/COLS
C     WITH THE BECTOR'S NUMBER, TAKE THE ROUTE WITH THE HIGHEST
C     BECTOR NUMBER.
C
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      FACTRY(I)=SUPPLY(I)
      DO 1 J=1,N
      COSTS(I,J)=COST(I,J)
    1 ALLOC(I,J)=0
      DO 2 J=1,N
    2 WAREHO(J)=DEMAND(J)
C
C     GENERATE THE VAM NUMBERS FOR ROWS
C
    3 DO 6 I=1,M
      IF(FACTRY(I).EQ.0)VAMROW(I)=0
      IF(FACTRY(I).EQ.0)GOTO 6
      LOWEST=MINO(COSTS(I,1),COSTS(I,2))
      LOWER=MAXO(COSTS(I,1),COSTS(I,2))
      DO 5 J=3,N
      IF(COSTS(I,J).GE.LOWEST)GOTO 4
      LOWER=LOWEST
      LOWEST=COSTS(I,J)
      GOTO 5
    4 IF(COSTS(I,J).GE.LOWER)GOTO 5
      LOWER=COSTS(I,J)
```

```
    5 CONTINUE
      IF (LOWER.EQ.HIGH) LOWER=0
      IF (LOWEST.EQ.HIGH) LOWEST=0
      VAMROW(I)=IABS(LOWER-LOWEST)
    6 CONTINUE
C
C     GENERATE THE VAM NUMBERS FOR COLUMNS
C
      DO 9 J=1,N
      IF (WAREHO(J) .EQ.0) VAMCOL(J)=0
      IF (WAREHO(J) .EQ.0) GOTO 9
      LOWEST=MINO(COSTS(1,J),COSTS(2,J))
      LOWER=MAXO(COSTS(1,J),COSTS(2,J))
      DO 8 I=3,M
      IF (COSTS(I,J) .GE.LOWEST) GOTO 7
      LOWER=LOWEST
      LOWEST=COSTS(I,J)
      GOTO 8
    7 IF (COSTS(I,J) .GE.LOWER) GOTO 8
      LOWER=COSTS(I,J)
    8 CONTINUE
      IF (LOWER.EQ.HIGH) LOWER=0
      IF (LOWEST.EQ.HIGH) LOWEST=0
      VAMCOL(J)=IABS(LOWER-LOWEST)
    9 CONTINUE
C
C     FIND THE HIGHEST VAM #   FROM VAMROW & VAMCOL INTO HIGH1 & HIGH2
C
      HIGH1=VAMROW(1)
      DO 10 I=2,M
      IF (VAMROW(I) .GT.HIGH1) HIGH1=VAMROW(I)
   10 CONTINUE
      HIGH2=VAMCOL(1)
      DO 11 J=2,N
      IF (VAMCOL(J) .GT.HIGH2) HIGH2=VAMCOL(J)
   11 CONTINUE
C
C     FIND THE HIGHEST FROM HIGH1 & HIGH2
C
      TOPVAM=MAXO(HIGH1,HIGH2)
      ROW=0
      COL=0
C
C     THE HIGHEST VAM #  IS UNIQUE ?
C
      VAMCT=0
      DO 12 I=1,M
      IF (VAMROW(I) .LT.TOPVAM) GOTO 12
      VAMCT=VAMCT+1
      ROW=I
   12 CONTINUE
      IF (VAMCT.GT.1) GOTO 20
      DO 13 J=1,N
      IF (VAMCOL(J) .LT.TOPVAM) GOTO 13
```

```
      VAMCT=VAMCT+1
      COL=J
   13 CONTINUE
      IF (VAMCT.EQ.1) GOTO 60
C
C     CHECK IF THERE ARE VAM NUMBERS THAT MAKE A TIE
C     USE BECTOR NUMBERS TO DETERMINE WHICH ROW OR COL
C
   20 TOPBEC=0
      ROW=0
      COL=0
      IF (TOPVAM.GT.HIGH1) GOTO 50
C
C     HIGHEST VAM NUMBER FOUND ON ROWS
C
      DO 41 I=1,M
      IF (VAMROW(I) .LT.TOPVAM) GOTO 41
      BECTOR=RANGE (M,N,I,0) *FACTRY (I)
      IF (BECTOR.LT.TOPBEC) GOTO 41
      TOPBEC=BECTOR
      ROW=I
   41 CONTINUE
C
C     HIGHEST VAM NUMBER FOUND ON COLS
C
   50 IF (TOPVAM.GT.HIGH2) GOTO 60
      DO 51 J=1,N
      IF (VAMCOL (J) .LT.TOPVAM) GOTO 51
      BECTOR=RANGE (M,N,0,J) *WAREHO (J)
      IF (BECTOR.LT.TOPBEC) GOTO 51
      TOPBEC=BECTOR
      COL=J
      ROW=0
   51 CONTINUE
C
C     DETERMINE THE HIGHEST BECTOR NUMBER IS ON ROW OR COL
C
   60 IF (COL.NE.0) GOTO 70
C
C     HIGHEST BECTOR NUMBER IS ON ROW
C     THEN PICKUP THE LOWEST ENTRY ON THAT ROW
C
   61 LOW=COSTS (ROW,1)
      COL=1
      DO 62 J=2,N
      IF (COSTS (ROW,J) .GE.LOW) GOTO 62
      LOW=COSTS (ROW,J)
      COL=J
   62 CONTINUE
      COSTS (ROW,COL) =HIGH
      ALLOC (ROW,COL) =MINO (FACTRY (ROW) ,WAREHO (COL) )
      FACTRY (ROW) =FACTRY (ROW) -ALLOC (ROW,COL)
      WAREHO (COL) =WAREHO (COL) -ALLOC (ROW,COL)
      IF (FACTRY (ROW) .GT.0) GOTO 61
```

```
      GOTO 95
C
C     HIGHEST BECTOR NUMBER IS ON COL
C     PICKUP THE LOWEST ENTRY ON THE COL
C
   70 LOW=COSTS(1,COL)
      ROW=1
      DO 71 I=2,M
      IF(COSTS(I,COL).GE.LOW)GOTO 71
      LOW=COSTS(I,COL)
      ROW=I
   71 CONTINUE
      COSTS(ROW,COL)=HIGH
      ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))
      FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)
      WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)
      IF(WAREHO(COL).GT.0)GOTO 70
      GOTO 95
   80 COSTS(ROW,COL)=HIGH
      ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))
      FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)
      WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)
C
C     CHECK TO SEE IF ALL CONDITIONS FOR DEMAND & SUPPLY ARE FULFILLED
C
   95 DO 96 I=1,M
   96 IF(FACTRY(I).GT.0)GOTO 3
      DO 97 J=1,N
   97 IF(WAREHO(J).GT.0)GOTO 3
C
C     CALCULATE THE TOTAL COST FOR THE B.F.S IN VAM ROUTE
C
      TOTAL=0.
      DO 100 I=1,M
      DO 100 J=1,N
  100 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))
      WRITE(6,101)TOTAL
  101 FORMAT('0','TOTAL COST IN  MODIFIED V.A.M. METHOD = ',E16.10)
C
C     OUTPUT THE ALLOCATION MATRIX
C
      DO 98 I=1,M
   98 WRITE(6,99)(ALLOC(I,J),J=1,N)
   99 FORMAT(1X,3314)
      RETURN
      END
```

Appendix Z

SUBROUTINE VAM2

```
      SUBROUTINE VAM2(M,N,ALLOC,COSTS,FACTRY,WAREHO,VAMROW,VAMCOL)
      IMPLICIT INTEGER(A-Z)
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
      INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),
     C        VAMROW(M),VAMCOL(N)
      INTEGER HIGH/2147483647/
      REAL TOTAL,ZERO/0./
C
C     CONVENTIONAL V.A.M. METHOD:
C
C     CALCULATE THE VAM NUMBERS BY TAKING THE DIFFERENCE BETWEEN THE
C     CHEAPEST ROUTE AND THE SECOND CHEAPEST ROUTE.
C     PICK UP THE HIGHEST VAM NUMBER AMONG THE ROWS AND COLUMNS
C     FILL UP THAT ROW OR COLUMN FROM THE CHEAPEST COST.
C     IF A TIE IS FOUND ON THE VAM NUMBERS, TEST THOSE ROWS/COLS
C     WITH THE CHEAPEST ROUTE, TAKE THE ROUTE WITH THE LOWEST COST.
C
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      FACTRY(I)=SUPPLY(I)
      DO 1 J=1,N
      COSTS(I,J)=COST(I,J)
    1 ALLOC(I,J)=0
      DO 2 J=1,N
    2 WAREHO(J)=DEMAND(J)
C
C     GENERATE THE VAM NUMBERS FOR ROWS
C
    3 DO 6 I=1,M
      IF(FACTRY(I).EQ.0)VAMROW(I)=0
      IF(FACTRY(I).EQ.0)GOTO 6
      LOWEST=MINO(COSTS(I,1),COSTS(I,2))
      LOWER=MAXO(COSTS(I,1),COSTS(I,2))
      DO 5 J=3,N
      IF(COSTS(I,J).GE.LOWEST)GOTO 4
      LOWER=LOWEST
      LOWEST=COSTS(I,J)
      GOTO 5
    4 IF(COSTS(I,J).GE.LOWER)GOTO 5
      LOWER=COSTS(I,J)
    5 CONTINUE
```

```
      IF (LOWER.EQ.HIGH) LOWER=0
      IF (LOWEST.EQ.HIGH) LOWEST=0
      VAMROW(I)=IABS(LOWER-LOWEST)
    6 CONTINUE
C
C     GENERATE THE VAM NUMBERS FOR COLUMNS
C
      DO 9 J=1,N
      IF (WAREHO(J).EQ.0) VAMCOL(J)=0
      IF (WAREHO(J).EQ.0) GOTO 9
      LOWEST=MINO(COSTS(1,J),COSTS(2,J))
      LOWER=MAXO(COSTS(1,J),COSTS(2,J))
      DO 8 I=3,M
      IF (COSTS(I,J).GE.LOWEST) GOTO 7
      LOWER=LOWEST
      LOWEST=COSTS(I,J)
      GOTO 8
    7 IF (COSTS(I,J).GE.LOWER) GOTO 8
      LOWER=COSTS(I,J)
    8 CONTINUE
      IF (LOWER.EQ.HIGH) LOWER=0
      IF (LOWEST.EQ.HIGH) LOWEST=0
      VAMCOL(J)=IABS(LOWER-LOWEST)
    9 CONTINUE
C
C     FIND THE HIGHEST VAM #   FROM VAMROW & VAMCOL INTO HIGH1 & HIGH2
C
      HIGH1=VAMROW(1)
      DO 10 I=2,M
      IF (VAMROW(I).GT.HIGH1) HIGH1=VAMROW(I)
   10 CONTINUE
      HIGH2=VAMCOL(1)
      DO 11 J=2,N
      IF (VAMCOL(J).GT.HIGH2) HIGH2=VAMCOL(J)
   11 CONTINUE
C
C     FIND THE HIGHEST FROM HIGH1 & HIGH2
C
      TOPVAM=MAXO(HIGH1,HIGH2)
      ROW=0
      COL=0
C
C     THE HIGHEST VAM #  IS UNIQUE ?
C
      VAMCT=0
      DO 12 I=1,M
      IF (VAMROW(I).LT.TOPVAM) GOTO 12
      VAMCT=VAMCT+1
      ROW=I
   12 CONTINUE
      IF (VAMCT.GT.1) GOTO 20
      DO 13 J=1,N
      IF (VAMCOL(J).LT.TOPVAM) GOTO 13
      VAMCT=VAMCT+1
```

```
      COL=J
   13 CONTINUE
      IF (VAMCT.EQ.1) GOTO 60
C
C     CHECK IF THERE ARE VAM NUMBERS THAT MAKE A TIE
C     USE CHEAPEST ROUTE TO DETERMINE WHICH ROW OR COL
C
   20 TOPCHP=HIGH
      ROW=0
      COL=0
      IF (TOPVAM.GT.HIGH1) GOTO 50
C
C     HIGHEST VAM NUMBER FOUND ON ROWS
C
      DO 41 I=1,M
      IF (VAMROW(I) .LT.TOPVAM) GOTO 41
      CHEAP=CHEAPT(M,N,COSTS,ALLOC,FACTRY,WAREHO,I,0)
      IF (CHEAP.GE.TOPCHP) GOTO 41
      TOPCHP=CHEAP
      ROW=I
   41 CONTINUE
C
C     HIGHEST VAM NUMBER FOUND ON COLS
C
   50 IF (TOPVAM.GT.HIGH2) GOTO 60
      DO 51 J=1,N
      IF (VAMCOL(J) .LT.TOPVAM) GOTO 51
      CHEAP=CHEAPT(M,N,COSTS,ALLOC,FACTRY,WAREHO,0,J)
      IF (CHEAP.GE.TOPCHP) GOTO 51
      TOPCHP=CHEAP
      COL=J
      ROW=0
   51 CONTINUE
C
C     HIGHEST VAM NUMBER WITH CHEAPEST ROUTE IS ON ROW OR COL ?
C
   60 IF (COL.NE.0) GOTO 70
C
C     HIGHEST VAM NUMBER ON CHEAPEST ROUTE IS ON ROW
C     THEN PICKUP THE LOWEST ENTRY ON THAT ROW
C
   61 LOW=COSTS(ROW,1)
      COL=1
      DO 62 J=2,N
      IF (COSTS(ROW,J) .GE.LOW) GOTO 62
      LOW=COSTS(ROW,J)
      COL=J
   62 CONTINUE
      GOTO 80
C
C     HIGHEST VAM NUMBER ON CHEAPEST ROUTE IS ON COL
C     PICKUP THE LOWEST ENTRY ON THE COL
C
   70 LOW=COSTS(1,COL)
```

```
      ROW=1
      DO 71 I=2,M
      IF (COSTS(I,COL).GE.LOW) GOTO 71
      LOW=COSTS(I,COL)
      ROW=I
   71 CONTINUE
   80 COSTS(ROW,COL)=HIGH
      ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))
      FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)
      WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)
C
C     CHECK TO SEE IF ALL CONDITIONS FOR DEMAND & SUPPLY ARE FULFILLED
C
      DO 96 I=1,M
   96 IF (FACTRY(I).GT.0) GOTO 3
      DO 97 J=1,N
   97 IF (WAREHO(J).GT.0) GOTO 3
C
C     CALCULATE THE TOTAL COST FOR THE B.F.S IN VAM ROUTE
C
      TOTAL=ZERO
      DO 100 I=1,M
      DO 100 J=1,N
  100 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))
      WRITE(6,101) TOTAL
  101 FORMAT('0','TOTAL COST IN  CONVENTIONAL V.A.M. METHOD = ',E16.10)
C
C     OUTPUT THE ALLOCATION MATRIX
C
      DO 98 I=1,M
   98 WRITE(6,99) (ALLOC(I,J),J=1,N)
   99 FORMAT(1X,3314)
      RETURN
      END
      INTEGER FUNCTION CHEAPT(M,N,COSTS,ALLOC,FACTRY,WAREHO,IROW,ICOL)
      IMPLICIT INTEGER(A-Z)
      COMMON COST(100,100),SUPPLY(100),DEMAND(100)
      INTEGER COSTS(M,N),ALLOC(M,N),FACTRY(M),WAREHO(N),
     C        C(100,100),A(100,100),F(100),W(100)
      INTEGER HIGH/2147483647/
C
C     PRESERVE THE ORIGINAL PARAMETERS
C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO
C
      DO 1 I=1,M
      F(I)=FACTRY(I)
      DO 1 J=1,N
      C(I,J)=COSTS(I,J)
    1 A(I,J)=ALLOC(I,J)
      DO 2 J=1,N
    2 W(J)=WAREHO(J)
C
      ROW=IROW
      COL=ICOL
```

```
C
C      FILL UP THE CHEAPEST CELL ON ROW/COL
C
       IF(COL.NE.0)GOTO 60
       LOW=C(ROW,1)
       COL=1
       DO 50 J=2,N
       IF(C(ROW,J).GE.LOW)GOTO 50
       LOW=C(ROW,J)
       COL=J
   50  CONTINUE
       GOTO 70
   60  LOW=C(1,COL)
       ROW=1
       DO 61 I=2,M
       IF(C(I,COL).GE.LOW)GOTO 61
       LOW=C(I,COL)
       ROW=I
   61  CONTINUE
       GOTO 70
C
C      FIND THE LOWEST COST ON COST MAP "C"
C
    3  LOWEST=C(1,1)
       ROW=1
       COL=1
       DO 4 I=1,M
       DO 4 J=1,N
       IF(C(I,J).GE.LOWEST)GOTO 4
       LOWEST=C(I,J)
       ROW=I
       COL=J
    4  CONTINUE
C
C      THE CHEAPEST ROUTE IS ASSIGNED WITH A HIGH VALUE NUMBER
C      AFTER BEING USED
C
   70  C(ROW,COL)=HIGH
       A(ROW,COL)=MINO(F(ROW),W(COL))
       F(ROW)=F(ROW)-A(ROW,COL)
       W(COL)=W(COL)-A(ROW,COL)
C
C      FIND THE NEXT CHEAPEST ROUTE IF THE REQUIREMENT OF WAREHOUSES
C      AND FACTORIES ARE NOT MET
C
       DO 5 I=1,M
    5  IF(F(I).GT.0)GOTO 3
       DO 6 J=1,N
    6  IF(W(J).GT.0)GOTO 3
       CHEAPT=0
       DO 20 I=1,M
       DO 20 J=1,N
   20  CHEAPT=CHEAPT+COST(I,J)*A(I,J)
       RETURN
```

```
END
```

Appendix AA

SAMPLE INPUT / OUTPUT

Input:

```
NJOB2M
8
4. 6.
8. 3.
7. 6.
8. 4.
2. 6.
1. 5.
3. 7.
9. 2.
```

Output:

```
 SEQUENCING   8 JOBS THROUGH 2 MACHINES


 JOB    PROCESSING TIME PROCESSING TIME

            MACHINE 1       MACHINE 2

  1            4.00            6.00
  2            8.00            3.00
  3            7.00            6.00
  4            8.00            4.00
  5            2.00            6.00
  6            1.00            5.00
  7            3.00            7.00
  8            9.00            2.00


  OPTIMAL SEQUENCE FOLLOWS:
    6   5   7   1   3   4   2   8

  TOTAL ELAPSED TIME     =     44.00

  IDLE TIME ON MACHINE 1 =      2.00

  IDLE TIME ON MACHINE 2 =      5.00
```

Input:

```
NJOB3M
5
4. 5. 5.
2. 2. 6.
8. 3. 8.
10. 3. 9.
5. 4. 7.
```

Output:

```
 SEQUENCING   5 JOBS THROUGH 3 MACHINES

 JOB    PROCESSING TIME PROCESSING TIME

           MACHINE 1       MACHINE 2       MACHINE 3

  1          4.00            5.00            5.00
  2          2.00            2.00            6.00
  3          8.00            3.00            8.00
  4         10.00            3.00            9.00
  5          5.00            4.00            7.00


  OPTIMAL SEQUENCE FOLLOWS:
    2   1   5   3   4

  TOTAL ELAPSED TIME     =     41.00

  IDLE TIME ON MACHINE 1 =     12.00

  IDLE TIME ON MACHINE 2 =     24.00

  IDLE TIME ON MACHINE 3 =      6.00
```

	MACHINE 1		MACHINE 2		MACHINE 3	
JOB	TIME IN	TIME OUT	TIME IN	TIME OUT	TIME IN	TIME OUT
2	0.00	2.00	2.00	4.00	4.00	10.00
1	2.00	6.00	6.00	11.00	11.00	16.00
5	6.00	11.00	11.00	15.00	16.00	23.00
3	11.00	19.00	19.00	22.00	23.00	31.00
4	19.00	29.00	29.00	32.00	32.00	41.00

Input:

```
MINSCT
8
7 2 3 4 9 11 5 4
```

Output:

```
 1 MACHINE       8 JOBS

 PROCESSING TIMES FOR EACH JOB:
    7.  2.  3.  4.  9. 11.  5.  4.


 OPTIMAL SEQUENCE TO MINIMIZE SUM OF COMPLETION TIMES
   2   3   4   8   7   1   5   6
```

Input:

```
MINWTS
6
2 4 7 6 3 2
4 6 2 1 5 3
```

Output:

```
 1 MACHINE       6 JOBS

 PROCESSING TIMES FOR EACH JOB:
    2.  4.  7.  6.  3.  2.

 WEIGHTS FOR EACH JOB:
    4.  6.  2.  1.  5.  3.

 OPTIMAL SEQUENCE TO MINIMIZE WEIGHTED SUM OF COMPLETION TIMES
   1   5   2   6   3   4
```

Input:

```
MINTRD
7
21 18 15 2 7 9 25
```

Output:

```
 1 MACHINE       7 JOBS

 DUE DATES FROM NOW FOR EACH JOB:
   21. 18. 15.  2.  7.  9. 25.

 OPTIMAL SEQUENCE TO MINIMIZE MAXIMUM TARDINESS
   4   5   6   3   2   1   7
```

Input:

```
MAXTRD
7
6 8 4 7 21 11 13
9 13 16 22 35 40 50
```

Output:

```
 1 MACHINE       7 JOBS

 PROCESSING TIMES FOR EACH JOB:
    6.  8.  4.  7. 21. 11. 13.

 DUE DATES FROM NOW FOR EACH JOB:
    9. 13. 16. 22. 35. 40. 50.

 OPTIMUM SEQUENCE TO MAXIMIZE MINIMUM LATENESS
   1   2   3   5   4   6   7
```

Input:

```
MINLAT
6
6 8 4 7 21 11
9 13 16 22 35 40
```

Output:

```
 1 MACHINE       6 JOBS

 PROCESSING TIMES FOR EACH JOB:
    6.  8.  4.  7. 21. 11.

 DUE DATES FROM NOW FOR EACH JOB:
    9. 13. 16. 22. 35. 40.

 LATE JOB IS   2
 LATE JOB IS   5

 OPTIMUM SEQUENCE TO MINIMIZE NUMBER OF LATE JOBS
   1   3   4   6   2   5

   4 JOBS ON TIME,   2 JOBS LATE
```

Input:

```
GUPTA
4 5
4 3 7 2 8
3 7 2 8 5
1 2 4 3 7
3 4 3 7 2
```

Output:

```
   5 MACHINES       4 JOBS
 PROCESSING TIME FOR EACH JOB ON PROCESS

 JOB   1:  4.  3.  7.  2.  8.
 JOB   2:  3.  7.  2.  8.  5.
 JOB   3:  1.  2.  4.  3.  7.
 JOB   4:  3.  4.  3.  7.  2.


 OPTIMUM SEQUENCE TO MINIMIZE TOTAL PROCESSING TIME IS:
   3   1   2   4
```

Input:

```
GUPTA
6 2
9 1
8 3
5 4
7 11
6 8
2 9
```

Output:

```
   2 MACHINES       6 JOBS

 PROCESSING TIME FOR EACH JOB ON PROCESS

 JOB    1:  9.  1.
 JOB    2:  8.  3.
 JOB    3:  5.  4.
 JOB    4:  7. 11.
 JOB    5:  6.  8.
 JOB    6:  2.  9.

 OPTIMUM SEQUENCE TO MINIMIZE TOTAL PROCESSING TIME IS:
   6   5   4   3   2   1
```

Input:

```
CAMPBL
9
3 20 0
1 70 40
1 30 60
4 0 70
1 10 30
2 30 10
2 20 70
4 0 40
3 40 0
```

Output:

```
 2 MACHINES        9 JOBS

 MACHINE        PROCESSING TIME
  ORDER          FOR EACH JOB
 --------       ---------------
 ONE ONLY            20.   0.
 ONE TWO             70.  40.
 ONE TWO             30.  60.
 TWO ONLY             0.  70.
 ONE TWO             10.  30.
 TWO ONE             30.  10.
 TWO ONE             20.  70.
 TWO ONLY             0.  40.
 ONE ONLY            40.   0.

 OPTIMAL ORDER ON MACHINE 1 IS:
   5   3   2   1   9   6   7

 OPTIMAL ORDER ON MACHINE 2 IS:
   6   7   4   8   5   3   2
```

Input:

```
GUPTA
4 5
4 3 7 2 8
3 7 2 8 5
1 2 4 3 7
3 4 3 7 2
```

Output:

```
   5 MACHINES        4 JOBS
 PROCESSING TIME FOR EACH JOB ON PROCESS

 JOB    1:  4.  3.  7.  2.  8.
 JOB    2:  3.  7.  2.  8.  5.
 JOB    3:  1.  2.  4.  3.  7.
 JOB    4:  3.  4.  3.  7.  2.


 OPTIMUM SEQUENCE TO MINIMIZE TOTAL PROCESSING TIME IS:
   3   1   2   4
```

Input:

```
HUNGAR
8 0
300 290 280 290 210 0 0 0
250 310 290 300 200 0 0 0
180 190 300 190 180 0 0 0
320 180 190 240 170 0 0 0
270 210 190 250 160 0 0 0
190 200 220 190 140 0 0 0
220 300 230 180 160 0 0 0
260 190 260 210 180 0 0 0
```

Output:

```
 ORIGINAL MATRIX:
      300      290      280      290      210        0        0        0
      250      310      290      300      200        0        0        0
      180      190      300      190      180        0        0        0
      320      180      190      240      170        0        0        0
      270      210      190      250      160        0        0        0
      190      200      220      190      140        0        0        0
      220      300      230      180      160        0        0        0
      260      190      260      210      180        0        0        0

THE OPTIMAL SOLUTION OF THE ASSIGNMENT PROBLEM'S TOTAL COST =   870

          THE ASSIGNMENT FOR   1 =        6
          THE ASSIGNMENT FOR   2 =        7
          THE ASSIGNMENT FOR   3 =        1
          THE ASSIGNMENT FOR   4 =        2
          THE ASSIGNMENT FOR   5 =        3
          THE ASSIGNMENT FOR   6 =        5
          THE ASSIGNMENT FOR   7 =        4
          THE ASSIGNMENT FOR   8 =        8
```

Input:

```
EASTMN
10
1000000 184 292 449 670 516 598 618 881 909
184 1000000 195 310 540 357 514 434 697 964
292 195 1000000 215 380 232 434 493 719 955
449 310 215 1000000 288 200 566 787 790 1020
670 540 380 288 1000000 211 436 814 632 974
516 357 232 200 211 1000000 381 642 697 952
598 514 434 566 436 381 1000000 295 224 541
618 434 493 787 814 642 295 1000000 320 341
881 697 719 790 632 697 224 320 1000000 318
909 964 955 1020 974 952 541 341 318 1000000
```

Output:

```
THE DISTANCE MATRIX:

 1000000      184      292      449      670      516      598      618      881      909
     184  1000000      195      310      540      357      514      434      697      964
     292      195  1000000      215      380      232      434      493      719      955
     449      310      215  1000000      288      200      566      787      790     1020
     670      540      380      288  1000000      211      436      814      632      974
     516      357      232      200      211  1000000      381      642      697      952
     598      514      434      566      436      381  1000000      295      224      541
     618      434      493      787      814      642      295  1000000      320      341
     881      697      719      790      632      697      224      320  1000000      318
     909      964      955     1020      974      952      541      341      318  1000000

 THE OPTIMAL FEASIBLE SOLUTION DISTANCE IS       2855.

 THE ASSIGNMENT FOR   1 IS           2.
 THE ASSIGNMENT FOR   2 IS           8.
 THE ASSIGNMENT FOR   3 IS           1.
 THE ASSIGNMENT FOR   4 IS           3.
 THE ASSIGNMENT FOR   5 IS           6.
 THE ASSIGNMENT FOR   6 IS           4.
 THE ASSIGNMENT FOR   7 IS           5.
 THE ASSIGNMENT FOR   8 IS          10.
 THE ASSIGNMENT FOR   9 IS           7.
 THE ASSIGNMENT FOR  10 IS           9.
```

Input:

```
PERT
7 10
1. 2. 2.2 2.45 6.
1. 3. 1.51 6.12 10.
1. 4. 5. 9.9 15.39
2. 5. 6. 7.5 12.
3. 4. 2. 5. 8.
3. 6. 5.51 10.12 14.
4. 5. 1.76 4.06 6.
4. 7. 4. 8.16 11.35
5. 7. 5. 10.38 13.49
6. 7. 2.51 5.62 11.
0 4 6 10 16 18 0
```

Output:

P E R T

I	J	A	M	B	T(I,J)	VT(I,J)
1.000	2.000	2.200	2.450	6.000	3.000	0.401
1.000	3.000	1.510	6.120	10.000	5.998	2.002
1.000	4.000	5.000	9.900	15.390	9.998	2.999
2.000	5.000	6.000	7.500	12.000	8.000	1.000
3.000	4.000	2.000	5.000	8.000	5.000	1.000
3.000	6.000	5.510	10.120	14.000	9.998	2.002
4.000	5.000	1.760	4.060	6.000	4.000	0.499
4.000	7.000	4.000	8.160	11.350	7.998	1.501
5.000	7.000	5.000	10.380	13.490	10.002	2.002
6.000	7.000	2.510	5.620	11.000	5.998	2.002

EVENT	EARLIEST TIME	LATEST TIME	SLACK
1	0.000	-0.000	-0.000
2	3.000	6.998	3.998
3	5.998	5.998	-0.000
4	10.998	10.998	-0.000
5	14.998	14.998	-0.000
6	15.997	19.002	3.005
7	25.000	25.000	0.000

CRITICAL PATH 1 3 4 5 7

EVENT	EARLIEST TIME	DUE DATE	PROBA-BILITY
2	3.000	4.000	0.943
3	5.998	6.000	0.500
4	10.998	10.000	0.282
5	14.998	16.000	0.704
6	15.997	18.000	0.842
7	25.000	25.000	0.500

Appendix AB

PROGRAM LISTING OF FORUM

```
REQUESTED OPTIONS (EXECUTE): ,,,,OBJ,,NOMAP,,GOSTMT,NOTERM,,SIZE(1024K),S,NAME(MAIN),OPT(0),LC(654),FLAG(I),

OPTIONS IN EFFECT:   NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                       OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(654)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

                C
                C       F O R U M       M O N I T O R
                C
ISN          1          COMMON COST(100,100),SUPPLY(100),DEMAND(100)
ISN          2          INTEGER IAS(500,15),IA(100,100),IB(100,100),NARCS(500)
ISN          3          REAL BR(500),R(500),C(500),P(500),TD(500),SUBT(500)
ISN          4          DIMENSION A(100,100),PAB(100,3),PBA(100,3),
                       *W(100),IO(100),IAB(100),IBA(100),JAB(100),JBA(100),WT(100),
                       *ICHK(100),TI1(100),TI2(100),TI3(100),TO1(100),TO2(100),TO3(100)
ISN          5          INTEGER VO,VT,M,N,EDGES
ISN          6          CHARACTER*6 NAMES(27),KEY,BLANK
ISN          7          DATA NAMES(1)/'BECTR1'/,NAMES(2)/'BECTR2'/,NAMES(3)/'BECTR3'/,
                       C NAMES(4)/'BECTR4'/,NAMES(5)/'BECTR5'/,NAMES(6)/'BECTR6'/,
                       C NAMES(7)/'CAMPBL'/,NAMES(8)/'CHEAP '/,NAMES(9)/'DIJKST'/,
                       C NAMES(10)/'EASTMN'/,NAMES(11)/'FLOYD '/,NAMES(12)/'GUPTA '/,
                       C NAMES(13)/'HUNGAR'/,NAMES(14)/'JONSON'/,NAMES(15)/'MAXTRD'/,
                       C NAMES(16)/'MINLAT'/,NAMES(17)/'NETPTH'/,NAMES(18)/'MINSCT'/,
                       C NAMES(19)/'MINTRD'/,NAMES(20)/'MINWTS'/,NAMES(21)/'NORTHW'/,
                       C NAMES(22)/'NJOB2M'/,NAMES(23)/'NJOB3M'/,NAMES(24)/'PERT  '/,
                       C NAMES(25)/'SHORTP'/,NAMES(26)/'VAM1  '/,NAMES(27)/'VAM2  '/,
                       C BLANK/'      '/
                C
                C       NUMRTN  -  NUMBER OF SUBROUTINES IN CURRENT VERSION OF FORUM
                C
ISN          8          NUMRTN=27
                C
ISN          9        1 READ(5,2,END=999)KEY
ISN         10        2 FORMAT(A6)
ISN         11          IF(KEY.EQ.BLANK)GOTO 1
ISN         12          DO 3 INDEX=1,NUMRTN
ISN         13        3 IF(KEY.EQ.NAMES(INDEX))GOTO 5
ISN         14          WRITE(6,4)KEY
ISN         15        4 FORMAT(1H-,'** ERROR ** SUBROUTINE ',A6,' NOT FOUND')
ISN         16          GOTO 999
                C
                C       BRANCH TO THE CORRESPONDING SUBROUTINES
                C
ISN         17        5 CALL $TRTM(ELAPSE)
ISN         18          GOTO(100,200,300,400,500,600,700,800,900,1000,1100,1200,1300,1400,
                       C 1500,1600,1700,1800,1900,2000,2100,2200,2300,2400,2500,2600,2700)
                       C ,INDEX
ISN         19        6 CALL $TPTM(ELAPSE)
ISN         20          CPU=ELAPSE
ISN         21          WRITE(6,7)KEY,CPU
ISN         22        7 FORMAT('0CPU TIME FOR ',A6,'=',F8.6,' SECONDS')
ISN         23          GOTO 1
                C
                C       TRANSPORTATION ALGORITHM  -  WEIGHTED PENALTY METHOD 1
                C
ISN         24   100    IF(COST(1,2).EQ.0)THEN
ISN         25                READ(5,*)M,N
ISN         26                READ(5,*)(SUPPLY(I),I=1,M)
ISN         27                READ(5,*)(DEMAND(J),J=1,N)
ISN         28                READ(5,*)((COST(I,J),J=1,N),I=1,M)
ISN         29          END IF
ISN         30          CALL BECTR1(M,N,IA,IB,IAB,IBA,JAB,JBA)
ISN         31          GOTO 6



                C
                C       TRANSPORTATION ALGORITHM  -  WEIGHTED PENALTY METHOD 2
                C
ISN         32   200    IF(COST(1,2).EQ.0)THEN
ISN         33                READ(5,*)M,N
ISN         34                READ(5,*)(SUPPLY(I),I=1,M)
ISN         35                READ(5,*)(DEMAND(J),J=1,N)
ISN         36                READ(5,*)((COST(I,J),J=1,N),I=1,M)
ISN         37          END IF
ISN         38          CALL BECTR2(M,N,IA,IB,IAB,IBA,JAB,JBA)
ISN         39          GOTO 6
                C
                C       TRANSPORTATION ALGORITHM  -  WEIGHTED PENALTY METHOD 3
                C
ISN         40   300    IF(COST(1,2).EQ.0)THEN
ISN         41                READ(5,*)M,N
ISN         42                READ(5,*)(SUPPLY(I),I=1,M)
ISN         43                READ(5,*)(DEMAND(J),J=1,N)
ISN         44                READ(5,*)((COST(I,J),J=1,N),I=1,M)
ISN         45          END IF
ISN         46          CALL BECTR3(M,N,IA,IB,IAB,IBA,JAB,JBA)
ISN         47          GOTO 6
                C
                C       TRANSPORTATION ALGORITHM  -  WEIGHTED PENALTY METHOD 4
                C
ISN         48   400    IF(COST(1,2).EQ.0)THEN
ISN         49                READ(5,*)M,N
ISN         50                READ(5,*)(SUPPLY(I),I=1,M)
ISN         51                READ(5,*)(DEMAND(J),J=1,N)
ISN         52                READ(5,*)((COST(I,J),J=1,N),I=1,M)
ISN         53          END IF
ISN         54          CALL BECTR4(M,N,IA,IB,IAB,IBA,JAB,JBA)
ISN         55          GOTO 6
                C
                C       TRANSPORTATION ALGORITHM  -  WEIGHTED PENALTY METHOD  5
                C
ISN         56   500    IF(COST(1,2).EQ.0)THEN
ISN         57                READ(5,*)M,N
ISN         58                READ(5,*)(SUPPLY(I),I=1,M)
ISN         59                READ(5,*)(DEMAND(J),J=1,N)
ISN         60                READ(5,*)((COST(I,J),J=1,N),I=1,M)
ISN         61          END IF
ISN         62          CALL BECTR5(M,N,IA,IB,IAB,IBA,JAB,JBA)
ISN         63          GOTO 6
                C
                C
                C       TRANSPORTATION ALGORITHM  -  WEIGHTED PENALTY METHOD 6
                C
ISN         64   600    IF(COST(1,2).EQ.0)THEN
ISN         65                READ(5,*)M,N
ISN         66                READ(5,*)(SUPPLY(I),I=1,M)
ISN         67                READ(5,*)(DEMAND(J),J=1,N)
ISN         68                READ(5,*)((COST(I,J),J=1,N),I=1,M)
ISN         69          END IF
ISN         70          CALL BECTR6(M,N,IA,IB,IAB,IBA,JAB,JBA)
ISN         71          GOTO 6
                C
                C
                C       MINIMIZE TOTAL PROCESSING TIME
                C       N TECHNOLOGICALLY ORDERED JOBS ON 2 MACHINES
                C
ISN         72   700    READ(5,*)N
ISN         73          CALL CAMPBL(N,P,W,IAB,IBA,JAB,JBA,PAB,PBA,ICHK)
ISN         74          GOTO 6
                C
                C       TRANSPORTATION ALGORITHM  -  CHEAPEST ROUTE METHOD
```

```
                C
ISN        75   800   IF(COST(1,2).EQ.0)THEN
ISN        76               READ(5,*)M,N
ISN        77               READ(5,*)(SUPPLY(I),I=1,M)
ISN        78               READ(5,*)(DEMAND(J),J=1,N)
ISN        79               READ(5,*)((COST(I,J),J=1,N),I=1,M)
ISN        80         END IF
ISN        81         CALL CHEAP(M,N,IA,IAB,IBA,IB)
ISN        82         GOTO 6
                C
                C       DIJKSTRA'S ALGORITHM - SHORTEST PATH PROBLEM
                C
ISN        83   900   IF(IA(1,2).EQ.0)THEN
ISN        84               READ(5,*)N,V0,VT
ISN        85               READ(5,*)((IA(I,J),J=1,N),I=1,N)
ISN        86         END IF
ISN        87         CALL DIJKST(N,V0,VT,IA,IAB,IBA,I0,EDGES)
ISN        88         GOTO 6
                C
                C       TRAVELING SALESMAN PROBLEM  -  EASTMAN'S ALGORITHM
                C
ISN        89   1000  READ(5,*)N
ISN        90         CALL EASTMN(N,IAS,IA,IB,ICHK,I0,WT,BR,R,C,P,TD,NARCS,SUBT)
ISN        91         GOTO 6
                C
                C       FLOYD-WARSHALL ALGORITHM -  ALL SHORTEST PATH ALGORITHM
                C
ISN        92   1100  IF(IA(1,2).EQ.0)THEN
ISN        93               READ(5,*)N
ISN        94               READ(5,*)(((IA(I,J),J=1,N),I=1,N)
ISN        95         END IF
ISN        96         DO 1101 I=1,N
ISN        97         DO 1101 J=1,N
ISN        98   1101  IB(I,J)=IA(I,J)
ISN        99         CALL FLOYD(IB,N)
ISN       100         GOTO 6
                C
                C       MINIMIZE TOTAL PROCESSING TIME
                C       N JOBS ON M MACHINES (FLOWSHOP SCHEDULING)
                C
ISN       101   1200  READ(5,*)N,M
ISN       102         CALL GUPTA(N,M,W,I0,WT,A,PAB,ICHK)
ISN       103         GOTO 6
                C
                C     ASSIGNMENT PROBLEM - HUNGARIAN ALGORITHM
                C
ISN       104   1300  READ(5,*)N,KODE
ISN       105         CALL HUNGAR(N,IA,IB,IAB,IBA,JAB,JBA,ITOTAL,KODE,0)
ISN       106         GOTO 6
                C
                C       JOHNSON'S ALGORITHM
                C       (TO USE - THROUGH GUPTA'S ALGORITHM)
                C
ISN       107   1400  GOTO 6
                C
                C       MAXIMIZE THE MINIMUM TARDINESS ON ONE PROCESS (SLACK RULE)
                C
ISN       108   1500  W(1)=0.
ISN       109         P(1)=0.
ISN       110         READ(5,*)N
ISN       111         CALL MAXTRD(N,P,I0,W)
ISN       112         GOTO 6
                C
                C       MINIMIZE THE NUMBER OF LATE JOBS,
                C       N JOBS ON 1 MACHINE
                C



ISN       113   1600  W(1)=0.
ISN       114         P(1)=0.
ISN       115         READ(5,*)N
ISN       116         CALL MINLAT(N,P,I0,W)
ISN       117         GOTO 6
                C
                C       NICHOLSON'S ALOGRITHM  -  NETWORK PATH (NETPTH)
                C
ISN       118   1700  IF(IA(1,2).EQ.0)THEN
ISN       119               READ(5,*)N,V0
ISN       120               READ(5,*)((IA(I,J),J=1,N),I=1,N)
ISN       121         END IF
ISN       122         CALL NETPTH(IA,N,V0,IAB,WT)
ISN       123         GOTO 6
                C
                C       MINIMIZE SUM OF COMPLETION TIMES  OR  SUM OF WAITING TIMES
                C       ON ONE PROCESS
                C
ISN       124   1800  P(1)=0.
ISN       125         READ(5,*)N
ISN       126         CALL MINSCT(N,P,I0,W)
ISN       127         GOTO 6
                C
                C       MINIMIZE THE MAXIMUM TARDINESS ON ONE PROCESS (DUE DATE RULE)
                C
ISN       128   1900  P(1)=0.
ISN       129         W(1)=0.
ISN       130         READ(5,*)N
ISN       131         CALL MINTRD(N,P,I0,W)
ISN       132         GOTO 6
                C
                C       MINIMIZE SUM OF COMPLETION TIMES  OR  SUM OF WAITING TIMES
                C       ON ONE PROCESS
                C
ISN       133   2000  P(1)=0.
ISN       134         READ(5,*)N
ISN       135         CALL MINWTS(N,P,I0,W,WT)
ISN       136         GOTO 6
                C
                C       TRANSPORTATION ALGORITHM  -  NORTH-WEST CORNER METHOD
                C
ISN       137   2100  IF(COST(1,2).EQ.0)THEN
ISN       138               READ(5,*)M,N
ISN       139               READ(5,*)(SUPPLY(I),I=1,M)
ISN       140               READ(5,*)(DEMAND(J),J=1,N)
ISN       141               READ(5,*)((COST(I,J),J=1,N),I=1,M)
ISN       142         END IF
ISN       143         CALL NORTHW(M,N,IA,IAB,IBA)
ISN       144         GOTO 6
                C
                C       N - JOBS  /  2 - MACHINES
                C
ISN       145   2200  READ(5,*)N
ISN       146         CALL NJOB2M(N,PAB,PBA,TI1,TI2,TO1,TO2,ICHK)
ISN       147         GOTO 6
                C
                C       N - JOBS  /  3 - MACHINES
                C
ISN       148   2300  READ(5,*)N
ISN       149         CALL NJOB3M(N,PAB,PBA,TI1,TI2,TI3,TO1,TO2,TO3,ICHK)
ISN       150         GOTO 6
                C
                C       P E R T
                C
ISN       151   2400  READ(5,*)M,N
ISN       152         CALL PERT(M,N,A,I0,TI1,TI2,TI3,TO1,TO2,ICHK)
```

```
ISN       153          GOTO 6
ISN       154    2500  IF(IA(1,2).EQ.0)THEN
ISN       155              READ(5,*)N,VO,VT
ISN       156              READ(5,*)((IA(I,J),J=1,N),I=1,N)
ISN       157          END IF
ISN       158          CALL SHORTP(N,VO,VT,IAB,IBA,EDGES)
ISN       159          GOTO 6
                C
                C      TRANSPORTATION ALGORITHM  -  VAM'S WEIGHTED PENALTY METHOD 1
                C
ISN       160    2600  IF(COST(1,2).EQ.0)THEN
ISN       161              READ(5,*)M,N
ISN       162              READ(5,*)(SUPPLY(I),I=1,M)
ISN       163              READ(5,*)(DEMAND(J),J=1,N)
ISN       164              READ(5,*)((COST(I,J),J=1,N),I=1,M)
ISN       165          END IF
ISN       166          CALL VAM1(M,N,IA,IB,IAB,IBA,JAB,JBA)
ISN       167          GOTO 6
                C
                C      TRANSPORTATION ALGORITHM  -  VAM'S WEIGHTED PENALTY METHOD 2
                C
ISN       168    2700  IF(COST(1,2).EQ.0)THEN
ISN       169              READ(5,*)M,N
ISN       170              READ(5,*)(SUPPLY(I),I=1,M)
ISN       171              READ(5,*)(DEMAND(J),J=1,N)
ISN       172              READ(5,*)((COST(I,J),J=1,N),I=1,M)
ISN       173          END IF
ISN       174          CALL VAM2(M,N,IA,IB,IAB,IBA,JAB,JBA)
ISN       175          GOTO 6
                C
ISN       176     999  STOP
ISN       177          END
```

NUMBER	MODULE	LEVEL	ISN	VS FORTRAN ERROR MESSAGES
IFX0010I	OPTN	0(1)		THE SECTION " " OF THE OPTION LIST IS UNRECOGNIZABLE. CHECK OPTION SPECIFICATIONS.
IFX0010I	OPTN	0(1)		THE SECTION " " OF THE OPTION LIST IS UNRECOGNIZABLE. CHECK OPTION SPECIFICATIONS.
IFX0010I	OPTN	0(1)		THE SECTION "OBJ" OF THE OPTION LIST IS UNRECOGNIZABLE. CHECK OPTION SPECIFICATIONS.
IFX0010I	OPTN	0(1)		THE SECTION "MAP" OF THE OPTION LIST IS UNRECOGNIZABLE. CHECK OPTION SPECIFICATIONS.
IFX0010I	OPTN	0(1)		THE SECTION "TERM" OF THE OPTION LIST IS UNRECOGNIZABLE. CHECK OPTION SPECIFICATIONS.
IFX0010I	OPTN	0(1)		THE SECTION "SIZE" OF THE OPTION LIST IS UNRECOGNIZABLE. CHECK OPTION SPECIFICATIONS.
IFX0012I	OPTN	0(1)		THE OPTIONS "LANGLVL(77) AND NAME" CONFLICT WITH EACH OTHER. "NAME" IGNORED. CHECK OPTION SPECIFICATIONS.

```
*STATISTICS*   SOURCE STATEMENTS = 177, PROGRAM SIZE = 180474 BYTES, PROGRAM NAME = MAIN     PAGE:     1.

*STATISTICS* 7 DIAGNOSTICS GENERATED. HIGHEST SEVERITY CODE IS 0.

****** END OF COMPILATION 1 ******
```



```
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(54)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE MINSCT(N,P,IO,W)                                          10.
                C                                                                            20.
                C       MINIMIZE SUM OF COMPLETION TIMES OR SUM OF WAITING TIMES              30.
                C       ON ONE PROCESS                                                       40.
                C                                                                            50.
                C       ASSUME A BATCH OF JOBS WITH KNOWN OR ESTIMATED PROCESSING TIMES.     60.
                C       SEQUENCE JOBS IN SUCH A WAY THAT                                     70.
                C       1) THE SUM OF COMPLETION TIMES OR                                    80.
                C       2) THE SUM OF WAITING TIMES IS MINIMIZED.                            90.
                C                                                                           100.
                C       ORDER THE JOBS FROM LEFT TO RIGHT IN ORDER OF INCREASING            110.
                C       PROCESSING TIME. STOP WHEN THE OPTIMAL SEQUENCE HAS BEEN FOUND.     120.
                C                                                                           130.
                C                                                                           140.
ISN         2           DIMENSION P(N),IO(N),W(N)                                           150.
ISN         3           IF(P(1).EQ.0.)READ(5,*)(P(I),I=1,N)                                 160.
ISN         5           WRITE(6,1)N,(P(I),I=1,N)                                            170.
ISN         6         1 FORMAT(1H-,'1 MACHINE      ',I3,' JOBS',//,                       180.
                     C' PROCESSING TIMES FOR EACH JOB:',5(/,2X,20F4.0) )                    190.
ISN         7           CALL SORTX(N,P,IO)                                                  200.
ISN         8           WRITE(6,2)                                                          210.
ISN         9         2 FORMAT('0OPTIMAL SEQUENCE TO MINIMIZE SUM OF COMPLETION TIMES')     220.
ISN        10           WRITE(6,3)(IO(I),I=1,N)                                             230.
ISN        11         3 FORMAT(2014,/)                                                      240.
ISN        12           RETURN                                                              250.
ISN        13           END                                                                 260.

*STATISTICS*   SOURCE STATEMENTS = 12, PROGRAM SIZE = 1066 BYTES, PROGRAM NAME = MINSCT     PAGE:     2.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 2 ******
```

```
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(54)

                *...*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE MINWTS(N,P,IO,W,WT)                                      270.
                C                                                                           280.
                C       MINIMIZE WEIGHTED SUM OF COMPLETION TIMES OR                        290.
                C       WEIGHTED SUM OF WAITING TIMES ON ONE PROCESS                        300.
                C                                                                           310.
                C       ASSUME THAT A GROUP OF JOBS WITH KNOWN OR ESTIMATED PROCESSING      320.
                C       TIMES P(I). SOME JOBS ARE MORE IMPORTANT THAN OTHERS, SO            330.
                C       ASSIGN A POSITIVE WEIGHT W(I) TO EACH OF THE JOBS. THE GREATER      340.
                C       THE WEIGHT, THE MORE IMPORTANT THE JOB. SEQUENCE THE JOBS IN        350.
                C       SUCH A WAY THAT THE WEIGHTED SUM OF COMPLETION TIMES IS             360.
                C       MINIMIZED.                                                          370.
                C                                                                           380.
                C       ORDER THE JOBS FROM LEFT TO RIGHT IN ORDER OF THE INCREASING        390.
                C       RATIO P(I)/W(I). STOP WHEN THIS IS THE OPTIMAL SEQUENCE.            400.
                C                                                                           410.
                C                                                                           420.
ISN         2           DIMENSION P(N),IO(N),W(N),WT(N)                                     430.
ISN         3           IF(P(1).EQ.0.)READ(5,*)(P(I),I=1,N)                                 440.
ISN         5           READ(5,*)(WT(I),I=1,N)                                              450.
ISN         6           WRITE(6,5)N,(P(I),I=1,N)                                            460.
ISN         7           WRITE(6,6)(WT(I),I=1,N)                                             470.
ISN         8           DO 1 I=1,N                                                          480.
ISN         9         1 P(I)=P(I)/WT(I)                                                     490.
ISN        10           CALL SORTX(N,P,IO)                                                  500.
ISN        11           WRITE(6,2)                                                          510.
ISN        12         2 FORMAT('0OPTIMAL SEQUENCE TO MINIMIZE WEIGHTED SUM OF COMPLETI',    520.
                       C      'ON TIMES')                                                  530.
ISN        13           WRITE(6,3)(IO(I),I=1,N)                                             540.
ISN        14         3 FORMAT(2014,/)                                                      550.
ISN        15           DO 4 I=1,N                                                          560.
ISN        16         4 P(I)=P(I)*WT(I)                                                     570.
ISN        17         5 FORMAT(1H-,'1 MACHINE      ',I3,' JOBS',//,                      580.
                       C' PROCESSING TIMES FOR EACH JOB:',5(/,2X,20F4.0) )                  590.
ISN        18         6 FORMAT(1HO,'WEIGHTS FOR EACH JOB:',5(/,2X,20F4.0) )                600.
ISN        19           RETURN                                                              610.
ISN        20           END                                                                 620.

*STATISTICS*    SOURCE STATEMENTS = 19, PROGRAM SIZE = 1688 BYTES, PROGRAM NAME = MINWTS     PAGE:     3.

*STATISTICS*     NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 3 ******
```



```
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(54)

                *...*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE MINTRD(N,P,IO,W)                                         630.
                C                                                                           640.
                C       TO MINIMIZE THE MAXIMUM TARDINESS ON ONE PROCESS                    650.
                C                                                                           660.
                C       ASSUME A BATCH OF JOBS WITH KNOW DUE DATES, COUNTED FROM TODAY,     670.
                C       AND KNOWN OR ESTIMATED PROCESSING TIMES. OUT OF ALL THE             680.
                C       POSSIBLE WAYS TO SEQUENCE THESE JOBS WE WISH TO CHOOSE THE          690.
                C       SEQUENCE THAT WILL MINIMIZE THE MAXIMUM TARDINESS ON ANY GIVEN JOB  700.
                C                                                                           710.
                C       ORDER THE JOBS FROM LEFT TO RIGHT IN ORDER OF INCREASING DUE DATE   720.
                C       STOP WHEN THE OPTIMAL SEQUENCE HAS BEEN FOUND                       730.
                C                                                                           740.
                C                                                                           750.
ISN         2           DIMENSION P(N),IO(N),W(N)                                           760.
ISN         3           IF(W(1).EQ.0.)READ(5,*)(W(I),I=1,N)                                 770.
ISN         5           WRITE(6,3)N,(W(I),I=1,N)                                            780.
ISN         6           CALL SORTX(N,W,IO)                                                  790.
ISN         7           WRITE(6,1)                                                          800.
ISN         8         1 FORMAT('0OPTIMAL SEQUENCE TO MINIMIZE MAXIMUM TARDINESS')           810.
ISN         9           WRITE(6,2)(IO(I),I=1,N)                                             820.
ISN        10         2 FORMAT(2014,/)                                                      830.
ISN        11         3 FORMAT(1H-,'1 MACHINE      ',I3,' JOBS',//,                      840.
                       C' DUE DATES FROM NOW FOR EACH JOB:',5(/,2X,20F4.0) )                850.
ISN        12           RETURN                                                              860.
ISN        13           END                                                                 870.

*STATISTICS*    SOURCE STATEMENTS = 12, PROGRAM SIZE = 1062 BYTES, PROGRAM NAME = MINTRD     PAGE:     4.

*STATISTICS*     NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 4 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE MAXTRD(N,P,IO,W)                                          880.
                C                                                                            890.
                C       MAXIMIZE THE MINIMUM TARDINESS ON ONE PROCESS (SLACK RULE)           900.
                C                                                                            910.
                C       ASSUME A BATCH OF JOBS WITH KNOW DUE DATES, COUNTED FROM TODAY,      920.
                C       AND KNOW OR ESTIMATED PROCESSING TIMES. OUT OF ALL THE POSSIBLE      930.
                C       WAYS TO SEQUENCE THESE JOBS, CHOOSE THE SEQUENCE THAT WILL           940.
                C       MAXIMIZE THE MINIMUM TARDINESS                                       950.
                C                                                                            960.
                C       ORDER THE JOBS FROM LEFT TO RIGHT IN ORDEROF INCREASING              970.
                C       (DUE DATE MINUS PROCESSING TIME). STOP WHEN THE OPTIMAL              980.
                C       SEQUENCE HAS BEEN FOUND.                                             990.
                C                                                                           1000.
                C                                                                           1010.
ISN         2           DIMENSION P(N),IO(N),W(N)                                       1020.
ISN         3           IF(P(1).EQ.0.)READ(5,*)(P(I),I=1,N)                             1030.
ISN         5           IF(W(1).EQ.0.)READ(5,*)(W(I),I=1,N)                             1040.
ISN         7           WRITE(6,5)N,(P(I),I=1,N)                                        1050.
ISN         8           WRITE(6,6)(W(I),I=1,N)                                          1060.
ISN         9           DO 1 I=1,N                                                      1070.
ISN        10         1 P(I)=W(I)-P(I)                                                  1080.
ISN        11           CALL SORTX(N,P,IO)                                              1090.
ISN        12           WRITE(6,2)                                                      1100.
ISN        13         2 FORMAT('0OPTIMUM SEQUENCE TO MAXIMIZE MINIMUM LATENESS')        1110.
ISN        14           WRITE(6,3)(IO(I),I=1,N)                                         1120.
ISN        15         3 FORMAT(20I4,/)                                                  1130.
ISN        16           DO 4 I=1,N                                                      1140.
ISN        17         4 P(I)=W(I)-P(I)                                                  1150.
ISN        18         5 FORMAT(1H-,'1 MACHINE      ',I3,' JOBS',//,                  1160.
                       C' PROCESSING TIMES FOR EACH JOB:',5(/,2X,20F4.0) )             1170.
ISN        19         6 FORMAT(1HO,'DUE DATES FROM NOW FOR EACH JOB:',5(/,2X,20F4.0) )  1180.
ISN        20           RETURN                                                          1190.
ISN        21           END                                                             1200.
```

STATISTICS SOURCE STATEMENTS = 19, PROGRAM SIZE = 1600 BYTES, PROGRAM NAME = MAXTRD PAGE: 5.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 5 ******

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE MINLAT(N,P,IO,W)                                     1210.
                C                                                                           1220.
                C       MINIMIZE THE NUMBER OF LATE JOBS, N JOBS ON ONE MACHINE             1230.
                C                                                                           1240.
                C       ASSUME A BATCH OF N JOBS WITH KNOWN DUE DATES D(I) AND              1250.
                C       KNOWN OR ESTIMATED PROCESSING TIMES P(I)                            1260.
                C                                                                           1270.
                C       ORDER THE JOBS FROM LEFT TO RIGHT IN ORDER OF INCREASING DUE DATES  1280.
                C       THAT IS, THE JOB WITH THE EARLIEST DUE DATE IS ORDERED FIRST AND    1290.
                C       THE JOB WITH LATEST DUE DATE IS ORDERED LAST. THIS IS THE CURRENT   1300.
                C       SEQUENCE                                                            1310.
                C                                                                           1320.
                C       USING THE CURRENT SEQUENCE, FIND THE FIRST LATE JOB. IF ONE IS      1330.
                C       FOUND, SKIP THIS STEP. OTHERWISE, STOP - THE SEQUENCE IS OPTIMAL    1340.
                C                                                                           1350.
                C       LOOK AT THE SUBSEQUENCE UP TO AND INCLUDING THE LATE JOB. FIND      1360.
                C       THE JOB IN THIS SUBSEQUENCE WITH THE LARGEST PROCESSING TIME AND    1370.
                C       REJECT IT. REPEAT THE PREVIOUS STEP                                 1380.
                C                                                                           1390.
                C                                                                           1400.
ISN         2           DIMENSION P(N),IO(N),W(N)                                       1410.
ISN         3           IF(P(1).EQ.0.)READ(5,*)(P(I),I=1,N)                             1420.
ISN         5           IF(W(1).EQ.0.)READ(5,*)(W(I),I=1,N)                             1430.
ISN         7           WRITE(6,10)N,(P(I),I=1,N)                                       1440.
ISN         8           WRITE(6,11)(W(I),I=1,N)                                         1450.
ISN         9           CALL SORTX(N,W,IO)                                              1460.
                C                                                                           1470.
                C       FIND LATE JOBS, IF ANY                                              1480.
                C                                                                           1490.
ISN        10           NN=N-1                                                          1500.
ISN        11           NJ=N                                                            1510.
ISN        12         1 T=0                                                             1520.
ISN        13           DO 2 I=1,NJ                                                     1530.
ISN        14           J=IO(I)                                                         1540.
ISN        15           T=T+P(J)                                                        1550.
ISN        16         2 IF(T.GT.W(J))GOTO 6                                             1560.
ISN        17           WRITE(6,3)                                                      1570.
ISN        18         3 FORMAT('0OPTIMUM SEQUENCE TO MINIMIZE NUMBER OF LATE JOBS')     1580.
ISN        19           WRITE(6,4)(IO(I),I=1,N)                                         1590.
ISN        20         4 FORMAT(20I4,/)                                                  1600.
ISN        21           NN=N-NJ                                                         1610.
ISN        22           WRITE(6,5)NJ,NN                                                 1620.
ISN        23         5 FORMAT(1HO,I3,' JOBS ON TIME, ',I3,' JOBS LATE')                1630.
ISN        24           RETURN                                                          1640.
                C                                                                           1650.
                C       REJECT LONGEST JOB                                                  1660.
                C                                                                           1670.
ISN        25         6 LATEST=1                                                        1680.
ISN        26           DO 7 J=1,I                                                      1690.
ISN        27           K=IO(J)                                                         1700.
ISN        28         7 IF(P(LATEST).LT.P(K))LATEST=J                                   1710.
ISN        30           J=LATEST                                                        1720.
ISN        31           LATEST=IO(J)                                                    1730.
ISN        32           NJ=NJ-1                                                         1740.
ISN        33           WRITE(6,8)LATEST                                                1750.
ISN        34         8 FORMAT('0LATE JOB IS ',I3)                                      1760.
ISN        35           DO 9 K=J,NN                                                     1770.
ISN        36         9 IO(K)=IO(K+1)                                                   1780.
ISN        37           IO(N)=LATEST                                                    1790.
```

```
ISN        38          GOTO 1                                                          1800.
ISN        39       10 FORMAT(1H-,'1 MACHINE      ',I3,' JOBS',//,                   1810.
                     C' PROCESSING TIMES FOR EACH JOB:',5(/,2X,20F4.0) )              1820.
ISN        40       11 FORMAT(1HO,'DUE DATES FROM NOW FOR EACH JOB:',5(/,2X,20F4.0) ) 1830.
ISN        41          END                                                             1840.

*STATISTICS*    SOURCE STATEMENTS = 38, PROGRAM SIZE = 2012 BYTES, PROGRAM NAME = MINLAT    PAGE:     6.

*STATISTICS*     NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 6 ******
```

```
LEVEL 1.2.0 (SEPT  82)            VS FORTRAN           DATE: 1984 APR 13      TIME: 11:49:31       NAME: MAIN     PAGE:     7

OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(654)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1          SUBROUTINE SORTX(N,X,I)                                        1850.
ISN         2          DIMENSION X(N),I(N)                                            1860.
ISN         3          K=1                                                            1870.
ISN         4          I(K)=1                                                         1880.
ISN         5        2 K=K+1                                                          1890.
ISN         6          IF(K.GT.N)RETURN                                               1900.
ISN         8          KK=K                                                           1910.
ISN         9          I(K)=K                                                         1920.
ISN        10        3 KK=KK-1                                                        1930.
ISN        11          IF(KK.EQ.0)GOTO 2                                              1940.
ISN        12          KKK=I(KK)                                                      1950.
ISN        13          IF(X(KKK).LE.X(K))GOTO 2                                       1960.
ISN        14          I(KK+1)=KKK                                                    1970.
ISN        15          I(KK)=K                                                        1980.
ISN        16          GOTO 3                                                         1990.
ISN        17          END                                                            2000.

*STATISTICS*    SOURCE STATEMENTS = 16, PROGRAM SIZE = 640 BYTES, PROGRAM NAME = SORTX    PAGE:     7.

*STATISTICS*     NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 7 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(54)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN         1           SUBROUTINE CAMPBL(N,IA,IB,IAB,IBA,JAB,JBA,PAB,PBA,ICHK)                2010.
                C                                                                              2020.
                C       ASSUME A BATCH OF JOBS TO BE PROCESSED ON 2 MACHINES A AND B           2030.
                C       SOME JOBS ARE TO BE PROCESSED ONLY ON MACHINE; SOME ONLY ON            2040.
                C       MACHINE B. SOME ARE TO BE PROCESSED ON BOTH MACHINES A AND B           2050.
                C       IN THE ORDER A,B                                                       2060.
                C       SOME ARE TO BE PROCESSED ON BOTH MACHINES A AND B IN THE ORDER         2070.
                C       B, A. THE GOAL IS TO MINIMIZE TOTAL PROCESSING TIME.                   2080.
                C                                                                              2090.
ISN         2           DIMENSION IA(N),IB(N),IAB(N),IBA(N),JAB(N),JBA(N),PAB(2,N),            2100.
                C                 PBA(2,N),ICHK(N)                                             2110.
ISN         3           INTEGER CASE                                                           2120.
ISN         4           WRITE(6,8)N                                                            2130.
ISN         5           KA=0                                                                   2140.
ISN         6           KB=0                                                                   2150.
ISN         7           KAB=0                                                                  2160.
ISN         8           KBA=0                                                                  2170.
ISN         9           DO 5 I=1,N                                                             2180.
ISN        10           READ(5,*)CASE,A,B                                                      2190.
                C                                                                              2200.
                C       CASE          ORDER                                                    2210.
                C        1            1, 2                                                     2220.
                C        2            2, 1                                                     2230.
                C        3            1, 0                                                     2240.
                C        4            0, 2                                                     2250.
                C                                                                              2260.
ISN        11           GOTO(1,2,3,4),CASE                                                     2270.
ISN        12         1 KAB=KAB+1                                                              2280.
ISN        13           IAB(KAB)=I                                                             2290.
ISN        14           PAB(1,KAB)=A                                                           2300.
ISN        15           PAB(2,KAB)=B                                                           2310.
ISN        16           WRITE(6,9)A,B                                                          2320.
ISN        17           GOTO 5                                                                 2330.
ISN        18         2 KBA=KBA+1                                                              2340.
ISN        19           IBA(KBA)=I                                                             2350.
ISN        20           PBA(1,KBA)=B                                                           2360.
ISN        21           PBA(2,KBA)=A                                                           2370.
ISN        22           WRITE(6,10)A,B                                                         2380.
ISN        23           GOTO 5                                                                 2390.
ISN        24         3 KA=KA+1                                                                2400.
ISN        25           IA(KA)=I                                                               2410.
ISN        26           WRITE(6,11)A,B                                                         2420.
ISN        27           GOTO 5                                                                 2430.
ISN        28         4 KB=KB+1                                                                2440.
ISN        29           IB(KB)=I                                                               2450.
ISN        30           WRITE(6,12)A,B                                                         2460.
ISN        31         5 CONTINUE                                                               2470.
                C                                                                              2480.
                C                                                                              2490.
ISN        32           CALL JONSON(KAB,JAB,IAB,PAB,ICHK)                                      2500.
ISN        33           CALL JONSON(KBA,JBA,IBA,PBA,ICHK)                                      2510.
ISN        34           K=1                                                                    2520.
ISN        35           WRITE(6,7)K,(JAB(I),I=1,KAB),(IA(I),I=1,KA),(JBA(I),I=1,KBA)           2530.
ISN        36           K=2                                                                    2540.
ISN        37           WRITE(6,7)K,(JBA(I),I=1,KBA),(IB(I),I=1,KB),(JAB(I),I=1,KAB)           2550.
ISN        38         7 FORMAT('0OPTIMAL ORDER ON MACHINE ',I1,' IS:',/(10(2X,I2)/))           2560.
ISN        39         8 FORMAT(1H-,'2 MACHINES     ',I3,' JOBS',//,                          2570.
                C        ' MACHINE',6X,'PROCESSING TIME',/'  ORDER',8X,'FOR EACH JOB',/,       2580.
                C        ' --------',5X,'---------------')                                     2581.


ISN        40         9 FORMAT(1H ,'ONE TWO',9X,2F5.0)                                         2590.
ISN        41        10 FORMAT(1H ,'TWO ONE',9X,2F5.0)                                         2600.
ISN        42        11 FORMAT(1H ,'ONE ONLY',8X,2F5.0)                                        2610.
ISN        43        12 FORMAT(1H ,'TWO ONLY',8X,2F5.0)                                        2620.
ISN        44           RETURN                                                                 2630.
ISN        45           END                                                                    2640.
```

STATISTICS SOURCE STATEMENTS = 45, PROGRAM SIZE = 2698 BYTES, PROGRAM NAME = CAMPBL PAGE: 8.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 8 ******

```
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN   )  LINECOUNT(54)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE JONSON(N,IO,JO,PAB,ICHK)                                  2650.
                C                                                                            2660.
                C       JOHNSON'S METHOD                                                     2670.
                C                                                                            2680.
ISN         2           DIMENSION IO(N),PAB(2,N),ICHK(N),JO(N)                               2690.
ISN         3           K=N                                                                  2700.
ISN         4           J=1                                                                  2710.
ISN         5           DO 1 I=1,N                                                           2720.
ISN         6         1 ICHK(I)=0                                                            2730.
ISN         7         2 M=0                                                                  2740.
ISN         8           DO 3 I=1,N                                                           2750.
ISN         9           IF(ICHK(I).EQ.1)GOTO 3                                               2760.
ISN        10           IF(M)4,4,5                                                           2770.
ISN        11         5 IF(PAB(1,I).GT.X.AND.PAB(2,I).GT.X)GOTO 3                            2780.
ISN        12         4 M=I                                                                  2790.
ISN        13           X=AMIN1(PAB(1,I),PAB(2,I))                                           2800.
ISN        14           L=1                                                                  2810.
ISN        15           IF(X.EQ.PAB(2,I))L=2                                                 2820.
ISN        17         3 CONTINUE                                                             2830.
ISN        18           IF(M.EQ.0)RETURN                                                     2840.
ISN        20           ICHK(M)=1                                                            2850.
ISN        21           IF(L-1)6,6,7                                                         2860.
ISN        22         6 IO(J)=JO(M)                                                          2870.
ISN        23           J=J+1                                                                2880.
ISN        24           GOTO 2                                                               2890.
ISN        25         7 IO(K)=JO(M)                                                          2900.
ISN        26           K=K-1                                                                2910.
ISN        27           GOTO 2                                                               2920.
ISN        28           END                                                                  2930.
```

STATISTICS SOURCE STATEMENTS = 26, PROGRAM SIZE = 1226 BYTES, PROGRAM NAME = JONSON PAGE: 9.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 9 ******


```
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN   )  LINECOUNT(54)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE GUPTA(NJ,NP,V,IO,JO,P,PAB,ICHK)                           2940.
                C                                                                            2950.
                C       GUPTA'S   METHOD    ( N JOB / M MACHINE )                            2960.
                C                                                                            2970.
ISN         2           DIMENSION P(NJ,NP),PAB(2,NJ),V(NJ),IO(NJ),JO(NJ),ICHK(NJ)            2980.
ISN         3           REAL BIG M/1.E9/                                                     2990.
ISN         4           WRITE(6,1)NP,NJ                                                      3000.
ISN         5         1 FORMAT(1H-,I3,' MACHINES      ',I3,' JOBS',/                       3010.
                     C  ' PROCESSING TIME FOR EACH JOB ON PROCESS',/)                        3020.
ISN         6           IF(NP.NE.2)THEN                                                      3030.
ISN         7                 DO 5 I=1,NJ                                                    3050.
ISN         8                 READ(5,*)(P(I,J),J=1,NP)                                       3060.
ISN         9                 WRITE(6,10)I,(P(I,J),J=1,NP)                                   3070.
ISN        10                 V(I)=BIG M                                                     3080.
ISN        11                 DO 6 J=1,NP-1                                                  3090.
ISN        12         6       V(I)=AMIN1(V(I),(P(I,J)+P(I,J+1)))                             3100.
ISN        13                 IF(P(I,1).LT.P(I,NP))V(I)=-V(I)                                3110.
ISN        15         5       V(I)=1./V(I)                                                   3120.
ISN        16                 CALL SORTX(NJ,V,IO)                                            3130.
ISN        17           ELSE                                                                 3140.
ISN        18                 DO 8 I=1,NJ                                                    3150.
ISN        19                 JO(I)=I                                                        3160.
ISN        20                 READ(5,*)PAB(1,I),PAB(2,I)                                     3170.
ISN        21         8       WRITE(6,10)I,PAB(1,I),PAB(2,I)                                 3180.
ISN        22                 CALL JONSON(NJ,IO,JO,PAB,ICHK)                                 3190.
ISN        23           END IF                                                               3200.
ISN        24        10 FORMAT(' JOB ',I3,':',30F4.0)                                        3210.
ISN        25           WRITE(6,11)(IO(I),I=1,NJ)                                            3220.
ISN        26        11 FORMAT('0OPTIMUM SEQUENCE TO MINIMIZE TOTAL PROCESSING TIME IS:',/3230.
                     C       2014,/)                                                         3240.
ISN        27           RETURN                                                               3250.
ISN        28           END                                                                  3260.
```

STATISTICS SOURCE STATEMENTS = 27, PROGRAM SIZE = 2314 BYTES, PROGRAM NAME = GUPTA PAGE: 10.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 10 ******

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(54)

```
                *....*...1.....2.....3.....4.....5.....6.....7.*.....8
ISN         1           SUBROUTINE HUNGAR(N,IIF,IEFF,IROW,ICOL,ICHECK,JCHECK,TOTAL,KODE,  10.
                     C                    ESTMAN)                                        11.
                    C                                                                    20.
                    C          H U N G A R I A N     A L G O R I T H M                   30.
                    C                                                                    40.
                    C          A METHOD FOR SOLVING THE GENERAL M-RESOURCE, M-ACTIVITY    50.
                    C          ASSIGNMENT PROBLEM FOR TOTAL EFFECTIVENESS. THE PROBLEM   60.
                    C          MAY BE EITHER MINIMIZATION OR MAXIMIZATION. THE PROGRAM   70.
                    C          USES INTEGER IN THE MATRICES.                             80.
                    C                                                                    90.
                    C          IIF     ASSINGMENT MATRIX                                 100.
                    C          TOTAL   OPTIMAL COST OF ASSIGNMENTS                       110.
                    C          KODE    0 - MINIMIZATION PROBLEM                          120.
                    C                  1 - MAXIMIZATION PROBLEM                          130.
                    C          ESTMAN SWITCH INDICATED CALL FROM SUBROUTINE EASTMN       131.
                    C                                                                    140.
ISN         2           INTEGER IIF(15,15),IEFF(N,N),IROW(N),ICOL(N),ICHECK(N),          150.
                     C        JCHECK(N),TOTAL,ESTMAN                                     160.
ISN         3           IF(ESTMAN.EQ.0)THEN                                             170.
ISN         4           WRITE(6,3)                                                      180.
ISN         5           DO 2 I=1,N                                                      190.
ISN         6           READ(5,*)(IIF(I,J),J=1,N)                                       200.
ISN         7           WRITE(6,1)(IIF(I,J),J=1,N)                                      210.
ISN         8         1 FORMAT(1X,10I8)                                                 220.
ISN         9         2 CONTINUE                                                        230.
ISN        10         3 FORMAT(1H-,'ORIGINAL MATRIX:')                                  240.
ISN        11           END IF                                                          250.
                    C                                                                    260.
                    C      CONSTRUCT THE EFFECTIVENESS MATRIX. IF TOTAL EFFECTIVENESS IS TO  270.
                    C      BE MAXIMIZED, LET THE MATRIX BE THE NEGATIVE OF THE ORIGINAL  280.
                    C      MATRIX; OTHERWISE, THE MATRIX EQUALS THE ORIGINAL MATRIX.     290.
                    C                                                                    300.
ISN        12           IF(KODE.EQ.0)THEN                                               310.
ISN        13                DO 110 I=1,N                                               320.
ISN        14                DO 110 J=1,N                                               330.
ISN        15       110      IEFF(I,J)=IIF(I,J)                                         340.
ISN        16           ELSE                                                            350.
ISN        17                DO 120 I=1,N                                               360.
ISN        18                DO 120 J=1,N                                               370.
ISN        19       120      IEFF(I,J)=-IIF(I,J)                                        380.
ISN        20           END IF                                                          390.
                    C                                                                    400.
                    C      FIND THE SMALLEST ELEMENT OF EACH ROW AND SUBTRACT IT FROM THE  410.
                    C      OTHER ELEMENTS OF THE SAME ROW                                420.
                    C                                                                    430.
ISN        21           DO 210 I=1,N                                                    440.
ISN        22           MINROW=IEFF(I,1)                                                450.
ISN        23           DO 220 J=1,N                                                    460.
ISN        24           IF(IEFF(I,J).LE.MINROW)MINROW=IEFF(I,J)                         470.
ISN        26       220 CONTINUE                                                        480.
ISN        27           DO 210 J1=1,N                                                   490.
ISN        28           IEFF(I,J1)=IEFF(I,J1)-MINROW                                    500.
ISN        29       210 CONTINUE                                                        510.
                    C                                                                    520.
                    C      FIND THE SMALLEST ELEMENT OF EACH COLUMN AND SUBTRACT IT FROM THE 530.
                    C      OTHER ELEMENTS IN THE SAME COLUMN                             540.
                    C                                                                    550.
ISN        30           DO 300 J=1,N                                                    560.
ISN        31           MINCOL=IEFF(1,J)                                                570.


ISN        32           DO 310 I=1,N                                                    580.
ISN        33           IF(IEFF(I,J).LE.MINCOL)MINCOL=IEFF(I,J)                         590.
ISN        35       310 CONTINUE                                                        600.
ISN        36           DO 300 I1=1,N                                                   610.
ISN        37           IEFF(I1,J)=IEFF(I1,J)-MINCOL                                    620.
ISN        38       300 CONTINUE                                                        630.
                    C                                                                    640.
                    C      ROW ASSIGNMENTS                                               650.
                    C                                                                    660.
ISN        39       400 DO 410 I=1,N                                                    670.
ISN        40           IROW(I)=0                                                       680.
ISN        41       410 ICOL(I)=0                                                       690.
ISN        42           NOMADE=0                                                        700.
ISN        43       420 LOOP=0                                                          710.
ISN        44           NZEROS=0                                                        720.
ISN        45           DO 430 I=1,N                                                    730.
ISN        46           NOZR=0                                                          740.
ISN        47           IF(IROW(I).EQ.0)THEN                                            750.
ISN        48           DO 440 J=1,N                                                    760.
ISN        49           IF(ICOL(J).EQ.0)THEN                                            770.
ISN        50           IF(IEFF(I,J).EQ.0)THEN                                          780.
ISN        51                NOZR=NOZR+1                                                790.
ISN        52                NZEROS=NZEROS+1                                            800.
ISN        53                NRPT=J                                                     810.
ISN        54                NREFF=I                                                    820.
ISN        55           END IF                                                          830.
ISN        56           END IF                                                          840.
ISN        57       440 CONTINUE                                                        850.
ISN        58           IF(NOZR.EQ.1)THEN                                               860.
ISN        59                IROW(I)=NRPT                                               870.
ISN        60                ICOL(NRPT)=1                                               880.
ISN        61                NOMADE=NOMADE+1                                            890.
ISN        62                LOOP=LOOP+1                                                900.
ISN        63           END IF                                                          910.
ISN        64           END IF                                                          920.
ISN        65       430 CONTINUE                                                        930.
                    C                                                                    940.
                    C      COLUMN ASSIGNMENTS                                            950.
                    C                                                                    960.
ISN        66           DO 500 I=1,N                                                    970.
ISN        67           NOZC=0                                                          980.
ISN        68           IF(ICOL(I).EQ.0)THEN                                            990.
ISN        69           DO 510 J=1,N                                                    1000.
ISN        70           IF(IROW(J).EQ.0)THEN                                            1010.
ISN        71           IF(IEFF(J,I).EQ.0)THEN                                          1020.
ISN        72                NOZC=NOZC+1                                                1030.
ISN        73                NZEROS=NZEROS+1                                            1040.
ISN        74                IRPT=J                                                     1050.
ISN        75           END IF                                                          1060.
ISN        76           END IF                                                          1070.
ISN        77       510 CONTINUE                                                        1080.
ISN        78           IF(NOZC.EQ.1)THEN                                               1090.
ISN        79                ICOL(I)=1                                                  1100.
ISN        80                IROW(IRPT)=I                                               1110.
ISN        81                NOMADE=NOMADE+1                                            1120.
ISN        82                LOOP=LOOP+1                                                1130.
ISN        83           END IF                                                          1140.
ISN        84           END IF                                                          1150.
ISN        85       500 CONTINUE                                                        1160.
                    C                                                                    1170.
                    C      IF COMPLETE ASSIGNMENT HAS BEEN MADE, RETURN                  1180.
                    C      IF AT LEAST ONE ASIGNMENT HAS BEEN MADE, RETURN TO ROW ASSIGNMENTS 1190.
                    C      TO MAKE ADDITIONAL ASSIGNMENTS.                               1200.
                    C                                                                    1210.
ISN        86           IF(NOMADE.EQ.N)GOTO 1500                                        1220.
ISN        87           IF(LOOP.NE.0)GOTO 420                                           1230.
```

```
ISN        88           IF(NZEROS.EQ.0)GOTO 800                                          1240.
                  C                                                                      1250.
                  C      PICK AN ARBITRARY ZERO AND RETURN TO ROW ASSIGNMENTS            1260.
                  C                                                                      1270.
ISN        89           IROW(NREFF)=NRPT                                                 1280.
ISN        90           NOMADE=NOMADE+1                                                  1290.
ISN        91           ICOL(NRPT)=1                                                     1300.
ISN        92           GOTO 420                                                         1310.
                  C                                                                      1320.
                  C      CHECK FOR UNASSIGNED ROW                                        1330.
                  C                                                                      1340.
ISN        93       800 DO 810 I=1,N                                                     1350.
ISN        94           ICHECK(I)=0                                                      1360.
ISN        95           JCHECK(I)=0                                                      1370.
ISN        96           IF(IROW(I).EQ.0)ICHECK(I)=I                                      1380.
ISN        98       810 CONTINUE                                                         1390.
                  C                                                                      1400.
                  C      CHECK COLUMNS THAT HAVE ZEROS IN CHECKED ROWS                   1410.
                  C                                                                      1420.
ISN        99       900 NCHECK=0                                                         1430.
ISN       100           DO 910 I=1,N                                                     1440.
ISN       101           IF(ICHECK(I).NE.0)THEN                                           1450.
ISN       102           DO 920 J=1,N                                                     1460.
ISN       103           IF(JCHECK(J).EQ.0)THEN                                           1470.
ISN       104           IF(IEFF(I,J).EQ.0)THEN                                           1480.
ISN       105                JCHECK(J)=J                                                 1490.
ISN       106                NCHECK=NCHECK+1                                             1500.
ISN       107           END IF                                                           1510.
ISN       108           END IF                                                           1520.
ISN       109       920 CONTINUE                                                         1530.
ISN       110           END IF                                                           1540.
ISN       111       910 CONTINUE                                                         1550.
                  C                                                                      1560.
                  C      CHECK ROWS THAT HAVE ASSIGNMENTS IN CHECKED COLUMNS             1570.
                  C      REPEAT UNTIL CHAIN OF CHECKING IS COMPLETED.                    1580.
                  C      ELEIMINATE ALL UNCHECKED ROWS AND ALL CHECKED COLUMNS           1590.
                  C                                                                      1600.
ISN       112           IF(NCHECK.NE.0)THEN                                              1610.
ISN       113           DO 1010 I=1,N                                                    1620.
ISN       114           IF(JCHECK(I).GT.0)THEN                                           1630.
ISN       115           DO 1020 J=1,N                                                    1640.
ISN       116           IF(JCHECK(I).EQ.IROW(J))THEN                                     1650.
ISN       117           ICHECK(J)=J                                                      1660.
ISN       118           NCHECK=NCHECK+1                                                  1670.
ISN       119           END IF                                                           1680.
ISN       120      1020 CONTINUE                                                         1690.
ISN       121           END IF                                                           1700.
ISN       122      1010 CONTINUE                                                         1710.
ISN       123           END IF                                                           1720.
ISN       124           IF(NCHECK.NE.0)GOTO 900                                          1730.
                  C                                                                      1740.
                  C     FIND THE MINIMUM UNMARKED ELEMENT                                1750.
                  C                                                                      1760.
ISN       125           MINELM=99999999                                                  1770.
ISN       126           DO 1310 I=1,N                                                    1780.
ISN       127           IF(ICHECK(I).NE.0)THEN                                           1790.
ISN       128           DO 1320 J=1,N                                                    1800.
ISN       129           IF(JCHECK(J).EQ.0)THEN                                           1810.
ISN       130                IF(IEFF(I,J).LT.MINELM)MINELM=IEFF(I,J)                     1820.
ISN       132           END IF                                                           1830.
ISN       133      1320 CONTINUE                                                         1840.
ISN       134           END IF                                                           1850.
ISN       135      1310 CONTINUE                                                         1860.
                  C                                                                      1870.
                  C      REDUCE MATRIX AND GOTO TO ROW REDUCTIONS                        1880.
                  C                                                                      1890.



ISN       136           DO 1400 I=1,N                                                    1900.
ISN       137           DO 1400 J=1,N                                                    1910.
ISN       138           IF(ICHECK(I).GE.0)THEN                                           1920.
ISN       139                IF(ICHECK(I).EQ.0)THEN                                      1930.
ISN       140                     IF(JCHECK(J).GT.0)IEFF(I,J)=IEFF(I,J)+MINELM           1940.
ISN       142                ELSE                                                        1950.
ISN       143                     IF(JCHECK(J).EQ.0)IEFF(I,J)=IEFF(I,J)-MINELM           1960.
ISN       145                END IF                                                      1970.
ISN       146           END IF                                                           1980.
ISN       147      1400 CONTINUE                                                         1990.
ISN       148           GOTO 400                                                         2000.
                  C                                                                      2010.
                  C      CALCULATE TOTAL AND RETURN TO MAIN PROGRAM                      2020.
                  C                                                                      2030.
ISN       149      1500 TOTAL=0                                                          2040.
ISN       150           DO 1510 I=1,N                                                    2050.
ISN       151           K=IROW(I)                                                        2060.
ISN       152      1510 TOTAL=TOTAL+IIF(I,K)                                             2070.
                  C                                                                      2071.
                  C                                                                      2072.
ISN       153           IF(ESTMAN.EQ.0)THEN                                              2080.
ISN       154           WRITE(6,1520)TOTAL                                               2090.
ISN       155      1520 FORMAT('1H0,5X,'THE OPTIMAL SOLUTION OF THE ASSIGNMENT PROBLEM''S T2100.
                       'OTAL COST = ',I8/)                                             2110.
ISN       156           DO 1530 I=1,N                                                    2120.
ISN       157      1530 WRITE(6,1540)I,IROW(I)                                           2130.
ISN       158      1540 FORMAT(10X,'THE ASSIGNMENT FOR ',I3,' = ',I8)                    2140.
ISN       159           END IF                                                           2150.
ISN       160           RETURN                                                           2160.
ISN       161           END                                                              2170.

*STATISTICS*   SOURCE STATEMENTS = 155, PROGRAM SIZE = 5628 BYTES, PROGRAM NAME = HUNGAR    PAGE:   11.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 11 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN         1           SUBROUTINE EASTMN(N,IIAS,IDIST,ISDIST,IAS,JCOMP,BFSD,BRANCH,R,C,P,
                     C                    TD,NARCS,SUBT)
                      C
                      C     E A S T M A N  '  S       A L G O R I T H M
                      C
                      C     SOLVE THE TRAVELING SALESMAN PROBLEM BY USING THE HUNGARIAN
                      C     ALGORITHM TO SOLVE THE ASSOCIATED ASSIGNMENT PROBLEM. SUBROUTINE
                      C     HUNGRY MUST BE SUPPLIED.
                      C
                      C     IDIST    DISTANCE MATRIX
                      C     IAS      ASSIGNMENT MATRIX
                      C     MD       MINIMUM DISTANCE OF THE ASSOCIATED ASSIGMENT PROBLEM
                      C     OPTSOL   DISTANCE OF OPTIMAL FEASIBLE SOLUTION
                      C
                      C
                      C     WHEN N > 15, ALL THE "15" COMPONENTS IN THE DIMENSION STATEMENT
                      C     HAVE TO CHANGE TO THE VALUE OF N
                      C     VARIABLE IDIST,KTEMP,ICOL,ICHECK,JCHECK CANNOT BE DYNAMICALLY
                      C     ALLOCATED (ALL THE VARIABLES WITH DIMENSION > 1 AND APPEAR ON
                      C     THE READ STATEMENT HAVE A DIFFERENT ADDRESS IN FORTRAN-77)
                      C
ISN         2           DIMENSION IAS(N),IDIST(15,15),ISDIST(N,N),JCOMP(N),BFSD(N),
                     C           IIAS(N*30,N),BRANCH(N*30),R(N*30),C(N*30),P(N*30),
                     C           TD(N*30),NARCS(N*30),SUBT(N*30),
                     C           KTEMP(15,15),ICOL(15),ICHECK(15),JCHECK(15)
ISN         3           INTEGER BIG M/999999999/
ISN         4           DO 2 I=1,N
ISN         5         2 READ(5,*)(IDIST(I,J),J=1,N)
ISN         6           KODE=0
ISN         7           WRITE(6,5)
ISN         8           DO 3 I=1,N
ISN         9         3 WRITE(6,4)(IDIST(I,J),J=1,N)
ISN        10         4 FORMAT(1OI8)
ISN        11         5 FORMAT(1H-,'THE DISTANCE MATRIX:')
                      C
                      C     SET MINIMUM TOTAL DISTANCE (OPTSOL) TO LARGE POSITIVE NUMBER
                      C
ISN        12           OPTSOL=1E10
                      C
                      C     CALL SUBROUTINE HUNGRY TO SOLVE THE ASSOCIATED ASSIGNMENT PROBLEM
                      C     BY THE HUNGARIAN METHOD
                      C     IF AT LEAST ONE SUBTOUR EXISTS, SKIP THIS STEP.
                      C     OTHERWISE, AN OPTIMAL SOLUTION OF THE ASSIGNMENT PROBLEM IS THE
                      C     OPTIMAL SOLUTION OF THE TRAVELING SALESMAN PROBLEM. STOP.
                      C
ISN        13           CALL HUNGAR(N,IDIST,KTEMP,IAS,ICOL,ICHECK,JCHECK,MD,KODE,1)
ISN        14           DO 71 I=1,N
ISN        15        71 JCOMP(I)=IAS(I)
ISN        16           DO 70 II=2,N-2
ISN        17           DO 72 I=1,N
ISN        18           J=JCOMP(I)
ISN        19           JCOMP(I)=IAS(J)
ISN        20           IF(JCOMP(I).EQ.I)GOTO 80
ISN        21        72 CONTINUE
ISN        22        70 CONTINUE
ISN        23           GOTO 200
                      C
                      C     A SUBTOUR EXISTS OF WHICH WE HAVE THE SHORTEST. DETERMINE THE
                      C     PARAMETERS OF THE SHORTEST SUBTOUR


                      C
ISN        24        80 IARS=II
ISN        25           INT=I
ISN        26           DO 701 KK=1,N
ISN        27       701 IIAS(1,KK)=IAS(KK)
ISN        28           K=1
                      C
                      C     BRANCH INTO K SUBLEVELS
                      C
ISN        29           P(1)=1
ISN        30           IPRED=1
ISN        31           GOTO 110
ISN        32       700 LL1=IPRED
ISN        33       111 IF(P(LL1).EQ.1)GOTO 112
ISN        34           LL2=R(LL1)
ISN        35           LL3=C(LL1)
ISN        36           ISDIST(LL2,LL3)=IDIST(LL2,LL3)
ISN        37           IDIST(LL2,LL3)=BIG M
ISN        38           LL1=P(LL1)
ISN        39           GOTO 111
ISN        40       112 LL2=R(LL1)
ISN        41           LL3=C(LL1)
ISN        42           ISDIST(LL2,LL3)=IDIST(LL2,LL3)
ISN        43           IDIST(LL2,LL3)=BIG M
ISN        44           IARS=NARCS(IPRED)
ISN        45           INT=SUBT(IPRED)
ISN        46       110 ITERM=INT
ISN        47           DO 90 IJ=1,IARS
ISN        48           IBEGIN=ITERM
ISN        49           ITERM=IIAS(IPRED,IBEGIN)
ISN        50           K=K+1
ISN        51           R(K)=IBEGIN
ISN        52           C(K)=ITERM
ISN        53           ITEMP=IDIST(IBEGIN,ITERM)
ISN        54           IDIST(IBEGIN,ITERM)=BIG M
                      C
                      C     SOLVE THE K SUBLEVELS AS NEW ASSIGNMENT PROBLEMS, AND LET
                      C     EACH SOLUTION DISTANCE BE A LOWER BOUND FOR THE CORESPONDING
                      C     SUBLEVEL
                      C
ISN        55           CALL HUNGAR(N,IDIST,KTEMP,IAS,ICOL,ICHECK,JCHECK,MD,KODE,1)
ISN        56           TD(K)=MD
ISN        57           P(K)=IPRED
ISN        58           IDIST(IBEGIN,ITERM)=ITEMP
ISN        59           DO 125 MM=1,N
ISN        60           IIAS(K,MM)=IAS(MM)
ISN        61       125 CONTINUE
ISN        62           DO 73 I=1,N
ISN        63           JCOMP(I)=IAS(I)
ISN        64        73 CONTINUE
ISN        65           DO 75 II=2,NN
ISN        66           DO 74 I=1,N
ISN        67           J=JCOMP(I)
ISN        68           JCOMP(I)=IAS(J)
ISN        69           IF(JCOMP(I).EQ.I)GOTO 131
ISN        70        74 CONTINUE
ISN        71        75 CONTINUE
                      C
                      C     IF THERE EXISTS 1 OR MORE FEASIBLE SOLUTIONS FROM PREVIOUS STEP
                      C     AND IF THE SMALLEST TOTAL DISTANCE FOR THAT FEASIBLE SOLUTION
                      C     IS SMALLER THAN OPTSOL, SET OPTSOL = STD AND SAVE THE
                      C     CORRESPONDING FEASIBLE SOLUTION
                      C
ISN        72           IF(OPTSOL.LT.TD(K))GOTO 130
ISN        73           OPTSOL=TD(K)
ISN        74           DO 129 JJ=1,N
```

```
ISN        75      129 BFSD(JJ)=IAS(JJ)                                                1210.
ISN        76      130 BRANCH(K)=0                                                     1220.
ISN        77          GOTO 90                                                         1230.
ISN        78      131 BRANCH(K)=1                                                     1240.
ISN        79          SUBT(K)=I                                                       1250.
ISN        80          NARCS(K)=II                                                     1260.
ISN        81       90 CONTINUE                                                        1270.
ISN        82          IF(IPRED.EQ.1)GOTO 160                                          1280.
ISN        83          LL1=IPRED                                                       1290.
ISN        84      150 IF(P(LL1).EQ.1)GOTO 152                                         1300.
ISN        85          LL2=R(LL1)                                                      1310.
ISN        86          LL3=C(LL1)                                                      1320.
ISN        87          LL1=P(LL1)                                                      1330.
ISN        88          IDIST(LL2,LL3)=ISDIST(LL2,LL3)                                  1340.
ISN        89          GOTO 150                                                        1350.
ISN        90      152 LL2=R(LL1)                                                      1360.
ISN        91          LL3=C(LL1)                                                      1370.
ISN        92          IDIST(LL2,LL3)=ISDIST(LL2,LL3)                                  1380.
                   C                                                                  1390.
                   C      IF OPTSOL IS LESS THAN THE LOWER BOUNDS ON ALL OF THE UNEXPLORED 1400.
                   C      SUBLEVELS, THE SOLUTION CORESPONDING TO OPTSOL IS AN OPTIMAL 1410.
                   C      SOLUTION, SO STOP. OTHERWISE, GO TO NEXT STEP               1420.
                   C                                                                  1430.
ISN        93      160 DO 132 I=2,K                                                    1440.
ISN        94          IF(BRANCH(I).EQ.0)GOTO 132                                      1450.
ISN        95          IF(OPTSOL.GT.TD(I))GOTO 134                                     1460.
ISN        96      132 CONTINUE                                                        1470.
ISN        97          GOTO 135                                                        1480.
                   C                                                                  1490.
                   C      SELECT THE UNEXPLORED SUBLEVEL WITH THE SMALLEST VALUE      1500.
                   C      FIND THE MINIMUM DISTANCE FOR THE UNFEASIBLE NODES          1510.
                   C      SET THE INITIAL MINIMAL NODE TO THE VALUE OF THE COUNTER IN THE 1520.
                   C      PREVIOUS LOOP, AND RETURN FOR FURTHER BRANCHING             1530.
                   C                                                                  1540.
ISN        98      134 MINND=I                                                         1550.
ISN        99          DO 142 J=I,K                                                    1560.
ISN       100          IF(BRANCH(J).EQ.0)GOTO 142                                      1570.
ISN       101          IF(TD(J).LT.TD(MINND))MINND=J                                   1580.
ISN       103      142 CONTINUE                                                        1590.
ISN       104          BRANCH(MINND)=0                                                 1600.
ISN       105          IPRED=MINND                                                     1610.
ISN       106          GOTO 700                                                        1620.
                   C                                                                  1630.
                   C      OUTPUT THE FINAL MINIMAL DISTANCE                           1640.
                   C      OUTPUT THE ASSIGNMENT FOR THE OPTIMAL SOLUTION              1650.
                   C                                                                  1660.
ISN       107      135 WRITE(6,222)OPTSOL                                              1670.
ISN       108      222 FORMAT(//,' THE OPTIMAL FEASIBLE SOLUTION DISTANCE IS',F12.0)   1680.
ISN       109          DO 136 I=1,N                                                    1690.
ISN       110          WRITE(6,220)I,BFSD(I)                                           1700.
ISN       111      220 FORMAT(' THE ASSIGNMENT FOR',I4,' IS ',F12.0)                   1710.
ISN       112      136 CONTINUE                                                        1720.
ISN       113          GOTO 999                                                        1730.
                   C                                                                  1740.
                   C      THE SOLUTION TO THE ASSIGNMENT PROBLEM IS THE OPTIMAL SOLUTION 1750.
                   C      TO THE TRAVELING SALESMAN PROBLEM                           1760.
                   C                                                                  1770.
ISN       114      200 WRITE(6,201)                                                    1780.
ISN       115      201 FORMAT(////5X,'THE SOLUTION HAS NO SUBTOURS',//5X,              1790.
                      C      'THE OPTIMAL SOLUTION OF THE ASSIGNMENT PROBLEM IS',/5X,  1800.
                      C      'ALSO AN OPTIMAL SOLUTION OF THE TRAVELING SALESMAN PROBLEM')1810.
ISN       116          WRITE(6,202)MD                                                  1820.
ISN       117      202 FORMAT(5X,'THE TOTAL MINIMAL DISTANCE IS',I8,//)                1830.
ISN       118          DO 203 I=1,N                                                    1840.
ISN       119      203 WRITE(6,204)I,IAS(I)                                            1850.
ISN       120      204 FORMAT(5X,'THE ASSIGNMENT FOR',I4,' IS ',I8)                    1860.


ISN       121      999 RETURN                                                          1870.
ISN       122          END                                                             1880.

*STATISTICS*    SOURCE STATEMENTS = 121, PROGRAM SIZE = 6984 BYTES, PROGRAM NAME = EASTMN     PAGE:   12.

*STATISTICS*     NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 12 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(54)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN         1           SUBROUTINE NJOB2M(N,T,S,TIM1,TIM2,TOM1,TOM2,OSVECT)                110.
                C                                                                          120.
                C       SEQUENCING  N-JOBS THROUGH 2 MACHINES                              130.
                C                                                                          140.
                C                                                                          150.
                C                                                                          160.
                C       T(I,1)  TIME TO PROCESS JOB I ON MACHINE 1                         170.
                C       T(I,2)  TIME TO PROCESS JOB I ON MACHINE 2                         180.
                C                                                                          190.
                C       OSVECT     OPTIMAL SEQUENCE FOR PROCESSING N JOBS THRU 2 MACHINES  200.
                C       TIM1,TOM1 TIME IN AND TIME OUT FOR MACHINE 1 FOR EACH JOB          210.
                C       TIM2,TOM2 TIME IN AND TIME OUT FOR MACHINE 2 FOR EACH JOB          220.
                C       ITM1       IDLE TIME ON MACHINE 1                                  230.
                C       ITM2       IDLE TIME ON MACHINE 2                                  240.
                C                                                                          250.
                C                                                                          260.
ISN         2           DIMENSION T(N,2),S(N,2),TIM1(N),TIM2(N),TOM1(N),TOM2(N)            270.
ISN         3           REAL ITM1,ITM2,BIG M/1.E16/                                        280.
ISN         4           INTEGER OSVECT(N)                                                  290.
ISN         5           DO 11 I=1,N                                                        300.
ISN         6        11 READ *,(T(I,J),J=1,2)                                              310.
ISN         7           WRITE(6,100)N                                                      320.
ISN         8       100 FORMAT(1X,'SEQUENCING',I4,' JOBS THROUGH 2 MACHINES',///,          330.
                      C        ' JOB    PROCESSING TIME PROCESSING TIME',//,              340.
                      C        12X,'MACHINE 1',7X,'MACHINE 2',/)                         350.
ISN         9           DO 1 I=1,N                                                         360.
ISN        10         1 WRITE(6,101)I,(T(I,J),J=1,2)                                      370.
ISN        11       101 FORMAT(I3,2F16.2)                                                 380.
                C                                                                          390.
                C       SAVE THE PROCESSING TIME VALUES INTO ARRAY S                       400.
                C                                                                          410.
ISN        12           DO 2 I=1,N                                                         420.
ISN        13           DO 2 J=1,2                                                         430.
ISN        14         2 S(I,J)=T(I,J)                                                      440.
ISN        15           K=1                                                                450.
ISN        16           L=1                                                                460.
                C                                                                          470.
                C       SELECT THE SMALLEST T(I,J) OF THOSE NOT-YET-ASSIGNED JOBS          480.
                C       CALCULATE THE OPTIMAL SEQUENCE AND PLACE IN OSVECT                 490.
                C                                                                          500.
ISN        17         8 IF(T(1,1).GT.T(1,2))THEN                                          510.
ISN        18               SMALL=T(1,2)                                                  520.
ISN        19               IZERO=1                                                       530.
ISN        20               JZERO=2                                                       540.
ISN        21           ELSE                                                              550.
ISN        22               SMALL=T(1,1)                                                  560.
ISN        23               IZERO=1                                                       570.
ISN        24               JZERO=1                                                       580.
ISN        25           END IF                                                            590.
                C                                                                          600.
ISN        26           DO 5 I=2,N                                                        610.
ISN        27           DO 5 J=1,2                                                        620.
ISN        28           IF(T(I,J).LT.SMALL)THEN                                           630.
ISN        29               SMALL=T(I,J)                                                  640.
ISN        30               IZERO=I                                                       650.
ISN        31               JZERO=J                                                       660.
ISN        32           END IF                                                            669.
ISN        33         5 CONTINUE                                                          670.
                C                                                                          680.



                C       IF JZERO = 1, ASSIGN JOB IZERO AS CLOSE TO FRONT AS POSSIBLE       690.
                C       OTHERWISE,    ASSIGN IT AS CLOSE TO END AS POSSIBLE                700.
                C                                                                          710.
ISN        34           IF(JZERO.EQ.1)THEN                                                720.
ISN        35               OSVECT(K)=IZERO                                               730.
ISN        36               K=K+1                                                         740.
ISN        37           ELSE                                                              750.
ISN        38               OSVECT(N-L+1)=IZERO                                           760.
ISN        39               L=L+1                                                         770.
ISN        40           END IF                                                            780.
                C                                                                          790.
                C       ELIMINATE JOB IZERO FROM FURTHER CONSIDERATION                     800.
                C                                                                          810.
ISN        41           T(IZERO,1)=BIG M                                                  820.
ISN        42           T(IZERO,2)=BIG M                                                  830.
                C                                                                          840.
                C       IF ALL JOBS HAVE BEEN ASSIGNED, GOTO THE FOLLOWING STEP            850.
                C       OTHERWISE, REPEAT AL PREVIOUS STEPS                                860.
                C                                                                          870.
ISN        43           IF(L+K.NE.N+2)GOTO 8                                              880.
                C                                                                          890.
                C       CALCULATE TIME IN & TIME OUT                                       900.
                C                                                                          910.
ISN        44           TIM1(1)=0                                                         920.
ISN        45           TOM1(1)=S(OSVECT(1),1)                                            930.
ISN        46           TIM2(1)=TOM1(1)                                                   940.
ISN        47           TOM2(1)=TIM2(1)+S(OSVECT(1),2)                                    950.
ISN        48           ITM2=TIM2(1)                                                      960.
                C                                                                          970.
                C       CALCULATE IDLE TIME FOR EACH MACHINE                               980.
                C                                                                          990.
ISN        49           DO 10 K=2,N                                                       1000.
ISN        50           TIM1(K)=TOM1(K-1)                                                 1010.
ISN        51           TOM1(K)=TIM1(K)+S(OSVECT(K),1)                                    1020.
ISN        52           IF(TOM2(K-1).GE.TOM1(K))THEN                                      1030.
ISN        53               TIM2(K)=TOM2(K-1)                                             1040.
ISN        54           ELSE                                                              1050.
ISN        55               TIM2(K)=TOM1(K)                                               1060.
ISN        56               ITM2=ITM2+TOM1(K)-TOM2(K-1)                                   1070.
ISN        57           END IF                                                            1080.
ISN        58        10 TOM2(K)=TIM2(K)+S(OSVECT(K),2)                                    1090.
ISN        59           ITM1=TOM2(N)-TOM1(N)                                              1100.
                C                                                                          1110.
                C        PRINT SEQUENCE FOR EACH MACHINE                                   1120.
                C                                                                          1130.
ISN        60           WRITE(6,102)                                                      1140.
ISN        61       102 FORMAT(1H-,' OPTIMAL SEQUENCE FOLLOWS:')                          1150.
ISN        62           WRITE(6,103)(OSVECT(I),I=1,N)                                     1160.
ISN        63       103 FORMAT(1X,25I4)                                                   1170.
ISN        64           WRITE(6,104)TOM2(N)                                               1180.
ISN        65       104 FORMAT(1HO,1X,'TOTAL ELAPSED TIME      =',F10.2)                  1190.
ISN        66           WRITE(6,105)ITM1,ITM2                                             1200.
ISN        67       105 FORMAT(1HO,1X,'IDLE TIME ON MACHINE 1 =',F10.2,//                 1210.
                      C           2X,'IDLE TIME ON MACHINE 2 =',F10.2)                   1220.
ISN        68           RETURN                                                            1230.
ISN        69           END                                                               1231.
```

STATISTICS SOURCE STATEMENTS = 69, PROGRAM SIZE = 3376 BYTES, PROGRAM NAME = NJOB2M PAGE: 13.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 13 ******

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(54)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE NJOB3M(N,T,S,TIM1,TIM2,TIM3,TOM1,TOM2,TOM3,OSVECT)      10.
                C                                                                           20.
                C       SEQUENCING  N-JOBS THROUGH 3 MACHINES                               30.
                C                                                                           40.
                C       DETERMINE THE OPTIMAL SEQUENCE OF N JOBS THROUGH 3 MACHINES         50.
                C       WHEN THE SMALLEST PROCESSING TIME ON MACHINE 1 AND/OR MACHINE 3     60.
                C       IS AT LEAST AS GREAT AS THE LARGEST PROCESSING TIME ON MACHINE 2    70.
                C                                                                           80.
                C                                                                           90.
                C       T(I,1)  TIME TO PROCESS JOB I ON MACHINE 1                          100.
                C       T(I,2)  TIME TO PROCESS JOB I ON MACHINE 2                          110.
                C       T(I,3)  TIME TO PROCESS JOB I ON MACHINE 3                          120.
                C                                                                           130.
                C       OSVECT    OPTIMAL SEQUENCE FOR PROCESSING N JOBS THRU 2 MACHINES    140.
                C       TIM1,TOM1 TIME IN AND TIME OUT FOR MACHINE 1 FOR EACH JOB           150.
                C       TIM2,TOM2 TIME IN AND TIME OUT FOR MACHINE 2 FOR EACH JOB           160.
                C       TIM3,TOM3 TIME IN AND TIME OUT FOR MACHINE 3 FOR EACH JOB           170.
                C       ITM1      IDLE TIME ON MACHINE 1                                    180.
                C       ITM2      IDLE TIME ON MACHINE 2                                    190.
                C       ITM3      IDLE TIME ON MACHINE 3                                    200.
                C                                                                           210.
                C                                                                           220.
ISN         2           DIMENSION T(N,3),S(N,3),TIM1(N),TIM2(N),                            230.
                C             TOM1(N),TOM2(N),TIM3(N),TOM3(N)                               240.
ISN         3           REAL ITM1,ITM2,ITM3,BIG M/1.E16/                                    250.
ISN         4           INTEGER OSVECT(N)                                                   260.
ISN         5           WRITE(6,100)N                                                       270.
ISN         6       100 FORMAT(1H-,'SEQUENCING',I4,' JOBS THROUGH 3 MACHINES',///,          280.
                C         ' JOB    PROCESSING TIME PROCESSING TIME',//,                    290.
                C         12X,'MACHINE 1',7X,'MACHINE 2',7X,'MACHINE 3',/)                300.
ISN         7           DO 11 I=1,N                                                         310.
ISN         8        11 READ *,(T(I,J),J=1,3)                                             320.
ISN         9           DO 1 I=1,N                                                          330.
ISN        10         1 WRITE(6,101)I,(T(I,J),J=1,3)                                       340.
ISN        11       101 FORMAT(I3,3F16.2)                                                  350.
                C                                                                           360.
                C       SAVE THE ORIGINAL PROCESSING TIME VALUES INTO ARRAY S               370.
                C                                                                           380.
ISN        12           DO 2 I=1,N                                                         390.
ISN        13           DO 2 J=1,3                                                         400.
ISN        14         2 S(I,J)=T(I,J)                                                      410.
                C                                                                           420.
                C       CONVERT TO A 2-MACHINE PROBLEM                                      430.
                C                                                                           440.
ISN        15           DO 3 I=1,N                                                         450.
ISN        16           T(I,1)=T(I,1)+T(I,2)                                               460.
ISN        17         3 T(I,2)=T(I,2)+T(I,3)                                               470.
ISN        18           K=1                                                                480.
ISN        19           L=1                                                                490.
                C                                                                           500.
                C       USE NJOB2M ALGORITHM TO FIND THE OPT. SEQ OF                        510.
                C       THE NEW N-JOB,2-MACHINE PROBLEM                                     520.
                C                                                                           530.
                C       SELECT THE SMALLEST T(I,J) OF THOSE NOT-YET-ASSIGNED JOBS           540.
                C       CALCULATE THE OPTIMAL SEQUENCE AND PLACE IN OSVECT                  550.
                C                                                                           560.
ISN        20         8 IF(T(1,1).GT.T(1,2))THEN                                           570.
ISN        21             SMALL=T(1,2)                                                     580.
ISN        22             IZERO=1                                                          590.
ISN        23                 JZERO=2                                                      600.
ISN        24           ELSE                                                             610.
ISN        25             SMALL=T(1,1)                                                     620.
ISN        26             IZERO=1                                                          630.
ISN        27             JZERO=1                                                          640.
ISN        28           END IF                                                           650.
                C                                                                           660.
ISN        29           DO 5 I=2,N                                                       670.
ISN        30           DO 5 J=1,2                                                       680.
ISN        31           IF(T(I,J).GE.SMALL)GOTO 5                                        690.
ISN        32           SMALL=T(I,J)                                                     700.
ISN        33           IZERO=I                                                          710.
ISN        34           JZERO=J                                                          720.
ISN        35         5 CONTINUE                                                         730.
                C                                                                           740.
                C       IF JZERO = 1, ASSIGN JOB IZERO AS CLOSE TO FRONT AS POSSIBLE        750.
                C       OTHERWISE,    ASSIGN IT AS CLOSE TO END AS POSSIBLE                 760.
                C                                                                           770.
ISN        36           IF(JZERO.EQ.1)THEN                                               780.
ISN        37             OSVECT(K)=IZERO                                                790.
ISN        38             K=K+1                                                          800.
ISN        39           ELSE                                                             810.
ISN        40             OSVECT(N-L+1)=IZERO                                            820.
ISN        41             L=L+1                                                          830.
ISN        42           END IF                                                           840.
                C                                                                           850.
                C       ELIMINATE JOB IZERO FROM FURTHER CONSIDERATION                      860.
                C                                                                           870.
ISN        43           T(IZERO,1)=BIG M                                                 880.
ISN        44           T(IZERO,2)=BIG M                                                 890.
                C                                                                           900.
                C       IF ALL JOBS HAVE BEEN ASSIGNED, GOTO THE FOLLOWING STEP          910.
                C       OTHERWISE, REPEAT AL PREVIOUS STEPS                              920.
                C                                                                           930.
ISN        45           IF(L+K.NE.N+2)GOTO 8                                             940.
                C                                                                           950.
                C       CALCULATE TIME IN & TIME OUT                                        960.
                C                                                                           970.
ISN        46           TIM1(1)=0                                                        980.
ISN        47           TOM1(1)=S(OSVECT(1),1)                                           990.
ISN        48           TIM2(1)=TOM1(1)                                                  1000.
ISN        49           TOM2(1)=TIM2(1)+S(OSVECT(1),2)                                   1010.
ISN        50           TIM3(1)=TOM2(1)                                                  1020.
ISN        51           TOM3(1)=TIM3(1)+S(OSVECT(1),3)                                   1030.
ISN        52           ITM2=TIM2(1)                                                     1040.
ISN        53           ITM3=TIM3(1)                                                     1050.
                C                                                                           1060.
                C       CALCULATE IDLE TIME FOR EACH MACHINE                                1070.
                C                                                                           1080.
ISN        54           DO 12 K=2,N                                                      1090.
ISN        55           TIM1(K)=TOM1(K-1)                                                1100.
ISN        56           TOM1(K)=TIM1(K)+S(OSVECT(K),1)                                   1110.
ISN        57           IF(TOM2(K-1).GE.TOM1(K))THEN                                     1120.
ISN        58               TIM2(K)=TOM2(K-1)                                            1130.
ISN        59           ELSE                                                             1140.
ISN        60               TIM2(K)=TOM1(K)                                              1150.
ISN        61               ITM2=ITM2+TOM1(K)-TOM2(K-1)                                  1160.
ISN        62           END IF                                                           1170.
ISN        63           TOM2(K)=TIM2(K)+S(OSVECT(K),2)                                   1180.
ISN        64           IF(TOM3(K-1).GE.TOM2(K))THEN                                     1190.
ISN        65               TIM3(K)=TOM3(K-1)                                            1200.
ISN        66           ELSE                                                             1210.
ISN        67               TIM3(K)=TOM2(K)                                              1220.
ISN        68               ITM3=ITM3+TOM2(K)-TOM3(K-1)                                  1230.
ISN        69           END IF                                                           1240.
ISN        70        12 TOM3(K)=TIM3(K)+S(OSVECT(K),3)                                   1250.
```

```
ISN        71          ITM1=TOM3(N)-TOM1(N)                                          1260.
ISN        72          ITM2=ITM2+TOM3(N)-TOM2(N)                                     1270.
                  C                                                                 1280.
                  C       PRINT SEQUENCE FOR EACH MACHINE                           1290.
                  C                                                                 1300.
ISN        73          WRITE(6,102)                                                  1310.
ISN        74        102 FORMAT(1H-,' OPTIMAL SEQUENCE FOLLOWS:')                    1320.
ISN        75          WRITE(6,103)(OSVECT(I),I=1,N)                                 1330.
ISN        76        103 FORMAT(1X,25I4)                                             1340.
ISN        77          WRITE(6,104)TOM3(N)                                           1350.
ISN        78        104 FORMAT(1HO,1X,'TOTAL ELAPSED TIME      =',F10.2)            1360.
ISN        79          WRITE(6,105)ITM1,ITM2,ITM3                                    1370.
ISN        80        105 FORMAT(1HO,1X,'IDLE TIME ON MACHINE 1 =',F10.2,//           1380.
                     C           2X,'IDLE TIME ON MACHINE 2 =',F10.2,//             1390.
                     C           2X,'IDLE TIME ON MACHINE 3 =',F10.2 )              1400.
ISN        81          WRITE(6,106)                                                  1410.
ISN        82        106 FORMAT(1H-,13X,'MACHINE 1',13X,'MACHINE 2',13X,'MACHINE 3',/ 1420.
                     C       11X,'TIME',5(7X,'TIME'),/1X,'JOB',3(8X,'IN',9X,'OUT'),/) 1430.
ISN        83          DO 14 I=1,N                                                   1440.
ISN        84         14 WRITE(6,120)OSVECT(I),ITM1(I),TOM1(I),ITM2(I),TOM2(I),TIM3(I), 1450.
                     C             TOM3(I)                                          1460.
ISN        85        120 FORMAT(I4,6F11.2)                                           1470.
ISN        86          RETURN                                                        1480.
ISN        87          END                                                           1490.

*STATISTICS*   SOURCE STATEMENTS = 87, PROGRAM SIZE = 4710 BYTES, PROGRAM NAME = NJOB3M     PAGE:   14.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 14 ******
```

```
LEVEL 1.2.0 (SEPT  82)          VS FORTRAN          DATE: 1984 APR 13      TIME: 11:49:32      NAME: MAIN     PAGE:   15

OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(54)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN        1          SUBROUTINE PERT(M,N,C,CPATH,SD,ET,LT,SL,SIGTE,FLAG)           100.
                  C                                                                 110.
                  C       P  E  R  T                                                120.
                  C                                                                 130.
                  C       OUTPUT:                                                   140.
                  C                                                                 150.
                  C       DETERMINE THE CRITICAL PATH IN A PERT NETWORK AND WILL PROVIDE 160.
                  C       THE PROBABILITY OF COMPLETING EACH EVENT ON OR BEFORE THE  170.
                  C       SCHEDULED COMPLETION TIME.                                180.
                  C                                                                 190.
                  C       INPUT:                                                    200.
                  C                                                                 210.
                  C       C(N,7):    DESCRIPTION OF THE NETORK                      220.
                  C                                                                 230.
                  C         C(I,1) - ACTIVITY FROM EVENT I                          240.
                  C         C(I,2) - ACTIVITY TO EVENT J                            250.
                  C         C(I,3) - OPTIMISTIC COMPLETION TIME FOR ACTIVITY (I,J)  260.
                  C         C(I,4) - MOST LIKELY COMPLETION TIME FOR ACTIVITY(I,J)  270.
                  C         C(I,5) - PESSIMISTIC COMPLETION TIME FOR ACTIVITY(I,J)  280.
                  C         C(I,6) - ESTIMATED MEAN COMPLETION TIME FOR ACTIVITY(I,J) 290.
                  C         C(I,7) - ESTIMATED VARIANCE OF COMPLETION TIME OF ACTIVITY(I,J)300.
                  C                                                                 310.
                  C       M          - MAXIMUM # OF EVENTS ON NETWORK               320.
                  C       N          - MAXIMUM # OF ACTIVITIES ON NETWORK           330.
                  C                                                                 340.
                  C       SD(J)      - SCHEDULED COMPLETION TIME FOR EVENT J        350.
                  C                                                                 360.
                  C       INTERMEDIATE VARIABLES:                                   370.
                  C                                                                 380.
                  C       ET(M)      - EARLIEST EXPECTED COMPLETION TIME OF EVENT J 390.
                  C       LT(M)      - LATEST ALLOWABLE EVENT COMPLETION TIME FOR EVENT I 400.
                  C       SIGTE(M) - SUM OF VARIANCES OF ACTIVITY TIME IN PATH USED TO 410.
                  C                    CALCULATE ET(J)                              420.
                  C       CPATH(N) - CRITICAL PATH                                  430.
                  C                                                                 440.
                  C                                                                 450.
ISN        2          INTEGER M,N,CPATH(N),FLAG(N)                                  460.
ISN        3          REAL C(N,7),SD(M),ET(M),LT(M),SL(M),SIGTE(M),                 470.
                     C     BIG M/1.E12/,ERROR/1.E-4/                                480.
ISN        4          DO 1 I=1,N                                                    490.
ISN        5          READ(5,*)(C(I,J),J=1,5)                                       500.
ISN        6        1 FLAG(I)=0                                                     510.
                  C                                                                 530.
                  C       CALCULATE ESTIMATED MEAN COMPLETION TIME OF ACTIVITY T(I,J) 540.
                  C       CALCULATE ESTIMATED VARIANCE OF COMPLETION TIME FOR EACH   550.
                  C                 EVENT VT(I,J)                                   560.
                  C                                                                 570.
ISN        7          WRITE(6,2)                                                    580.
ISN        8        2 FORMAT(1H-,31X 'P  E  R  T',//,                               590.
                     C6X,'I',9X,'J',10X,'A',9X,'M',9X,'B',6X,'T(I,J)   VT(I,J)'/)  591.
ISN        9          DO 3 I=1,N                                                    600.
ISN       10          C(I,6)=(C(I,3)+4.*C(I,4)+C(I,5))/6.                           610.
ISN       11          C(I,7)=(C(I,5)**2 - 2*C(I,5)*C(I,3) + C(I,3)**2)/36.          620.
ISN       12        3 WRITE(6,4)(C(I,J),J=1,7)                                      630.
ISN       13        4 FORMAT(7F10.3)                                                640.
                  C                                                                 650.
                  C       CALCALATE EARLIEST EXPECTED COMPLETION TIME FOR EVENT J - ET(J) 660.
                  C                                                                 670.
ISN       14          ET(1)=0.                                                      680.
```

```
ISN        15          SIGTE(1)=0.                                                    690.
ISN        16          DO 6 J=2,M                                                     700.
ISN        17          TMAX=0.                                                        710.
ISN        18          SIGMAX=0.                                                      720.
ISN        19          DO 5 K=1,N                                                     730.
ISN        20          IF(C(K,2).EQ.J)THEN                                            740.
ISN        21                ET(J)=ET(C(K,1))+C(K,6)                                  750.
ISN        22                SIGTE(J)=SIGTE(C(K,1))+C(K,7)                            760.
ISN        23                IF(ET(J).GT.TMAX)THEN                                    770.
ISN        24                     TMAX=ET(J)                                          780.
ISN        25                     SIGMAX=SIGTE(J)                                     790.
ISN        26                END IF                                                   800.
ISN        27              END IF                                                     810.
ISN        28            5 CONTINUE                                                   820.
ISN        29              ET(J)=TMAX                                                 830.
ISN        30              SIGTE(J)=SIGMAX                                            840.
ISN        31            6 CONTINUE                                                   850.
                     C                                                              860.
                     C       CALCULATE LATEST ALLOWABLE COMPLETION TIME FOR EVENT J - LT(J)  870.
                     C                                                              880.
ISN        32              LT(M)=ET(M)                                                890.
ISN        33              SD(M)=LT(M)                                                900.
ISN        34              DO 8 I=M-1,1,-1                                            910.
ISN        35              TMIN=BIG M                                                 930.
ISN        36              DO 7 K=N,1,-1                                              940.
ISN        37              IF(C(K,1).EQ.I)THEN                                        950.
ISN        38                   LT(I)=LT(C(K,2))-C(K,6)                              970.
ISN        39                   IF(LT(I).LT.TMIN)TMIN=LT(I)                          980.
ISN        41              END IF                                                     990.
ISN        42            7 CONTINUE                                                   1000.
ISN        43              LT(I)=TMIN                                                 1010.
ISN        44            8 CONTINUE                                                   1020.
                     C                                                              1030.
                     C        CALCULATE SLACK TIME FOR EACH EVENT J - SL(J)          1040.
                     C                                                              1050.
ISN        45              WRITE(6,9)                                                 1060.
ISN        46            9 FORMAT(//,19X,'EARLIEST',8X,'LATEST',                    1070.
                      C         /,7X,'EVENT',11X,'TIME',11X,'TIME',10X,'SLACK')     1080.
ISN        47              DO 10 J=1,M                                                1090.
ISN        48              SL(J)=LT(J)-ET(J)                                          1100.
ISN        49           10 WRITE(6,11)J,ET(J),LT(J),SL(J)                             1110.
ISN        50           11 FORMAT(/7X,I5,3F15.3)                                      1120.
                     C                                                              1130.
                     C       FIND ACTIVITIES ON CRITICAL PATH                       1140.
                     C                                                              1150.
ISN        51              DO 12 K=1,N                                                1160.
ISN        52              I=C(K,1)                                                   1170.
ISN        53              J=C(K,2)                                                   1180.
ISN        54              IF(ABS(LT(I)-ET(I)-SL(M)).LE.ERROR)THEN                    1190.
ISN        55              IF(ABS(LT(J)-ET(J)-SL(M)).LE.ERROR)THEN                    1200.
ISN        56              IF(ABS(ET(J)-ET(I)-LT(J)+LT(I)).LE.ERROR.AND.              1210.
                        C    ABS(LT(J)-LT(I)-C(K,6)).LE.ERROR)     FLAG(K)=1        1220.
ISN        58              END IF                                                     1230.
ISN        59              END IF                                                     1240.
ISN        60           12 CONTINUE                                                   1250.
ISN        61              CPATH(1)=1                                                 1260.
ISN        62              KP=1                                                       1270.
ISN        63              DO 14 K=1,N                                                1280.
ISN        64              I=C(K,1)                                                   1290.
ISN        65              J=C(K,2)                                                   1300.
ISN        66              IF(FLAG(K).NE.0)THEN                                       1310.
ISN        67              IF(I.EQ.CPATH(KP))THEN                                     1320.
ISN        68                   KP=KP+1                                             1330.
ISN        69                   CPATH(KP)=J                                         1340.
ISN        70              END IF                                                     1350.
ISN        71              END IF                                                     1360.


ISN        72           14 CONTINUE                                                   1370.
ISN        73              WRITE(6,15)(CPATH(I),I=1,KP)                               1380.
ISN        74           15 FORMAT(//5X,'CRITICAL PATH',15I4)                          1390.
                     C                                                              1400.
                     C       CALCULATE CRITICAL VALUE OF CUMULATIVE N(0,1) DISTRIBUTION  1410.
                     C       TO DETERMINE PROBABILITY OF COMPLETING EVENT BEFORE SCHEDULED  1420.
                     C       COMPLETION TIME                                        1430.
                     C                                                              1440.
                     C       MDNOR - NORMAL PROBABILITY DISTRIBUTION FUNCTION (IMSL LIBRARY)  1450.
                     C                                                              1460.
ISN        75              READ(5,*,END=999)(SD(I),I=1,M)                             1461.
ISN        76              WRITE(6,16)                                                1470.
ISN        77           16 FORMAT(//19X,'EARLIEST',12X,'DUE',9X,'PROBA-',             1480.
                        C         /7X,'EVENT',11X,'TIME',11X,'DATE',9X,'BILITY')    1490.
ISN        78              DO 17 K=2,M                                                1500.
ISN        79              X=(SD(K)-ET(K))/SQRT(SIGTE(K))                             1510.
ISN        80              CALL MDNOR(X,PROB)                                         1520.
ISN        81           17 WRITE(6,11)K,ET(K),SD(K),PROB                              1530.
ISN        82          999 RETURN                                                     1540.
ISN        83              END                                                        1550.

*STATISTICS*    SOURCE STATEMENTS = 81, PROGRAM SIZE = 4994 BYTES, PROGRAM NAME = PERT     PAGE:   15.

*STATISTICS*     NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 15 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN         1           SUBROUTINE DIJKST(M,VO,W,A,FROM,TO,LENGTH,EDGES)
                C
                C     THIS SUBROUTINE DETERMINES THE SHORTEST DISTANCE FROM ININITIAL
                C     VERTEX VO TO A FINAL VERTEX W IN A NON-NEGATIVELY WEIGHTED,
                C     CONNECTED NETWORK G. USING AN ALGORITHM DUE TO DIJKSTRA
                C     (NUMERISCHE MATHEMATIK, VOL.1, PP.269-271.)
                C     THIS IMPLEMENTATION IS A MODIFICATION OF A PROGRAM DUE TO
                C     V. KEVIN M. WHITNEY(COMM. ACM, 1972 APRIL), ALGORITHM 422.
                C     THE OUTPUT OF THIS SUBROUTINE IS A SHORTEST PATH TREE ROOTED
                C     AT VO
                C
                C  **  INPUT PARAMETERS **
                C
                C    A(I,J)      LENGTH OF THE EDGE BETWEEN VERTICES I & J
                C                IF THERE IS NO EDGE BETWEEN I AND J THEN A(I,J)=1000000
                C                AN ARBITRARY LARGE NUMBER
                C
                C    VO          INITIAL VERTEX
                C
                C    W           FINAL VERTEX
                C
                C    M           THE VERTICES OF G ARE NUMBERED 1,2,..,M
                C                WHERE 2 < M < 100
                C
                C  **  OUTPUT PARAMETERS **
                C
                C    FROM(I)     CONTAIN THE I-TH EDGE IN THE SHORTEST PATH TREE
                C
                C    TO(I)       FROM VERTEX -FROM(I)- TO VERTEX -TO(I)-
                C
                C    LENGTH(I)   OF LENGTH -LENGTH(I)-
                C
                C  ** VARIABLES **
                C
                C    DIST(I)     THE SHORTEST DISTANCE FROM UNDET(I) TO THE PARTIAL
                C                SHORTEST PATH TREE
                C
                C    NEXT        NEXT VERTEX TO BE ADDED TO THE SHORTEST PATH TREE
                C
                C    NUMUN       NUMBER OF UNDETERMINED VERTICES
                C
                C    UNDET(I)    LIST OF UNDETERMINED VERTICES
                C
                C    VERTEX(I)   VERTEX OF PARTIAL SHORTEST PATH TREE ON SHORTEST PATH
                C                FROM UNDET(I) TO VO
                C
ISN         2           INTEGER M,A(100,M),VO,W,FROM(M),TO(M),LENGTH(M),EDGES
ISN         3           INTEGER DIST(100),NEXT,NUMUN,UNDET(100),VERTEX(100)
                C
                C     INITIALIZE CONDITIONS
                C
ISN         4           EDGES=0
ISN         5           NEXT=VO
ISN         6           NUMUN=M-1
                C
                C     LABEL ALL VERTICES AS "UNDETERMINED"
                C     SET ALL VERTICES AS UNCOLORED VERTICES
                C
                C     INITIALIZE VARIABLE DIST


                C
ISN         7           DO 100 I=1,M
ISN         8                UNDET(I)=I
ISN         9                DIST(I)=A(VO,I)
ISN        10                VERTEX(I)=VO
ISN        11      100 CONTINUE
ISN        12          UNDET(VO)=M
ISN        13          DIST(VO)=DIST(M)
ISN        14          GOTO 350
                C
                C     ITERATE M-1 TIMES
                C
ISN        15      200 DO 300 I=1,NUMUN
ISN        16               J=UNDET(I)
ISN        17               JK=L+A(NEXT,J)
                C
                C     UPDATE SHORTEST DISTANCE TO EACH UNDETERMINED VERTEX
                C
ISN        18               IF(DIST(I).LE.JK)GOTO 300
ISN        19                    VERTEX(I)=NEXT
ISN        20                    DIST(I)=JK
ISN        21      300 CONTINUE
                C
                C     PICK A SHORTEST DISTANCE TO AN UNDETERMINED VERTEX
                C
ISN        22      350 K=1
ISN        23          L=DIST(1)
ISN        24          DO 400 I=1,NUMUN
ISN        25               IF(DIST(I).LT.L)THEN
ISN        26                    L=DIST(I)
ISN        27                    K=I
ISN        28               END IF
ISN        29      400 CONTINUE
                C
                C     ADD EDGE TO SHORTEST PATH TREE
                C
ISN        30          EDGES=EDGES+1
ISN        31          FROM(EDGES)=VERTEX(K)
ISN        32          TO(EDGES)=UNDET(K)
ISN        33          LENGTH(EDGES)=L
ISN        34          NEXT=UNDET(K)
                C
                C     HAVE WE REACHED W ?
                C
ISN        35          IF(NEXT.EQ.W)GOTO 500
                C
                C     DELETE NEWLY DETERMINED VERTEX FROM UNDETERMINED LIST
                C
ISN        36          DIST(K)=DIST(NUMUN)
ISN        37          UNDET(K)=UNDET(NUMUN)
ISN        38          VERTEX(K)=VERTEX(NUMUN)
                C
                C     ANY UNDETERMINED VERTICES LEFT
                C
ISN        39          NUMUN=NUMUN-1
ISN        40          GOTO 200
                C
ISN        41      500 WRITE(6,600)VO,W,LENGTH(EDGES)
ISN        42      600 FORMAT('-','THE SHORTEST PATH FROM',I3,' TO',I3,' =',I3)
ISN        43          RETURN
ISN        44          END
```

STATISTICS SOURCE STATEMENTS = 44, PROGRAM SIZE = 2872 BYTES, PROGRAM NAME = DIJKST PAGE: 16.

STATISTICS NO DIAGNOSTICS GENERATED.

```
****** END OF COMPILATION 16 ******
```

```
LEVEL 1.2.0 (SEPT  82)          VS FORTRAN          DATE: 1984 APR 13      TIME: 11:49:33       NAME: MAIN     PAGE:    17

OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(654)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE FLOYD(A,M)
                C
                C       FLOYD-WARSHALL METHOD
                C
                C       A  -  ADJACENCY MATRIX
                C       M  -  SIZE OF THE MATRIX A
                C
ISN         2           INTEGER M,A(100,M),S,BIG M/999999999/
ISN         3           DO 1 I=1,M
ISN         4           DO 1 J=1,M
                C
                C       BIG M   REPRESENTS THE NON-EXISTING ARC
                C
ISN         5           IF(A(J,I).GE.BIG M)GOTO 1
ISN         6                DO 2 K=1,M
ISN         7                IF(A(I,K).GE.BIG M)GOTO 2
                C
                C           A(J,K) = MINIMUM(  A(J,K),  A(J,I)+A(I,K)  )
                C
ISN         8                S=A(J,I)+A(I,K)
ISN         9                IF(S.LT.A(J,K))A(J,K)=S
ISN        11            2   CONTINUE
ISN        12            1 CONTINUE
ISN        13            WRITE(6,5)
ISN        14            DO 3 I=1,M
ISN        15          3 WRITE(6,4)(A(I,J),J=1,M)
ISN        16          4 FORMAT(1X,2013)
ISN        17          5 FORMAT(1H-,'ALL SHORTEST PATH MATRIX:',//)
ISN        18            RETURN
ISN        19            END

*STATISTICS*    SOURCE STATEMENTS = 18, PROGRAM SIZE = 1082 BYTES, PROGRAM NAME = FLOYD     PAGE:    17.

*STATISTICS*     NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 17 ******
```

```
LEVEL 1.2.0 (SEPT  82)          VS FORTRAN          DATE: 1984 APR 13     TIME: 11:49:33      NAME: MAIN     PAGE:   18

OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(654)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE NETPTH(A,M,VO,PRECED,MINCOS)                                  10.
                C                                                                              11.
                C       MINPTH  -  SHORTEST PATH BETWEEN START NODE AND ALL OTHER NODES         12.
                C                  OF A NETWORK                                                13.
                C                                                                              14.
ISN         2           IMPLICIT INTEGER(A-Z)                                                  20.
ISN         3           INTEGER A(100,M),PRECED(M),MINCOS(M),SCAN(100),                        30.
                     C         BIG M/2147483647/                                               31.
ISN         4           DO 1 I=1,M                                                             50.
ISN         5                SCAN(I)=0                                                         60.
ISN         6                PRECED(I)=0                                                       70.
ISN         7                MINCOS(I)=BIG M                                                   80.
ISN         8             1 CONTINUE                                                           90.
ISN         9           MINCOS(VO)=0                                                           100.
ISN        10           SCAN(VO)=1                                                             110.
                C                                                                              120.
                C       ITERATE BEGINS                                                         130.
                C                                                                              140.
ISN        11         100 DO 50 I=1,M                                                         150.
ISN        12                IF(SCAN(I).EQ.0)GOTO 50                                          160.
ISN        13                     MINI=MINCOS(I)                                              170.
ISN        14                     DO 40 J=1,M                                                 180.
ISN        15                          JCOST=A(I,J)+MINI                                      190.
ISN        16                          IF(JCOST.LT.MINCOS(J))THEN                             200.
ISN        17                               MINCOS(J)=JCOST                                   210.
ISN        18                               SCAN(J)=1                                         220.
ISN        19                               PRECED(J)=I                                       230.
ISN        20                          END IF                                                 231.
ISN        21          40             CONTINUE                                               240.
ISN        22                         SCAN(I)=0                                              250.
ISN        23                         GOTO 100                                               260.
ISN        24          50 CONTINUE                                                           270.
ISN        25             WRITE(6,60)VO,(MINCOS(I),I=1,M)                                    271.
ISN        26          60 FORMAT(1H-,'MINIMUM COST FROM NODE',I3,' TO ALL OTHER NODES:',/,   272.
                        C       5(40I3,/) )                                                   273.
ISN        27           RETURN                                                               280.
ISN        28           END                                                                  290.

*STATISTICS*    SOURCE STATEMENTS = 28, PROGRAM SIZE = 1678 BYTES, PROGRAM NAME = NETPTH     PAGE:   18.

*STATISTICS*     NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 18 ******
```

```
LEVEL 1.2.0 (SEPT  82)          VS FORTRAN          DATE: 1984 APR 13     TIME: 11:49:33      NAME: MAIN     PAGE:   19

OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(654)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE SHORTP(M,VO,W,FROM,TO,EDGES)                                10.
                C                                                                              11.
                C       SHORTP  -  THE LIST OF NODES ON THE SHORTEST PATH                      12.
                C                  FROM START NODE TO END NODE OF A NETWORK                    13.
                C                                                                              14.
ISN         2           INTEGER FROM(100),TO(100),VO,W,M,EDGES                                 20.
ISN         3           J=W                                                                    30.
ISN         4           TO(M)=W                                                                40.
ISN         5           K=M-1                                                                  50.
                C                                                                              60.
                C       REPEAT UNTIL VO IS NOT FOUND IN FROM(I)                                70.
                C                                                                              80.
ISN         6         100 I=FROM(J)                                                            90.
ISN         7           IF(I.NE.VO)THEN                                                        100.
ISN         8                J=I                                                             110.
ISN         9                TO(K)=I                                                         120.
ISN        10                K=K-1                                                           130.
ISN        11                GOTO 100                                                        140.
ISN        12           END IF                                                               150.
                C                                                                              160.
ISN        13           TO(1)=VO                                                             170.
ISN        14           J=K+1                                                                180.
ISN        15           K=2                                                                  190.
ISN        16           DO 50 L=J,M                                                          200.
ISN        17                TO(K)=TO(L)                                                     210.
ISN        18                K=K+1                                                           220.
ISN        19         50 CONTINUE                                                           230.
                C                                                                              231.
                C       EMPTY THE REMAINING LIST                                               232.
                C                                                                              233.
ISN        20           DO 20 J=K,M                                                          240.
ISN        21         20 TO(J)=0                                                             250.
ISN        22           EDGES=K-1                                                            251.
ISN        23           WRITE(6,30)VO,W,(TO(I),I=1,EDGES)                                    252.
ISN        24         30 FORMAT(1H-,'SHORTEST PATH FROM NODE',I3,' TO',I3,/,                 253.
                        C       5(40I3,/) )                                                   254.
ISN        25           RETURN                                                               260.
ISN        26           END                                                                  270.

*STATISTICS*    SOURCE STATEMENTS = 26, PROGRAM SIZE = 908 BYTES, PROGRAM NAME = SHORTP     PAGE:   19.

*STATISTICS*     NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 19 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(54)

```
                *....*...1.....2.....3.....4.....5.....6.....7.*.....8
ISN         1           SUBROUTINE NORTHW(M,N,ALLOC,FACTRY,WAREHO)                          2760.
ISN         2           IMPLICIT INTEGER(A-Z)                                               2770.
ISN         3           INTEGER ALLOC(M,N),FACTRY(M),WAREHO(N)                              2780.
ISN         4           REAL TOTAL                                                          2791.
ISN         5           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                        2800.
                C                                                                           2810.
                C       NORTH-WEST CORNER METHOD:                                           2820.
                C            ALWAYS START FROM POSITION (1,1) AS A CONVENTIONAL START PT    2830.
                C            (AS A MATTER OF FACT, START FROM SOUTH-EAST IS PERFECTLY        2840.
                C             ALL RIGHT, JUST AN ARBITARY POSITION)                         2850.
                C                                                                           2860.
                C                                                                           2870.
                C       PRESERVE THE ORIGINAL PARAMETERS                                    2880.
                C       COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO        2890.
                C                                                                           2900.
ISN         6           DO 1 I=1,M                                                          2910.
ISN         7           FACTRY(I)=SUPPLY(I)                                                 2911.
ISN         8           DO 1 J=1,N                                                          2920.
ISN         9         1 ALLOC(I,J)=0                                                        2930.
ISN        10           DO 2 J=1,N                                                          2960.
ISN        11         2 WAREHO(J)=DEMAND(J)                                                 2970.
ISN        12           COL=1                                                               2980.
ISN        13           ROW=1                                                               2990.
                C                                                                           3000.
                C       RANK = MINIMUM(ROW,COL)                                             3010.
                C                                                                           3020.
ISN        14           RANK=MINO(M,N)                                                      3030.
ISN        15           DO 10 IT=1,RANK                                                     3040.
                C                                                                           3050.
                C       SUMCOL = SUM OF FACTORY(1) TO FACTORY(IT)                           3060.
                C       SUMROW = SUM OF WAREHOUSE(1) TO WAREHOSE(IT)                        3070.
                C                                                                           3080.
ISN        16           SUMCOL=0                                                            3090.
ISN        17           DO 3 I=1,ROW                                                        3100.
ISN        18         3 SUMCOL=SUMCOL+ALLOC(I,COL)                                          3110.
ISN        19           SUMROW=0                                                            3120.
ISN        20           DO 4 J=1,COL                                                        3130.
ISN        21         4 SUMROW=SUMROW+ALLOC(ROW,J)                                          3140.
                C                                                                           3150.
                C     TAKE THE MINIMUM OF SUPPLY & DEMAND AT ALLOCATION(I,J)                3160.
                C                                                                           3170.
ISN        22           ALLOC(ROW,COL)=MINO(FACTRY(ROW)-SUMROW,WAREHO(COL)-SUMCOL)          3180.
                C                                                                           3190.
                C       HIT LAST ROW OR COL?                                                3200.
                C                                                                           3210.
                C       IF(ROW.EQ.RANK.AND.M.LE.N .OR. COL.EQ.RANK.AND.N.LE.M)GOTO 11       3220.
ISN        23           IF(IT.EQ.RANK)GOTO 11                                               3230.
                C                                                                           3240.
                C       FILL UP ALLOC(I+1,J) OR ALLOC(I,J+1) DEPENDS ON THE RIM CONDITION   3250.
                C                                                                           3260.
ISN        24           IF(ALLOC(ROW,COL).LT.FACTRY(ROW))GOTO 7                             3270.
                C                                                                           3280.
                C      FULFIL THE WAREHOUSE(I) REQUIREMENT                                  3290.
                C                                                                           3300.
ISN        25         5 SUMCOL=SUMCOL+ALLOC(ROW,COL)                                        3310.
ISN        26           IF(SUMCOL.EQ.WAREHO(COL))GOTO 6                                     3320.
ISN        27           ALLOC(ROW+1,COL)=MINO(WAREHO(COL)-SUMCOL,FACTRY(ROW+1))             3330.
ISN        28           ROW=ROW+1                                                           3340.
ISN        29           GOTO 5                                                              3350.


ISN        30         6 COL=COL+1                                                           3360.
ISN        31           GOTO 10                                                             3370.
                C                                                                           3380.
                C       SEND OUT FACTORY(I) PRODUCTION                                      3390.
                C                                                                           3400.
ISN        32         7 SUMROW=SUMROW+ALLOC(ROW,COL)                                        3410.
ISN        33           IF(SUMROW.EQ.FACTRY(ROW))GOTO 8                                     3420.
ISN        34           ALLOC(ROW,COL+1)=MINO(FACTRY(ROW)-SUMROW,WAREHO(COL+1))             3430.
ISN        35           COL=COL+1                                                           3440.
ISN        36           GOTO 7                                                              3450.
ISN        37         8 ROW=ROW+1                                                           3460.
ISN        38        10 CONTINUE                                                            3470.
                C                                                                           3480.
                C       FILL UP THE REMAINING ROW                                           3490.
                C                                                                           3500.
ISN        39        11 SUM=0                                                              3510.
ISN        40           IF(M.GT.N)GOTO 14                                                   3520.
ISN        41           DO 12 J=1,COL                                                       3530.
ISN        42           RP1=COL+1                                                           3540.
ISN        43        12 SUM=SUM+ALLOC(RANK,J)                                              3550.
ISN        44           DO 13 J=RP1,N                                                       3560.
ISN        45           ALLOC(RANK,J)=MINO(FACTRY(RANK)-SUM,WAREHO(J))                      3570.
ISN        46        13 SUM=SUM+ALLOC(RANK,J)                                              3580.
ISN        47           GOTO 17                                                             3590.
                C                                                                           3600.
                C       FILL UP THE REMAINING COL                                           3610.
                C                                                                           3620.
ISN        48        14 DO 15 I=1,ROW                                                      3630.
ISN        49        15 SUM=SUM+ALLOC(I,RANK)                                              3640.
ISN        50           RP1=ROW+1                                                           3650.
ISN        51           DO 16 I=RP1,M                                                       3660.
ISN        52           ALLOC(I,RANK)=MINO(WAREHO(RANK)-SUM,FACTRY(I))                      3670.
ISN        53        16 SUM=SUM+ALLOC(I,RANK)                                              3680.
                C                                                                           3690.
ISN        54        17 CONTINUE                                                           3700.
                C                                                                           3710.
                C       CALCULATE THE TOTAL COST FOR THE B.F.S IN NORTH-WEST METHOD          3720.
                C                                                                           3730.
ISN        55           TOTAL=0.                                                            3740.
ISN        56           DO 20 I=1,M                                                         3750.
ISN        57           DO 20 J=1,N                                                         3760.
ISN        58        20 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))                            3770.
ISN        59           WRITE(6,21)TOTAL                                                    3780.
ISN        60        21 FORMAT('0','TOTAL COST IN  NORTH-WEST CORNER METHOD = ',E16.10)    3790.
                C                                                                           3800.
                C       OUTPUT THE ALLOCATION MATRIX                                        3810.
                C                                                                           3820.
ISN        61           DO 18 I=1,M                                                         3830.
ISN        62        18 WRITE(6,19)(ALLOC(I,J),J=1,N)                                      3840.
ISN        63        19 FORMAT(1X,33I4)                                                    3850.
ISN        64           RETURN                                                              3860.
ISN        65           END                                                                 3870.
```

STATISTICS SOURCE STATEMENTS = 65, PROGRAM SIZE = 3120 BYTES, PROGRAM NAME = NORTHW PAGE: 20.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 20 ******

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(54)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN         1           SUBROUTINE CHEAP(M,N,ALLOC,FACTRY,WAREHO,COSTS)                    3880.
ISN         2           IMPLICIT INTEGER(A-Z)                                              3890.
ISN         3           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                       3900.
ISN         4           INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N)                  3910.
ISN         5           INTEGER HIGH/2147483647/                                           3920.
ISN         6           REAL TOTAL                                                         3930.
                C                                                                          3940.
                C       CHEAPEST ROUTE METHOD:                                             3950.
                C                                                                          3960.
                C       FILL UP THE CELL WITH THE LOWEST COST FIRST AND SO ON              3970.
                C                                                                          3980.
                C                                                                          3990.
                C       PRESERVE THE ORIGINAL PARAMETERS                                   4000.
                C       COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO       4010.
                C                                                                          4020.
ISN         7           DO 1 I=1,M                                                         4030.
ISN         8           FACTRY(I)=SUPPLY(I)                                                4031.
ISN         9           DO 1 J=1,N                                                         4040.
ISN        10           COSTS(I,J)=COST(I,J)                                               4050.
ISN        11         1 ALLOC(I,J)=0                                                       4060.
ISN        12           DO 2 J=1,N                                                         4080.
ISN        13         2 WAREHO(J)=DEMAND(J)                                                4100.
                C                                                                          4110.
                C       FIND THE LOWEST COST ON COST MAP "COSTS"                           4120.
                C                                                                          4130.
ISN        14         3 LOWEST=COSTS(1,1)                                                  4140.
ISN        15           ROW=1                                                              4150.
ISN        16           COL=1                                                              4160.
ISN        17           DO 4 I=1,M                                                         4170.
ISN        18           DO 4 J=1,N                                                         4180.
ISN        19           IF(COSTS(I,J).GE.LOWEST)GOTO 4                                     4190.
ISN        20           LOWEST=COSTS(I,J)                                                  4200.
ISN        21           ROW=I                                                              4210.
ISN        22           COL=J                                                              4220.
ISN        23         4 CONTINUE                                                           4230.
                C                                                                          4240.
                C       THE CHEAPEST ROUTE IS ASSIGNED WITH A HIGH VALUE NUMBER            4250.
                C       AFTER BEING USED                                                   4260.
                C                                                                          4270.
ISN        24           COSTS(ROW,COL)=HIGH                                                4280.
ISN        25           ALLOC(ROW,COL)=MIN0(FACTRY(ROW),WAREHO(COL))                       4290.
ISN        26           FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                             4300.
ISN        27           WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                             4310.
                C                                                                          4320.
                C       FIND THE NEXT CHEAPEST ROUTE IF THE REQUIREMENT OF WAREHOUSES      4330.
                C       AND FACTORIES ARE NOT MET                                          4340.
                C                                                                          4350.
ISN        28           DO 5 I=1,M                                                         4360.
ISN        29         5 IF(FACTRY(I).GT.0)GOTO 3                                           4370.
ISN        30           DO 6 J=1,N                                                         4380.
ISN        31         6 IF(WAREHO(J).GT.0)GOTO 3                                           4390.
                                                                                           4400.
                C                                                                          4410.
                C                                                                          4420.
                C       CALCULATE THE TOTAL COST FOR THE B.F.S IN CHEAPEST ROUTE            4430.
                C                                                                          4440.
ISN        32           TOTAL=0.                                                           4450.
ISN        33           DO 20 I=1,M                                                        4460.
ISN        34           DO 20 J=1,N                                                        4470.


ISN        35        20 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))                            4480.
ISN        36           WRITE(6,21)TOTAL                                                   4490.
ISN        37        21 FORMAT('0','TOTAL COST IN  CHEAPEST ROUTE METHOD = ',E16.10)       4500.
                C                                                                          4510.
                C       OUTPUT THE ALLOCATION MATRIX                                       4520.
                C                                                                          4530.
ISN        38           DO 18 I=1,M                                                        4540.
ISN        39        18 WRITE(6,19)(ALLOC(I,J),J=1,N)                                      4550.
ISN        40        19 FORMAT(1X,33I4)                                                    4560.
ISN        41           RETURN                                                             4570.
ISN        42           END                                                                4580.

*STATISTICS*   SOURCE STATEMENTS = 42, PROGRAM SIZE = 2424 BYTES, PROGRAM NAME = CHEAP    PAGE:   21.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 21 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.....2.....3.....4.....5.....6.....7.*.....8
ISN         1           SUBROUTINE VAM1(M,N,ALLOC,COSTS,FACTRY,WAREHO,VAMROW,VAMCOL)      4590.
ISN         2           IMPLICIT INTEGER(A-Z)                                             4600.
ISN         3           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                      4610.
ISN         4           INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),                4620.
                       C          VAMROW(M),VAMCOL(N)                                    4630.
ISN         5           INTEGER HIGH/2147483647/                                          4640.
ISN         6           REAL TOTAL                                                        4650.
                    C                                                                    4660.
                    C     MODIFIED V.A.M. METHOD:                                        4670.
                    C                                                                    4680.
                    C     CALCULATE THE VAM NUMBERS BY TAKING THE DIFFERENCE BETWEEN THE 4690.
                    C     CHEAPEST ROUTE AND THE SECOND CHEAPEST ROUTE.                  4700.
                    C     PICK UP THE HIGHEST VAM NUMBER AMONG THE ROWS AND COLUMNS      4710.
                    C     FILL UP THAT ROW OR COLUMN FROM THE CHEAPEST COST.             4720.
                    C     IF A TIE IS FOUND ON THE VAM NUMBERS, TEST THOSE ROWS/COLS     4730.
                    C     WITH THE BECTOR'S NUMBER, TAKE THE ROUTE WITH THE HIGHEST      4740.
                    C     BECTOR NUMBER.                                                 4750.
                    C                                                                    4760.
                    C                                                                    4770.
                    C     PRESERVE THE ORIGINAL PARAMETERS                               4780.
                    C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO   4790.
                    C                                                                    4800.
ISN         7           DO 1 I=1,M                                                       4810.
ISN         8           FACTRY(I)=SUPPLY(I)                                              4811.
ISN         9           DO 1 J=1,N                                                       4820.
ISN        10           COSTS(I,J)=COST(I,J)                                             4830.
ISN        11         1 ALLOC(I,J)=0                                                     4840.
ISN        12           DO 2 J=1,N                                                       4870.
ISN        13         2 WAREHO(J)=DEMAND(J)                                            4880.
                    C                                                                    4890.
                    C     GENERATE THE VAM NUMBERS FOR ROWS                              4900.
                    C                                                                    4910.
ISN        14         3 DO 6 I=1,M                                                       4920.
ISN        15           IF(FACTRY(I).EQ.0)VAMROW(I)=0                                    4930.
ISN        17           IF(FACTRY(I).EQ.0)GOTO 6                                         4940.
ISN        18           LOWEST=MINO(COSTS(I,1),COSTS(I,2))                               4950.
ISN        19           LOWER=MAXO(COSTS(I,1),COSTS(I,2))                                4980.
ISN        20           DO 5 J=3,N                                                       4970.
ISN        21           IF(COSTS(I,J).GE.LOWEST)GOTO 4                                   4980.
ISN        22           LOWER=LOWEST                                                     4990.
ISN        23           LOWEST=COSTS(I,J)                                                5000.
ISN        24           GOTO 5                                                           5010.
ISN        25         4 IF(COSTS(I,J).GE.LOWER)GOTO 5                                    5020.
ISN        26           LOWER=COSTS(I,J)                                                 5030.
ISN        27         5 CONTINUE                                                         5040.
ISN        28           IF(LOWER.EQ.HIGH)LOWER=0                                         5050.
ISN        30           IF(LOWEST.EQ.HIGH)LOWEST=0                                       5060.
ISN        32           VAMROW(I)=IABS(LOWER-LOWEST)                                     5070.
ISN        33         6 CONTINUE                                                         5080.
                    C                                                                    5090.
                    C     GENERATE THE VAM NUMBERS FOR COLUMNS                           5100.
                    C                                                                    5110.
ISN        34           DO 9 J=1,N                                                       5120.
ISN        35           IF(WAREHO(J).EQ.0)VAMCOL(J)=0                                    5130.
ISN        37           IF(WAREHO(J).EQ.0)GOTO 9                                         5140.
ISN        38           LOWEST=MINO(COSTS(1,J),COSTS(2,J))                               5150.
ISN        39           LOWER=MAXO(COSTS(1,J),COSTS(2,J))                                5160.
ISN        40           DO 8 I=3,M                                                       5170.
ISN        41           IF(COSTS(I,J).GE.LOWEST)GOTO 7                                   5180.


ISN        42           LOWER=LOWEST                                                     5190.
ISN        43           LOWEST=COSTS(I,J)                                                5200.
ISN        44           GOTO 8                                                           5210.
ISN        45         7 IF(COSTS(I,J).GE.LOWER)GOTO 8                                    5220.
ISN        46           LOWER=COSTS(I,J)                                                 5230.
ISN        47         8 CONTINUE                                                         5240.
ISN        48           IF(LOWER.EQ.HIGH)LOWER=0                                         5250.
ISN        50           IF(LOWEST.EQ.HIGH)LOWEST=0                                       5260.
ISN        52           VAMCOL(J)=IABS(LOWER-LOWEST)                                     5270.
ISN        53         9 CONTINUE                                                         5280.
                    C                                                                    5290.
                    C     FIND THE HIGHEST VAM #   FROM VAMROW & VAMCOL INTO HIGH1 & HIGH2 5300.
                    C                                                                    5310.
ISN        54           HIGH1=VAMROW(1)                                                  5320.
ISN        55           DO 10 I=2,M                                                      5330.
ISN        56           IF(VAMROW(I).GT.HIGH1)HIGH1=VAMROW(I)                            5340.
ISN        58        10 CONTINUE                                                         5350.
ISN        59           HIGH2=VAMCOL(1)                                                  5360.
ISN        60           DO 11 J=2,N                                                      5370.
ISN        61           IF(VAMCOL(J).GT.HIGH2)HIGH2=VAMCOL(J)                            5380.
ISN        63        11 CONTINUE                                                         5390.
                    C                                                                    5400.
                    C     FIND THE HIGHEST FROM HIGH1 & HIGH2                            5410.
                    C                                                                    5420.
ISN        64           TOPVAM=MAXO(HIGH1,HIGH2)                                         5430.
ISN        65           ROW=0                                                            5440.
ISN        66           COL=0                                                            5450.
                    C                                                                    5460.
                    C     THE HIGHEST VAM #  IS UNIQUE ?                                 5470.
                    C                                                                    5480.
ISN        67           VAMCT=0                                                          5490.
ISN        68           DO 12 I=1,M                                                      5500.
ISN        69           IF(VAMROW(I).LT.TOPVAM)GOTO 12                                   5510.
ISN        70           VAMCT=VAMCT+1                                                    5520.
ISN        71           ROW=I                                                            5530.
ISN        72        12 CONTINUE                                                         5540.
ISN        73           IF(VAMCT.GT.1)GOTO 20                                            5550.
ISN        74           DO 13 J=1,N                                                      5560.
ISN        75           IF(VAMCOL(J).LT.TOPVAM)GOTO 13                                   5570.
ISN        76           VAMCT=VAMCT+1                                                    5580.
ISN        77           COL=J                                                            5590.
ISN        78        13 CONTINUE                                                         5600.
ISN        79           IF(VAMCT.EQ.1)GOTO 60                                            5610.
                    C                                                                    5620.
                    C     CHECK IF THERE ARE VAM NUMBERS THAT MAKE A TIE                 5630.
                    C     USE BECTOR NUMBERS TO DETERMINE WHICH ROW OR COL               5640.
                    C                                                                    5650.
ISN        80        20 TOPBEC=0                                                        5660.
ISN        81           ROW=0                                                            5670.
ISN        82           COL=0                                                            5680.
ISN        83           IF(TOPVAM.GT.HIGH1)GOTO 50                                       5690.
                    C                                                                    5700.
                    C     HIGHEST VAM NUMBER FOUND ON ROWS                               5710.
                    C                                                                    5720.
ISN        84           DO 41 I=1,M                                                      5730.
ISN        85           IF(VAMROW(I).LT.TOPVAM)GOTO 41                                   5740.
ISN        86           BECTOR=RANGE(M,N,I,0)*FACTRY(I)                                  5750.
ISN        87           IF(BECTOR.LT.TOPBEC)GOTO 41                                      5760.
ISN        88           TOPBEC=BECTOR                                                    5770.
ISN        88           ROW=I                                                            5780.
ISN        90        41 CONTINUE                                                         5790.
                    C                                                                    5800.
                    C     HIGHEST VAM NUMBER FOUND ON COLS                               5810.
                    C                                                                    5820.
ISN        91        50 IF(TOPVAM.GT.HIGH2)GOTO 60                                      5830.
ISN        92           DO 51 J=1,N                                                      5840.
```

```
ISN        93          IF(VAMCOL(J).LT.TOPVAM)GOTO 51                                  5850.
ISN        94          BECTOR=RANGE(M,N,O,J)*WAREHO(J)                                 5860.
ISN        95          IF(BECTOR.LT.TOPBEC)GOTO 51                                     5870.
ISN        96          TOPBEC=BECTOR                                                   5880.
ISN        97          COL=J                                                           5890.
ISN        98          ROW=O                                                           5900.
ISN        99       51 CONTINUE                                                        5910.
                 C                                                                    5920.
                 C      DETERMINE THE HIGHEST BECTOR NUMBER IS ON ROW OR COL          5930.
                 C                                                                    5940.
ISN       100       60 IF(COL.NE.O)GOTO 70                                             5950.
                 C                                                                    5960.
                 C      HIGHEST BECTOR NUMBER IS ON ROW                               5970.
                 C      THEN PICKUP THE LOWEST ENTRY ON THAT ROW                      5980.
                 C                                                                    5990.
ISN       101       61 LOW=COSTS(ROW,1)                                                6000.
ISN       102          COL=1                                                          6010.
ISN       103          DO 62 J=2,N                                                    6020.
ISN       104          IF(COSTS(ROW,J).GE.LOW)GOTO 62                                 6030.
ISN       105          LOW=COSTS(ROW,J)                                               6040.
ISN       106          COL=J                                                          6050.
ISN       107       62 CONTINUE                                                       6060.
ISN       108          COSTS(ROW,COL)=HIGH                                            6070.
ISN       109          ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))                   6080.
ISN       110          FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                         6090.
ISN       111          WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                         6100.
ISN       112          IF(FACTRY(ROW).GT.O)GOTO 61                                    6110.
ISN       113          GOTO 95                                                        6120.
                 C                                                                    6130.
                 C      HIGHEST BECTOR NUMBER IS ON COL                               6140.
                 C      PICKUP THE LOWEST ENTRY ON THE COL                            6150.
                 C                                                                    6160.
ISN       114       70 LOW=COSTS(1,COL)                                               6170.
ISN       115          ROW=1                                                          6180.
ISN       116          DO 71 I=2,M                                                    6190.
ISN       117          IF(COSTS(I,COL).GE.LOW)GOTO 71                                 6200.
ISN       118          LOW=COSTS(I,COL)                                               6210.
ISN       119          ROW=I                                                          6220.
ISN       120       71 CONTINUE                                                       6230.
ISN       121          COSTS(ROW,COL)=HIGH                                            6240.
ISN       122          ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))                   6250.
ISN       123          FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                         6260.
ISN       124          WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                         6270.
ISN       125          IF(WAREHO(COL).GT.O)GOTO 70                                    6280.
ISN       126          GOTO 95                                                        6290.
ISN       127       80 COSTS(ROW,COL)=HIGH                                            6300.
ISN       128          ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))                   6310.
ISN       129          FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                         6320.
ISN       130          WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                         6330.
                 C                                                                    6340.
                 C      CHECK TO SEE IF ALL CONDITIONS FOR DEMAND & SUPPLY ARE FULFILLED  6350.
                 C                                                                    6360.
ISN       131       95 DO 96 I=1,M                                                    6370.
ISN       132       96 IF(FACTRY(I).GT.O)GOTO 3                                       6380.
ISN       133          DO 97 J=1,N                                                    6390.
ISN       134       97 IF(WAREHO(J).GT.O)GOTO 3                                       6400.
                 C                                                                    6410.
                 C      CALCULATE THE TOTAL COST FOR THE B.F.S IN VAM ROUTE           6420.
                 C                                                                    6430.
ISN       135          TOTAL=O.                                                       6440.
ISN       136          DO 100 I=1,M                                                   6450.
ISN       137          DO 100 J=1,N                                                   6460.
ISN       138      100 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))                        6470.
ISN       139          WRITE(6,101)TOTAL                                              6480.
ISN       140      101 FORMAT('O','TOTAL COST IN  MODIFIED V.A.M. METHOD = ',E16.10)  6490.
                 C                                                                    6500.



                 C      OUTPUT THE ALLOCATION MATRIX                                  6510.
                 C                                                                    6520.
ISN       141          DO 98 I=1,M                                                    6530.
ISN       142       98 WRITE(6,99)(ALLOC(I,J),J=1,N)                                  6540.
ISN       143       99 FORMAT(1X,3314)                                                6550.
ISN       144          RETURN                                                         6560.
ISN       145          END                                                            6570.

*STATISTICS*   SOURCE STATEMENTS = 137, PROGRAM SIZE = 5820 BYTES, PROGRAM NAME = VAM1     PAGE:   22.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 22 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.....2.....3.....4.....5.....6.....7.*.....8
ISN         1           SUBROUTINE VAM2(M,N,ALLOC,COSTS,FACTRY,WAREHO,VAMROW,VAMCOL)      6580.
ISN         2           IMPLICIT INTEGER(A-Z)                                            6590.
ISN         3           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                     6600.
ISN         4           INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),               6610.
                       C         VAMROW(M),VAMCOL(N)                                    6620.
ISN         5           INTEGER HIGH/2147483647/                                         6630.
ISN         6           REAL TOTAL,ZERO/0./                                              6640.
                C                                                                       6650.
                C     CONVENTIONAL V.A.M. METHOD:                                       6660.
                C                                                                       6670.
                C     CALCULATE THE VAM NUMBERS BY TAKING THE DIFFERENCE BETWEEN THE    6680.
                C     CHEAPEST ROUTE AND THE SECOND CHEAPEST ROUTE.                     6690.
                C     PICK UP THE HIGHEST VAM NUMBER AMONG THE ROWS AND COLUMNS         6700.
                C     FILL UP THAT ROW OR COLUMN FROM THE CHEAPEST COST.                6710.
                C     IF A TIE IS FOUND ON THE VAM NUMBERS, TEST THOSE ROWS/COLS        6720.
                C     WITH THE CHEAPEST ROUTE, TAKE THE ROUTE WITH THE LOWEST COST.     6730.
                C                                                                       6740.
                C                                                                       6750.
                C     PRESERVE THE ORIGINAL PARAMETERS                                  6760.
                C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO      6770.
                C                                                                       6780.
ISN         7           DO 1 I=1,M                                                      6790.
ISN         8           FACTRY(I)=SUPPLY(I)                                             6791.
ISN         9           DO 1 J=1,N                                                      6800.
ISN        10           COSTS(I,J)=COST(I,J)                                            6810.
ISN        11         1 ALLOC(I,J)=0                                                    6820.
ISN        12           DO 2 J=1,N                                                      6850.
ISN        13         2 WAREHO(J)=DEMAND(J)                                           6860.
                C                                                                       6870.
                C     GENERATE THE VAM NUMBERS FOR ROWS                                 6880.
                C                                                                       6890.
ISN        14         3 DO 6 I=1,M                                                    6900.
ISN        15           IF(FACTRY(I).EQ.0)VAMROW(I)=0                                   6910.
ISN        17           IF(FACTRY(I).EQ.0)GOTO 6                                        6920.
ISN        18           LOWEST=MINO(COSTS(I,1),COSTS(I,2))                              6930.
ISN        19           LOWER=MAXO(COSTS(I,1),COSTS(I,2))                               6940.
ISN        20           DO 5 J=3,N                                                      6950.
ISN        21           IF(COSTS(I,J).GE.LOWEST)GOTO 4                                  6960.
ISN        22           LOWER=LOWEST                                                    6970.
ISN        23           LOWEST=COSTS(I,J)                                               6980.
ISN        24           GOTO 5                                                          6990.
ISN        25         4 IF(COSTS(I,J).GE.LOWER)GOTO 5                                   7000.
ISN        26           LOWER=COSTS(I,J)                                                7010.
ISN        27         5 CONTINUE                                                        7020.
ISN        28           IF(LOWER.EQ.HIGH)LOWER=0                                        7030.
ISN        30           IF(LOWEST.EQ.HIGH)LOWEST=0                                      7040.
ISN        32           VAMROW(I)=IABS(LOWER-LOWEST)                                    7050.
ISN        33         6 CONTINUE                                                        7060.
                C                                                                       7070.
                C     GENERATE THE VAM NUMBERS FOR COLUMNS                              7080.
                C                                                                       7090.
ISN        34           DO 9 J=1,N                                                      7100.
ISN        35           IF(WAREHO(J).EQ.0)VAMCOL(J)=0                                   7110.
ISN        37           IF(WAREHO(J).EQ.0)GOTO 9                                        7120.
ISN        38           LOWEST=MINO(COSTS(1,J),COSTS(2,J))                              7130.
ISN        39           LOWER=MAXO(COSTS(1,J),COSTS(2,J))                               7140.
ISN        40           DO 8 I=3,M                                                      7150.
ISN        41           IF(COSTS(I,J).GE.LOWEST)GOTO 7                                  7160.
ISN        42           LOWER=LOWEST                                                    7170.

ISN        43           LOWEST=COSTS(I,J)                                               7180.
ISN        44           GOTO 8                                                          7190.
ISN        45         7 IF(COSTS(I,J).GE.LOWER)GOTO 8                                   7200.
ISN        46           LOWER=COSTS(I,J)                                                7210.
ISN        47         8 CONTINUE                                                        7220.
ISN        48           IF(LOWER.EQ.HIGH)LOWER=0                                        7230.
ISN        50           IF(LOWEST.EQ.HIGH)LOWEST=0                                      7240.
ISN        52           VAMCOL(J)=IABS(LOWER-LOWEST)                                    7250.
ISN        53         9 CONTINUE                                                        7260.
                C                                                                       7270.
                C     FIND THE HIGHEST VAM #   FROM VAMROW & VAMCOL INTO HIGH1 & HIGH2  7280.
                C                                                                       7290.
ISN        54           HIGH1=VAMROW(1)                                                 7300.
ISN        55           DO 10 I=2,M                                                     7310.
ISN        58           IF(VAMROW(I).GT.HIGH1)HIGH1=VAMROW(I)                           7320.
ISN        58        10 CONTINUE                                                       7330.
ISN        59           HIGH2=VAMCOL(1)                                                 7340.
ISN        60           DO 11 J=2,N                                                     7350.
ISN        61           IF(VAMCOL(J).GT.HIGH2)HIGH2=VAMCOL(J)                           7360.
ISN        63        11 CONTINUE                                                       7370.
                C                                                                       7380.
                C     FIND THE HIGHEST FROM HIGH1 & HIGH2                               7390.
                C                                                                       7400.
ISN        64           TOPVAM=MAXO(HIGH1,HIGH2)                                        7410.
ISN        65           ROW=0                                                           7420.
ISN        66           COL=0                                                           7430.
                C                                                                       7440.
                C     THE HIGHEST VAM #  IS UNIQUE ?                                    7450.
                C                                                                       7460.
ISN        67           VAMCT=0                                                         7470.
ISN        68           DO 12 I=1,M                                                     7480.
ISN        69           IF(VAMROW(I).LT.TOPVAM)GOTO 12                                  7490.
ISN        70           VAMCT=VAMCT+1                                                   7500.
ISN        71           ROW=I                                                           7510.
ISN        72        12 CONTINUE                                                       7520.
ISN        73           IF(VAMCT.GT.1)GOTO 20                                           7530.
ISN        74           DO 13 J=1,N                                                     7540.
ISN        75           IF(VAMCOL(J).LT.TOPVAM)GOTO 13                                  7550.
ISN        76           VAMCT=VAMCT+1                                                   7560.
ISN        77           COL=J                                                           7570.
ISN        78        13 CONTINUE                                                       7580.
ISN        79           IF(VAMCT.EQ.1)GOTO 60                                           7590.
                C                                                                       7600.
                C     CHECK IF THERE ARE VAM NUMBERS THAT MAKE A TIE                    7610.
                C     USE CHEAPEST ROUTE TO DETERMINE WHICH ROW OR COL                  7620.
                C                                                                       7630.
ISN        80        20 TOPCHP=HIGH                                                    7640.
ISN        81           ROW=0                                                           7650.
ISN        82           COL=0                                                           7660.
ISN        83           IF(TOPVAM.GT.HIGH1)GOTO 50                                      7670.
                C                                                                       7680.
                C     HIGHEST VAM NUMBER FOUND ON ROWS                                  7690.
                C                                                                       7700.
ISN        84           DO 41 I=1,M                                                     7710.
ISN        85           IF(VAMROW(I).LT.TOPVAM)GOTO 41                                  7720.
ISN        86           CHEAP=CHEAPT(M,N,COSTS,ALLOC,FACTRY,WAREHO,I,0)                 7730.
ISN        87           IF(CHEAP.GE.TOPCHP)GOTO 41                                      7740.
ISN        88           TOPCHP=CHEAP                                                    7750.
ISN        89           ROW=I                                                           7760.
ISN        90        41 CONTINUE                                                       7770.
                C                                                                       7780.
                C     HIGHEST VAM NUMBER FOUND ON COLS                                  7790.
                C                                                                       7800.
ISN        91        50 IF(TOPVAM.GT.HIGH2)GOTO 60                                     7810.
ISN        92           DO 51 J=1,N                                                     7820.
ISN        93           IF(VAMCOL(J).LT.TOPVAM)GOTO 51                                  7830.
```

```
ISN        94          CHEAP=CHEAPT(M,N,COSTS,ALLOC,FACTRY,WAREHO,O,J)                    7840.
ISN        95          IF(CHEAP.GE.TOPCHP)GOTO 51                                         7850.
ISN        96          TOPCHP=CHEAP                                                       7860.
ISN        97          COL=J                                                              7870.
ISN        98          ROW=O                                                              7880.
ISN        99       51 CONTINUE                                                           7890.
                 C                                                                       7900.
                 C     HIGHEST VAM NUMBER WITH CHEAPEST ROUTE IS ON ROW OR COL ?         7910.
                 C                                                                       7920.
ISN       100       60 IF(COL.NE.O)GOTO 70                                                7930.
                 C                                                                       7940.
                 C     HIGHEST VAM NUMBER ON CHEAPEST ROUTE IS ON ROW                    7950.
                 C     THEN PICKUP THE LOWEST ENTRY ON THAT ROW                          7960.
                 C                                                                       7970.
ISN       101       61 LOW=COSTS(ROW,1)                                                   7980.
ISN       102          COL=1                                                              7990.
ISN       103          DO 62 J=2,N                                                        8000.
ISN       104          IF(COSTS(ROW,J).GE.LOW)GOTO 62                                     8010.
ISN       105          LOW=COSTS(ROW,J)                                                   8020.
ISN       106          COL=J                                                              8030.
ISN       107       62 CONTINUE                                                           8040.
ISN       108          GOTO 80                                                            8050.
                 C                                                                       8060.
                 C     HIGHEST VAM NUMBER ON CHEAPEST ROUTE IS ON COL                    8070.
                 C     PICKUP THE LOWEST ENTRY ON THE COL                                8080.
                 C                                                                       8090.
ISN       109       70 LOW=COSTS(1,COL)                                                   8100.
ISN       110          ROW=1                                                              8110.
ISN       111          DO 71 I=2,M                                                        8120.
ISN       112          IF(COSTS(I,COL).GE.LOW)GOTO 71                                     8130.
ISN       113          LOW=COSTS(I,COL)                                                   8140.
ISN       114          ROW=I                                                              8150.
ISN       115       71 CONTINUE                                                           8160.
ISN       116       80 COSTS(ROW,COL)=HIGH                                                8170.
ISN       117          ALLOC(ROW,COL)=MINO(FACTRY(ROW),WAREHO(COL))                       8180.
ISN       118          FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                             8190.
ISN       119          WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                             8200.
                 C                                                                       8210.
                 C     CHECK TO SEE IF ALL CONDITIONS FOR DEMAND & SUPPLY ARE FULFILLED  8220.
                 C                                                                       8230.
ISN       120          DO 96 I=1,M                                                        8240.
ISN       121       96 IF(FACTRY(I).GT.O)GOTO 3                                           8250.
ISN       122          DO 97 J=1,N                                                        8260.
ISN       123       97 IF(WAREHO(J).GT.O)GOTO 3                                           8270.
                 C                                                                       8280.
                 C     CALCULATE THE TOTAL COST FOR THE B.F.S IN VAM ROUTE                8290.
                 C                                                                       8300.
ISN       124          TOTAL=ZERO                                                         8310.
ISN       125          DO 100 I=1,M                                                       8320.
ISN       126          DO 100 J=1,N                                                       8330.
ISN       127      100 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))                            8340.
ISN       128          WRITE(6,101)TOTAL                                                  8350.
ISN       129      101 FORMAT('O','TOTAL COST IN  CONVENTIONAL V.A.M. METHOD = ',E16.10) 8360.
                 C                                                                       8370.
                 C     OUTPUT THE ALLOCATION MATRIX                                      8380.
                 C                                                                       8390.
ISN       130          DO 98 I=1,M                                                        8400.
ISN       131       98 WRITE(6,99)(ALLOC(I,J),J=1,N)                                      8410.
ISN       132       99 FORMAT(1X,3314)                                                    8420.
ISN       133          RETURN                                                             8430.
ISN       134          END                                                                8440.

*STATISTICS*    SOURCE STATEMENTS = 126, PROGRAM SIZE = 5170 BYTES, PROGRAM NAME = VAM2     PAGE:   23.

*STATISTICS*     NO DIAGNOSTICS GENERATED.


****** END OF COMPILATION 23 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN         1           INTEGER FUNCTION CHEAPT(M,N,COSTS,ALLOC,FACTRY,WAREHO,IROW,ICOL)  8450.
ISN         2           IMPLICIT INTEGER(A-Z)                                             8460.
ISN         3           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                      8470.
ISN         4           INTEGER COSTS(M,N),ALLOC(M,N),FACTRY(M),WAREHO(N),                8480.
                       C         C(100,100),A(100,100),F(100),W(100)                     8490.
ISN         5           INTEGER HIGH/2147483647/                                          8500.
                C                                                                        8510.
                C     PRESERVE THE ORIGINAL PARAMETERS                                   8520.
                C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO       8530.
                C                                                                        8540.
ISN         6           DO 1 I=1,M                                                        8550.
ISN         7           F(I)=FACTRY(I)                                                    8551.
ISN         8           DO 1 J=1,N                                                        8560.
ISN         9           C(I,J)=COSTS(I,J)                                                 8570.
ISN        10         1 A(I,J)=ALLOC(I,J)                                                 8580.
ISN        11           DO 2 J=1,N                                                      8610.
ISN        12         2 W(J)=WAREHO(J)                                                  8620.
                C                                                                        8630.
ISN        13           ROW=IROW                                                        8640.
ISN        14           COL=ICOL                                                        8650.
                C                                                                        8660.
                C     FILL UP THE CHEAPEST CELL ON ROW/COL                               8670.
                C                                                                        8680.
ISN        15           IF(COL.NE.0)GOTO 60                                             8690.
ISN        16           LOW=C(ROW,1)                                                    8700.
ISN        17           COL=1                                                           8710.
ISN        18           DO 50 J=2,N                                                     8720.
ISN        19           IF(C(ROW,J).GE.LOW)GOTO 50                                      8730.
ISN        20           LOW=C(ROW,J)                                                    8740.
ISN        21           COL=J                                                           8750.
ISN        22        50 CONTINUE                                                        8760.
ISN        23           GOTO 70                                                         8770.
ISN        24        60 LOW=C(1,COL)                                                    8780.
ISN        25           ROW=1                                                           8790.
ISN        26           DO 61 I=2,M                                                     8800.
ISN        27           IF(C(I,COL).GE.LOW)GOTO 61                                      8810.
ISN        28           LOW=C(I,COL)                                                    8820.
ISN        29           ROW=I                                                           8830.
ISN        30        61 CONTINUE                                                        8840.
ISN        31           GOTO 70                                                         8850.
                C                                                                        8860.
                C     FIND THE LOWEST COST ON COST MAP "C"                               8870.
                C                                                                        8880.
ISN        32         3 LOWEST=C(1,1)                                                   8890.
ISN        33           ROW=1                                                           8900.
ISN        34           COL=1                                                           8910.
ISN        35           DO 4 I=1,M                                                      8920.
ISN        36           DO 4 J=1,N                                                      8930.
ISN        37           IF(C(I,J).GE.LOWEST)GOTO 4                                      8940.
ISN        38           LOWEST=C(I,J)                                                   8950.
ISN        39           ROW=I                                                           8960.
ISN        40           COL=J                                                           8970.
ISN        41         4 CONTINUE                                                        8980.
                C                                                                        8990.
                C     THE CHEAPEST ROUTE IS ASSIGNED WITH A HIGH VALUE NUMBER            9000.
                C     AFTER BEING USED                                                   9010.
                C                                                                        9020.
ISN        42        70 C(ROW,COL)=HIGH                                                 9030.
ISN        43           A(ROW,COL)=MIN0(F(ROW),W(COL))                                  9040.


ISN        44           F(ROW)=F(ROW)-A(ROW,COL)                                        9050.
ISN        45           W(COL)=W(COL)-A(ROW,COL)                                        9060.
                C                                                                        9070.
                C     FIND THE NEXT CHEAPEST ROUTE IF THE REQUIREMENT OF WAREHOUSES      9080.
                C     AND FACTORIES ARE NOT MET                                          9090.
                C                                                                        9100.
ISN        46           DO 5 I=1,M                                                      9110.
ISN        47         5 IF(F(I).GT.0)GOTO 3                                             9120.
ISN        48           DO 6 J=1,N                                                      9130.
ISN        49         6 IF(W(J).GT.0)GOTO 3                                             9140.
ISN        50           CHEAPT=0                                                        9150.
ISN        51           DO 20 I=1,M                                                     9160.
ISN        52           DO 20 J=1,N                                                     9170.
ISN        53        20 CHEAPT=CHEAPT+COST(I,J)*A(I,J)                                  9180.
ISN        54           RETURN                                                          9190.
ISN        55           END                                                             9200.
```

STATISTICS SOURCE STATEMENTS = 55, PROGRAM SIZE = 83330 BYTES, PROGRAM NAME = CHEAPT PAGE: 24.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 24 ******

```
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(654)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE BECTR1(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)     9210.
ISN         2           IMPLICIT INTEGER(A-Z)                                             9220.
ISN         3           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                      9230.
ISN         4           INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),                9240.
                       C         BECTOR(N),DSCEND(N)                                      9250.
ISN         5           INTEGER HIGH/2147483647/                                          9260.
ISN         6           REAL TOTAL                                                        9270.
                    C                                                                    9280.
                    C     B E C T O R   M E T H O D     # 1                                9290.
                    C                                                                    9300.
                    C     CALCULATE THE RANGE R, THE DIFFERENCE BETWEEN THE CHEAPEST &   9310.
                    C     THE MOST EXPENSIVE ROUTE.                                      9320.
                    C     MULTIPLY EACH R BY THE # OF UNITS REQUIRED AT THAT DESTINATION 9330.
                    C     TO GET B, THE BECTOR'S NUMBER.                                 9340.
                    C     ASSIGN FIRST IN THE COLUMN WITH THE LARGEST B, THEN SECOND     9350.
                    C     LARGEST B AND SO ON.                                           9360.
                    C                                                                    9370.
                    C                                                                    9380.
                    C                                                                    9390.
                    C     PRESERVE THE ORIGINAL PARAMETERS                               9400.
                    C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO   9410.
                    C                                                                    9420.
ISN         7           DO 1 I=1,M                                                       9430.
ISN         8           FACTRY(I)=SUPPLY(I)                                              9431.
ISN         9           DO 1 J=1,N                                                       9440.
ISN        10           COSTS(I,J)=COST(I,J)                                             9450.
ISN        11         1 ALLOC(I,J)=0                                                     9460.
ISN        12           DO 2 J=1,N                                                       9490.
ISN        13         2 WAREHO(J)=DEMAND(J)                                            9500.
                    C                                                                    9510.
                    C     GENERATE THE BECTOR NUMBERS FOR COLUMNS (WAREHOUSES)           9520.
                    C                                                                    9530.
ISN        14           DO 3 J=1,N                                                       9540.
ISN        15         3 BECTOR(J)=WAREHO(J)*RANGE(M,N,0,J)                             9550.
                    C                                                                    9560.
                    C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER                   9570.
                    C                                                                    9580.
ISN        16           CALL SORT(N,BECTOR,DSCEND)                                       9590.
                    C                                                                    9600.
                    C     FILL UP THE CELL WITH LOWEST COST                              9610.
                    C                                                                    9620.
ISN        17           DO 6 J=1,N                                                       9630.
ISN        18           COL=DSCEND(J)                                                    9640.
                    C                                                                    9650.
ISN        19         4 LOWEST=COSTS(1,COL)                                            9660.
ISN        20           ROW=1                                                            9670.
ISN        21           DO 5 I=2,M                                                       9680.
ISN        22           IF(COSTS(I,COL).GE.LOWEST)GOTO 5                                 9690.
ISN        23           ROW=I                                                            9700.
ISN        24           LOWEST=COSTS(ROW,COL)                                            9710.
ISN        25         5 CONTINUE                                                         9720.
                    C                                                                    9730.
ISN        26           COSTS(ROW,COL)=HIGH                                              9740.
ISN        27           ALLOC(ROW,COL)=MIN0(FACTRY(ROW),WAREHO(COL))                     9750.
ISN        28           FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                           9760.
ISN        29           WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                           9770.
ISN        30           IF(WAREHO(COL).GT.0)GOTO 4                                       9780.
ISN        31         6 CONTINUE                                                         9790.
                    C                                                                    9800.



                    C                                                                    9810.
                    C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD       9820.
                    C                                                                    9830.
ISN        32           TOTAL=0.                                                         9840.
ISN        33           DO 30 I=1,M                                                      9850.
ISN        34           DO 30 J=1,N                                                      9860.
ISN        35        30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))                          9870.
ISN        36           WRITE(6,31)TOTAL                                                 9880.
ISN        37        31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 1 = ',E16.10)        9890.
                    C                                                                    9900.
                    C     OUTPUT THE ALLOCATION MATRIX                                   9910.
                    C                                                                    9920.
ISN        38           DO 28 I=1,M                                                      9930.
ISN        39        28 WRITE(6,29)(ALLOC(I,J),J=1,N)                                    9940.
ISN        40        29 FORMAT(1X,3314)                                                 9950.
ISN        41           RETURN                                                           9960.
ISN        42           END                                                              9970.

*STATISTICS*   SOURCE STATEMENTS = 42, PROGRAM SIZE = 2662 BYTES, PROGRAM NAME = BECTR1    PAGE:   25.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 25 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN         1           SUBROUTINE BECTR2(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)     9980.
ISN         2           IMPLICIT INTEGER(A-Z)                                             9990.
ISN         3           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                      10000.
ISN         4           INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),                10010.
                       C         BECTOR(M),DSCEND(M)                                     10020.
ISN         5           INTEGER HIGH/2147483647/                                          10030.
ISN         6           REAL TOTAL                                                        10040.
                C                                                                        10050.
                C     B E C T O R   M E T H O D     # 2                                   10060.
                C                                                                        10070.
                C     CALCULATE THE RANGE R, THE DIFFERENCE BETWEEN THE CHEAPEST &      10080.
                C     THE MOST EXPENSIVE ROUTE.                                         10090.
                C     MULTIPLY EACH R BY THE # OF UNITS REQUIRED AT THAT SOURCE          10100.
                C     TO GET B, THE BECTOR'S NUMBER.                                    10110.
                C     ASSIGN FIRST IN THE ROW WITH THE LARGEST B, THEN SECOND           10120.
                C     LARGEST B AND SO ON.                                              10130.
                C                                                                       10140.
                C                                                                       10150.
                C                                                                       10160.
                C     PRESERVE THE ORIGINAL PARAMETERS                                  10170.
                C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO      10180.
                C                                                                       10190.
ISN         7           DO 1 I=1,M                                                      10200.
ISN         8           FACTRY(I)=SUPPLY(I)                                             10201.
ISN         9           DO 1 J=1,N                                                      10210.
ISN        10           COSTS(I,J)=COST(I,J)                                            10220.
ISN        11         1 ALLOC(I,J)=0                                                    10230.
ISN        12           DO 2 J=1,N                                                      10260.
ISN        13         2 WAREHO(J)=DEMAND(J)                                           10270.
                C                                                                       10280.
                C     GENERATE THE BECTOR NUMBERS FOR ROWS (FACTORIES)                  10290.
                C                                                                       10300.
ISN        14           DO 3 I=1,M                                                      10310.
ISN        15         3 BECTOR(I)=FACTRY(I)*RANGE(M,N,I,0)                            10320.
                C                                                                       10330.
                C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER                      10340.
                C                                                                       10350.
ISN        16           CALL SORT(M,BECTOR,DSCEND)                                      10360.
                C                                                                       10370.
                C     FILL UP THE CELL WITH LOWEST COST                                 10380.
                C                                                                       10390.
ISN        17           DO 6 I=1,M                                                      10400.
ISN        18           ROW=DSCEND(I)                                                   10410.
                C                                                                       10420.
ISN        19         4 LOWEST=COSTS(ROW,1)                                           10430.
ISN        20           COL=1                                                           10440.
ISN        21           DO 5 J=2,N                                                      10450.
ISN        22           IF(COSTS(ROW,J).GE.LOWEST)GOTO 5                                10460.
ISN        23           COL=J                                                           10470.
ISN        24           LOWEST=COSTS(ROW,COL)                                           10480.
ISN        25         5 CONTINUE                                                        10490.
                C                                                                       10500.
ISN        26           COSTS(ROW,COL)=HIGH                                             10510.
ISN        27           ALLOC(ROW,COL)=MIN0(FACTRY(ROW),WAREHO(COL))                    10520.
ISN        28           FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                          10530.
ISN        29           WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                          10540.
ISN        30           IF(FACTRY(ROW).GT.0)GOTO 4                                      10550.
ISN        31         6 CONTINUE                                                        10560.
                C                                                                       10570.
                C                                                                       10580.
                C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD          10590.
                C                                                                       10600.
ISN        32           TOTAL=0.                                                        10610.
ISN        33           DO 30 I=1,M                                                     10620.
ISN        34           DO 30 J=1,N                                                     10630.
ISN        35        30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))                        10640.
ISN        36           WRITE(6,31)TOTAL                                                10650.
ISN        37        31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 2 = ',E16.10)      10660.
                C                                                                       10670.
                C     OUTPUT THE ALLOCATION MATRIX                                      10680.
                C                                                                       10690.
ISN        38           DO 28 I=1,M                                                     10700.
ISN        39        28 WRITE(6,29)(ALLOC(I,J),J=1,N)                                  10710.
ISN        40        29 FORMAT(1X,3314)                                                10720.
ISN        41           RETURN                                                          10730.
ISN        42           END                                                             10740.

*STATISTICS*   SOURCE STATEMENTS = 42, PROGRAM SIZE = 2688 BYTES, PROGRAM NAME = BECTR2    PAGE:   26.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 26 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE BECTR3(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)     10750.
ISN         2           IMPLICIT INTEGER(A-Z)                                             10760.
ISN         3           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                      10770.
ISN         4           INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),                10780.
                       C          BECTOR(N),DSCEND(N)                                     10790.
ISN         5           INTEGER HIGH/2147483647/                                          10800.
ISN         6           REAL TOTAL,MEAN                                                   10810.
                C                                                                        10820.
                C     B E C T O R   M E T H O D     # 3                                   10830.
                C                                                                        10840.
                C     CALCULATE THE MEAN M, THE AVERAGE AMONG THE ROUTES ON THAT COL.    10850.
                C     MULTIPLY EACH M BY THE # OF UNITS REQUIRED AT THAT DESTINATION     10860.
                C     TO GET B, THE BECTOR'S NUMBER.                                     10870.
                C     ASSIGN FIRST IN THE COLUMN WITH THE LARGEST B, THEN SECOND         10880.
                C     LARGEST B AND SO ON.                                               10890.
                C                                                                        10900.
                C                                                                        10910.
                C                                                                        10920.
                C     PRESERVE THE ORIGINAL PARAMETERS                                   10930.
                C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO       10940.
                C                                                                        10950.
ISN         7           DO 1 I=1,M                                                        10960.
ISN         8           FACTRY(I)=SUPPLY(I)                                               10961.
ISN         9           DO 1 J=1,N                                                        10970.
ISN        10           COSTS(I,J)=COST(I,J)                                              10980.
ISN        11         1 ALLOC(I,J)=0                                                      10990.
ISN        12           DO 2 J=1,N                                                        11020.
ISN        13         2 WAREHO(J)=DEMAND(J)                                               11030.
                C                                                                        11040.
                C     GENERATE THE BECTOR NUMBERS FOR COLUMNS (WAREHOUSES)               11050.
                C                                                                        11060.
ISN        14           DO 3 J=1,N                                                        11070.
ISN        15         3 BECTOR(J)=WAREHO(J)*MEAN(M,N,0,J)                                 11080.
                C                                                                        11090.
                C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER                       11100.
                C                                                                        11110.
ISN        16           CALL SORT(N,BECTOR,DSCEND)                                        11120.
                C                                                                        11130.
                C     FILL UP THE CELL WITH LOWEST COST                                  11140.
                C                                                                        11150.
ISN        17           DO 6 J=1,N                                                        11160.
ISN        18           COL=DSCEND(J)                                                     11170.
                C                                                                        11180.
ISN        19         4 LOWEST=COSTS(1,COL)                                               11190.
ISN        20           ROW=1                                                             11200.
ISN        21           DO 5 I=2,M                                                        11210.
ISN        22           IF(COSTS(I,COL).GE.LOWEST)GOTO 5                                  11220.
ISN        23           ROW=I                                                             11230.
ISN        24           LOWEST=COSTS(ROW,COL)                                             11240.
ISN        25         5 CONTINUE                                                          11250.
                C                                                                        11260.
ISN        26           COSTS(ROW,COL)=HIGH                                               11270.
ISN        27           ALLOC(ROW,COL)=MIN0(FACTRY(ROW),WAREHO(COL))                      11280.
ISN        28           FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                            11290.
ISN        29           WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                            11300.
ISN        30           IF(WAREHO(COL).GT.0)GOTO 4                                        11310.
ISN        31         6 CONTINUE                                                          11320.
                C                                                                        11330.
                C                                                                        11340.



                C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD           11350.
                C                                                                        11360.
ISN        32           TOTAL=0.                                                          11370.
ISN        33           DO 30 I=1,M                                                       11380.
ISN        34           DO 30 J=1,N                                                       11390.
ISN        35        30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))                          11400.
ISN        36           WRITE(6,31)TOTAL                                                  11410.
ISN        37        31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 3 = ',E16.10)        11420.
                C                                                                        11430.
                C     OUTPUT THE ALLOCATION MATRIX                                       11440.
                C                                                                        11450.
ISN        38           DO 28 I=1,M                                                       11460.
ISN        39        28 WRITE(6,29)(ALLOC(I,J),J=1,N)                                    11470.
ISN        40        29 FORMAT(1X,33I4)                                                  11480.
ISN        41           RETURN                                                            11490.
ISN        42           END                                                               11500.
```

STATISTICS SOURCE STATEMENTS = 42, PROGRAM SIZE = 2718 BYTES, PROGRAM NAME = BECTR3 PAGE: 27.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 27 ******

```
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF  GOSTMT NODECK  SOURCE NOTERM  OBJECT FIXED  NOTEST
                    OPTIMIZE(0)  LANGLVL(77) NOFIPS  FLAG(I)  NAME(MAIN  )  LINECOUNT(54)

                *....*...1.....2.....3.....4.....5.....6.....7.*.....8

ISN         1           SUBROUTINE BECTR4(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)     11510.
ISN         2           IMPLICIT INTEGER(A-Z)                                             11520.
ISN         3           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                      11530.
ISN         4           INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),                11540.
                       C          BECTOR(M),DSCEND(M)                                     11550.
ISN         5           INTEGER HIGH/2147483647/                                          11560.
ISN         6           REAL TOTAL,MEAN                                                   11570.
                    C                                                                     11580.
                    C     B E C T O R   M E T H O D     # 4                                 11590.
                    C                                                                     11600.
                    C     CALCULATE THE MEAN M, THE AVERAGE AMONG THE ROUTES ON THAT ROW.  11610.
                    C     MULTIPLY EACH M BY THE # OF UNITS REQUIRED AT THAT SOURCE        11620.
                    C     TO GET B, THE BECTOR'S NUMBER.                                  11630.
                    C     ASSIGN FIRST IN THE ROW WITH THE LARGEST B, THEN SECOND         11640.
                    C     LARGEST B AND SO ON.                                            11650.
                    C                                                                     11660.
                    C                                                                     11670.
                    C                                                                     11680.
                    C     PRESERVE THE ORIGINAL PARAMETERS                                11690.
                    C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO    11700.
                    C                                                                     11710.
ISN         7           DO 1 I=1,M                                                        11720.
ISN         8           FACTRY(I)=SUPPLY(I)                                               11721.
ISN         9           DO 1 J=1,N                                                        11730.
ISN        10           COSTS(I,J)=COST(I,J)                                              11740.
ISN        11         1 ALLOC(I,J)=0                                                      11750.
ISN        12           DO 2 J=1,N                                                        11760.
ISN        13         2 WAREHO(J)=DEMAND(J)                                               11790.
                    C                                                                     11800.
                    C     GENERATE THE BECTOR NUMBERS FOR ROWS (FACTORIES)                11810.
                    C                                                                     11820.
ISN        14           DO 3 I=1,M                                                        11830.
ISN        15         3 BECTOR(I)=FACTRY(I)*MEAN(M,N,I,0)                                 11840.
                    C                                                                     11850.
                    C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER                    11860.
                    C                                                                     11870.
ISN        16           CALL SORT(M,BECTOR,DSCEND)                                        11880.
                    C                                                                     11890.
                    C     FILL UP THE CELL WITH LOWEST COST                               11900.
                    C                                                                     11910.
ISN        17           DO 6 I=1,M                                                        11920.
ISN        18           ROW=DSCEND(I)                                                     11930.
                    C                                                                     11940.
ISN        19         4 LOWEST=COSTS(ROW,1)                                               11950.
ISN        20           COL=1                                                             11960.
ISN        21           DO 5 J=2,N                                                        11970.
ISN        22           IF(COSTS(ROW,J).GE.LOWEST)GOTO 5                                  11980.
ISN        23           COL=J                                                             11990.
ISN        24           LOWEST=COSTS(ROW,COL)                                             12000.
ISN        25         5 CONTINUE                                                          12010.
                    C                                                                     12020.
ISN        26           COSTS(ROW,COL)=HIGH                                               12030.
ISN        27           ALLOC(ROW,COL)=MIN0(FACTRY(ROW),WAREHO(COL))                      12040.
ISN        28           FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                            12050.
ISN        29           WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                            12060.
ISN        30           IF(FACTRY(ROW).GT.0)GOTO 4                                        12070.
ISN        31         6 CONTINUE                                                          12080.
                    C                                                                     12090.
                    C                                                                     12100.



                    C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD        12110.
                    C                                                                     12120.
ISN        32           TOTAL=0.                                                          12130.
ISN        33           DO 30 I=1,M                                                       12140.
ISN        34           DO 30 J=1,N                                                       12150.
ISN        35        30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))                           12160.
ISN        36           WRITE(6,31)TOTAL                                                  12170.
ISN        37        31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 4 = ',E16.10)         12180.
                    C                                                                     12190.
                    C     OUTPUT THE ALLOCATION MATRIX                                    12200.
                    C                                                                     12210.
ISN        38           DO 28 I=1,M                                                       12220.
ISN        39        28 WRITE(6,29)(ALLOC(I,J),J=1,N)                                     12230.
ISN        40        29 FORMAT(1X,33I4)                                                  12240.
ISN        41           RETURN                                                            12250.
ISN        42           END                                                               12260.

*STATISTICS*   SOURCE STATEMENTS = 42, PROGRAM SIZE = 2744 BYTES, PROGRAM NAME = BECTR4    PAGE:   28.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 28 ******
```

```
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(654)

                *....*...1.....2.....3.....4.....5.....6.....7.*.....8

ISN         1           SUBROUTINE BECTR5(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)     12270.
ISN         2           IMPLICIT INTEGER(A-Z)                                             12280.
ISN         3           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                      12290.
ISN         4           INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),                12300.
                       C         BECTOR(N),DSCEND(N)                                     12310.
ISN         5           INTEGER HIGH/2147483647/                                          12320.
ISN         6           REAL TOTAL,COEFF                                                  12330.
                C                                                                        12340.
                C     B E C T O R   M E T H O D     # 5                                   12350.
                C                                                                        12360.
                C     CALCULATE THE COEFFICIENT OF VARIATION C, THE QUOTIENT OF THE      12370.
                C     STD. DEV AND MEAN OF THE COLUMN.                                   12380.
                C     MULTIPLY EACH C BY THE # OF UNITS REQUIRED AT THAT DESTINATION     12390.
                C     TO GET B, THE BECTOR'S NUMBER.                                     12400.
                C     ASSIGN FIRST IN THE COLUMN WITH THE LARGEST B, THEN SECOND         12410.
                C     LARGEST B AND SO ON.                                               12420.
                C                                                                        12430.
                C                                                                        12440.
                C                                                                        12450.
                C     PRESERVE THE ORIGINAL PARAMETERS                                   12460.
                C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO       12470.
                C                                                                        12480.
ISN         7           DO 1 I=1,M                                                        12490.
ISN         8           FACTRY(I)=SUPPLY(I)                                               12491.
ISN         9           DO 1 J=1,N                                                        12500.
ISN        10           COSTS(I,J)=COST(I,J)                                              12510.
ISN        11         1 ALLOC(I,J)=0                                                      12520.
ISN        12           DO 2 J=1,N                                                        12550.
ISN        13         2 WAREHO(J)=DEMAND(J)                                             12560.
                C                                                                        12570.
                C     GENERATE THE BECTOR NUMBERS FOR COLUMNS (WAREHOUSES)               12580.
                C                                                                        12590.
ISN        14           DO 3 J=1,N                                                        12600.
ISN        15         3 BECTOR(J)=WAREHO(J)*COEFF(M,N,0,J)                              12610.
                C                                                                        12620.
                C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER                       12630.
                C                                                                        12640.
ISN        16           CALL SORT(N,BECTOR,DSCEND)                                        12650.
                C                                                                        12660.
                C     FILL UP THE CELL WITH LOWEST COST                                  12670.
                C                                                                        12680.
ISN        17           DO 6 J=1,N                                                        12690.
ISN        18           COL=DSCEND(J)                                                     12700.
                C                                                                        12710.
ISN        19         4 LOWEST=COSTS(1,COL)                                             12720.
ISN        20           ROW=1                                                           12730.
ISN        21           DO 5 I=2,M                                                      12740.
ISN        22           IF(COSTS(I,COL).GE.LOWEST)GOTO 5                                12750.
ISN        23           ROW=I                                                           12760.
ISN        24           LOWEST=COSTS(ROW,COL)                                           12770.
ISN        25         5 CONTINUE                                                        12780.
                C                                                                        12790.
ISN        26           COSTS(ROW,COL)=HIGH                                             12800.
ISN        27           ALLOC(ROW,COL)=MIN0(FACTRY(ROW),WAREHO(COL))                    12810.
ISN        28           FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                          12820.
ISN        29           WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                          12830.
ISN        30           IF(WAREHO(COL).GT.0)GOTO 4                                      12840.
ISN        31         6 CONTINUE                                                        12850.
                C                                                                        12860.



                C                                                                        12870.
                C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD           12880.
                C                                                                        12890.
ISN        32           TOTAL=0.                                                        12900.
ISN        33           DO 30 I=1,M                                                     12910.
ISN        34           DO 30 J=1,N                                                     12920.
ISN        35        30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))                         12930.
ISN        36           WRITE(6,31)TOTAL                                                12940.
ISN        37        31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 5 = ',E16.10)       12950.
                C                                                                        12960.
                C     OUTPUT THE ALLOCATION MATRIX                                       12970.
                C                                                                        12980.
ISN        38           DO 28 I=1,M                                                     12990.
ISN        39        28 WRITE(6,29)(ALLOC(I,J),J=1,N)                                   13000.
ISN        40        29 FORMAT(1X,3314)                                                 13010.
ISN        41           RETURN                                                          13020.
ISN        42           END                                                             13030.

*STATISTICS*   SOURCE STATEMENTS = 42, PROGRAM SIZE = 2718 BYTES, PROGRAM NAME = BECTR5     PAGE:   29.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 29 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN         1           SUBROUTINE BECTR6(M,N,ALLOC,COSTS,FACTRY,WAREHO,BECTOR,DSCEND)     13040.
ISN         2           IMPLICIT INTEGER(A-Z)                                              13050.
ISN         3           COMMON COST(100,100),SUPPLY(100),DEMAND(100)                       13060.
ISN         4           INTEGER ALLOC(M,N),COSTS(M,N),FACTRY(M),WAREHO(N),                 13070.
                       C          BECTOR(M),DSCEND(M)                                     13080.
ISN         5           INTEGER HIGH/2147483647/                                           13090.
ISN         6           REAL TOTAL,COEFF                                                   13100.
                C                                                                         13110.
                C     B E C T O R   M E T H O D     # 6                                     13120.
                C                                                                         13130.
                C     CALCULATE THE COEFFICIENT OF VARIATION C, THE QUOTIENT OF THE       13140.
                C     THE STD. DEV AND MEAN OF THE ROW.                                   13150.
                C     MULTIPLY EACH C BY THE # OF UNITS REQUIRED AT THAT SOURCE           13160.
                C     TO GET B, THE BECTOR'S NUMBER.                                      13170.
                C     ASSIGN FIRST IN THE ROW WITH THE LARGEST B, THEN SECOND             13180.
                C     LARGEST B AND SO ON.                                                13190.
                C                                                                         13200.
                C                                                                         13210.
                C                                                                         13220.
                C     PRESERVE THE ORIGINAL PARAMETERS                                    13230.
                C     COST, SUPPLY & DEMAND ARE COPIED INTO COSTS, FACTRY & WAREHO        13240.
                C                                                                         13250.
ISN         7           DO 1 I=1,M                                                        13260.
ISN         8           FACTRY(I)=SUPPLY(I)                                               13261.
ISN         9           DO 1 J=1,N                                                        13270.
ISN        10           COSTS(I,J)=COST(I,J)                                              13280.
ISN        11         1 ALLOC(I,J)=0                                                      13290.
ISN        12           DO 2 J=1,N                                                        13320.
ISN        13         2 WAREHO(J)=DEMAND(J)                                               13330.
                C                                                                         13340.
                C     GENERATE THE BECTOR NUMBERS FOR ROWS (FACTORIES)                    13350.
                C                                                                         13360.
ISN        14           DO 3 I=1,M                                                        13370.
ISN        15         3 BECTOR(I)=FACTRY(I)*COEFF(M,N,I,0)                                13380.
                C                                                                         13390.
                C     SORT THE BECTOR'S NUMBER IN DESCENDING ORDER                        13400.
                C                                                                         13410.
ISN        16           CALL SORT(M,BECTOR,DSCEND)                                        13420.
                C                                                                         13430.
                C     FILL UP THE CELL WITH LOWEST COST                                   13440.
                C                                                                         13450.
ISN        17           DO 6 I=1,M                                                        13460.
ISN        18           ROW=DSCEND(I)                                                     13470.
                C                                                                         13480.
ISN        19         4 LOWEST=COSTS(ROW,1)                                               13490.
ISN        20           COL=1                                                             13500.
ISN        21           DO 5 J=2,N                                                        13510.
ISN        22           IF(COSTS(ROW,J).GE.LOWEST)GOTO 5                                  13520.
ISN        23           COL=J                                                             13530.
ISN        24           LOWEST=COSTS(ROW,COL)                                             13540.
ISN        25         5 CONTINUE                                                          13550.
                C                                                                         13560.
ISN        26           COSTS(ROW,COL)=HIGH                                               13570.
ISN        27           ALLOC(ROW,COL)=MIN0(FACTRY(ROW),WAREHO(COL))                      13580.
ISN        28           FACTRY(ROW)=FACTRY(ROW)-ALLOC(ROW,COL)                            13590.
ISN        29           WAREHO(COL)=WAREHO(COL)-ALLOC(ROW,COL)                            13600.
ISN        30           IF(FACTRY(ROW).GT.0)GOTO 4                                        13610.
ISN        31         6 CONTINUE                                                          13620.
                C                                                                         13630.



                C                                                                         13640.
                C     CALCULATE THE TOTAL COST FOR THE B.F.S IN BECTOR'S METHOD            13650.
                C                                                                         13660.
ISN        32           TOTAL=0.                                                          13670.
ISN        33           DO 30 I=1,M                                                       13680.
ISN        34           DO 30 J=1,N                                                       13690.
ISN        35        30 TOTAL=TOTAL+FLOAT(COST(I,J)*ALLOC(I,J))                           13700.
ISN        36           WRITE(6,31)TOTAL                                                  13710.
ISN        37        31 FORMAT('0','TOTAL COST IN  BECTOR''S METHOD 6 = ',E16.10)         13720.
                C                                                                         13730.
                C     OUTPUT THE ALLOCATION MATRIX                                        13740.
                C                                                                         13750.
ISN        38           DO 28 I=1,M                                                       13760.
ISN        39        28 WRITE(6,29)(ALLOC(I,J),J=1,N)                                     13770.
ISN        40        29 FORMAT(1X,33I4)                                                   13780.
ISN        41           RETURN                                                            13790.
ISN        42           END                                                               13800.
```

STATISTICS SOURCE STATEMENTS = 42, PROGRAM SIZE = 2744 BYTES, PROGRAM NAME = BECTR6 PAGE: 30.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 30 ******

```
LEVEL 1.2.0 (SEPT  82)          VS FORTRAN          DATE: 1984 APR 13      TIME: 11:49:37      NAME: MAIN     PAGE:   31

OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN   )  LINECOUNT(654)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           SUBROUTINE SORT(NUM,BECTOR,DSCEND)                                    1.
                C                                                                             2.
                C       USING STRAIGHT INSERTION ALGORITHM TO SORT BECTOR NUMBER               3.
                C       INTO DESCENDING ORDER                                                 4.
                C                                                                             5.
                C       DSCEND IS THE POSITION OF THE UNSORTED BECTOR NUMBERS                 6.
                C                                                                             7.
ISN         2           INTEGER BECTOR(NUM),DSCEND(NUM)                                       8.
ISN         3           DO 1 I=1,NUM                                                          9.
ISN         4         1 DSCEND(I)=I                                                           10.
ISN         5           DO 4 J=2,NUM                                                          11.
ISN         6           M=J-1                                                                 12.
ISN         7           KEY=BECTOR(J)                                                         13.
ISN         8           INDEX=DSCEND(J)                                                       14.
                C                                                                             15.
                C                                                                             16.
ISN         9           DO 3 I=1,M                                                            17.
ISN        10           IF(KEY.LE.BECTOR(I))GOTO 3                                            18.
                C                                                                             19.
ISN        11                DO 2 K=I,M                                                      20.
ISN        12                IPOS=M-K+I                                                      21.
ISN        13                DSCEND(IPOS+1) = DSCEND(IPOS)                                   22.
ISN        14         2      BECTOR(IPOS+1) = BECTOR(IPOS)                                   23.
                C                                                                             24.
ISN        15                BECTOR(I) = KEY                                                 25.
ISN        16                DSCEND(I) = INDEX                                               26.
ISN        17                GOTO 4                                                          27.
ISN        18         3    CONTINUE                                                         28.
ISN        19         4    CONTINUE                                                         29.
ISN        20           RETURN                                                              30.
ISN        21           END                                                                 31.

*STATISTICS*   SOURCE STATEMENTS = 21, PROGRAM SIZE = 940 BYTES, PROGRAM NAME = SORT     PAGE:   31.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 31 ******
```

```
LEVEL 1.2.0 (SEPT  82)          VS FORTRAN          DATE: 1984 APR 13      TIME: 11:49:37      NAME: MAIN     PAGE:   32

OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF   GOSTMT NODECK   SOURCE NOTERM   OBJECT FIXED  NOTEST
                      OPTIMIZE(0)  LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN   )  LINECOUNT(654)

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           INTEGER FUNCTION RANGE(M,N,ROW,COL)                                   32.
                C                                                                             33.
                C       FUNCTION   RANGE   - FIND THE RANGE(MAX-MIN) OF A CERTAIN             34.
                C       ROW OR COL FROM THE COST MATRIX                                       35.
                C                                                                             36.
                C       IF  ROW ~= 0  THEN COL = 0                                            37.
                C       IF  COL ~= 0  THEN ROW = 0                                            38.
                C                                                                             39.
ISN         2           IMPLICIT INTEGER(A-Z)                                                 40.
ISN         3           COMMON COST(200,200),SUPPLY(200),DEMAND(200)                          41.
ISN         4           MAX=0                                                                 42.
ISN         5           MIN=2147483647                                                        43.
ISN         6           IF(COL.NE.0)GOTO 2                                                    44.
ISN         7           DO 1 J=1,N                                                            45.
ISN         8           IF(COST(ROW,J).GT.MAX)MAX=COST(ROW,J)                                 46.
ISN        10           IF(COST(ROW,J).LT.MIN)MIN=COST(ROW,J)                                 47.
ISN        12         1 CONTINUE                                                              48.
ISN        13           GOTO 4                                                                49.
ISN        14         2 DO 3 I=1,M                                                            50.
ISN        15           IF(COST(I,COL).GT.MAX)MAX=COST(I,COL)                                 51.
ISN        17           IF(COST(I,COL).LT.MIN)MIN=COST(I,COL)                                 52.
ISN        19         3 CONTINUE                                                              53.
ISN        20         4 RANGE=MAX-MIN                                                         54.
ISN        21           RETURN                                                                55.
ISN        22           END                                                                   56.

*STATISTICS*   SOURCE STATEMENTS = 18, PROGRAM SIZE = 844 BYTES, PROGRAM NAME = RANGE     PAGE:   32.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 32 ******
```

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           REAL FUNCTION MEAN(M,N,ROW,COL)                                        57.
                C                                                                              58.
                C       FUNCTION   MEAN    - FIND THE MEAN(SUM OF X / N) OF A CERTAIN          59.
                C       ROW OR COL FROM THE COST MATRIX                                        60.
                C                                                                              61.
                C       IF  ROW ~= 0  THEN COL = 0                                             62.
                C       IF  COL ~= 0  THEN ROW = 0                                             63.
                C                                                                              64.
ISN         2           IMPLICIT INTEGER(A-Z)                                                  65.
ISN         3           COMMON COST(200,200),SUPPLY(200),DEMAND(200)                           66.
ISN         4           REAL FLOAT                                                             67.
ISN         5           SUM=0                                                                  68.
ISN         6           IF(COL.NE.0)GOTO 2                                                     69.
ISN         7           DO 1 J=1,N                                                             70.
ISN         8           SUM=SUM+COST(ROW,J)                                                    71.
ISN         9         1 CONTINUE                                                               72.
ISN        10           MEAN=SUM/FLOAT(N)                                                      73.
ISN        11           GOTO 4                                                                 74.
ISN        12         2 DO 3 I=1,M                                                             75.
ISN        13           SUM=SUM+COST(I,COL)                                                    76.
ISN        14         3 CONTINUE                                                               77.
ISN        15           MEAN=SUM/FLOAT(M)                                                      78.
ISN        16         4 RETURN                                                                 79.
ISN        17           END                                                                    80.
```

STATISTICS SOURCE STATEMENTS = 17, PROGRAM SIZE = 648 BYTES, PROGRAM NAME = MEAN PAGE: 33.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 33 ******

OPTIONS IN EFFECT: NOLIST NOMAP NOXREF GOSTMT NODECK SOURCE NOTERM OBJECT FIXED NOTEST
OPTIMIZE(0) LANGLVL(77) NOFIPS FLAG(I) NAME(MAIN) LINECOUNT(654)

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN         1           REAL FUNCTION COEFF(M,N,ROW,COL)                                       81.
                C                                                                              82.
                C       FUNCTION   COEFF   - FIND THE COEFF. OF VARIATION( S/MEAN ) OF         83.
                C       A CERTAIN ROW OR COL FROM THE COST MATRIX                              84.
                C                                                                              85.
                C       IF  ROW ~= 0  THEN COL = 0                                             86.
                C       IF  COL ~= 0  THEN ROW = 0                                             87.
                C                                                                              88.
ISN         2           IMPLICIT INTEGER(A-Z)                                                  89.
ISN         3           COMMON COST(200,200),SUPPLY(200),DEMAND(200)                           90.
ISN         4           REAL MEAN,STDEV,FLOAT                                                  91.
ISN         5           SUMX=0                                                                 92.
ISN         6           SUMXSQ=0                                                               93.
ISN         7           IF(COL.NE.0)GOTO 2                                                     94.
ISN         8           DO 1 J=1,N                                                             95.
ISN         9           SUMX=SUMX+COST(ROW,J)                                                  96.
ISN        10           SUMXSQ=SUMXSQ+COST(ROW,J)**2                                           97.
ISN        11         1 CONTINUE                                                               98.
ISN        12           MEAN=SUMX/FLOAT(N)                                                     99.
ISN        13           STDEV=SQRT((SUMXSQ-SUMX**2/FLOAT(N))/FLOAT(N-1))                       100.
ISN        14           GOTO 4                                                                 101.
ISN        15         2 DO 3 I=1,M                                                             102.
ISN        16           SUMX=SUMX+COST(I,COL)                                                  103.
ISN        17           SUMXSQ=SUMXSQ+COST(I,COL)**2                                           104.
ISN        18         3 CONTINUE                                                               105.
ISN        19           MEAN=SUMX/FLOAT(M)                                                     106.
ISN        20           STDEV=SQRT((SUMXSQ-SUMX**2/FLOAT(M))/FLOAT(M-1))                       107.
ISN        21         4 COEFF=STDEV/MEAN                                                       108.
ISN        22           RETURN                                                                 109.
ISN        23           END                                                                    110.
```

STATISTICS SOURCE STATEMENTS = 23, PROGRAM SIZE = 1104 BYTES, PROGRAM NAME = COEFF PAGE: 34.

STATISTICS NO DIAGNOSTICS GENERATED.

****** END OF COMPILATION 34 ******

```
OPTIONS USED - PRINT,NOMAP,NOLET,CALL,RES,NOTERM,SIZE=581632,NAME=**GO
               EP=MAIN

   TOTAL LENGTH     8D2C8
   ENTRY ADDRESS    DC010
```

```
2 MACHINES        9 JOBS

MACHINE        PROCESSING TIME
 ORDER          FOR EACH JOB
--------       ---------------
ONE ONLY             20.    0.
ONE TWO              70.   40.
ONE TWO              30.   60.
TWO ONLY              0.   70.
ONE TWO              10.   30.
TWO ONE              30.   10.
TWO ONE              20.   70.
TWO ONLY              0.   40.
ONE ONLY             40.    0.

OPTIMAL ORDER ON MACHINE 1 IS:
  5   3   2   1   9   6   7

OPTIMAL ORDER ON MACHINE 2 IS:
  6   7   4   8   5   3   2

CPU TIME FOR CAMPBL=0.003354 SECONDS


THE DISTANCE MATRIX:
1000000      184      292      449      670      516      598      618      881      909
    184  1000000      195      310      540      357      514      434      697      964
    292      195  1000000      215      380      232      434      493      719      955
    449      310      215  1000000      288      200      566      787      790     1020
    670      540      380      288  1000000      211      436      814      632      974
    516      357      232      200      211  1000000      381      642      697      952
    598      514      434      566      436      381  1000000      295      224      541
    618      434      493      787      814      642      295  1000000      320      341
    881      697      719      790      632      697      224      320  1000000      318
    909      964      955     1020      974      852      541      341      318  1000000


THE OPTIMAL FEASIBLE SOLUTION DISTANCE IS        2500.
THE ASSIGNMENT FOR    1 IS             2.
THE ASSIGNMENT FOR    2 IS             3.
THE ASSIGNMENT FOR    3 IS             1.
THE ASSIGNMENT FOR    4 IS             6.
THE ASSIGNMENT FOR    5 IS             4.
THE ASSIGNMENT FOR    6 IS             5.
THE ASSIGNMENT FOR    7 IS             9.
THE ASSIGNMENT FOR    8 IS            10.
THE ASSIGNMENT FOR    9 IS             7.
THE ASSIGNMENT FOR   10 IS             8.

CPU TIME FOR EASTMN=0.017316 SECONDS


  5 MACHINES        4 JOBS
PROCESSING TIME FOR EACH JOB ON PROCESS

JOB    1:  4.  3.  7.  2.  8.
JOB    2:  3.  7.  2.  8.  5.
JOB    3:  1.  2.  4.  3.  7.
JOB    4:  3.  4.  3.  7.  2.

OPTIMUM SEQUENCE TO MINIMIZE TOTAL PROCESSING TIME IS:
  3   1   2   4



CPU TIME FOR GUPTA =0.002444 SECONDS


ORIGINAL MATRIX:
     300      290      280      290      210        0        0        0
     250      310      290      300      200        0        0        0
     180      190      300      190      180        0        0        0
     320      180      190      240      170        0        0        0
     270      210      190      250      160        0        0        0
     190      200      220      190      140        0        0        0
     220      300      230      180      150        0        0        0
     260      190      260      210      180        0        0        0

     THE OPTIMAL SOLUTION OF THE ASSIGNMENT PROBLEM'S TOTAL COST =      870

          THE ASSIGNMENT FOR    1 =         6
          THE ASSIGNMENT FOR    2 =         7
          THE ASSIGNMENT FOR    3 =         1
          THE ASSIGNMENT FOR    4 =         2
          THE ASSIGNMENT FOR    5 =         3
          THE ASSIGNMENT FOR    6 =         5
          THE ASSIGNMENT FOR    7 =         4
          THE ASSIGNMENT FOR    8 =         8

CPU TIME FOR HUNGAR=0.005616 SECONDS


1 MACHINE        7 JOBS

PROCESSING TIMES FOR EACH JOB:
   6.  8.  4.  7. 21. 11. 13.

DUE DATES FROM NOW FOR EACH JOB:
   9. 13. 16. 22. 35. 40. 50.

OPTIMUM SEQUENCE TO MAXIMIZE MINIMUM LATENESS
  1   2   3   5   4   6   7

CPU TIME FOR MAXTRD=0.001716 SECONDS


1 MACHINE        6 JOBS

PROCESSING TIMES FOR EACH JOB:
   6.  8.  4.  7. 21. 11.

DUE DATES FROM NOW FOR EACH JOB:
   9. 13. 16. 22. 35. 40.

LATE JOB IS   2

LATE JOB IS   5

OPTIMUM SEQUENCE TO MINIMIZE NUMBER OF LATE JOBS
  1   3   4   6   2   5

  4 JOBS ON TIME,   2 JOBS LATE

CPU TIME FOR MINLAT=0.002444 SECONDS


1 MACHINE        8 JOBS

PROCESSING TIMES FOR EACH JOB:
   7.  2.  3.  4.  9. 11.  5.  4.
```

```
OPTIMAL SEQUENCE TO MINIMIZE SUM OF COMPLETION TIMES
  2   3   4   8   7   1   5   6

CPU TIME FOR MINSCT=0.001196 SECONDS


1 MACHINE        7 JOBS

DUE DATES FROM NOW FOR EACH JOB:
  21. 18. 15.  2.  7.  9. 25.

OPTIMAL SEQUENCE TO MINIMIZE MAXIMUM TARDINESS
  4   5   6   3   2   1   7

CPU TIME FOR MINTRD=0.001092 SECONDS


1 MACHINE        6 JOBS

PROCESSING TIMES FOR EACH JOB:
   2.  4.  7.  6.  3.  2.

WEIGHTS FOR EACH JOB:
   4.  6.  2.  1.  5.  3.

OPTIMAL SEQUENCE TO MINIMIZE WEIGHTED SUM OF COMPLETION TIMES
  1   5   2   6   3   4

CPU TIME FOR MINWTS=0.001508 SECONDS
SEQUENCING   8 JOBS THROUGH 2 MACHINES


JOB     PROCESSING TIME PROCESSING TIME

            MACHINE 1        MACHINE 2

 1             4.00             6.00
 2             8.00             3.00
 3             7.00             6.00
 4             8.00             4.00
 5             2.00             6.00
 6             1.00             5.00
 7             3.00             7.00
 8             9.00             2.00


 OPTIMAL SEQUENCE FOLLOWS:
   6   5   7   1   3   4   2   8

 TOTAL ELAPSED TIME      =      44.00

 IDLE TIME ON MACHINE 1 =       2.00

 IDLE TIME ON MACHINE 2 =       5.00

CPU TIME FOR NJOB2M=0.003432 SECONDS


SEQUENCING   5 JOBS THROUGH 3 MACHINES


JOB     PROCESSING TIME PROCESSING TIME

            MACHINE 1        MACHINE 2        MACHINE 3

 1             4.00             5.00             5.00


 2             2.00             2.00             6.00
 3             8.00             3.00             8.00
 4            10.00             3.00             9.00
 5             5.00             4.00             7.00


 OPTIMAL SEQUENCE FOLLOWS:
   2   1   5   3   4

 TOTAL ELAPSED TIME      =      41.00

 IDLE TIME ON MACHINE 1 =      12.00

 IDLE TIME ON MACHINE 2 =      24.00

 IDLE TIME ON MACHINE 3 =       6.00

              MACHINE 1              MACHINE 2              MACHINE 3
          TIME        TIME       TIME        TIME       TIME        TIME
JOB         IN         OUT         IN         OUT         IN         OUT

  2       0.00        2.00       2.00        4.00       4.00       10.00
  1       2.00        6.00       6.00       11.00      11.00       16.00
  5       6.00       11.00      11.00       15.00      16.00       23.00
  3      11.00       19.00      19.00       22.00      23.00       31.00
  4      19.00       29.00      29.00       32.00      32.00       41.00

CPU TIME FOR NJOB3M=0.004472 SECONDS


                                P  E  R  T

     I          J          A          M          B        T(I,J)    VT(I,J)

 1.000      2.000      2.200      2.450      6.000      3.000      0.401
 1.000      3.000      1.510      6.120     10.000      5.998      2.002
 1.000      4.000      5.000      9.900     15.390      9.998      2.998
 2.000      5.000      6.000      7.500     12.000      8.000      1.000
 3.000      4.000      2.000      5.000      8.000      5.000      1.000
 3.000      6.000      5.510     10.120     14.000      9.998      2.002
 4.000      5.000      1.760      4.060      6.000      4.000      0.499
 4.000      7.000      4.000      8.160     11.350      7.998      1.501
 5.000      7.000      5.000     10.380     13.490     10.002      2.002
 6.000      7.000      2.510      5.620     11.000      5.998      2.002

                EARLIEST          LATEST
     EVENT          TIME            TIME          SLACK

         1         0.000          -0.000         -0.000

         2         3.000           6.998          3.998

         3         5.998           5.998         -0.000

         4        10.998          10.998         -0.000

         5        14.998          14.998         -0.000

         6        15.997          19.002          3.005

         7        25.000          25.000          0.000

    CRITICAL PATH   1   3   4   5   7
```

EVENT	EARLIEST TIME	DUE DATE	PROBA-BILITY
2	3.000	4.000	0.843
3	5.998	6.000	0.500
4	10.998	10.000	0.282
5	14.998	16.000	0.704
6	15.997	18.000	0.842
7	25.000	0.000	0.000

CPU TIME FOR PERT =0.010296 SECONDS

THE SHORTEST PATH FROM 16 TO 5 = 12

CPU TIME FOR DIJKST=0.008996 SECONDS

ALL SHORTEST PATH MATRIX:

0	3	5	6	7	2	7	6	7	8	6	5	8	11	10	7	8	9	10	11
3	0	2	3	4	4	4	3	4	5	8	7	9	8	7	9	10	10	9	8
5	2	0	1	2	6	3	1	2	3	10	8	7	6	5	10	9	8	7	6
6	3	1	0	1	7	4	2	3	4	11	9	8	7	6	11	10	9	8	7
7	4	2	1	0	8	5	3	4	5	12	10	9	8	7	12	11	10	9	8
2	4	6	7	8	0	8	7	8	9	4	3	6	9	10	5	6	7	8	9
7	4	3	4	5	8	0	2	3	4	9	5	8	7	6	8	8	9	8	7
6	3	1	2	3	7	2	0	1	2	10	7	6	5	4	9	8	7	6	5
7	4	2	3	4	8	3	1	0	1	9	8	7	4	3	8	7	6	5	4
8	5	3	4	5	9	4	2	1	0	8	9	6	3	2	7	6	5	4	3
6	8	10	11	12	4	9	10	9	8	0	4	5	5	6	1	2	3	4	5
5	7	8	9	10	3	5	7	8	9	4	0	3	6	8	3	4	5	6	7
8	9	7	8	9	6	8	6	7	8	5	3	0	3	5	4	3	2	3	4
11	8	6	7	8	9	7	5	4	3	5	6	3	0	3	4	3	2	2	3
10	7	5	6	7	10	6	4	3	2	6	8	5	3	0	5	4	3	2	1
7	9	10	11	12	5	8	9	8	7	1	3	4	4	5	0	1	2	3	4
8	10	9	10	11	6	9	8	7	6	2	4	3	3	4	1	0	1	2	3
9	10	8	9	10	7	9	7	6	5	3	5	2	2	3	2	1	0	1	2
10	9	7	8	9	8	8	6	5	4	4	6	3	2	2	3	2	1	0	1
11	8	6	7	8	9	7	5	4	3	5	7	4	3	1	4	3	2	1	0

CPU TIME FOR FLOYD =0.039884 SECONDS

MINIMUM COST FROM NODE 16 TO ALL OTHER NODES:

7	9	10	11	12	5	8	9	8	7	1	3	4	4	5	0	1	2	3	4

CPU TIME FOR NETPTH=0.003068 SECONDS

SHORTEST PATH FROM NODE 16 TO 5

16	17	18	14	10	9	8	3	4	5

CPU TIME FOR SHORTP=0.000416 SECONDS

REFERENCES

Ackoff, Russell L., Sasieni, Maurice W.,
Fundamentals of Operations Research, John Wiley & Sons, 1968.

Aron, Joel D.,
The Program Development Process, Addison Wesley, 1974.

Baker, Kenneth R.,
Introduction to Sequencing & Scheduling, John Wiley & Sons, 1974.

Bazaraa, Mokhtar S., Jarvis, John J.,
Linear Programming and Network Flows, John Wiley & Sons, 1977.

Bector, C.R.,
Weighted Penalty Method (Hybrid Method) in Transportation Algorithms, 1982. (unpublished working paper)
Dept. of Act. & Mngt. Sc., University of Manitoba,
Winninpeg, R3T 2N2.

Blaauw, Gerrit A.,
Digital System Implementation, Prentice-Hall, Series in Automatic Computation, 1976.

Boehm, Barry,
Software Engineering, IEEE Transactions on Computers, 25, 12, December, 1976. pp.1226-1241.

Bondy, J.A., Murty, U.S.R.,
Graph Theory with Applications, North Holland, 1980.

Boothroyd, J.,
Algorithm Supplement, Computer Journal, vol. 10, 1968. pp. 306-308.

Breuer, M.A.,
Design Automation of Digital Systems, vol. 1, Theory and Techniques, Prentice-Hall, Englewood Cliffs, N.J. 1972.

Briggs, William A.,
Minimum Excess Cost Curve, Commication ACM 6, Dec., 1963.

Campbell, Herbert G., Dudek, Richard A., Smith, Milton L.,
A Heuristic Algorithm for the n Job, m Machine Sequencing Problem,
Management Science, vol. 16, No. 10, June, 1970.

Carey, T.T., Mason R.E.A.,
Information System Prototyping: Techniques, Tools and Methodologies. INFOR, Canadian Journal of Operational Research and Information Processing, Vol. 21, Number 3, August 1983.

Conway, Richard W., Maxwell, William L., Miller, Louis W.,
Theory of Scheduling, Addison-Wesley, 1967.

Cooper, L., Bhat, U.N., LeBlanc, L.J.,
Introduction to Operations Research Models, W.B. Saunders, 1977.

Eastman, W.L.,
Linear Programming with Pattern Constraints, The Computer Laboratory, Report No. B1-20, Harvard Press, 1958.

Gillett, Billy E.,
Introduction to Operations Research, Series in Industrial Engineering & Management Science, McGraw-Hill, 1976.

Gue, Ronald L., Thomas, Michael E.,
Mathematical Methods in Operations Research, The MacMillan Co., Collier-Macmillan Ltd., 1968.

Gupta, Jatinder N.D.,
A Functional Heuristic Algorithm for the Flowshop Scheduling Problem, Operational Research Quarterly, vol. 22, pp.199-205, March, 1971.

Hicks, Philip E.,
Introduction to Industrial Engineering and Management Science, McGraw-Hill, 1977.

Horowitz, Ellis, Sahni, Sartaj,
Fundamentals of Data Structures, Computer Science Press Inc., 1982.

IBM,
A Systems Analysis and Documentation Technique, GT20-8102-0, White Plains, N.Y., 1962.

IBM,
HIPO: A design Aid and Documentation Technique, GC20-1851, White Plains, N.Y., 1974.

Johnson, Lynwood A., Montgomery, Douglas C.,
Operations Research in Production Planning, Scheduling & Inventory Control, New York, Wiley & Sons, 1974.

Johnson, S.M.,
Optimal Two and Three Stage Production Schedules with Set-Up times Included, Naval Research Logistics Quarterly, Vol.1, pp.61-68, 1954.

Katzan, Harry Jr.,
Fortran 77, Computer Science Series, Van Nostrand Reinhold, 1978.

Kelley J.,
Critical-Path Planning and Scheduling: Mathematical Basis, Operations Research, 9, No. 3, pp.296-320, June 1961.

Knuth, Donald E.,
Sorting and Searching, The Art of Computer Programming, vol. 3, Addison Wesley, 1972.

Lai D.,
Weighted Penalty Methods and VAM's Weighted Penalty Methods in Transportation Algorithms, 1983. (term paper)
available from Dr. C.R. Bector, Dept. of Act. & Mngt. Sc., University of Manitoba, Winnipeg, R3T 2N2.

Lawler, Eugene,
Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston, 1976.

Lee, Sang M., Moore, Laurence J., Taylor, Bernard W., Management Science, Wm.C. Brown Co., 1981.

Linger, R.C., Mills, H.D.,
Structured Programming, Addison-Wesley, Reading, Mass., 1972.

Little, J.D.C., Murty, K.G., Sweeny D.W., Karel, C.,
Algorithm for the Traveling Salesman Problem, Operations Research, 11, No. 6, pp.979-989, November 1963.

Loomba, Narendra Paul, Turban, Efraim,
Applied Programming for Management, Holt, Rinehart & Winston Inc., 1974.

Minieka, Edward,
Optimization Algorithms for Networks and Graphs, Industrial Engineering Series vol. 1, Marcel Dekker Inc., New York, 1978.

Mital, K.V.,
Optimization Methods (in operations research and systems analysis), A Halsted Press Book, John Wiley & Sons, 1976.

Moder, Joseph J., Elmaghraby, Salah E.,
Handbook of Operations Research, Van Nostrand Reinhold Co., 1978.

Moore, P.G.,
Basic Operational Research, A Pitman International Text, Second Edition, 1976.

Morlok, Edward K.,
Introduction to Transportation Engineering and Planning, McGraw-Hill, 1978.

Muth, John F.,
Remark on Algorithm 217 [H], Minimum Excess Cost Curve, Communication ACM 6, Dec. 1967.

Nicholson, T.A.J.,
Finding the Shortest Route between Two Points in a Network, Computer Journal, vol. 9, pp.275-280, 1960.

Orr, Kenneth T.,
Structed Systems Development, Yourden Press, New York, 1977.

Palmer, R.D.,
Sequencing Jobs through a Multi-Stage Process in the Minimum Total Time --- A quick method of obtaining a Near Optimum, Operational Research Quarterly, vol.16, No.1, pp.100-107, March 1965.

Siemens, Nicolai, Marting, C.H., Greenwood, Frank,
Operations Research, The Free Press, 1973.

Shimon, Even,
Graph Algorithms, Computer Science Press, 1979.

Steenbrink, Peter A.,
Optimization of Transport Networks, A Wiley-Interscience Publication, John Wiley & Sons, 1973.

Wagner, Harvey M.,
Principles of Management Science with Applications to Executive Decisions, Prentice-Hall, 1970.

Warnier, Jean Dominique,
Logical Construction of Programs, Van Nostrand Reinhold Company, New York, 1974.

Wiest, Jerome D., Levy, Ferdinand K.,
A Management Guide to PERT/CPM, Prentice-Hall, 1969.

Woolsey, Robert E.D., Swanson, Huntington S.,
Operations Research for Immediate Application - A Quick & Dirty Manual, Harper & Row, 1975.

Yourdon, Edward,
Techniques of Program Structure and Design, Prentice-Hall, 1975.

Zierer T.K., Mitchell W.A., White T.R.,
Practical Applications of Linear Programming to Shell's Distribution Problems, ORSA/TIMS reports, Oct. 16-18, 1974.