USER'S GUIDE TO MATCOV

by

Clovis Perin

under supervision of

Prof. K. G. Murty

Technical Report 80-5

Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48109

September 1980

Research effort is partially supported by the Department of Industrial and Operations Egineering of The University of Michigan and by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant no. AFOSR 78-3646. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. This paper was typed by Mrs. Geraldine Cox.

TABLE OF CONTENTS

1. INTRODUCTION

This report discusses a FORTRAN implementation of the algorithm presented in [1] for the minimum cost 1-matching/covering problem.

Given a network $G = (N,A,c)$ where N is a nonempty set of n nodes, A is a set of m edges (an edge is an unordered pair of distinct nodes), and $c = [c_{i;j}]$ is a vector of edge costs, the minimum cost 1-matching/covering problem is formulated as follows:

Find an m-vector $x = [x_{i;j}]$

$$\text{To minimize} \quad \Sigma_{i;j}\, c_{i;j}\, x_{i;j}$$

$$\text{Subject to} \quad \Sigma_j\, x_{i;j} \begin{cases} \leqq 1 & \text{for } i \in N^{\leqq} \\ = 1 & \text{for } i \in N^{=} \\ \geqq 1 & \text{for } i \in N^{\geqq} \\ \text{unconstrained} & \text{for } i \in N^{o} \end{cases}$$

$$x_{i;j} = 0,\, 1 \qquad \text{for } (i;j) \in A$$

where N^o, $N^{\leqq}$, $N^{=}$, $N^{\geqq}$ is a given partition of N.

It is assumed that the reader is already familiar with the solution of this problem, which is discussed in [1].

2. PROGRAM INPUT

Every node and edge of the network is uniquely identified by a user assigned node/edge number. Each pseudonode created during the execution os the program is automatically associated with a unique pseudonode number. These numbers need not be serial, but they should lie within the ranges specified later on.

The program starts by reading the network which is defined by the data below:

a) node data (node number, subset the node belongs to);

b) edge data (edge number, pair of nodes joined by the edge, and edge-cost).

Successive iterations of post-optimality can be carried out. Changes introduced at each iteration define a new network, and these changes are always made on the latest network. Changes in the network are classified into the following types:

a) edge-cost change (edge number, and new edge-cost);

b) partition change (node number, and new subset);

c) edge elimination (edge number);

d) node elimination (node number);

e) new node introduction (node number, and subset);

f) new edge introduction (edge number, pair of nodes joined by the edge, edge-cost).

In short, the program input has the following structure:

i) network data (node and edge data);

ii) first post-optimality iteration (change data);

iii) second post-optimality iteration (change data);

and so on.

Just as a remark, this implementation is able to work with multi-networks; i.e., it is possible to have two or more edges joining the same pair of nodes. Therefore, every edge must be uniquely identified by an user assigned edge number (the cost of two parallel edges may be the same).

The description of the input data is based on the variables given below.

<u>title</u> - string with at most 48 characters, choosen by the user to identify the problem.

<u>n</u> - number of nodes ($n \leq 100$).

<u>m</u> - number of edges ($m \leq 3000$).

<u>option</u> - program option (see section 5 for details)

option = 1 : program prints statistics only;

option = 2 : program prints final solution only (suggested for most of the applications);

option = 3 : program prints progress report, final solution, and statistics;

option = 4 : program prints input data, progress report, final solution, and statistics.

<u>node</u> - node number (integer between 1 and 100); must be unique for each node.

set - indicates the subset that node belongs to, under the code:

set = 0: node $\in N^o$

set = 1: node $\in N^=$

set = 2: node $\in N^{\leqq}$

set = 3: node $\in N^{\geqq}$

edge - edge number (integer between 1 and 3000); must be unique for each edge.

cost - edge cost (real number between -9999.9 and +9999.9).

$node_1$, $node_2$ - numbers of the pair of nodes joined by edge.

k_1 - total number of edges which have the cost coefficient changed during a post-optimality iteration.

k_2 - total number of set changes.

k_3 - total number of edge eliminations.

k_4 - total number of node eliminations.

k_5 - total number of node introductions.

k_6 - total number of edge introductions.

The user must provide the input data in FORTRAN free format; i.e., blanks or commas can be used as separator of data, and each line below must start in a new line of the input file.

Input Data

title	
m, n, option	
node, set	(one line per node)
edge, $node_1$, $node_2$, cost	(one line per edge)
k_1, k_2, k_3, k_4, k_5, k_6	
edge, newcost	(one line per cost change)
node, newset	(one line per set change)
edge	(one line per edge elimination)
node	(one line per node elimination)
node, set	(one line per node introduction)
edge, $node_1$, $node_2$, cost	(one line per edge introduction)
...	
0, 0, 0, 0, 0, 0	(last line)

The 2nd, 3rd., and 4th. lines define the original network. From the 5th. line to the 11th. line above, a post-optimality iteration is defined. Every post-optimal iteration should be given under such a general form. When several iterations are used, the network at the beginning of iteration k is the same as the network at the end of iteration k-1. Finally, the last line of the input data should be a line with zix zeros separated by commas or blanks.

3. PROGRAM EXECUTION

This section is appropriate for users of the Michigan Terminal System (MTS) of the University of Michigan; users of different installations should prepare the necessary modifications.

Batch Mode

On batch mode, the deck of cards to be punched should have the following structure:

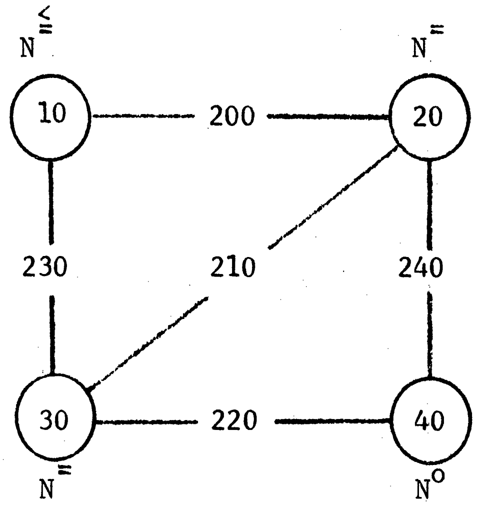
```
$SIG ccid
password
$RUN K45V:MATCOV T=time
   .
   .    input data
   .
$ENDFILE
$SIG
```

where ccid, and password are assigned for each user of MTS. As an order of magnitude for time, MATCOV is able to solve a network with 50 nodes and 1000 edges in less than 5 s.

Example 1

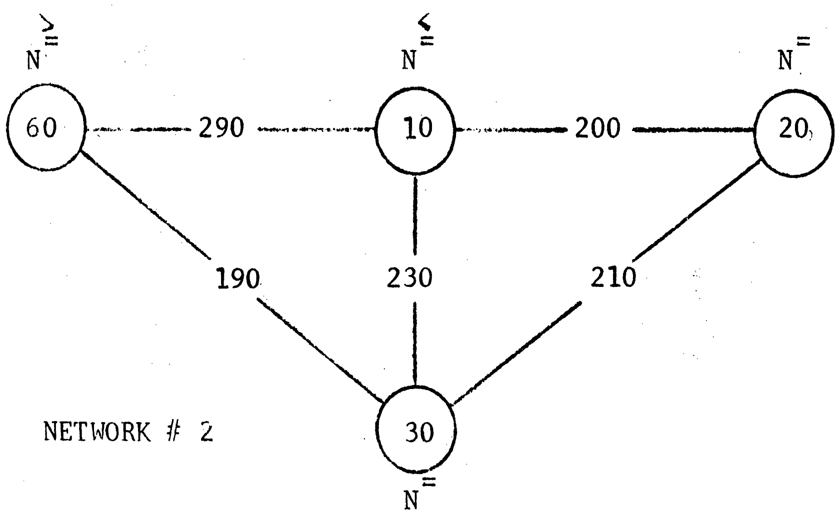
Consider the network in Figure I. The input data to be used in the solution of these three problems is shown in Output I. Note that the original network has 4 nodes and 5 edges, the first modified network has 4 nodes and 5 edges, and the last network has 3 nodes and 2 edges. Output II shows the optimum solution for the original network. The first post-optimality iteration was performed with 1 edge cost change, 1 edge elimination, 1 node elimination, 1 node introduction, and 2 edge introduction; edge 240 = (20;40) was automatically eliminated from the network because it became incident with a nonexisting node (node 40 was eliminated). The optimum solution for the first modified network is given in Output III. In the second post-optimality iteration, one edge and one node were explicitly removed from the network, and two edges were automatically removed because they were incident with eliminated node. The final solution for the second modified network, which is infeasible, is presented in Output IV; the last network corresponds to an infeasible minimum cost 1-matching/covering problem.

NETWORK # 1



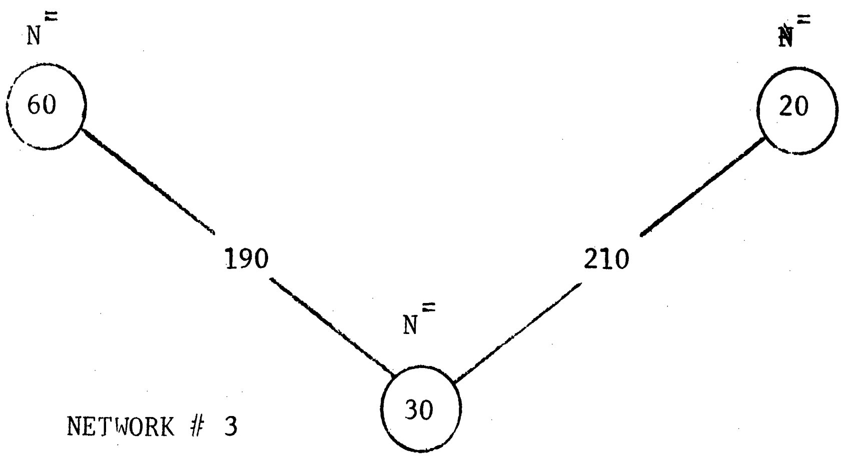
edge	cost
200	20.0
210	-50.0
220	-4.0
230	2.57
240	-2.0

NETWORK # 2



edge	cost
190	10.0
200	20.0
210	-50.0
230	3.0
290	-10.0

NETWORK # 3



edge	cost
190	10.0
210	-50.0

Figure I - Example 1

```
$SIG ccid
password
$RUN K45V:MATCOV T=1

                 EXAMPLE 1

  4    5    2

 10    2
 20    1
 40    0
 30    1

240   20   40    -2.0
220   30   40    -4.0
200   20   10    20.0
210   20   30   -50.0
230   10   30     2.57

  1    0    1    1    1    2

230    3.0
220
 40
 60    3
290   10   60   -10.0
190   60   30   +10.0

  0    1    0    1    0    0

 60    1
 10

  0    0    0    0    0    0

$ENDFILE

$SIG
```

Output I - list of the deck of cards for solving Example 1

```
*****   MINIMUM COST 1-MATCHING/COVERING PROBLEM   *****

                     EXAMPLE 1

             *****  SOLUTION  *****

        NETWORK:      4 NODES,     5 EDGES

           OBJECTIVE VALUE:     -50.00

LIST OF NODES (NODE,SET,PRICE) :
     10 2    0.0        20 1  -25.0        40 2    0.0        30 1  -25.0

LIST OF EDGES (EDGE,NODE1,NODE2,COST) :
   240 40 20   -2.0    220 40 30   -4.0    200 10 20   20.0    210 30 20  -50.0    230 30 10    2.6

SOLUTION EDGES:
     210
```

Output II - Optimum solution for network # 1 of Example 1

```
*****  POST-OPTIMALITY ANALYSIS  *****

CHANGES:             1 EDGE COSTS,    2 NODE SETS
ELIMINATIONS:        1 EDGES,         1 NODES
INTRODUCTIONS:       1 NODES,         2 EDGES


*****  SOLUTION  *****

NETWORK:     4 NODES,     5 EDGES


OBJECTIVE VALUE:    -60.00


LIST OF NODES (NODE,SET,PRICE) :
        10 2  -10.0         20 1  -25.0           30 1  -25.0           60 3    0.0

LIST OF EDGES (EDGE,NODE1,NODE2,COST) :
    200 10 20   20.0       210 30 20  -50.0     230 30 10    3.0      290 10 60  -10.0      190 60 30    10.0

SOLUTION EDGES:
      210         290
```

Output III - optimum solution for network # 2 of Example 1.

```
*****  POST-OPTIMALITY ANALYSIS  *****

CHANGES:          0 EDGE COSTS,    1 NODE SETS
ELIMINATIONS:     0 EDGES,         1 NODES
INTRODUCTIONS:    0 NODES,         0 EDGES

          *****  SOLUTION  *****

     NETWORK:     3 NODES,    2 EDGES

          =  PROBLEM INFEASIBLE  =

     OBJECTIVE VALUE:     -15.00

LIST OF NODES (NODE,SET,PRICE) :
      20 1  -25.0          30 1  -25.0          60 1   35.0

LIST OF EDGES (EDGE,NODE1,NODE2,COST) :
    210 30 20  -50.0        190 60 30    10.0

SOLUTION EDGES:
     210
```


Output IV - Infeasible solution for network # 3 of Example 1

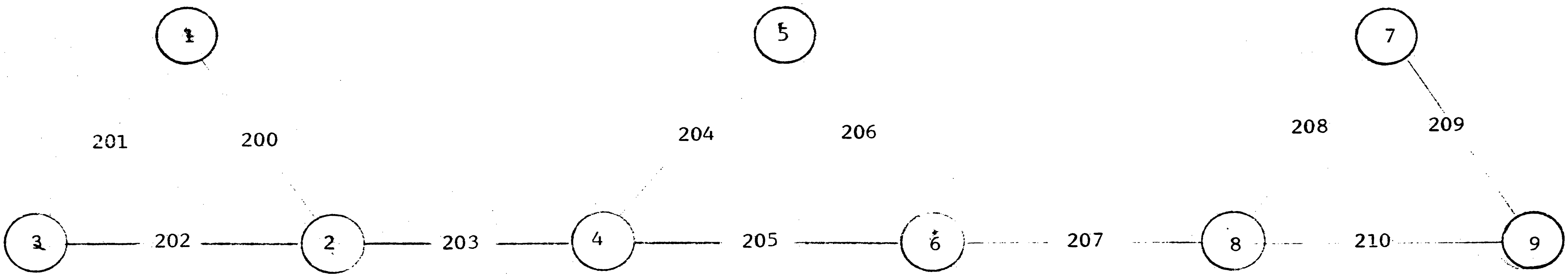
Terminal Mode

On terminal mode, it is convenient to create a file containing the input data previously to the execution of the program. In the MTS statement below, infile indicates such a file, and outfile represents the file to which the output will be directed, such as *PRINT*. As an order of magnitude for time, MATCOV is able to solve a network with 50 nodes and 1000 edges in less than 5 s.

```
$RUN K45V:MATCOV 5=infile 6=outfile T=time
```

Example 2

Consider the networks in Figure II. The input data for such an example is listed in Output V. All three networks have 9 nodes and 11 edges. In the first network, all nodes are in the subset $N^{=}$. During the first post-optimality iteration, node 9 is transferred to subset $N^{\leqq}$. During the second post-optimality iteration, node 9 is transferred to subset $N^{\geqq}$. The final solution for the original network, which is infeasible, is presented in Output VI; it contains 2 pseudonodes. Output VII shows the optimum solution for the first modified network, which contains just one pseudonode. Finally, the solution of the last network has two pseudonodes and it is shown in Output VIII.



edge 203 has cost 10.0

edge 207 has cost 80.0

remaining edges have cost 2.0

NETWORK # 1 - all nodes are in $N^{=}$

NETWORK # 2 - node 9 is in $N^{\leq}$; remaining nodes are in $N^{=}$

NETWROK # 3 - node 9 is in $N^{\geq}$; remaining nodes are in $N^{=}$

Figure II - Networks of Example 2.

EXAMPLE 2

9	11	2			
1	1				
2	1				
3	1				
4	1				
5	1				
6	1				
7	1				
8	1				
9	1				
200	1	2	2.0		
201	1	3	2.0		
202	2	3	2.0		
203	2	4	10.0		
204	4	5	2.0		
205	4	6	2.0		
206	5	6	2.0		
207	6	9	80.0		
208	7	8	2.0		
209	7	9	2.0		
210	8	9	2.0		
0	1	0	0	0	0
9	2				
0	1	0	0	0	0
9	3				
0	0	0	0	0	0

Output V - list of the input data corresponding to the Example 2

```
*****    MINIMUM COST 1-MATCHING/COVERING PROBLEM    *****

                         EXAMPLE 2

                   *****   SOLUTION   *****

              NETWORK:      9 NODES,     11 EDGES

                  =  PROBLEM INFEASIBLE  =

               OBJECTIVE VALUE:      95.00

LIST OF NODES (NODE,SET,PRICE) :
           1 1     1.0          2 1     1.0          3 1     1.0          4 1     1.0          5 1     1.0
           6 1     1.0          7 1     1.0          8 1     1.0          9 1     1.0

LIST OF EDGES (EDGE,NODE1,NODE2,COST) :
     200   2   1     2.0     201   3   1     2.0     202   3   2     2.0     203   4   2    10.0     204   5   4     2.0
     205   6   4     2.0     206   6   5     2.0     207   8   6    80.0     208   8   7     2.0     209   9   7     2.0
     210   9   8     2.0

PSEUDO  PRICE   SIMPLE BLOSSOM
   101    8.0       2    3    1
   103   78.0       8    7    9

SOLUTION EDGES:
     203         206         201         208
```

Output VI - Final solution for Network # 1 of Example 2

```
*****  POST-OPTIMALITY ANALYSIS  *****

CHANGES:           2 EDGE COSTS,    1 NODE SETS
ELIMINATIONS:      0 EDGES,         0 NODES
INTRODUCTIONS:     0 NODES,         0 EDGES

*****  SOLUTION  *****

NETWORK:      9 NODES,    11 EDGES

OBJECTIVE VALUE:      16.00

LIST OF NODES (NODE,SET,PRICE) :
          1 1    1.0          2 1    1.0          3 1    1.0          4 1    1.0          5 1    1.0
          6 1    1.0          7 1    1.0          8 1    1.0          9 2    0.0

LIST OF EDGES (EDGE,NODE1,NODE2,COST) :
     200  2  1    2.0     201  3  1    2.0     202  3  2    2.0     203  4  2   10.0     204  5  4    2.0
     205  6  4    2.0     206  6  5    2.0     207  8  6   80.0     208  8  7    2.0     209  9  7    2.0
     210  9  8    2.0

PSEUDO  PRICE   SIMPLE BLOSSOM
  101    8.0      2    3    1

SOLUTION EDGES:
     203        206        208        201
```


Output VII - Optimum solution for Network # 2 of Example 2

```
*****  POST-OPTIMALITY ANALYSIS  *****

CHANGES:          0 EDGE COSTS,    1 NODE SETS
ELIMINATIONS:     0 EDGES,         0 NODES
INTRODUCTIONS:    0 NODES,         0 EDGES

*****  SOLUTION  *****

NETWORK:      9 NODES,    11 EDGES

OBJECTIVE VALUE:     18.00

LIST OF NODES (NODE,SET,PRICE) :
       1 1    1.0         2 1    1.0         3 1    1.0         4 1    1.0         5 1    1.0
       6 1    1.0         7 1    1.0         8 1    1.0         9 3   -0.0

LIST OF EDGES (EDGE,NODE1,NODE2,COST) :
   200  2  1    2.0    201  3  1    2.0    202  3  2    2.0    203  4  2   10.0    204  5  4    2.0
   205  6  4    2.0    206  6  5    2.0    207  9  6   80.0    208  8  7    2.0    209  9  7    2.0
   210  9  8    2.0

PSEUDO  PRICE   SIMPLE BLOSSOM
  101    8.0      2    3    1
  103    1.0      9    7    8

SOLUTION EDGES:
     203          205          201          209          210
```

Output VIII - Optimum solution for network # 3 of Example 2

5. PROGRAM OUTPUT

There are several possible outputs in accordance with the value assigned by the user to the variable OPTION. They are classified into:

option = 1 : statistical output only;

option = 2 : final solution output only;

option = 3 : progress,final solution, and statistical outputs;

option = 4 : input data, progress, final solution, and statiscs;

see section 9 for options providing debugging output.

The final solution output is printed at the end of each iteration; there is one for each network being solved (this output is produced by all options except by option 1). Such a report consists of the following information:

a) message "PROBLEM IS INFEASIBLE", if appropriate;

b) total number of nodes and total number of edges;

c) dual objective value of the final solution (this is the optimum objective value whenever the problem is feasible);

d) list of all nodes, comprising for each node,

i) node number,

ii) subset code (0, 1, 2, 3 for N^{o}, $N^{=}$, $N^{\leqq}$, $N^{\geqq}$, respectively),

iii) node price (dual variable);

e) list of pseudonodes (if appropriate), comprising for each pseudonode,

i) pseudonode number,

ii) pseudonode price (dual variable),

iii) list of nodes in the simple blossom;

f) list of edges, comprising for each edge,

i) edge number,

ii) and iii) pair of nodes joined by the edge,

iv) edge cost;

g) list of all edges in the (primal) solution.

Statistics are provided for every option, except by option 2. It is expected to be used by researchers who are interested in comparing performance of codes. This report provides a table with the following information:

a) Dual Solution Changes - number of times the dual solution change step was executed during the iteration;

b) Paths rematched - number of alternating paths rematched during all augmentations of the iteration.

c) Nodes Remated - number of nodes in those alternating paths;

d) Nodes Scanned - number of nodes removed from the list of unscanned nodes;

e) Edges Scanned - number of (equality) edges which were scanned;

f) Nodes Labelled - number of outer and inner labels assigned;

g) Shrinkings - number of simple blossom shrinkings;

h) Unshrinkings - number of pseudonode unshrinkings;

i) LINK/MATE's Used - number of starage positions used for LINK/MATE information (the value of P2 was estimated by this statistic);

j) CPU MSEC (INPUT) - time in cpu msec used for initialization and input operations;

k) CPU MSEC (SOLUTION) - time in cpu msec used for solving the problem;

l) CPU MSEC (OUTPUT) - time in cpu msec used for performing the final solution transformation and printing the output.

The progress output is available by option 3 and option 4; it contains a number of messages indicating important marks that the algorithm achieves during the solution, such as the following:

number of exposed nodes at initialization;

node scanning;

node labelling;

augmentation;

simple blossom shrinking;

pseudonode unshrinking;

dual solution change;

node/edge inspection;

etc.

In this report, several symbols are used; among them we have the following:

<R> - indicates the root of an alternating tree

<+> - indicates an outer node

<-> - indicates an inner node

< > - indicates an unlabelled node

<=> - indicates an equality edge

edge = (node$_1$,node$_2$) - indicates an edge and the pair of nodes joined by the edge;

DUAL SOLUTION CHANGE: XXXX.X - indicates the increase in the dual objective value per exposed node (variable h in [1]).

Only option 4 provides a listing of the input data at the time data is entering; such output is useful for detection of errors in the input data.

5. ERROR MESSAGES

This section deals with the events that cause error messages and the subsequent halt in the execution of the program. It is expected that the message printed to be sufficient to lead to the error; however, since no indication is given to the input line the error was detected, it may be useful to rerun the program using option 4. For some errors, it may be necessary to augment the size of the program and recompile it (see section 10 for such cases).

The complete list of error messages is given below.

UNEXPECTED END OF FILE

self explanatory.

ILLEGAL PARAMETER

$n < 0$, $k_1 < 0$, $k_3 < 0$, $k_5 < 0$,

$m < 0$, $k_2 < 0$, $k_4 < 0$, $k_6 < 0$.

ILLEGAL NODE

node < 1,

node > N2 (100),

duplication of an existing node,

edge incident with a nonexisting node,

elimination of a nonexisting node.

ILLEGAL EDGE

edge < 1

edge > M2 - L2 (3000)

duplication of an existing edge

elimination of a nonexisting edge.

ILLEGAL NODE SET

set < 0

set > 3

NODE STORAGE SPACE EXCEEDED [see section 10]

n > N2 (100)

introducing more than N2 (100) nodes.

EDGE STORAGE SPACE EXCEEDED [see section 10]

m > M2 - L2 (3000)

introducing more than M2-L2 (3000) edges.

LINK/MATE STORAGE SPACE EXCEEDED [see section 10]

HINK > P2 (1500)

ERROR NO. XXX

not used

6. CPU TIMES

Minimum cost 1-matching/covering problems were randomly generated by the program NET.OBJ (the source code called NET.FTN is listed in the appendix). The input for such a program should have the following structure (FORTRAN free format):

```
title
n, m, #sets,  mincost, maxcost, seed
k1, k2, k3, k4, k5, k6
k1, k2, k3, k4, k5, k6
    .
    .
    .
0, 0, 0, 0, 0, 0
```

where title, m, n, k_1 to k_6 have the meaning presented in section 2. The parameter #sets indicates the number of subsets to be used as the partition for the set of nodes; #sets = 1 indicates that all the nodes should be in the set $N^=$; #sets = 3 indicates that all nodes should be evenly distributed among the sets $N^=$, $N^\leqq$, $N^\geqq$ (no node in N^o); #sets = 4 indicates that all nodes should be evenly distributed among the sets $N^=$, $N^=$, $N^=$, N^o. The parameters mincost, maxcost represent the range in which the edge costs are to be uniformily distributed. The parameter seed is used as an initialization number for the random number generator; different networks are generated just by changing seed. The number of post-optimality iterations to be generated are indicated by the number of nonnull lines with the parameters k_1 to k_6. All problems are generated with program option = 1, which means that the execution of MATCOV will produce only statistics.

It should be noted here that the output produced by NET.OBJ can be used directly as input for MATCOV, execpt when there are node eliminations because this may imply that some edges become incident with nonexisting nodes; such edges are automatically eliminated by MATCOV from the network. Since NET.OBJ does not take into account such automatic edge eliminations, it may generate inconsistent changes afterwards.

Table I shows a set of average cpu times obtained from outputs of MATCOV under the statistic CPU MSEC (SOLUTION). These cpu times do not include the cpu time spent during the reading of the input data and during the printing of the final solution. All problems in Table I correspond to networks with 50 nodes; 1000 edges; edge costs (integers) generated in the intervals [-10,10], [-100,100], [100,200], [-200,-200]; and partitions with 1, 3, 4 sets of nodes. Three problems were generated for each cell of Table I.

Table II presents the average cpu times obtained for problems with edge costs generated in the range [-100,100]; partitions with 1, 3, 4 stes of nodes; and (#nodes,#edges) taking the values (20,200), (40,800), (60,1800). Three problems were generated for each cell of Table II. Note that #nodes = $(\#edges)^2 / 2$ for all cases.

Partition	Edge Cost Range			
	[-10, 10]	[-100, 100]	[100, 200]	[-200, -100]
$N^{=}$	902	907	1064	887
$N^{=}, N^{=}, N^{=}$	485	505	1372	477
$N^{=}, N^{=}, N^{=}, N^{o}$	355	354	905	354

Table I - cpu msec (no readings, no writings) for solving minimum cost 1-matching/covering problems with 50 nodes and 1000 edges using AMDAHL 470/V7. Figures are the average of 3 different problems.

Partition	(# Nodes, # Edges)		
	(20, 200)	(40, 800)	(60, 1800)
$N^{=}$	63	472	1796
$N^{=}, N^{=}, N^{=}$	39	252	887
$N^{=}, N^{=}, N^{=}, N^{o}$	34	199	640

Table II - cpu msec (no readings, no writings) for solving minimum cost 1-matching/covering problems with edge cost ranging in the interval [-100,100] using AMDAHL 470/V7. Figures are the avarage of 3 different problems.

Table III indicates the average cpu times obtained for networks with 50 nodes, 1000 edges, edge cost range [-100,100], and partittions with all 4 sets. Post-optimality analysis was applied to the problems in order to have 1000 edge cost changes. Due to the way problems are generated, some edges may have their edge costs changed more than one time in the same post-optimality iteration; this means that some edges may keep their edge costs during one iteration. Two post-optimality iterations were applied for each problem; each cell of Table III corresponds to the average of 3 problems. Cpu times spent for reading the input data, for initialization, and for preparing a post-optimality iteration are shown as CPU TIME (INPUT); they can be compared against the CPU TIME (SOLUTION) spent for solving the problems.

	CPU TIME (MSEC)	
	Input	Solution
Original Network	361	315
1000 edge cost changes	301	119
1000 edge cost changes	299	113

Table III - cpu times for input and solution of problems with 50 nodes and 1000 edges, edge cost range [-100,100] using AMDAHL 470/V8. Problems had twice 1000 edge cost changes. Figures are for the average of 3 different set of problems.

7. DATA STRUCTURE

Intenally, every node is identified by an unique number between N0 (1) and N2 (100); every pseudonode by a number between L0 (101) and L2 (150); and every edge by a number between M0 (151) and M2 (3150). So, the external and internal representation of nodes and pseudonodes is based on the same node/pseudnnode number. However, the internal representation of an edge is obtained by the expression

$$\text{internal edge number} = \text{external edge number} + L2.$$

Moreover, each edge is decomposed into two arcs joining the same pair of nodes, but with different directions; one of the arcs is identified by the edge number corresponding to its edge; the other arc is identified by the complement of that number to M9 = 2.M2 + 1 = 6301.

$$arc_1 = edge$$

$$arc_2 = M9 - edge$$

In this way, every arc is uniquely identified by an arc number in the range M0 (151) to M9 - L2 - 1 (6150).

In the context of this section, all node/pseudonode/edge/arc numbers refer to the internal representation, unless it is explicitly sated as external.

Main Variables

N - number of nodes

M - number of edges

N2 = 100 - maximum node number

L2 = 150 - maximum pseudonode number

M2 = 3150 - maximum edge number

P2 = 1500 - maximum position in LINK/MATE

INF = 9999.9 - maximum edge cost

TOLER = 10^{-50} - tolerance for numerical comparisons

VALUE - dual objective value

ROOT$ - number of exposed nodes

The variables NODE, PNODE (pseudonode), EDGE (edge or arc), IND (index), BLSM (node in a simple blossom), BASE (base of a simple blossom), APICE (apex of a blossom), BNODE or BCINOD (BCI of a blossom), PT (pointer) are used throughout the program with numerical appendices (e.g., NODE2); all these variables have always the meaning indicated, except NODE which is normally used as node or pseudonode, but it may indicate an edge or an arc. The variables ROOT$, NODE$, EDGE$, SHRK$, UNSH$, AUGM$, DUAL$, RMAT$, LABL$, LINK$, TIME1, TIME2, TIME3 have statistical usage. The values assigned to the variables PRINT, TRACE depend on the program option.

List of existing node/edge numbers

LIST(ind) - three-stack vector containing all node/edge numbers which correspond to existing elements in the network.

node numbers:	head = N0	tail = N1
pseudonode numbers:	head = L0	tail = L1
edge numbers:	head = M0	tail = M1

The heads N0, L0, M0 are fixed during the execution; the tails N1, M1 are fixed for each post-optimality iteration; the tail L1 depends on the number of pseudonodes in the solution.

INDEX(node/edge) - vector of indices for LIST

N0 ≤ INDEX(node) ≤ N1	N0 ≤ node ≤ N2
L0 ≤ INDEX(pnode) ≤ L1	L0 ≤ pnode ≤ L2
M0 ≤ INDEX(edge) ≤ M1	M0 ≤ edge ≤ M2

INDEX(node/edge) = 0   implies a nonexisting node/edge

INDEX(node/edge) > 0   implies an index to LIST

for existing node/edge: LIST(INDEX(node/edge)) = node/edge

for a valid ind: INDEX(LIST(ind)) = ind

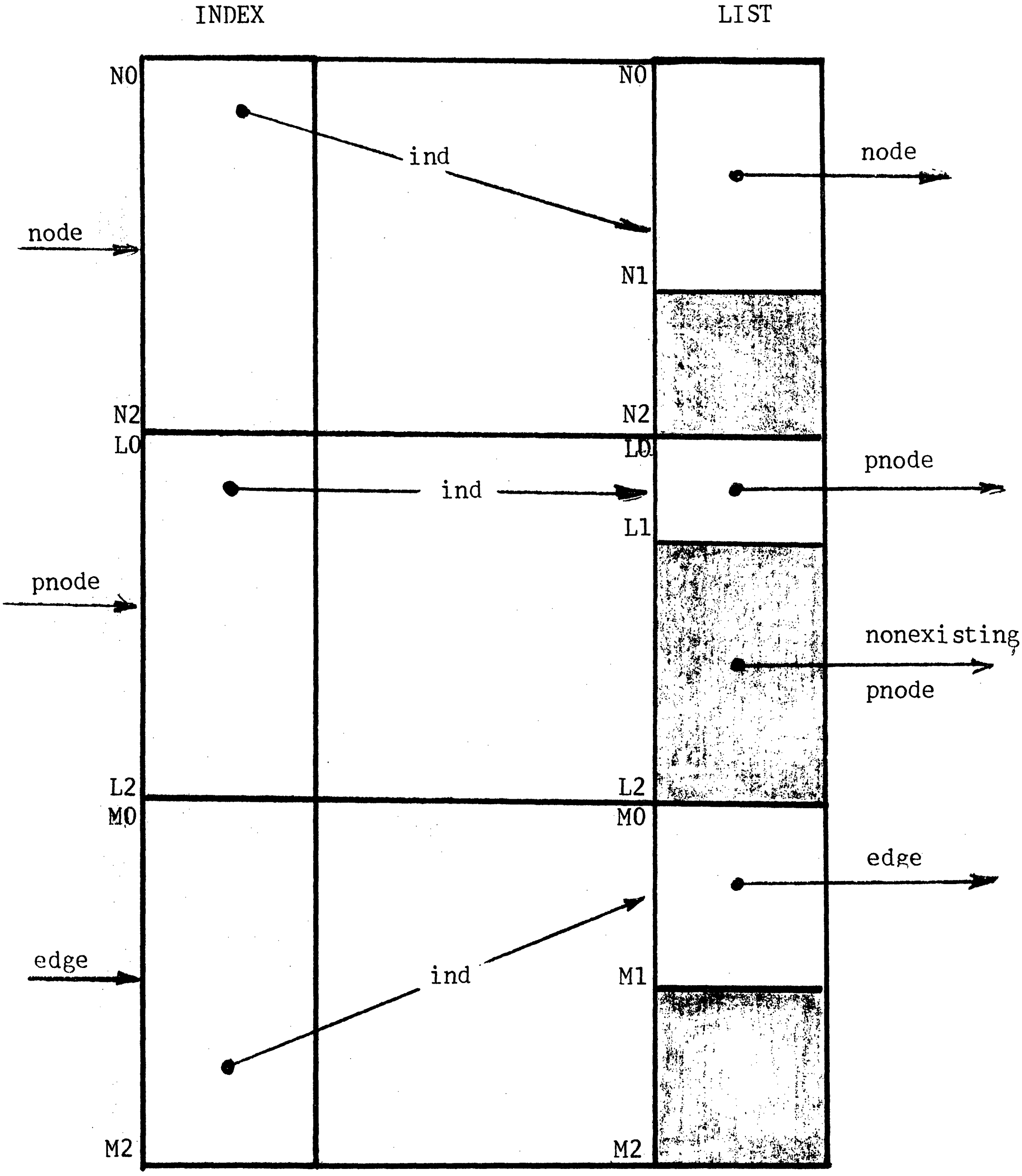


Figure III - list of existing node/edge numbers

Network Specifications

SET(node) - indicates the partition subset that node belongs to, under the code:

0 - node of N^{o}

1 - node of $N^{=}$

2 - node of $N^{\leqq}$

3 - node of $N^{\geqq}$

PRICE(edge) - edge cost of edge

HEAD(arc) - indicates the node that arc is heading to.

in order to determine the two nodes joined by an edge, use the formulas:

$$arc_1 = edge$$
$$arc_2 = M9 - edge$$
$$node_1 = HEAD(arc_1)$$
$$node_2 = HEAD(arc_2)$$

Arc Lists

LINK/MATE(pt) - pair of vectors used for storing the following lists:

incidency-list(node) - contains all current equality arcs heading from node (incidency in the current equality network);

mating-list(node) - contains all current solution arcs heading from node;

internal-list(pnode) - contains equality arcs joining two nodes inside pnode

LINK(node) - head of

MATE(node) - head of mating-list(node)

LINK(pnode) - head of internal-list(pnode)

MATE(pnode) - base of pnode

MATE(pt) - contains the arc number of an element in one of those three lists

LINK(pt) - points to LINK/MATE indicating the next element in the list

Obs.: MATE(node) = 0 implies that there is no solution arc heading from node

MATE(node) > 0 implies that arc = MATE(node) is the only solution arc heading from node

MATE(node) < 0 implies that pt = -MATE(node) is the position in LINK/MATE containing the first element of mating-list(node).

if APEX(PSEUDO(node)) ≠ node then MATE(node) is meaningless; in words, the solution edges inside a pseudonode are determined only at the time the final solution is being printed.

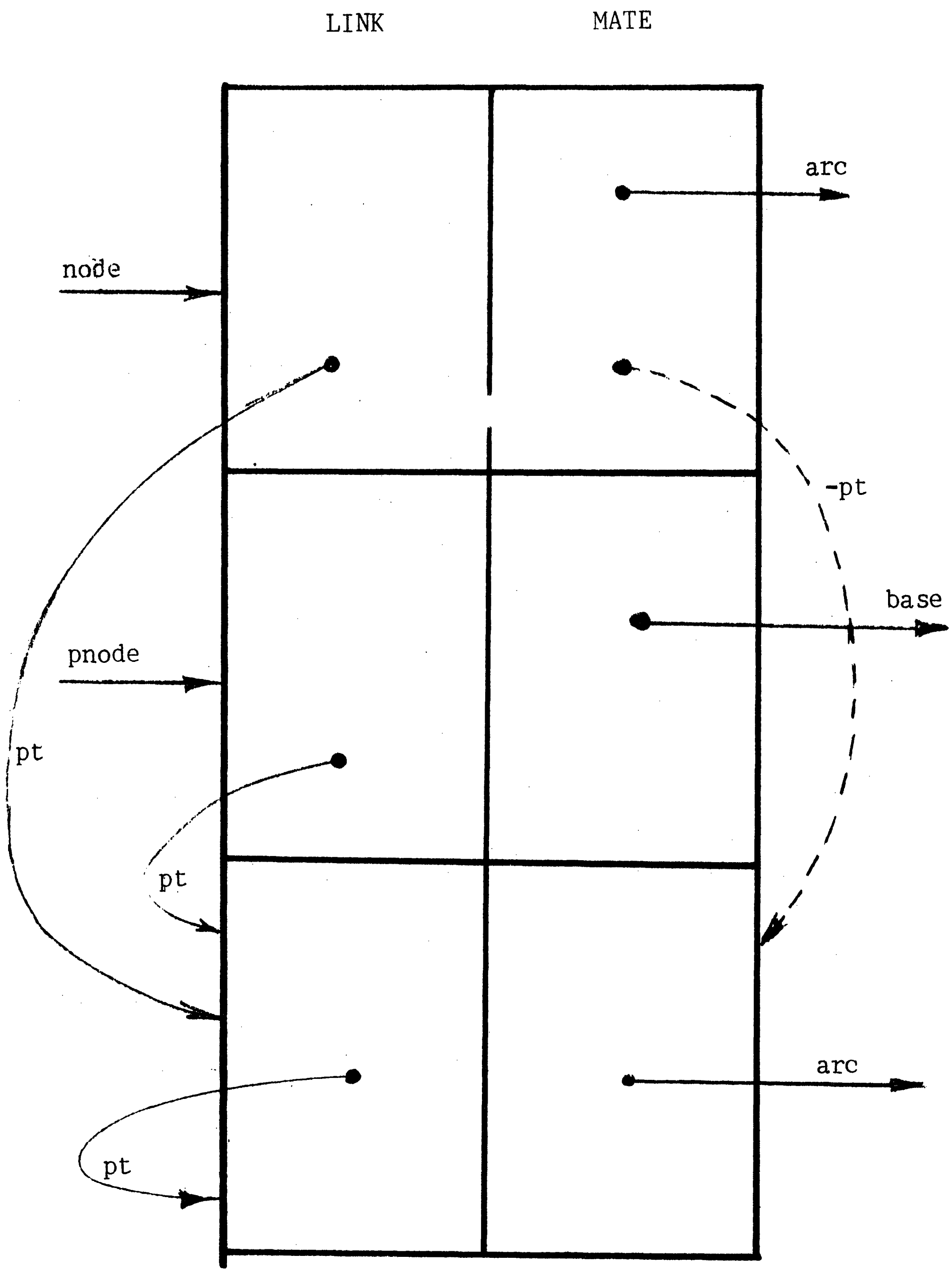


Figure IV - LINK/MATE pair of vectors

Labelling Structure (current nodes)

LABLO(node) - label of node (OUTER, INNER, UNLABD)

PREDA(node) - predecessor arc; this is the second arc on the path from node to the root; defined only for outer nodes

HEAD(node) - points to the apex of the root of the alternating tree

TREE(node) - single-linked list containing all nodes of an alternating tree. The head of such a list id the apex of the root of the alternating tree.

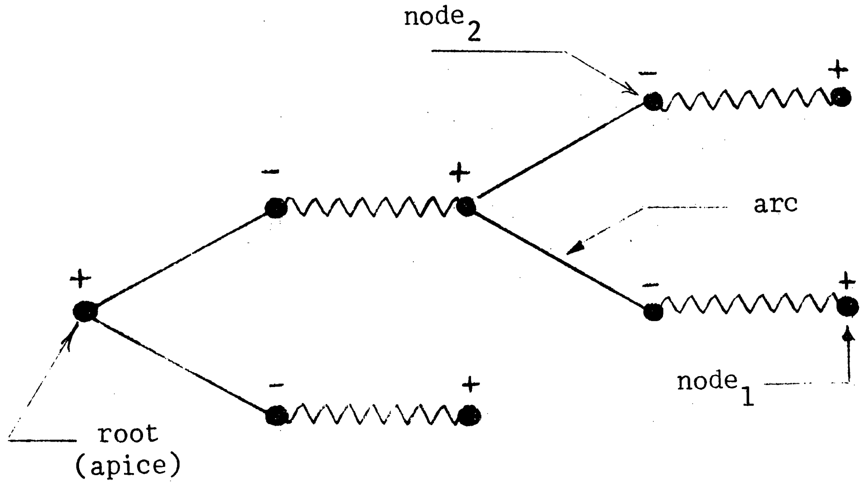
Simple Blossom Structure (noncurrent nodes)

LABLO(node) - pseudonode number of the simple blossom containing node.

HEAD(node) - node number of the next node in the cyclic list containing all nodes in a simple blossom

PREDA(node) - arc number of the arc heading from node to HEAD(node)

MATE(pnode) - base of the simple blossom associated with pnode



<u>Alternating tree</u>

LABLO($node_1$) = outer

LABLO($node_2$) = inner

LABLO(root) = outer

HEAD($node_1$) = apice

HEAD($node_2$) = apice

HEAD(root) = apice

apice = APEX(root)

PREDA($node_1$) = arc

PREDA($node_2$) = null

PREDA(root) = null

Figure V - Labelling Structure

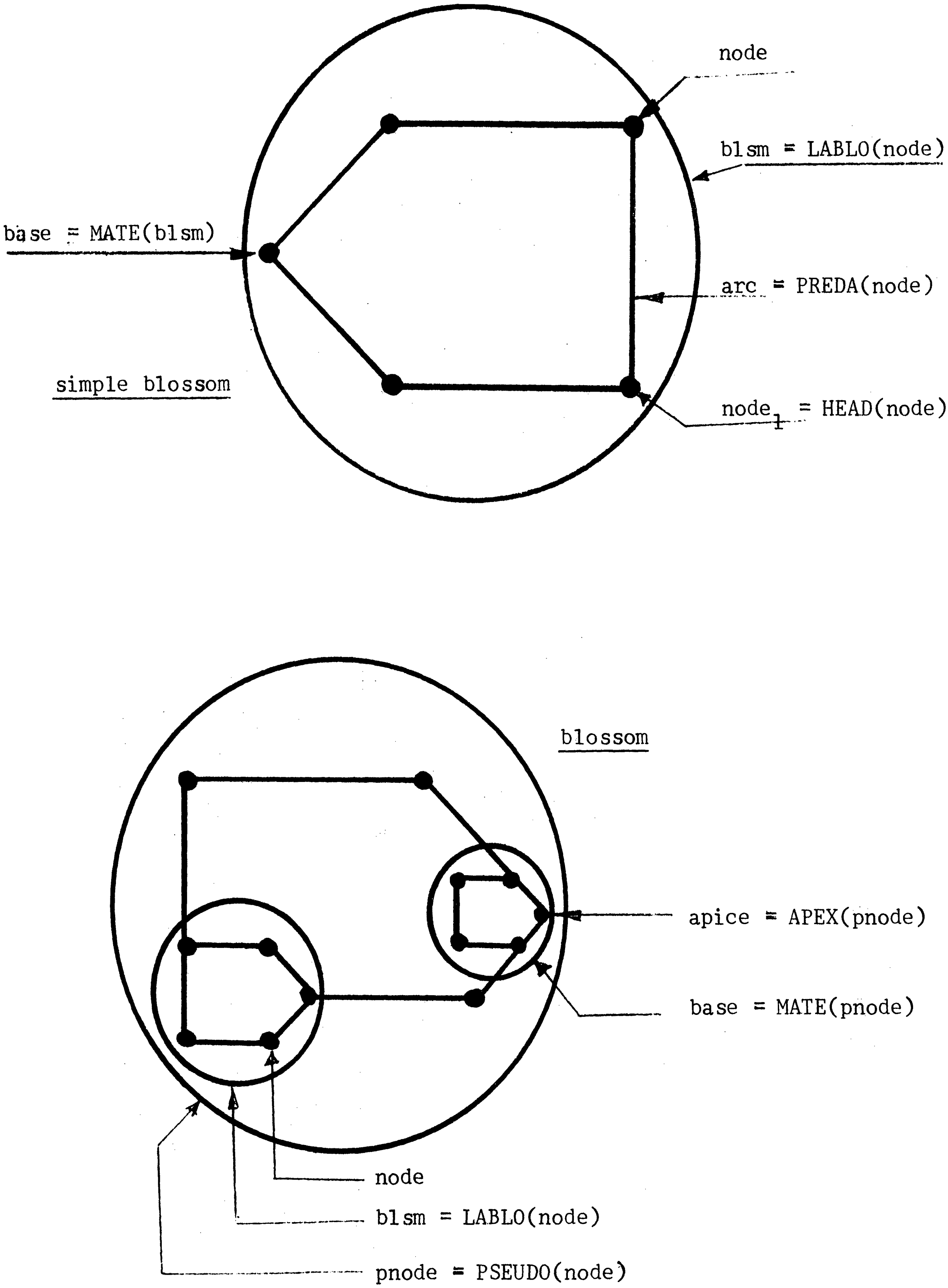


Figure VI - Blossom and Simple Blossom Structure

Blossom Structure (noncurrent nodes)

PSEUDO(node) - node in the current network which contains node;

PSEUDO(node) = node implies node is current

APEX(pnode) - points to an original node inside pnode;

for original nodes: APEX(node) = node

PRICE(node) - corresponds to the dual variables; let y(i), z(s) be the dual variables defined in [1]; let node ≡ i, pnode ≡ s

$$PRICE(i) = \begin{cases} y(i) & \text{for } i \in N^{o} \cup N^{\leqq} \\ y(i) + \Sigma_s[\ |z(s)|\ /2 : s \text{ contains } i\ ] & \text{for } i \in N^{=} \\ y(i) + \Sigma_s[\ |z(s)|\ : s \text{ contains } i\ ] & \text{for } i \in N^{\geqq} \end{cases}$$

$$PRICE(s) = |z(s)|\ /\ 2 \quad \text{for } s \in S$$

At the time the final solution is to be printed, the values of y(i) and z(s) are computed from the above expressions and printed.

BCI(node) - points to the Blossom Constraint Identifier.

BCI(node) = node node $\in N^{\leqq} \cup N^{\geqq} \cup N^{o}$

BCI(node) = null node $\in N^{=}$

BCI(pnode) = node node $\in N^{\leqq} \cup N^{\geqq}$

BCI(pnode) = null all nodes inside pnode are in $N^{=}$

MPRICE(node) - maximum value for PRICE(BCI(node))

MPRICE(node) = 0 for node $\in N^{\leqq} \cup N^{o}$

MPRICE(node) = INF for node $\in N^{=}$ or current node $\in N^{\geqq}$

MPRICE(node) = variable for noncurrent node $\in N^{\equiv}$

MPRICE(pnode) = MPRICE(BCI(node))

For noncurrent node of $N^{\geqq}$ the value of MPRICE is assigned at the time the node becomes noncurrent by setting

MPRICE(node) = 2.PRICE(node)

Special Lists

NSCAN(pt) - cyclic queue of unscanned nodes

Head: HSCAN

Tail: TSCAN

Usable size: N9 = N + 1

Nodes to be scanned are removed from the head of NSCAN; nodes just outer labelled are introduced in the tail of NSCAN; inner nodes of just erased alternating trees are introduced in head of NSCAN; pnodes to be scanned just removed from NSCAN have the nodes in their simple blossoms introduced in the head of NSCAN.

INSP(pt) stack of nodes/edges to be inspected after a dual solution change step have been performed

head: HINSP

usable size : M + N

NORIG(pt) - stack of nodes inside a blossom

Head: HORIG

Usable size: 3 N / 2

8. PROGRAM STRUCTURE

This section discusses the main characteristics of each subroutine in the FORTRAN program MATCOV.

MATCOV (main program)

```
   call GETNET
10 call GRWFOR
   call PUTSOL
   call POSTOP
   go to 10
```

Subroutine GETNET (Get network)

read the original network
initialize the variables
plant the alternating forest
special calls: MATCH
main divisions: i) initialization; ii) read node data and set node variab.; iii) read edge data and set edge variables; iv) plant alternating tree.

Subroutine GRWFOR (Grow Forest)

solve a network with an alternating tree
controls the scanning, dual solution change, and inspection processes;
starts unshrinking and augmentation processes;
checks both termination criteria: optimality and infeasibility;
special calls: SCAN, MINIM, RMATCH, NSHRINK
main divisions: i) scanning; ii) dual solution change; iii) inspection

Subroutine PUTSOL (Put Solution)

transform the final solution: internal to external representation
prints the final solution
no special calls
main divisions: i) get solution edges and transform node prices; ii) get simple blossoms and solution edges inside pseudonodes; iii) print statistics

Subroutine POSTOP (Post-Optimality)

controls the post-optimality process;
reads and performs the changes in the network
plants the alternating forest
special calls: NMATCH, OPEN, DECRSE, MATCH, INCRSE, INCLUD
main divisions: i) introduction; ii) edge cost changes; iii) node set changes; iv) edge elimination; v) node elimination; vi) node introduction; vii) edge introduction; viii) plants alternating forest

Subroutine SCAN (Scan arc)

Scan an equality arc whose tail is an outer node
starts shrinking, unshrinking, and augmentation processes;
assigns inner and outer labels;
special calls: SHRINK, RMATCH, NSHRNK
main divisions: i) initialization; ii) shrinking check; iii) double augmentation check; iv) type 2 augmentation check; v) type 0 augmentation check; vi) type 1 augmentation check; vii) node labelling; viii) type 3 augmentation check; ix) unshrinking check.

Subroutine SHRINK (Shrink a simple blossom)

Given an edge joining two outer nodes of the same alternating tree, this subroutine finds the simple blossom containing that edge and shrinks it into a pseudonode
special calls: ORIG, STPSDO, RMATCH
main divisions: i) get simple blossom; ii) shrink simple blossom and determine BCI; iii) type 3 augmentation check.

Subroutine NSHRNK (Unshrink a pseudonode)

Given an inner labelled pseudonode with null pseudonode price , this subroutine unshrinks it into its simple blossom.
special calls: STPSDO, STAPEX
main divisions: i) get alternating path in simple blossom; ii) unshrink pseudonode; iii) forest growing check;

Subroutine RMATCH (Rematch alternating path)

special calls: STAPEX
main divisions: i) remate the nodes in the alternating path ii) erase the alternating tree

Subroutine <u>MATCH</u>

Match two nodes joined by a given edge

Subroutine <u>NMATCH</u> (Unmatch)

Eliminate the mate connections of two nodes joined by a given edge.

Subroutine <u>MINIM</u> (Minimization)

Checks and updatds the inspection list

Subroutine <u>STPSDO</u> (Set Pseudo)

Update the variable PSEUDO of all nodes inside a pseudonode
special call: ORIG

Subroutine <u>STAPEX</u> (Set Apex)

Update the basis and apices of all nodes inside a pseudonode with a new apex.

Subroutine <u>ORIG</u> (Original nodes)

Get the nodes in the simple blossom of a given pseudonode. It is useful for obtaining the original nodes inside a blossom.

Subroutine <u>OPEN</u>

Unshrink all pseudonodes containing a given noncurrent node
Remove all solution edges incident with a given current node
special calls; NMATCH, ORIG, STPSDO, STAPEX
main divisions: i) open pseudonodes containing a noncurrent node; ii) remove the solution edges incident with a current node.

Subroutine <u>INCLUD</u> (Include edge)

Include a new edge in the network
special calls: OPEN, DECRSE, MATCH

Subroutine DECRSE (Decrease node price)

Attempts to decrease the node price of a given current node by a given amount

Subroutine INCRSE (Increase node price)

Increases the price of a given current node to zero.
Special calls: OPEN, DECRSE

Subroutine ERROR

Prints error messages.

Subroutine DUMP

Dumps node and edge tables

9. ADDITIONAL OUTPUTS

This section deals with program options which produces outputs with debugging purpose.

Options 5 to 9 activate the subroutine DUMP which prints tables with node, pseudonode, and edge internal information at specific points during the execution of the program, as follows:

option	subroutine	timming
5	PUTSOL	before and after each final solution transformation
6	POSTOP	before and after each post-optimality iteration
7	GETNET	after the initialization of the original network
8	GRWFOR	after each dual solution change step
9	GRWFOR	before and after each scanning step

Options corresponding to nonpositive integers causes messages indicating all subroutine calls and respective parameters to be printed during the periods of time given below. Also, dumps are provided at the biginning and at the end of these periods. The word trace is used as a referrence to such actions.

option = 0 : traces while there are no exposed nodes during the solution of the original network;

option = -r : traces while there are r exposed nodes during the solution of the original network;

option = -N2 : traces since initialization until the first augmentation;

option = -(k.N2+r) : traces while there are r exposed nodes during the k-th post-optimality iteration;

option = -(k.N2) : traces since the beginning of the (k-1)-th post-optimality iteration until an augmentation occurs.

Finally, option =10 is useful in determining the period of time a trace is desirable. Such an option prints a message stating the number of exposed nodes every time an augr ntation is performed.

A remark should be made here with respect to the dumps and traces which are produced by the options described in this section. The information is always given in terms of the internal representation and of the internal values. This means (cf. section 8):

a) edges are internally identified by (edge number + L2); in the current version of the program, L2 = 150; so, an edge number 200 is internally represented by 350 = 200 + 150;

b) node prices of original nodes inside a pseudonode are internally represented by:

$$\text{PRICE}(i) \begin{cases} = y(i) & \text{for } i \in N^{\leqq} \\ = y(i) - \Sigma_s [\ z(s) : s \text{ contains } i\ ] & \text{for } i \in N^{=} \\ = y(i) - \Sigma_s [\ 2z(s) : s \text{ contains } i\ ] & \text{for } i \in N^{\geqq} \end{cases}$$

c) solution edges inside a pseudonode are not provided.

10. MODIFYING THE PROGRAM SIZE

The design of MATCOV allows for modifications in the total number of nodes (N2 = 100), in the total number of edges (M2-L2 = 3000), in the total number of LINK/MATE's (P2 = 1500), and in the range of the edge costs ([-INF,INF]; INF = 9999.9) to be easily performed, as explained below. Such modification should be made on the file PD.FTN which contains the source code in FORTRAN for MATCOV and it is listed in the appendix.

a) Modifying the total number of nodes

Let n' denote the new maximum number of nodes that will replace the current value 100. In BLOCKDATA assign N2 = n', L2 = (3n'+1)/2. Every vector with dimension 100 should be redimensioned with n'. Every vector with dimension 101 should be redimensioned with (n'+1). Every vector with dimension 150 should be redimensioned with (3n'+1)/2. It may be necessary to rewrite some FORMAT statements. It should be noted here that the maximum number of edges will be modified also; if necessary, follow the instructions in (b).

b) Modifying the total number of edges

Let m' denote the new maximum number of edges that will replace the current value M2 - L2 (3000). In BLOCKDATA assign M2 = L2 + m'. Every vector with dimension 3150 should be redimensioned to L2 + m'. Every vector with dimension 6150 should be redimensioned to L2 + 2m'. It may be necessary to rewrite some FORMAT statements.

c) Modifying the total number of LINK/MATE positions

Let p' denote the new maximum number of LINK/MATE positions that will replace the current value P2 (1500). In BLOCKDATA assign P2 = p'. Every vector with dimension 1500 should be redimensioned to p'.

d) Modifying the value of INF or TOLER

In BLOCKDATA assign INF (TOLER) to its new value. Note that this variable is in the current version of the program a REAL*4 variable.

11. REFERENCE

C. Perin, "Matching and Edge Covering Algorithms", Ph.D. Dissertation, Department of Industrial and Operations Engineering, The University of Michigan, 1980.

APPENDIX

```
C    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C      NETWORK GENERATOR
C      THE UNIVERSITY OF MICHIGAN
C      WRITTEN BY CLOVIS PERIN
C      AUGUST 1980
C
C
C      THIS PROGRAM GENERATES A NETWORK FORMATTED IN ACCORDANCE WITH
C      THE INPUT OF MATCOV
C
C    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
       REAL MINCST, MAXCST
       INTEGER SET(100), EDGE, OPTION, HEAD1(3150), HEAD2(3150)
       INTEGER FREE(1) / '*' /
       LOGICAL SAVE(3150)/3150*.FALSE./
C
C      READ N (#NODES), M(#EDGES), NSETS(#SETS), MINCST, MAXCST
C      (EDGE COSTS RANGE FROM MINCST TO MAXCST), IX (SEED)
       WRITE(6,1)
       READ (5,2)
       WRITE(7,2)
       WRITE(6,3)
       READ (5,FREE)  N, M, NSETS, MINCST, MAXCST, IX
       IF (N.LT.0.OR.N.GT.100)        GO TO 999
       IF (M.LT.0.OR.M.GT.3000)       GO TO 999
       IF (NSETS.LT.1.OR.NSET.GT.4)   GO TO 999
       IF (MINCST.GT.MAXCST)          GO TO 999
       WRITE(7,4)      N, M
C
C      GENERATE NODES -- 1 TO N
       DO 10 NODE = 1, N
            SAVE(NODE) = .TRUE.
            SET(NODE)  = RAND(IX) * NSETS  + 1
            IF (SET(NODE).GT.3) SET(NODE) = 0
   10       WRITE(7,5) NODE, SET(NODE)
       DELTA = MAXCST - MINCST
       M1    = 150 + M
C
C      GENERATE EDGES -- 151 TO 151+M
       DO 30 EDGE = 151, M1
            SAVE(EDGE)   = .TRUE.
            NODE1        = 1 + RAND(IX)  * N
   20       NODE2        = 1 + RAND(IX)  * N
            IF (NODE1.EQ.NODE2) GO TO 20
            HEAD1(EDGE) = NODE1
            HEAD2(EDGE) = NODE2
            CST         = RAND(IX)  * DELTA + MINCST
   30       WRITE(7,5) EDGE, NODE1, NODE2, CST
C
C      GENERATE INPUT FOR POST-OPTIMALITY ANALYSIS
       N1 = N
       M1 = M
C
C      GET PARAMETERS K1 TO K6
  100  WRITE(6,6)
       READ(5,FREE)    K1, K2, K3, K4, K5, K6
       IF (K1.LT.0.OR.K2.LT.0.OR.K3.LT.0.OR.K4.LT.0.OR.K5.LT.0.OR.K6.
      &                LT.0) GO TO 999
```

```
      WRITE(7,7)      K1, K2, K3, K4, K5, K6
      IF             ( K1+ K2+ K3+ K4+ K5+ K6 .EQ. 0 ) STOP
C
C     GENERATE K1 CHANGES OF EDGE COSTS
      IF (K1.EQ.0) GO TO 200
      DO 120 K = 1, K1
  110      EDGE = 151 + RAND(IX) * M
           IF (.NOT.SAVE(EDGE))          GO TO 110
           IF (.NOT.SAVE(HEAD1(EDGE))) GO TO 110
           IF (.NOT.SAVE(HEAD2(EDGE))) GO TO 110
           CST  = RAND(IX) * DELTA + MINCST
  120      WRITE(7,8) EDGE, CST
C
C     GENERATE K2 CHANGES OF NODE SUBSETS
  200 IF (K2.EQ.0) GO TO 300
      DO 220 K = 1, K2
  210      NODE       = 1 + RAND(IX) * N
           IF (.NOT.SAVE(NODE)) GO TO 210
           SET(NODE) = RAND(IX) * NSETS + 1
           IF (SET(NODE).GT.3) SET(NODE) = 0
  220      WRITE(7,5) NODE, SET(NODE)
C
C     GENRATE K3 EDGE DELETIONS
  300 IF (K3.EQ.0)  GO TO 400
      IF (K3.GT.M1) GO TO 999
      DO 320 K = 1, K3
  310      EDGE        = 151 + RAND(IX)* M
           IF (.NOT.SAVE(EDGE))          GO TO 310
           IF (.NOT.SAVE(HEAD1(EDGE))) GO TO 310
           IF (.NOT.SAVE(HEAD2(EDGE))) GO TO 310
           SAVE(EDGE) = .FALSE.
  320      WRITE(7,5) EDGE
      M1 = M1 - K3
C
C     GENERATE K4 NODE DELETIONS
  400 IF (K4.EQ.0)  GO TO 500
      IF (K4.GT.N1) GO TO 999
      DO 420 K = 1, K4
  410      NODE        = 1 + RAND(IX) * N
           IF (.NOT.SAVE(NODE)) GO TO 410
           SAVE(NODE) =.FALSE.
  420      WRITE(7,5) NODE
      N1 = N1 - K4
C
C     GENERATE K5 NEW NODE INTRODUCTIONS
  500 IF (K5.EQ.0)      GO TO 600
      IF (N+K5.GT.100) GO TO 999
      DO 510 K = 1, K5
           N           = N + 1
           NODE        = N
           SAVE(NODE) = .TRUE.
           SET(NODE)  = RAND(IX) * NSETS + 1
           IF (SET(NODE).GT.3) SET(NODE) = 0
  510      WRITE(7,5) NODE, SET(NODE)
C
C     GENERATE K6 NEW EDGE INTRODUCTIONS
  600 IF (K6.EQ.0)        GO TO 100
      IF (M+K6.GT.3000) GO TO 999
      DO 630 K = 1, K6
           M             = M + 1
```

```
            EDGE          = M + 151
            SAVE(EDGE)    = .TRUE.
  610       NODE1         = 1 + RAND(IX) * N
            IF (.NOT.SAVE(NODE1)) GO TO 610
  620       NODE2         = 1 + RAND(IX) * N
            IF (.NOT.SAVE(NODE2)) GO TO 620
            IF (NODE1.EQ.NODE2)   GO TO 620
            HEAD1(EDGE) = NODE1
            HEAD2(EDGE) = NODE2
            CST         = RAND(IX) * DELTA + MINCST
  630       WRITE(7,5) EDGE, NODE1, NODE2, CST
C
C     RETURN FOR A NEW POST-OPTIMALITY ITERATION
      GO TO 100
C
C     ERROR
  999 WRITE(6,99)
      STOP
C
C     FORMATS
    1 FORMAT(' ENTER TITLE')
    2 FORMAT(48H123456789012345678901234567890123456789012345678    )
    3 FORMAT(' ENTER N, M, #SETS, MINCST, MAXCST, IX')
    4 FORMAT(2I5,' 1')
    5 FORMAT(3I5,F10.4)
    6 FORMAT(' ENTER K1, K2, K3, K4, K5, K6')
    7 FORMAT(6I5)
    8 FORMAT(I5,F10.4)
   99 FORMAT(' *****   ERROR   *****')
      END
C
C     GENERATE RANDOM NUMBERS
      FUNCTION RAND(IX)
      IX = IX * 65539
      IF (IX.LT.0) IX = IX + 2147483647 + 1
      RAND = IX * .4656613E-9
      RETURN
      END
```

```
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),BACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C
C
C     ------------------------------------------------------------------
C
C     THIS PROGRAM COMPUTES A MINIMUM COST 1-MATCHING/COVERING FOR A
C     NETWORK WITH
C
C                   N   NODES   AND   M   EDGES
C
C     WHERE   EVERY   EDGE   HAS   AN   ASSOCIATED     EDGE COST     AND
C     N<, N=, N>, NO        IS A GIVEN PARTITION OF THE SET OF NODES:
C
C          N< - SUBSET OF MATCHED NODES              (CODE = 2)
C          N= - SUBSET OF PERFECT MATCHED NODES      (CODE = 1)
C          N> - SUBSET OF COVERED NODES              (CODE = 3)
C          NO - SUBSET OF UNRESTRICTED NODES         (CODE = 4)
C
C     THIS  PROGRAM  ALLOWS  FOR  POST-OPTIMALITY  ANALYSIS.  IT  IS
C     POSSIBLE  TO HAVE A  SEQUENCE OF  POST-OPTIMALITY  ITERATIONS.
C     EACH ITERATION IS SUBDIVIDED INTO THE FOLLOWING PHASES:
C
C          A) EDGE COST CHANGE,
C          B) NODE SUBSET CHANGE,
C          C) EDGE ELIMINATION,
C          D) NODE ELIMINATION,
C          E) NEW NODE INTRODUCTION,
C          F) NEW EDGE INTRODUCTION.
C
C     LET K1, K2, K3, K4, K5, K6 DESIGNATE THE NUMBER OF NODE/EDGES
C     TO BE WORKED OUT AT EACH OF THE ABOVE PHASES.
C
C
C     PROGRAM INPUT (FORTRAN FREE-FORMAT):
```

```
C
C      1) TITLE                                        (MAX 48 CHARACTERS)
C      2) N, M, OPTION                   (#NODES,#EDGES,PROGRAM OPTION)
C
C      3) NODE,SUBSET                    <N TIMES>           (NODE DATA)
C      4) EDGE,NODE1,NODE2,COST          <M TIMES>           (EDGE DATA)
C
C      5) K1, K2, K3, K4, K5, K6
C      6) EDGE, NEWCOST                  <K1 TIMES>   (EDGE COST CHANGE)
C      7) NODE, NEWSUBSET                <K2 TIMES> (NODE SUBSET CHANGE)
C      8) EDGE                           <K3 TIMES>   (EDGE ELIMINATION)
C      9) NODE                           <K4 TIMES>   (NODE ELIMINATION)
C     10) NEWNODE, SUBSET                <K5 TIMES> (NODE INTRODUCTION)
C     11) NEWEDGE,NODE1,NODE2,COST       <K6 TIMES> (EDGE INTRODUCTION)
C
C     12) 0  0  0  0  0  0                               (TERMINATION)
C
C
C    OBS.: 3) TO 4) DEFINE THE NETWORK;
C
C          5) TO 11) DEFINE A POST-OPTIMALITY ITERATION,
C                    THEY ARE REPEATED FOR EACH ITERATION;
C
C    PROGRAM EXECUTION:
C              $RUN MATCOV SCARDS=XXX SPRINT=YYY T=ZZZ
C
C
C    EXAMPLE:
C
C              $R MATCOV T=1
C          1)  EXAMPLE
C          2)    4     5     3
C
C          3)   10     2
C               20     1
C               40     0
C               30     1
C          4)  240    20    40   -2.0
C              220    30    40   -4.0
C              200    20    10   20.0
C              210    20    30  -50.0
C              230    10    30    2.57
C
C          5)    1     0     1     1     1     2
C          6)  230    3.0
C          7)
C          8)  220
C          9)   40
C         10)   60     3
C         11)  290    10    60  -10.0
C              190    60    30  +10.0
C
C         12)    0     0     0     0     0     0
C
C
C    EXAMPLE DESCRIPTION:
C
C    1)   TITLE= "EXAMPLE"
C    2)   N= 4 NODES, M= 5 EDGES, OPTION= 3,
C
```

```
C      3)    NODES:
C                 NODE 10 IS IN N<
C                 NODE 20 IS IN N=
C                 NODE 30 IS IN N=
C                 NODE 40 IS IN NO
C      4)    EDGES:
C                 EDGE 200 = (10;20) WITH COST  20.0
C                 EDGE 210 = (20;30) WITH COST -50.0
C                 EDGE 220 = (30;40) WITH COST  -4.0
C                 EDGE 230 = (10;30) WITH COST   2.57
C                 EDGE 240 = (20;40) WITH COST  -2.0
C
C      5)    K1= 1 EDGE COST CHANGES,
C            K2= 0 SUBSET CHANGES
C            K3= 1 EDGE ELIMINATION
C            K4= 1 NODE ELIMINATION
C            K5= 1 NODE INTRODUCTION
C            K6= 2 EDGE INTRODUCTION
C      6)    CHANGE COST OF EDGE 230 TO 3.0
C      7)    DO NOT CHANGE ANY NODE SUBSET
C      8)    ELIMINATE EDGE 220
C      9)    ELIMINATE NODE  40
C     10)    INTRODUCE NODE  60 IN N>
C     11)    INTRODUCE EDGES 290 = (10;60) WITH COST -10.0
C                            190 = (30;60) WITH COST -10.0
C
C     12)    TERMINATE
C
C     AFTER 5) TO 11) THE NEW NETWORK IS:
C         NODES: NODE 10 IN N<
C                NODE 20 IN N=
C                NODE 30 IN N=
C                NODE 60 IN N>
C
C
C         EDGES: EDGE 190 = (30;60) WITH COST  10.0
C                EDGE 200 = (10;20) WITH COST  20.0
C                EDGE 210 = (20;30) WITH COST -50.0
C                EDGE 230 = (10;30) WITH COST   2.57
C                EDGE 290 = (10;60) WITH COST -10.0
C
C         OBS.: EDGE 240 = (20;40) WAS AUTOMATICALLY ELIMINATED FROM
C               THE NETWORK AT THE TIME NODE 40 WAS ELIMINATED
C               BECAUSE IT BECAME INCIDENT WITH A NONEXISTING  NODE.
C
C
C      PROGRAM OPTIONS:
C
C      OPTION = 1 - PRINTS STATISTICS ONLY
C
C      OPTION = 2 - PRINTS INPUT AND SOLUTION (SHORT OUTPUT)
C
C      OPTION = 3 - PRINTS INPUT, HISTORY, AND SOLUTION (LONG OUTPUT)
C
C      OPTION > 3 - DUMPS NODE INFORMATION AT SPECIFIC POINTS
C                                                  (DEBUG PURPOSES)
C      OPTION =-R - TRACES WHILE THERE ARE R ROOTS   (DEBUG PURPOSES)
C
C      ------------------------------------------------------------
C
```

```
      CALL GETNET
   10 CALL GRWFOR
      CALL PUTSOL
      CALL POSTOP
      GO TO 10
C
      END
      BLOCKDATA
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /BLOCK DATA/
C
C     DATA STRUCTURE DESCRIPTION
C
C     N = #NODES        M = #EDGES       L = MAX #PSEUDOS
C
C     LIST(IND) = NODE/EDGE,   NODE/EDGE EXISTS
C     LIST(IND) = NULL         NODE/EDGE DOES NOT EXIST
C
C     LIST( INDEX(NODE/EDGE) ) = NODE/EDGE   IFF NODE/EDGE EXISTS
C
C     EXISTING NODES:    FROM  LIST(N0)  TO  LIST(N1)
C     EXISTING PSEUDOS:  FROM  LIST(L0)  TO  LIST(L1)
C     EXISTING EDGES:    FROM  LIST(M0)  TO  LIST(M1)
C
C     MAX NODE   = N2 >= N1
C     MAX PSEUDO = L2 >= L1
C     MAX EDGE   = M2 >= M1
C
C     VALUE = DUAL OBJECTIVE VALUE
C     ROOT$ = # EXPOSED NODES
C
C     SET(NODE) = 0, 1, 2, 3     (N0, N=, N<, N>, RESPECTIVELY)
C     LABLO(NODE) = OUTER, INNER, UNLABD     FOR CURRENT NODES
C     LABLO(NODE) = SIMPLE BLOSSOM           FOR NONCURRENT NODES
C
C     PSEUDO(NODE) - CURRENT NODE CONTAINING NODE (OR NODE)
C
C     APEX(NODE) - ORIGINAL NODE INSIDE NODE (OR NODE)
C
C     PRED(NODE) - PREDECESSOR ARC IN ALTERNATING PATH TO ROOT
C
C     TREE - LISTS OF NODES IN A TREE (HEADS = EXPOSED NODES)
C
C     BCI(NODE) - BLOSSOM CONSTRAINT IDENTIFIER (OR NODE)
C
C     MPRICE(NODE) - MAXIMUM PRICE OF BCI(NODE)
C
C     HEAD(NODE) = ROOT OF TREE
C     HEAD(ARC) = NODE POINTED BY THE ARC
C
C     PRICE(NODE) = NODE DUAL VARIABLE
C     PRICE(EDGE) = EDGE COST
C
C     NSCAN - CIRCULAR LIST OF UNSCANNED NODES
```

```
C
C      INSP - LIST OF NODE/EDGES TO BE INSPECTED
C
C      NORIG - LIST OF NODES IN A SIMPLE BLOSSOM (SEE ORIG)
C
C      LINK/MATE - LISTS OF EQUALITY/MATE EDGES
C           HEADS FOR CURRENT EQUALITY EDGES = LINK(NODE)
C           HEADS FOR NONCURRENT EQUALITY EDGES = LINK(PNODE)
C           HEADS FOR MATE EDGES = -MATE(NODE)
C                LINK(PT) = NEXT ELEMENT IN THE LIST
C                MATE(PT) = EDGE
C                PT  <=  HINK  <=  P2  =  MAX ELEMENT
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C      NODE VECTORS
      DATA SET/100*0/,LABLO/150*0/,PSEUDO/150*0/,APEX/150*0/,PRED/15
     &               0*0/,TREE/150*0/,MPRICE/150*0./,BCI/150*0/
C
C      NODE/EDGE VECTORS
      DATA HEAD/6150*0/,PRICE/3150*0.0/
C
C      LIST OF EXISTING NODE/EDGES
      DATA LIST/3150*0/,INDEX/3150*0/
C
C      LINK/MATE VECTORS
      DATA LINK/1000*0/,MATE/1000*0/
C
C      HEAD/TAILS POINTERS
      DATA HSCAN,TSCAN,HINSP,HORIG,HINK/5*0/
C
C      CONTROL VARIABLES
      DATA VALUE,DELTA,MCOST,TRACE,PRINT/3*0.0,2*.FALSE./
C
C      STATISTICAL VARIALBLES
      DATA ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$,LABL$,LIN
     &               K$/10*0/
C
C      CONSTANTS
      DATA INF,TOLER,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET3,NULL/999
     &               9.9,1.E-50,1.,-1,0,0,1,2,3,0/
C
C      PROBLEM PARAMETERS
```

```
DATA N2, L2, M2, P2 / 100, 150, 3150, 1500 /
END
```

```
      SUBROUTINE GETNET
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /GET NETWORK/
C     READ NETWORK DATA
C     INITIALIZE VARIABLES
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,NO,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
      REAL MCOST,COST,TITLE(12)
      LOGICAL FREE(1) /'*'/
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C     READ PROBLEM PARAMETERS
      CALL TIME(0)
      READ(5,1,END=999) TITLE
    1 FORMAT(12A4)
      READ(5,FREE,END=999) N, M, OPTION
      TRACE = .FALSE.
      PRINT = .FALSE.
      IF (OPTION.EQ.-N2) TRACE = .TRUE.
      IF (0.GT.OPTION.AND.OPTION.GE.-N2) PRINT = .TRUE.
      IF (OPTION.EQ.3.OR.OPTION.EQ.4)    PRINT = .TRUE.
      WRITE(6,2)
    2 FORMAT(1H1,30X,'*****   MINIMUM COST 1-MATCHING/COVERING PROBL
     &               EM   *****'//)
      WRITE(6,3) TITLE
    3 FORMAT(40X,12A4)
      IF ( N.LE.0 .OR. M.LE.0 ) CALL ERROR(2)
      IF ( N.GT.N2 )            CALL ERROR(3)
      IF ( M.GT.M2-L2 )         CALL ERROR(4)
C
C     INITIALIZE PARAMETRIC VARIABLES
      N0 = 1
      N1 = N
      N9 = N  + 1
      L  = N  / 2
      L0 = N2 + 1
      L1 = N2
      M0 = L2 + 1
```

```
      M1 = L2 + M
      M9 = 2  * M2 + 1
      P0 = L2 + 1
C
C     INITIALIZE LINK/MATE STORAGE SPACE
      DO 10 PT = P0, P2
   10     LINK(PT)  = PT + 1
      LINK(P2) = NULL
      HINK     = M0
C
C     READ NODE DATA
C     INITIALIZE NODE VECTORS
      DO 20 IND = N0, N1
          READ(5,FREE,END=999)  NODE,SET(NODE)
          IF (TRACE.OR.OPTION.EQ.4) WRITE(6,4) NODE,SET(NODE)
    4     FORMAT(' NODE DATA :               ',2I5)
          IF (NODE.LT.N0.OR.NODE.GT.N2.OR.INDEX(NODE).NE.NULL) CALL
     &            ERROR(6)
          IF (SET(NODE).LT.0.OR.SET(NODE).GT.3) CALL ERROR(7)
          INDEX(NODE)  = IND
          LIST(IND)    = NODE
          PSEUDO(NODE) = NODE
          APEX(NODE)   = NODE
          LABLO(NODE)  = UNLABD
          ST           = SET(NODE)
          IF (ST.EQ.SET1.OR.ST.EQ.SET2) PRICE(NODE) = -INF/2
          VALUE        = VALUE + PRICE(NODE)
          BCI(NODE)    = NODE
          IF (ST.EQ.SET1) BCI(NODE) = L2
          IF (ST.EQ.SET1) MPRICE(NODE) = INF
   20     CONTINUE
      MPRICE(L2) = INF
      H      = 0
C
C     READ EDGE DATA
C     INITIALIZE EDGE VECTORS
C     LINK EQUALITY EDGES
C     MATCH EDGES IN A-
      DO 40 IND = M0, M1
          READ(5,FREE,END=999)  EDGE, NODE1, NODE2, COST
          IF (TRACE.OR.OPTION.EQ.4) WRITE(6,5) EDGE, NODE1, NODE2,
     &           COST
    5     FORMAT(' EDGE DATA :              ',3I5,F7.1)
          EDGE1 = EDGE + L2
          IF (EDGE1.LT.M0.OR.EDGE1.GT.M2.OR.INDEX(EDGE1).NE.NULL) C
     &           ALL ERROR(7)
          IF (NODE1.LT.N0.OR.NODE1.GT.N2.OR.INDEX(NODE1).EQ.NULL) C
     &           ALL ERROR(6)
          IF (NODE2.LT.N0.OR.NODE2.GT.N2.OR.INDEX(NODE2).EQ.NULL) C
     &           ALL ERROR(6)
          IF (COST.GT.INF.OR.COST.LT.-INF)                          C
     &           ALL ERROR(9)
          EDGE2          = M9 - EDGE1
          HEAD(EDGE1)    = NODE2
          HEAD(EDGE2)    = NODE1
          PRICE(EDGE1)   = COST
          INDEX(EDGE1)   = IND
          LIST(IND)      = EDGE1
          COST           = PRICE(EDGE1) - PRICE(NODE1) - PRICE(NODE2)
          IF (COST.GT.+TOLER) GO TO 40
```

```
            IF (COST.LT.-TOLER) GO TO 30
C                "EDGE1" IS AN EQUALITY EDGE
                 IF (HINK.EQ.NULL) CALL ERROR(5)
                 PT           = HINK
                 HINK         = LINK(HINK)
                 MATE(PT)     = EDGE1
                 LINK(PT)     = LINK(NODE1)
                 LINK(NODE1)  = PT
                 IF (HINK.EQ.NULL) CALL ERROR(5)
                 IF (HINK.GT.LINK$) LINK$ = HINK
                 PT           = HINK
                 HINK         = LINK(HINK)
                 MATE(PT)     = EDGE2
                 LINK(PT)     = LINK(NODE2)
                 LINK(NODE2)  = PT
                 GO TO 40
C           "EDGE1" IS AN EGE OF A-
   30       CALL MATCH(EDGE1)
            VALUE       = VALUE + PRICE(EDGE1)
            H           = H + 1
            STACK(H)    = EDGE1
   40       CONTINUE
      IF (PRINT.AND.H.GT.1) WRITE(6,6) (STACK(I),HEAD(STACK(I)),HEAD(
     &              M9-STACK(I)),PRICE(STACK(I)),I=1,H)
    6 FORMAT(//' EDGES OF A- (EDGE,NODE1,NODE2,COST) :'/5(I12,2I3,F6
     &              .1))
C
C     INITIALIZE PSEUDONODE LIST
      DO 50 IND = L0, L2
   50       LIST(IND) = IND
C
C     PLANT ALTERNATING FOREST
      DO 60 IND = N0, N1
            NODE         = LIST(IND)
            IF (SET(NODE).EQ.SET0.OR.MATE(NODE).NE.NULL) GO TO 60
            LABLO(NODE)  = OUTER
            HEAD(NODE)   = NODE
            HSCAN        = HSCAN + 1
            NSCAN(HSCAN) = NODE
            ROOT$        = ROOT$ + 1
   60       CONTINUE
      LABL$ = ROOT$
      IF (PRINT) WRITE(6,7) ROOT$
    7 FORMAT(//' ALTERNATING FOREST PLANTED WITH',I4,' ROOTS')
C
      IF (OPTION.EQ.7) CALL DUMP
      CALL TIME(1,0,TIME1)
      IF (OPTION.NE.-ROOT$) RETURN
      TRACE = .TRUE.
      CALL DUMP
      RETURN
C
C     UNEXPECTED END-OF-FILE
  999 CALL ERROR(1)
      STOP
      END
```

```
      SUBROUTINE MINIM(NODE,NDELTA)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /COMPUTE MINIMUM DELTA/
C     COMPARE DELTA AND NEWDELTA AND CONTROL INSPECTION LIST
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
      REAL NDELTA
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C     IF (TRACE) WRITE(6,1) NODE, NDELTA
C   1 FORMAT('  #TRACE# MINIM  (',I4,',',F6.1,')')
C
C     NEWDELTA > DELTA -- RETURN
C     NEWDELTA = DELTA -- SAVE NODE/EDGE FOR INSPECTION
C     NEWDELTA < DELTA -- ERASE LIST, SAVE NODE/EDGE FOR INSPECTION
      IF (NDELTA.GT.DELTA+TOLER) RETURN
      IF (NDELTA.GE.DELTA-TOLER) GO TO 10
          HINSP = 0
          DELTA = NDELTA
   10 HINSP       = HINSP + 1
      INSP(HINSP) = NODE
      RETURN
      END
```

```
      SUBROUTINE MATCH(EDGE1)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /MATCH NODES JOINED BY "EDGE1"/
C     FOR BOTH NODES DO:
C          IF "MATE(NODE) = NULL" -- "MATE(NODE) = EDGE1"
C          IF "NODE(NODE) > NULL" -- "-MATE(NODE)" BECOMES A HEAD
C          IF "NODE(NODE) < NULL" -- INTRODUCE "EDGE1" IN LIST
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) EDGE1
    1 FORMAT('  #TRACE# MATCH  (',I4,')')
C
C     DO 10 TO 40 FOR BOTH NODES JOINED BY "EDGE1"
      EDGE = EDGE1
   10 NODE = HEAD(M9-EDGE)
      IF (MATE(NODE).NE.NULL) GO TO 20
C          NO MATES
           MATE(NODE) = EDGE
           GO TO 40
   20 IF (MATE(NODE).LT.NULL) GO TO 30
C          ONE MATE -- "-MATE(NODE)" BECOMES A LIST HEAD
           IF (HINK.EQ.NULL) CALL ERROR(5)
           PT         = HINK
           HINK       = LINK(HINK)
           MATE(PT)   = MATE(NODE)
           LINK(PT)   = NULL
           MATE(NODE) = -PT
C     INTRODUCE "EDGE" IN THE MATE LIST WITH HEAD "-MATE(NODE)"
   30 IF (HINK.EQ.NULL) CALL ERROR(5)
      IF (HINK.GT.LINK$) LINK$ = HINK
      PT         = HINK
      HINK       = LINK(HINK)
      MATE(PT)   = EDGE
      LINK(PT)   = -MATE(NODE)
      MATE(NODE) = -PT
C
```

```
C     GET SECOND NODE OR RETURN
   40 IF (EDGE.NE.EDGE1) RETURN
      EDGE = M9 -EDGE1
      GO TO 10
C
      END
```

```
      SUBROUTINE NMATCH(EDGE1)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /UNMATCH NODES JOINED BY "EDGE1"/
C     FOR BOTH NODES DO:
C          IF EDGE IS NOT A MATE, RETURN
C          IF EDGE IS THE ONLY MATE, MAKE "MATE(NODE)" NULL
C          IF EDGE IS ONE OF THE MATES, REMOVE IT FROM THE LIST WITH
C               HEAD "-MATE(NODE)",  AND IF SUCH A LIST CONTAINS NOW
C               JUST ONE EDGE, TRANSFORM "MATE(NODE)" INTO A POINTER
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) EDGE1
    1 FORMAT('  #TRACE# NMATCH (',I4,')')
      IF (EDGE1.EQ.NULL) RETURN
C
C     DO 10 TO 50 FOR BOTH NODES JOINED BY "EDGE1"
      EDGE = EDGE1
   10 NODE = HEAD(M9-EDGE)
      PT   = MATE(NODE)
      IF (PT.LT.NULL) GO TO 20
C     "MATE(NODE)"IS A POINTER
C     "MATE(NODE)" POINTS TO EDGE -- POINT IT TO NULL
      IF (PT.EQ.EDGE) MATE(NODE) = NULL
      GO TO 50
C
C     "-MATE(NODE)" IS A LIST HEAD
   20 PT = -PT
      IF (MATE(PT).NE.EDGE) GO TO 30
C          "EDGE" IS THE FIRST ELEMENT LIST
           MATE(NODE) = -LINK(PT)
           GO TO 40
C     "EDGE" IS NOT THE FIRST ELEMENT LIST
   30 PT1  = PT
      PT   = LINK(PT1)
C     "EDGE" IS NOT IN THE LIST -- DONE
      IF (PT.EQ.NULL) GO TO 50
```

```
      IF (MATE(PT).NE.EDGE) GO TO 30
C          "EDGE" IS IN THE LIST
           LINK(PT1) = LINK(PT)
   40      MATE(PT)  = NULL
           LINK(PT)  = HINK
           HINK      = PT
C
C     GET SECOND NODE OR RETURN
   50 IF (EDGE.NE.EDGE1) RETURN
      EDGE = M9 - EDGE
      GO TO 10
C
      END
```

```
      SUBROUTINE RMATCH(NODE)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /REMATCH ALTERNATING PATH/
C     "PSEUDO(NODE)" IS AN OUTER NODE
C     REMATCH ITS ALTERNATING PATH
C     ERASE ITS ALTERNATING TREE
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &                150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) NODE
    1 FORMAT('  #TRACE# RMATCH (',I4,')')
      AUGM$     = AUGM$ + 1
      RMAT$     = RMAT$ + 1
C
C     UPDATE "NODE" VECTORS
      PNODE     = PSEUDO(NODE)
      IF (NODE.NE.PNODE) CALL STAPEX(NODE)
      K         = 1
      STACK(K)  = PNODE
C
C     DO 10 FOR ALL OUTER NODES IN THE PATH
      PNODE1    = PNODE
   10 EDGE1     = PRED(PNODE1)
C     "PRED(NODE1) = NULL" -- "PNODE1" IS A ROOT
      IF (EDGE1.EQ.NULL) GO TO 100
C         "PNODE2" IS INNER LABELLED
C         "PNODE2" IS OUTER LABELLED
C         UPDATE "NODE1" AND "NODE2" VECTORS
          RMAT$         = RMAT$ + 2
          EDGE2         = M9 - EDGE1
          NODE1         = HEAD(EDGE1)
          NODE2         = HEAD(EDGE2)
          PNODE1        = PSEUDO(NODE1)
          PNODE2        = PSEUDO(NODE2)
          K             = K + 4
          STACK(K-3)    = PNODE1
          STACK(K-2)    = MATE(NODE1)
```

```
            STACK(K-1)   = PNODE2
            STACK(K)     = EDGE1
            MATE(NODE1) = EDGE2
            MATE(NODE2) = EDGE1
            IF (NODE1.NE.PNODE1) CALL STAPEX(NODE1)
            IF (NODE2.NE.PNODE2) CALL STAPEX(NODE2)
            GO TO 10
C
C     ERASE THE ALTERNATING TREE
C     DO 110 FOR EVERY NODE IN THE TREE
  100 ROOT$ = ROOT$ - 1
      J     = 0
      NODE2 = HEAD(PNODE)
  110 NODE1 = NODE2
      IF ( NODE1 .EQ. NULL ) GO TO 120
C           SAVE "NODE2"
C           UPDATE "NODE1" VECTORS
            NODE2         = TREE(NODE1)
            PNODE1        = PSEUDO(NODE1)
            TREE(NODE1)   = NULL
            J             = J + 1
            PACK(J)       = NODE1
            IF ( PNODE1.NE.NODE1 ) GO TO 110
C           "NODE1" IS A CURRENT NODE
            LABEL         = LABLC(NODE1)
            LABLO(NODE1)  = UNLABD
            HEAD(NODE1)   = NULL
            PRED(NODE1)   = NULL
            IF ( LABEL.NE.INNER ) GO TO 110
C           "NODE1" IS INNER LABELLED
C           STACK IT FOR SCANNING LATER
            NSCAN(TSCAN) = - NODE1
            TSCAN         = TSCAN - 1
            IF (TSCAN.LT.1) TSCAN = N9
            GO TO 110
C
C     PRINT PATH AND TREE
  120 IF (PRINT) WRITE(6,2) (STACK(I),I=1,K)
    2 FORMAT(' REMATCH ALTERN. PATH :',18I5/(24X,18I5))
      IF (PRINT) WRITE(6,3) (PACK(I),I=1,J)
    3 FORMAT(' ERASE ALTERNAT. TREE :',18I5/(24X,18I5))
      IF (OPTION.EQ.10) WRITE(6,4) ROOT$
    4 FORMAT('  #PRE-TRACE# ALTERNATING FOREST WITH',I4,' ROOTS')
      IF (TRACE) GO TO 130
      IF (OPTION.NE.-ROOT$) RETURN
      TRACE  = .TRUE.
      PRINT  = .TRUE.
      RETURN
  130 TRACE  = .FALSE.
      PRINT  = .TRUE.
      OPTION = 3
      RETURN
      END
```

```
      SUBROUTINE SHRINK(EDGE)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /SHRINK SIMPLE BLOSSOM/
C     "EDGE" JOINS TWO OUTER NODES IN THE SIMPLE BLOSSOM
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
      REAL DD, DD1, DD2
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) EDGE
    1 FORMAT('  #TRACE# SHRINK (',I4,')')
      SHRK$ = SHRK$ + 1
      LABL$ = LABL$ + 1
C
C     GET FIRST ALTERNATING PATH
      H        = 1
      PACK(H)  = 0
      EDGE1    = EDGE
   10     NODE1     = PSEUDO(HEAD(EDGE1))
          H         = H + 1
          PACK(H)   = NODE1
          EDGE1     = PRED(NODE1)
      IF (EDGE1.NE.NULL) GO TO 10
C
C     GET SECOND ALTERNATING PATH
      T        = 50
      PACK(T)  = -1
      EDGE2    = M9 - EDGE
   20     NODE2     = PSEUDO(HEAD(EDGE2))
          T         = T - 1
          PACK(T)   = NODE2
          EDGE2     = PRED(NODE2)
      IF (EDGE2.NE.NULL) GO TO 20
C
C     ELIMINATE COMMON NODES IN THE PATHS
   30     H = H - 1
          T = T + 1
      IF (PACK(H).EQ.PACK(T)) GO TO 30
```

```
C
C     CREATE "PNODE"
      L1              = L1 + 1
      PNODE           = LIST(L1)
      INDEX(PNODE)    = L1
      BASE            = PACK(H+1)
      LABLO(PNODE)    = OUTER
      PSEUDO(PNODE)   = PNODE
      PRICE(PNODE)    = 0
      LINK(PNODE)     = NULL
      MATE(PNODE)     = BASE
      APEX(PNODE)     = APEX(BASE)
      PRED(PNODE)     = PRED(BASE)
      ROOT            = HEAD(BASE)
      HEAD(PNODE)     = ROOT
      TREE(PNODE)     = TREE(ROOT)
      TREE(ROOT)      = PNODE
C
C     UPDATE NODE VECTORS OF THE FIRST PATH
      PNODE1 = PACK(2)
   40 IF (PNODE1.EQ.BASE) GO TO 50
           EDGE1          = MATE(APEX(PNODE1))
           PNODE2         = PSEUDO(HEAD(EDGE1))
           EDGE2          = PRED(PNODE1)
           PRED(PNODE1)   = EDGE1
           HEAD(PNODE1)   = PNODE2
           PNODE1         = PSEUDO(HEAD(EDGE2))
           PRED(PNODE2)   = EDGE2
           HEAD(PNODE2)   = PNODE1
           GO TO 40
C
C     UPDATE NODE VECTORS OF THE SECOND PATH
   50 PNODE1        = PACK(49)
      EDGE2         = PRED(PNODE1)
      PRED(PNODE1)  = EDGE
      HEAD(PNODE1)  = PACK(2)
   60 IF (PNODE1.EQ.BASE) GO TO 70
           EDGE1          = MATE(APEX(PNODE1))
           PNODE2         = PSEUDO(HEAD(EDGE1))
           HEAD(PNODE2)   = PNODE1
           PRED(PNODE2)   = M9 - EDGE1
           PNODE1         = PSEUDO(HEAD(EDGE2))
           HEAD(PNODE1)   = PNODE2
           EDGE1          = PRED(PNODE1)
           PRED(PNODE1)   = M9 - EDGE2
           EDGE2          = EDGE1
           GO TO 60
C
C     DETERMINE "BCI(PNODE)"
   70 BCINOD    = L2
      DD        = INF
      J         = 1
      STACK(J)  = PNODE
      CALL ORIG(PNODE)
      DO 80 H = 1, HORIG
           NODE          = NORIG(H)
           LABLO(NODE)   = PNODE
           J             = J + 1
           STACK(J)      = NODE
           BNODE         = BCI(NODE)
```

```
          IF  (SET(BNODE).EQ.SET3.AND.BNODE.EQ.NODE)  MPRICE(BNODE)  =
     &              2 * PRICE(BNODE)
          DD1           = MPRICE(NODE)  -  PRICE(BNODE)
          IF  (DD1.GE.DD)  GO TO 80
          DD            = DD1
          BCINOD        = BNODE
   80     CONTINUE
C
C     PRINT SIMPLE BLOSSOM
      BCI(PNODE)      = BCINOD
      MPRICE(PNODE) = MPRICE(BCINOD)
      IF  (PRINT)  WRITE(6,2)  (STACK(I), I=1,J)
    2 FORMAT(' SHRINK SIMPLE BLOSSOM',I6,' WITH NODES',I3,' (BASE)',
     &              10I4/(47X,10I4))
C     SET PSEUDO OF ALL NODES INSIDE "PNODE"
      CALL STPSDO(PNODE)
      IF  (DD.GT.0)  RETURN
C         TYPE 3 AUGMENTATION
          MATE(BCINOD) = NULL
          IF  (PRINT)  WRITE(6,3)  ROOT,  PNODE
    3     FORMAT(' TYPE 3 AUGMENTATION :',I6,'<R> ...',I3,'<+>')
          CALL RMATCH(BCINOD)
          RETURN
      END
```

```
      SUBROUTINE NSHRNK(PNODE)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /UNSHRINK "PNODE"/
C     "PNODE" IS AN INNER LABELLED PSEUDONODE WITH NULL NODE PRICE
C     "PNODE" "PNODE0" ARE MATED BY "EDGE0"
C     "BLSM0" AND "BASE" ARE THE EXTREME OF THE ALTERNATING PATH
C          TO BE CONSTRUCTED ON THE SIMPLE BLOSSOM OF "PNODE"
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
      INTEGER LAB(3)/'<+>','<->','< >'/
      LOGICAL DONT
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) PNODE
    1 FORMAT('  #TRACE# NSHRNK (',I4,')')
      UNSH$  = UNSH$ + 1
C
C     ELIMINATE "PNODE" FROM TREE
      NODE = HEAD(PNODE)
   10 IF (TREE(NODE).EQ.PNODE) GO TO 20
            NODE = TREE(NODE)
            GO TO 10
   20 TREE(NODE)   = TREE(PNODE)
      TREE(PNODE)  = NULL
C     DETERMINE "BASE", "PNODE0", AND "BLSM0"
      BASE    = MATE(PNODE)
      PNODE0  = PSEUDO(HEAD(MATE(APEX(PNODE))))
      EDGE0   = PRED(PNODE0)
      BLSM0   = HEAD(M9-EDGE0)
   30 IF (LABLO(BLSM0).EQ.PNODE) GO TO 40
            BLSM0 = LABLO(BLSM0)
            GO TO 30
C
C     DETERMINE NUMBER OF NODES BETWEEN "BASE" AND "BLSM0"
   40 BLSM = BASE
      PT   = 1
   50 IF (BLSM.EQ.BLSM0) GO TO 60
            PT    = PT + 1
```

```
            EDGE1 = PRED(BLSM)
            BLSM1 = BLSM
            BLSM  = HEAD(BLSM)
            GO TO 50
C       EVEN NUMBER OF NODES -- RIGHT DIRECTION
   60 IF (PT.EQ.PT/2*2.OR.PT.EQ.1) GO TO 80
C           ODD NUMBER -- REDIRECT THE SIMPLE BLOSSOM
   70       EDGE2      = PRED(BLSM)
            BLSM2      = HEAD(BLSM)
            PRED(BLSM) = M9 - EDGE1
            HEAD(BLSM) = BLSM1
            EDGE1      = EDGE2
            BLSM1      = BLSM
            BLSM       = BLSM2
      IF (BLSM.NE.BLSM0) GO TO 70
C
C       UPDATE NODE VECTORS FOR NODES IN THE ALTERNATING PATH
   80 EDGE       = EDGE0
      BLSM1      = BLSM0
      ROOT       = HEAD(PNODE)
      BLSM2      = TREE(ROOT)
      TREE(ROOT) = BASE
      J          = 1
      STACK(J)   = BLSM1
      PACK(J)    = 2
   90       LABLO(BLSM1)        = INNER
            LABL$               = LABL$ + 1
            CALL STPSDO(BLSM1)
            TREE(BLSM1)         = BLSM2
            BLSM2               = HEAD(BLSM1)
            HEAD(BLSM1)         = ROOT
            EDGE1               = PRED(BLSM1)
            PRED(BLSM1)         = NULL
      IF (BLSM1.EQ.BASE) GO TO 100
            LABLO(BLSM2)        = OUTER
            LABL$               = LABL$ + 1
            CALL STPSDO(BLSM2)
            EDGE2               = M9 - EDGE1
            NODE1               = HEAD(EDGE2)
            NODE2               = HEAD(EDGE1)
            MATE(NODE1)         = EDGE1
            MATE(NODE2)         = EDGE2
            CALL STAPEX(NODE1)
            CALL STAPEX(NODE2)
            TREE(BLSM2)         = BLSM1
            BLSM1               = HEAD(BLSM2)
            HEAD(BLSM2)         = ROOT
            EDGE2               = PRED(BLSM2)
            PRED(BLSM2)         = EDGE
            EDGE                = M9 - EDGE2
            HSCAN               = HSCAN + 1
            IF (HSCAN.GT.N9) HSCAN = 1
            NSCAN(HSCAN)        = BLSM2
            J                   = J + 2
            STACK(J-1)          = BLSM2
            STACK(J)            = BLSM1
            PACK(J-1)           = 1
            PACK(J)             = 2
            GO TO 90
C
```

```
C     UPDATE NODE VECTORS FOR NODE NOT IN THE ALTERNATING PATH
  100 PRED(PNODE0) = EDGE
      DONT         = .FALSE.
  110 IF (BLSM2.EQ.BLSM0) GO TO 120
            BLSM1         = HEAD(BLSM2)
            LABLO(BLSM2) = UNLABD
            LABLO(BLSM1) = UNLABD
            CALL STPSDO(BLSM2)
            CALL STPSDO(BLSM1)
            EDGE1         = PRED(BLSM2)
            EDGE2         = M9 - EDGE1
            NODE2         = HEAD(EDGE1)
            NODE1         = HEAD(EDGE2)
            MATE(NODE2)  = EDGE2
            MATE(NODE1)  = EDGE1
            CALL STAPEX(NODE1)
            CALL STAPEX(NODE2)
            PRED(BLSM2)  = NULL
            PRED(BLSM1)  = NULL
            NSCAN(TSCAN) = -BLSM2
            TSCAN         = TSCAN - 1
            IF (TSCAN.LT.1) TSCAN = N9
            NSCAN(TSCAN) = -BLSM1
            TSCAN         = TSCAN - 1
            IF (TSCAN.LT.1) TSCAN = N9
            J             = J + 2
            STACK(J-1)   = BLSM2
            PACK(J-1)    = 3
            STACK(J)     = BLSM1
            PACK(J)      = 3
            HEAD(BLSM2)  = NULL
            BLSM2         = HEAD(BLSM1)
            HEAD(BLSM1)  = NULL
            GO TO 110
C
C     UPDATE THE EDGE VECTORS FOR EDGES LINKED TO PNODE
  120 PT1          = LINK(PNODE)
      LINK(PNODE) = NULL
  130 PT = PT1
      IF (PT.EQ.NULL) GO TO 150
            PT1          = LINK(PT)
            EDGE = MATE(PT)
            IF (EDGE.EQ.NULL) GO TO 140
            NODE         = HEAD(EDGE)
            LINK(PT)    = LINK(NODE)
            LINK(NODE) = PT
            GO TO 130
  140       LINK(PT) = HINK
            HINK     = PT
            GO TO 130
C
C     UPDATE PSEUDONODE LIST
  150 NODE           = LIST(L1)
      IND            = INDEX(PNODE)
      LIST(L1)       = PNODE
      LIST(IND)      = NODE
      INDEX(NODE)    = IND
      INDEX(PNODE)   = NULL
      L1             = L1 - 1
      IF (PRINT) WRITE(6,2) PNODE, (STACK(I),LAB(PACK(I)
```

```
2 FORMAT(' UNSHRINK PSEUDONODE :  ',I4,' WITH NODES',10(I5,A3)/(
 &              39X,10(I5,A3)))
  RETURN
  END
```

```
      SUBROUTINE STPSDO(PNODE)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /SET PSEUDO FOR NODES INSIDE "PNODE"/
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) PNODE
    1 FORMAT('  #TRACE# STPSDO (',I4,')')
      PSEUDO(PNODE) = PNODE
C     "PNODE" IS AN ORIGINAL NODE -- RETURN
      IF (PNODE.LE.N2) RETURN
C
C     SET PSEUDO FOR ALL ORIGINAL NODES INSIDE "PNODE"
      CALL ORIG(PNODE)
   10 IF (HORIG.EQ.0) RETURN
            NODE         = NORIG(HORIG)
            HORIG        = HORIG - 1
            PSEUDO(NODE) = PNODE
            IF (NODE.GT.N2) CALL ORIG(NODE)
            GO TO 10
C
      END
```

```
      SUBROUTINE STAPEX(NODE)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /SET "NODE" AS APEX/
C     "NODE" IS AN ORIGINAL NODE
C     SET "NODE" AS APEX OF ALL PSEUDONODES
C     SET "BASE" AS BASE OF ALL PSEUDONODES
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) NODE
    1 FORMAT('  #TRACE# STAPEX (',I4,')')
C
C     "NODE" IS CONSTANT FOR ALL PSEUDOS
C     "BASE" IS VARIABLE
C     "BLSM" IS THE SIMPLE BLOSSOM WITH BASE "BASE"
      BASE = NODE
   10 BLSM = LABLO(BASE)
      IF (BLSM.LE.N2) RETURN
          MATE(BLSM) = BASE
          APEX(BLSM) = NODE
          BASE = BLSM
          GO TO 10
      END
```

```
      SUBROUTINE SCAN(EDGE1)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /SCAN EDGE/
C     "EDGE" IS AN EQUALTIY EDGE
C     "NODE1"  <--  "EDGE"  --<  "NODE2"
C     "NODE3" IS THE MATE OF "NODE2"
C
C     CHECK AUGMENTATION
C     CHECK SHRINKING
C     CHECK NODE LABELLING
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
      REAL COST
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) EDGE1
    1 FORMAT('  #TRACE# SCNEDG (',I4,')')
      EDGE$ = EDGE$ + 1
C
C     DETERMINE "NODE1", "NODE2"
      EDGE2  = M9 - EDGE1
      NODE2  = HEAD(EDGE1)
      NODE1  = HEAD(EDGE2)
      PNODE1 = PSEUDO(NODE1)
      PNODE2 = PSEUDO(NODE2)
      APEX1  = APEX(PNODE1)
      APEX2  = APEX(PNODE2)
      ROOT1  = HEAD(PNODE1)
      ROOT2  = HEAD(PNODE2)
      LABEL1 = LABLO(PNODE1)
      LABEL2 = LABLO(PNODE2)
C
C     START CHECKS
      EDGE = EDGE1 - L2
      IF (PRINT) WRITE(6,9) EDGE, NODE1, NODE2
    9 FORMAT(' SCAN EDGE :          ',I5,' = (',I3,';',I3,')')
C     NONCURRENT EDGE -- RETURN
      IF ( PNODE1 .EQ. PNODE2 ) RETURN
```

```
      IF ( LABEL2 .EQ. UNLABD ) GO TO 20
C     "NODE1", "NODE2" ARE BOTH OUTER NODES
      IF ( ROOT1  .NE. ROOT2  ) GO TO 10
C           "NODE1", "NODE2" ARE IN THE SAME TREE -- SHRINK
            CALL SHRINK(EDGE1)
            RETURN
C           "NODE1", "NODE2" ARE IN DIFFERENT TREES -- AUGMENT
   10       MATE(NODE1)    = EDGE1
            MATE(NODE2)    = EDGE2
            IF (PRINT) WRITE(6,2) ROOT1,PNODE1,PNODE2,ROOT2
    2       FORMAT(' DOUBLE AUGMENTATION :',I6,'<R> ...',I3,'<+>',I4,
     &             '<+> ...',I3,'<R>')
            CALL RMATCH(NODE1)
            CALL RMATCH(NODE2)
            RETURN
C
C     "NODE2" IS UNLABELLED
   20 IF ( PRICE(PNODE2).GT.TOLER .OR. PNODE2 .GT.N1 )      GO TO 30
      IF ( SET(PNODE2).NE.SET0 .AND. SET(PNODE2).NE.SET3 ) GO TO 30
C           "NODE2" CAN BE A TYPE 2 NODE-- AUGMENT
            MATE(NODE1) = NULL
            CALL MATCH(EDGE1)
            IF (PRINT) WRITE(6,3) ROOT1,PNODE1,PNODE2
    3       FORMAT(' TYPE 2 AUGMENTATION :',I6,'<R> ...',I3,'<+>',I4,
     &             '< >')
            CALL RMATCH(NODE1)
            RETURN
C
   30 IF (MATE(APEX2).NE.NULL) GO TO 40
C           "NODE2" IS A TYPE 0 NODE -- AUGMENT
            MATE(NODE1)    = EDGE1
            MATE(NODE2)    = EDGE2
            IF (PRINT) WRITE(6,4) ROOT1,PNODE1,PNODE2
    4       FORMAT(' TYPE 0 AUGMENTATION :',I6,'<R> ...',I3,'<+>',I4,
     &             '< >')
            IF (NODE2.NE.PNODE2) CALL STAPEX(NODE2)
            CALL RMATCH(NODE1)
            RETURN
C
   40 EDGE3 = MATE(APEX2)
      APEX3 = HEAD(EDGE3)
      IF ( MATE(APEX3).GE.NULL ) GO TO 50
C           "NODE2" IS A TYPE 1 NODE -- AUGMENT
            CALL NMATCH(EDGE3)
            MATE(NODE1)    = EDGE1
            MATE(NODE2)    = EDGE2
            IF (PRINT) WRITE(6,5) ROOT1,PNODE1,PNODE2
    5       FORMAT(' TYPE 1 AUGMENTATION :',I6,'<R> ...',I3,'<+>',I4,
     &             '< >')
            IF (NODE2.NE.PNODE2) CALL STAPEX(NODE2)
            CALL RMATCH(NODE1)
            RETURN
C
C     LABEL "NODE2" INNER
C     LABEL "NODE3" OUTER
   50 PNODE3          = PSEUDO(APEX3)
      LABLO(PNODE2)   = INNER
      HEAD(PNODE2)    = ROOT1
      LABLO(PNODE3)   = OUTER
      TREE(PNODE2)    = PNODE3
```

```
      TREE(PNODE3)    = TREE(ROOT1)
      TREE(ROOT1)     = PNODE2
      PRED(PNODE3)    = EDGE2
      HEAD(PNODE3)    = ROOT1
      LABL$           = LABL$ + 2
      IF (PRINT) WRITE(6,6) ROOT1,PNODE1,PNODE2, PNODE3
    6 FORMAT(' NODE LABELING :        ',I6,'<R> ...',I3,'<+>',I4,'<->'
     &                ,I4,'<+>')
C
      NODE3 = BCI(PNODE3)
      IF ( PRICE(NODE3).NE.MPRICE(PNODE3) ) GO TO 60
C            "NODE3" CAN BECOME A TYPE 0 NODE -- AUGMENT
             MATE(NODE3) = NULL
             IF (PRINT) WRITE(6,7) ROOT1,PNODE3
    7        FORMAT(' TYPE 3 AUGMENTATION :',I6,'<R> ...',I3,'<+>')
             CALL RMATCH(NODE3)
             RETURN
C
   60 IF ( PRICE(PNODE2).NE.0.0 .OR. PNODE2.LE.N ) GO TO 70
C            "PNODE2" IS A INNER PSEUDO WITH NULL NODE PRICE --
C            -- UNSHRINK IT
             CALL NSHRNK(PNODE2)
             RETURN
C
C     PUSH "PNODE3" INTO THE LIST OF UNSCANNED NODES
   70 HSCAN          = HSCAN + 1
      IF (HSCAN.GT.N9) HSCAN = 1
      NSCAN(HSCAN) = PNODE3
      RETURN
      END
```

```
      SUBROUTINE GRWFOR
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C
C     /GROW ALTERNATING FOREST/
C     SCAN UNSCANNED NODES
C     PERFORM DUAL SOLUTION CHANGE
C     CHECK TERMINATION OR INFEASIBILITY
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
      REAL COST
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
   10 IF (TRACE) WRITE(6,1)
    1 FORMAT('  #TRACE# GRWFOR')
      IF (OPTION.EQ.9) CALL DUMP
C
C     DO 20 TO 80 FOR EVERY NODE IN "NSCAN"
   20 IF (HSCAN.EQ.TSCAN) GO TO 100
      TSCAN = TSCAN + 1
      IF (TSCAN.GT.N9) TSCAN = 1
      NODE  = NSCAN(TSCAN)
      LABEL = OUTER
      IF (NODE.GT.0) GO TO 30
C             "NODE > 0"  --  OUTER NODE
C             "NODE < 0"  --  EX-INNER NODE
              LABEL = UNLABD
              NODE  = -NODE
C     CHECK WHETHER "NODE" IS A DESTROYED PSEUDO
   30 IF (INDEX(NODE).EQ.NULL) GO TO 20
      PNODE = PSEUDO(NODE)
C     CHECK WHETHER LABEL IS STILL OUTER OR EX-INNER (UNLABELLED)
      IF (LABLO(PNODE).NE.LABEL) GO TO 20
      IF (PRINT) WRITE(6,7)NODE
    7 FORMAT(' SCAN NODE :               ',I4)
      NODE$ = NODE$ + 1
      PT    = NODE
C     BIFURCATE NODE : PSEUDO
      IF (NODE.LE.N2) GO TO 50
```

```
C
C     "NODE" IS A PSEUDONODE
C     PUSH ITS SIMPLE BLOSSOM INTO "NSCAN"
      LABEL = 2 * LABEL - 1
      BASE  = MATE(NODE)
      BLSM  = BASE
   40       NSCAN(TSCAN) = LABEL * BLSM
            TSCAN        = TSCAN - 1
            IF (TSCAN.LT.1) TSCAN = N9
            BLSM         = HEAD(BLSM)
      IF (BLSM.NE.BASE) GO TO 40
      GO TO 20
C
C     "NODE" IS AN ORIGINAL NODE
C     SCAN ALL EQUALTIY EDGES INCIDENT WITH "NODE"
   50 PT1  = PT
   60 PT   = LINK(PT1)
      IF ( PT .EQ. NULL ) GO TO 20
            EDGE1  = MATE(PT)
C           CHECK FOR DESTROYED EDGE
            IF (EDGE1.EQ.NULL) GO TO 70
C           DETERMINE DIRECTION OF ARC
            IF (LABEL.EQ.UNLABD) EDGE1 = M9 - EDGE1
            EDGE2  = M9 - EDGE1
            EDGE   = EDGE1
            IF (EDGE.GT.EDGE2) EDGE = EDGE2
            NODE1  = HEAD(EDGE2)
            NODE2  = HEAD(EDGE1)
            PNODE1 = PSEUDO(NODE1)
            PNODE2 = PSEUDO(NODE2)
            COST   = PRICE(EDGE) - PRICE(NODE1) - PRICE(NODE2)
C           CHECK   OUTER >-- EDGE --> UNLABELLED
            IF (LABLO(PNODE1).NE.OUTER ) GO TO 50
            IF (LABLO(PNODE2).EQ.INNER ) GO TO 50
C           CHECK FOR EQUALITY EDGE
            IF (  COST.GT.TOLER ) GO TO 70
C           CHECK FOR CURRENT EDGE
            IF (PNODE1.EQ.PNODE2) GO TO 80
            CALL SCAN(EDGE1)
C           CHECK AUGMENTATION
            IF (LABLO(PSEUDO(NODE1)).NE.OUTER.AND.LABEL.EQ.OUTER) GO
     &              TO 20
            GO TO 50
C
C           RETURN "LINK/MATE(PT)" TO HINK
   70       LINK(PT1)  = LINK(PT)
            MATE(PT)   = NULL
            LINK(PT)   = HINK
            HINK       = PT
            GO TO 60
C
C           TRANSFER "LINK/MATE(PT)" TO "PNODE1"
   80       LINK(PT1)     = LINK(PT)
            LINK(PT)      = LINK(PNODE1)
            LINK(PNODE1)  = PT
            GO TO 60
C
C     NO MORE AUGMENTATIONS, SHRINKINGS, UNSHRINKINGS, OR LABELLINGS
C     START DUAL SOLUTION CHANGE
  100 IF (TRACE)        CALL DUMP
```

```
      IF (OPTION.EQ.9) CALL DUMP
C     CHECK TERMINATION
      IF (ROOT$.EQ.0)  RETURN
C     INITIALIZE CONTROL VARIABLES
      DELTA = INF
      HINSP = 0
C
C     CHECK ORIGINAL NODES
      DO 110 IND = N0, N1
             NODE  = LIST(IND)
             LABEL = LABLO(NODE)
             PNODE = PSEUDO(NODE)
             IF ( PNODE.NE.NODE ) GO TO 110
C            ONLY CURRENT ORIGINAL NODES
C            CHECK MAXIMUM INCREASE
             IF ( SET(NODE).EQ.SET2 .AND. LABEL.EQ.OUTER ) CALL MINIM
     &                ( NODE, -PRICE(NODE) )
C            CHECK MAXIMUM DECREASE
             IF ( SET(NODE).EQ.SET3 .AND. LABEL.EQ.INNER ) CALL MINIM
     &                ( NODE,  PRICE(NODE) )
  110        CONTINUE
C
C     CHECK PSEUDONODES
      IF (L0.GT.L1) GO TO 130
      DO 120 IND = L0, L1
            NODE = LIST(IND)
            LABEL = LABLO(NODE)
C           CHECK MAXIMUM DECREASE
            IF ( LABEL.EQ.INNER ) CALL MINIM(NODE,PRICE(NODE))
            BNODE = BCI(NODE)
C           CHECK MAXIMUM INCREASE
            IF ( LABEL.EQ.OUTER .AND. BNODE.LT.N2 ) CALL MINIM( NODE,
     &                MPRICE(BNODE)-PRICE(BNODE) )
  120       CONTINUE
C
C     CHECK EDGES
  130 DO 140 IND = M0, M1
            EDGE   = LIST(IND)
            NODE1  = HEAD(EDGE)
            NODE2  = HEAD(M9-EDGE)
            PNODE1 = PSEUDO(NODE1)
            PNODE2 = PSEUDO(NODE2)
            IF (PNODE1.EQ.PNODE2) GO TO 140
C           ONLY CURRENT EDGES
            LABEL1 = LABLO(PNODE1)
            LABEL2 = LABLO(PNODE2)
            IF ( LABEL1.NE.OUTER .AND .LABEL2.NE.OUTER ) GO TO 140
C           "AT LEAST ONE OF "LABEL1" OR "LABEL2" IS OUTER
C           CHECK MAXIMUM INCREASE
            COST = PRICE(EDGE) - PRICE(NODE1) - PRICE(NODE2)
            IF ( LABEL1.EQ.OUTER .AND. LABEL2.EQ.UNLABD )
     1           CALL MINIM(EDGE,COST)
            IF ( LABEL1.EQ.UNLABD .AND. LABEL2.EQ.OUTER )
     1           CALL MINIM(EDGE,COST)
            IF ( LABEL1.EQ.OUTER .AND. LABEL2.EQ.OUTER )
     1           CALL MINIM(EDGE,COST/2)
  140       CONTINUE
C
C     NO NODE/EDGE TO INSPECT -- RETURN
      IF (HINSP.EQ.0) RETURN
```

```
C
C     UPDATE DUAL SOLUTION
      DUAL$ = DUAL$ + 1
      IF (PRINT) WRITE(6,2) DELTA
    2 FORMAT(' DUAL SOLUTION CHANGE:  ',F6.1)
C
C     UPDATE ORIGINAL NODE PRICES
      DO 210 IND = N0, N1
            NODE        = LIST(IND)
  210       PRICE(NODE) = PRICE(NODE) + DELTA*LABLO(PSEUDO(NODE))
C
C     UPDATE PSEUDONODE PRICES
      IF (L0.GT.L1) GO TO 300
      DO 220 IND = L0, L1
           NODE = LIST(IND)
           IF (NODE.EQ.PSEUDO(NODE))
     1         PRICE(NODE) = PRICE(NODE) + DELTA*LABLO(NODE)
  220      CONTINUE
C
C     UPDATE DUAL OBJECTIVE VALUE
  300 VALUE = VALUE + ROOT$*DELTA
C
C     START INSPECTING EVERY NODE/EDGE IN "INSP"
      DO 340 PT = 1, HINSP
           NODE = INSP(PT)
C          BIFURCATE  NODE : EDGE
           IF (NODE.GE.M0) GO TO 330
           IF (PRINT) WRITE(6,3) NODE
    3      FORMAT(' INSPECT NODE :        ',I6)
           LABEL = LABLO(NODE)
           IF (LABEL.EQ.INNER) GO TO 310
           IF (LABEL.NE.OUTER) GO TO 340
C              ONLY OUTER NODES -- AUGMENT
               BNODE       = BCI(NODE)
               MATE(BNODE) = NULL
               IF (PRINT) WRITE(6,4) HEAD(NODE),NODE
    4          FORMAT(' TYPE 3 AUGMENTATION :',I6,'<R> ...',I3,'<+>
     &         ')
               CALL RMATCH(BNODE)
               GO TO 340
C          ONLY INNER NODES
C          BIFURCATE  NODE : PSEUDO
  310      IF (NODE.GT.N2) GO TO 320
C              ONLY INNER ORIGINAL NODES -- AUGMENT
               EDGE        = PRED(HEAD(MATE(NODE)))
               CALL MATCH(EDGE)
               APEX1       = HEAD(EDGE)
               PNODE       = PSEUDO(APEX1)
               IF (PRINT) WRITE(6,5) HEAD(NODE),PNODE,NODE
    5          FORMAT(' TYPE 2 AUGMENTATION :',I6,'<R> ...',I3,'<+>
     &         ',I4,'< >')
               CALL RMATCH(APEX1)
               GO TO 340
C          ONLY INNER PSEUDONODES -- UNSHRINK
  320      CALL NSHRNK(NODE)
           GO TO 340
C
C          ONLY EDGES
C          EQUALITY EDGE
  330      EDGE1        = NODE
```

```
          EDGE2        = M9 - EDGE1
          NODE1        = HEAD(EDGE2)
          NODE2        = HEAD(EDGE1)
          IF (HINK.EQ.NULL) CALL ERROR(5)
          PTR          = HINK
          HINK         = LINK(HINK)
          MATE(PTR)    = EDGE1
          LINK(PTR)    = LINK(NODE1)
          LINK(NODE1)  = PTR
          IF (HINK.EQ.NULL) CALL ERROR(5)
          IF (HINK.GT.LINK$) LINK$ = HINK
          PTR          = HINK
          HINK         = LINK(HINK)
          MATE(PTR)    = EDGE2
          LINK(PTR)    = LINK(NODE2)
          LINK(NODE2)  = PTR
          LABEL1       = LABLO(PSEUDO(NODE1))
          LABEL2       = LABLO(PSEUDO(NODE2))
          EDGE = EDGE1 - L2
          IF (PRINT) WRITE(6,6) EDGE, NODE1, NODE2
    6     FORMAT(' INSPECT EDGE :          ',I5,' = (',I3,';',I3,')')
C         CHECK   OUTER >-- EDGE --> OUTER
C                 OUTER >-- EDGE --> UNLABD
C                UNLABD >-- EDGE --> OUTER
          IF (LABEL1.EQ.INNER.OR. LABEL2.EQ.INNER) GO TO 340
          IF (LABEL1.NE.OUTER.AND.LABEL2.NE.OUTER) GO TO 340
          IF (LABEL1.EQ.OUTER) CALL SCAN(EDGE1)
          IF (LABEL1.NE.OUTER) CALL SCAN(EDGE2)
  340     CONTINUE
C
C     CHECK TERMINATION
      IF (TRACE)        CALL DUMP
      IF (OPTION.EQ.8) CALL DUMP
      IF (ROOT$.GT.0)   GO TO 10
      RETURN
      END
```

```
      SUBROUTINE ORIG(PNODE)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /GET SIMPLE BLOSSOM, EVENTUALLY ALL ORIGINAL NODES/
C     PUSH INTO NORIG ALL NODES IN THE SIMPLE BLOSSOM OF "PNODE"
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) PNODE
    1 FORMAT('  #TRACE# ORIG   (',I4,')')
C
C     "PNODE" IS ORIGINAL NODE? -- RETURN
      IF (PNODE.LE.N2) RETURN
C
C     PUSH SIMPLE BLOSSOM INTO NORIG
      BASE = MATE(PNODE)
      NODE = BASE
   10     HORIG         = HORIG + 1
          NORIG(HORIG)  = NODE
          NODE          = HEAD(NODE)
      IF (NODE.NE.BASE) GO TO 10
C
      RETURN
      END
```

```
      SUBROUTINE PUTSOL
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /PUT SOLUTION/
C     TRANSFORM NODE PRICES
C     TRANSFORM TYPE D CONFIGURATION
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABID(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
      INTEGER EDLIST(3000)
      REAL SIGMA, NPRICE(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      CALL TIME(1,0,TIME2)
      IF (OPTION.EQ.5) CALL DUMP
      IF (OPTION.EQ.1) GO TO 310
C
C     PRINT HEADING
      K = 0
      WRITE(6,1) N, M
    1 FORMAT(//43X,'      *****  SOLUTION  *****'//43X,'NETWORK: ',I5
     &             ,' NODES,',I5,' EDGES')
C
C     CHECK INFEASIBILITY
      IF (DELTA.GE.INF) WRITE(6,2)
    2 FORMAT(//43X,'      =  PROBLEM INFEASIBLE  =')
C
C     PRINT DUAL OBJECTIVE VALUE
      WRITE(6,3) VALUE
    3 FORMAT(//43X,'  OBJECTIVE VALUE: ',F9.2)
C
C     FOR ALL NODES
      DO 70 IND = N0, N1
           NODE = LIST(IND)
           NPRICE(NODE) = PRICE(NODE)
C
C          CHECK CURRENCY
           IF (PSEUDO(NODE).NE.NODE) GO TO 40
C          GET MATES (SOLUTION EDGES) OF CURRENT NODES
           PT   = -MATE(NODE)
```

```
          IF (PT) 30, 70, 10
C              SEVERAL MATES
   10          EDGE = MATE(PT)
               IF (EDGE.GT.M2) GO TO 20
                    K        = K + 1
                    STACK(K) = EDGE - L2
   20          PT = LINK(PT)
          IF (PT.NE.NULL) GO TO 10
          GO TO 70
C         EXACTLY ONE MATE
   30     IF (-PT.GT.M2) GO TO 70
               K        = K + 1
               STACK(K) = -PT - L2
               GO TO 70
C         NONCURRENT NODES
   40     IF (NODE.NE.APEX(PSEUDO(NODE))) GO TO 50
C         IF APEX --> GET MATE (SOLUTION EDGE)
          EDGE = MATE(NODE)
          IF (EDGE.EQ.NULL.OR.EDGE.GT.M2) GO TO 50
               K        = K + 1
               STACK(K) = EDGE  - L2
C         TRANSFORM NODE PRICES
   50     IF (SET(NODE).EQ.SET2) GO TO 70
C         N< OK!
C         N= PRICE DECREASED ON EVERY PSEUDO
C         N> PRICE DOUBLE DECREASED ON EVERY PSEUDO
          BLSM = NODE
          SIGN = 1
          IF (SET(NODE).EQ.SET3) SIGN = 2
   60          BLSM = LABLO(BLSM)
               IF (BLSM.LE.N2) GO TO 70
               NPRICE(NODE) = NPRICE(NODE) - SIGN * PRICE(BLSM)
               GO TO 60
   70     CONTINUE
C
C      PRINT NODES AND EDGES
       WRITE(6,4)  (LIST(I),SET(LIST(I)),NPRICE(LIST(I)),I=N0,N1)
     4 FORMAT(//' LIST OF NODES (NODE,SET,PRICE) :'/5(I15,I2,F7.1))
       DO 80 IND = M0, M1
   80     EDLIST(IND) = LIST(IND) - L2
       WRITE(6,5)  (EDLIST(I),HEAD(LIST(I)),HEAD(M9-LIST(I)),PRICE(LIS
     &             T(I)),I=M0,M1)
     5 FORMAT(//' LIST OF EDGES (EDGE,NODE1,NODE2,COST) :'/5(I11,2I3,
     &             F7.1))
C
C      CHECK EXISTENCE OF PSEUDONODES
       IF (L0.GT.L1) GO TO 200
       WRITE(6,6)
     6 FORMAT(//' PSEUDO  PRICE   SIMPLE BLOSSOM')
C
C      CORRECT SOLUTION EDGES ON EVERY SIMPLE BLOSSOM
       DO 150 IND = L0, L1
          PNODE = LIST(IND)
          BASE  = MATE(PNODE)
C         GET SIMPLE BLOSSOM
          I     = 0
          NODE  = BASE
  110          I        = I + 1
               PACK(I)  = NODE
               NODE     = HEAD(NODE)
```

```
      IF (NODE.NE.BASE) GO TO 110
      WRITE(6,7) PNODE,PRICE(PNODE),(PACK(J),J=1,I)
    7 FORMAT(I7,F7.1,2X,20I5/(19X,20I5))
C     CHECK TYPE C SIMPLE BLOSSOM
      APICE = APEX(NODE)
      IF (BASE.GT.N2.OR.SET(APICE).NE.SET3.OR.MATE(APICE).NE.NU
     &    LL.OR.NPRICE(APICE).GT.TOLER) GO TO 130
C     TYPE C SIMPLE BLOSSOM
C     GET SOLUTION EDGES
      BLSM1 = BASE
  120     BLSM2 = HEAD(BLSM1)
          K = K + 1
          STACK(K) = PRED(BLSM1)
          IF (STACK(K).GT.M2) STACK(K) = M9 - STACK(K)
          STACK(K) = STACK(K) - L2
          IF (BLSM2.EQ.BASE) GO TO 150
          BLSM1 = HEAD(BLSM2)
          GO TO 120
C     TYPE A, TYPE B, TYPE D, OR ROOTED SIMPLE BLOSSOM
C     GET SOLUTION EDGES
  130 BLSM1 = HEAD(BASE)
  140     K     = K + 1
          STACK(K) = PRED(BLSM1)
          IF (STACK(K).GT.M2) STACK(K) = M9 - STACK(K)
          STACK(K) = STACK(K) - L2
          BLSM1 = HEAD(HEAD(BLSM1))
      IF (BLSM1.NE.BASE) GO TO 140
  150 CONTINUE
C
C     PRINT SOLUTION EDGES
  200 WRITE(6,8) (STACK(I),I=1,K)
    8 FORMAT(//' SOLUTION EDGES:'/(10I12))
C
C     PRINT STATISTICS
  300 IF (OPTION.EQ.2) RETURN
      WRITE(6,11)
   11 FORMAT(///45X,'     =  STATISTICS  =')
  310 CALL TIME(1,0,TIME3)
      TIME3 = TIME3 - TIME2
      TIME2 = TIME2 - TIME1
      WRITE(6,12) DUAL$,AUGM$,RMAT$,LABL$,NODE$,EDGE$,SHRK$,UNSH$,LI
     &            NK$,TIME1,TIME2,TIME3
   12 FORMAT(//
     119X,I5,' DUAL SOL. CHANGES   ',    I5,' PATHS REMATCHED      ',
     2    I5,' NODES REMATED       '/19X,I5,' NODES LABELLED       ',
     3    I5,' NODES SCANNED       ',    I5,' EDGES SCANNED        '/
     419X,I5,' SHRINKINGS          ',    I5,' UNSHRINKINGS         ',
     5    I5,' LINK/MATES USED     '/19X,I5,' CPU MSEC (INPUT)     ',
     5    I5,' CPU MSEC (SOLUTION) ',    I5,' CPU MSEC (OUTPUT)    ')
      IF (OPTION.EQ.5) CALL DUMP
      RETURN
      END
```

```
      SUBROUTINE DUMP
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /DUMP NODE/EDGE INFORMATION/
C
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C     DUMP NODE INFORMATION
      WRITE(6,1)
    1 FORMAT(///'  NODE   SET LABEL PSEUD  HEAD  PRED  TREE  APEX
     &              BCI  PRICE MPRICE  LINK(S)')
      J = 0
      DO 60 IND = N0, N1
           NODE     = LIST(IND)
           IF (NODE.EQ.NULL) GO TO 60
           PT       = LINK(NODE)
           PACK(1)  = 0
           K        = 0
   10      IF (PT.EQ.NULL) GO TO 20
                K          = K + 1
                PACK(K)    = MATE(PT)
                PT         = LINK(PT)
                GO TO 10
   20      IF (APEX(PSEUDO(NODE)).NE.NODE) GO TO 50
           PT = -MATE(NODE)
           IF (PT) 30, 50, 40
   30           J          = J + 1
                STACK(J)   = -PT
                GO TO 50
   40      J          = J + 1
           STACK(J)   = MATE(PT)
           PT         = LINK(PT)
           IF (PT.NE.NULL) GO TO 40
   50      WRITE(6,2) NODE,SET(NODE),LABLO(NODE),PSEUDO(NODE),HEAD(N
     &              ODE),PRED(NODE),TREE(NODE),APEX(NODE),BCI(NODE),
    2      FORMAT(9I6,2F7.1,1X,10I5/(69X,10I5))
```

```
   60       CONTINUE
C
C     DUMP PSEUDO INFORMATION
      IF (LO.GT.L1) GO TO 200
      WRITE(6,3)
    3 FORMAT(//' PNODE  BASE IABEL PSEUD  HEAD  PRED  TREE  APEX   B
     &              CI  PRICE MPRICE  LINK(S)')
      DO 120 IND = LO, L1
           NODE    = LIST(IND)
           PT      = LINK(NODE)
           PACK(1) = 0
           K       = 0
  110      IF (PT.EQ.NULL) GO TO 120
                K        = K + 1
                PACK(K)  = MATE(PT)
                PT       = LINK(PT`
                GO TO 110
  120      WRITE(6,2) NODE,MATE(NODE),LABLO(NODE),PSEUDO(NODE),HEAD(
     &              NODE),PREC(NODE),TREE(NODE),APEX(NODE),BCI(NODE)
C
C     DUMP EDGE INFORMTION
  200 IF (J.EQ.0) RETURN
      WRITE(6,4) (STACK(I),HEAD(STACK(I)),HEAD(M9-STACK(I)),I=1,J)
    4 FORMAT(//'MATES : (EDGE,NODE1,NODE2)'/5(I9,2I4))
      RETURN
      END
```

```
      SUBROUTINE ERROR(I)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /PRINT MESSAGE OF ERROR/
C
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      GO TO (10,20,30,40,50,60,70,80,90,100), I
      GO TO 110
C
   10 WRITE(6,1)
    1 FORMAT(' *****     UNEXPECTED END OF FILE     *****')
      STOP
C
   20 WRITE(6,2)
    2 FORMAT(' *****     ILLEGAL PARAMETER     *****')
      STOP
C
   30 WRITE(6,3)
    3 FORMAT(' *****     NODE STORAGE SPACE EXCEEDED     *****')
      STOP
C
   40 WRITE(6,4)
    4 FORMAT(' *****     EDGE STORAGE SPACE EXCEEDED     *****')
      STOP
C
   50 WRITE(6,5)
    5 FORMAT(' *****     LINK/MATE STORAGE SPACE EXCEEDED     *****')
      STOP
C
   60 WRITE(6,6)
    6 FORMAT(' *****     ILLEGAL NODE     *****')
      STOP
C
   70 WRITE(6,7)
    7 FORMAT(' *****     ILLEGAL EDGE     *****')
```

```
      STOP
C
   80 WRITE(6,8)
    8 FORMAT(' *****      ILLEGAL NODE SET      *****')
      STOP
C
   90 WRITE(6,9)
    9 FORMAT(' *****      ILLEGAL COST      *****')
      STOP
C
  100 CONTINUE
  110 WRITE(6,12) I
   12 FORMAT(' *****      ERROR NO. ',I3,'    *****')
      STOP
      END
```

```
      SUBROUTINE POSTOP
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /POST-OPTIMALITY ANALYSIS/
C     K1  EDGE COST CHANGES
C     K2  SUBSET CHANGES
C     K3  EDGE ELIMINATIONS
C     K4  NODE ELIMINATIONS
C     K5  NODE INCLUSIONS
C     K6  EDGE INCLUSIONS
C
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)

      LOGICAL FREE(1) /'*'/
      REAL NPRICE, COST
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      CALL TIME(0)
      IF (.NOT.TRACE) GO TO 10
          WRITE(6,1)
    1     FORMAT('  #TRACE# POSTOP')
          TRACE  = .FALSE.
          PRINT  = .TRUE.
          OPTION = 3
   10 IF (0.GT.OPTION.AND.OPTION.GE.-N2) OPTION = 3
C
C     GET NUMBER OF MODIFICATIONS
      READ(5,FREE,END=1000) K1,K2,K3,K4,K5,K6
      IF (K1+K2+K3+K4+K5+K6.EQ.0) STOP
      IF (K1.LT.0.OR.K2.LT.0.OR.K3.LT.0.OR.K4.LT.0.OR.K5.LT.0.OR.K6.
     &               LT.0) CALL ERROR(2)
      WRITE(6,2) K1, K2, K3, K4, K5, K6
    2 FORMAT(1H1,40X,'*****  POST-OPTIMALITY ANALYSIS  *****'//
     139X,'CHANGES:        ',I4,' EDGE COSTS,',I4,' NODE SETS'/
     239X,'ELIMINATIONS: ',I4,' EDGES,      ',I4,' NODES'/
     339X,'INTRODUCTIONS:',I4,' NODES,      ',I4,' EDGES')
      IF (OPTION.GE.0)   GO TO 100
```

```
            OPTION = CPTION + N2
            IF (OPTION.EQ.-N2) TRACE = .TRUE.
            IF (0.GT.OPTION.AND.OPTION.GE.-N2) PRINT = .TRUE.
C
C     EDGE COST CHANGES
  100 IF (K1.LT.1) GO TO 200
      DO 110 K = 1, K1
      IF (TRACE) CALL DUMP
C     GET EDGE AND NEW COST
      READ(5,FREE,END=999)  EDGE, NPRICE
      IF (TRACE.OR.OPTION.EQ.4) WRITE(6,4) EDGE, NPRICE
    4 FORMAT(' EDGE COST CHANGE :    ',I5,F7.1)
      STACK(K)     = EDGE
      EDGE = EDGE + L2
      IF (EDGE.LT.M0.OR.EDGE.GT.M2.OR.INDEX(EDGE).EQ.NULL) CALL ERRO
     &              R(7)
      IF (NPRICE.GT.INF.OR.NPRICE.LT.-INF)                   CALL ERRO
     &              R(9)
      IF ( NPRICE.EQ.PRICE(EDGE) ) GO TO 110
      EDGE1        = EDGE
      EDGE2        = M9 - EDGE
      NODE2        = HEAD(EDGE1)
      NODE1        = HEAD(EDGE2)
      PRICE(EDGE)  = NPRICE
      COST         = PRICE(EDGE) - PRICE(NODE1) - PRICE(NODE2)
      CALL NMATCH(EDGE)
      IF (PSEUDO(NODE1).NE.PSEUDO(NODE2).AND.COST.GE.-TOLER) GO TO 1
     &              10
C     EDGE IS OUT-OF-KILTER
C     TRY TO DECREASE PRICE(NODE1)
      CALL OPEN(NODE1)
      CALL DECRSE(EDGE1,COST)
      IF (COST.GT.-TOLER) GO TO 110
C     TRY TO DECREAS PRICE(NODE2)
      CALL OPEN(NODE2)
      CALL DECRSE(EDGE2,COST)
      IF (COST.GT.-TOLER) GO TO 110
C     EDGE OF A-
      CALL MATCH(EDGE)
      VALUE = VALUE + COST
  110 CONTINUE
      IF (PRINT) WRITE(6,5) (STACK(K),HEAD(STACK(K)),HEAD(M9-STACK(K
     &              )),PRICE(STACK(K)),K=1,K1)
    5 FORMAT(//' EDGE COST CHANGE (EDGE,NODE1,NODE2,NEWCOST):'/5(I9,
     &              2I3,F6.1))
C
C     SUBSET CHANGE
  200 IF (K2.LT.1) GO TO 300
      DO 250 K = 1, K2
      IF (TRACE) CALL DUMP
C     GET NODE AND NEWSET
      READ(5,FREE,END=999)  NODE, NSET
      IF (TRACE.OR.OPTION.EQ.4) WRITE(6,6) NODE, NSET
    6 FORMAT(' CHANGE NODE SET :     ',2I5)
      STACK(K) = NODE
      IF (NSET.LT.SET0.OR.NSET.GT.SET3 ) CALL ERROR(8)
      IF (NODE.LT.N0.OR.NODE.GT.N2.OR.INDEX(NODE).EQ.NULL) CALL ERRO
     &              R(6)
      IF ( NSET.EQ.SET(NODE) ) GO TO 250
C     UPDATE SET(NODE0, MPRICE(NODE), AND BCI(NODE)
```

```
      J              = SET(NODE) + 1
      SET(NODE)      = NSET
      MPRICE(NODE) = 0
      IF  (SET(NODE).EQ.SET1)  MPRICE(NODE)  =  INF
      IF  (SET(NODE).EQ.SET3)  MPRICE(NODE)  =  INF
      BCI(NODE)      = NODE
      IF  (SET(NODE).EQ.SET1)  BCI(NODE)  =  L2
      PNODE          = PSEUDO(NODE)
C     FORCE NODE TO BECOME CURRENT
      IF (PNODE.NE.NCDE) CALL OPEN(NODE)
C     BRANCH IN ACCORDANCE WITH OLD SUBSET
      GO TO (210,220,230,240), J
C
C     OLD SUBSET WAS SET0
C     SET3 IS OK
  210 IF ( NSET.EQ.SET3 ) GO TO 250
C     SET1, SET2 : UNMATCH ALL MATES OF NODE
  212 PT            = MATE(NODE)
      IF  (PT) 214, 218, 216
  214       EDGE    = MATE(-PT)
            CALL NMATCH(EDGE)
            GO TO 212
  216 CALL NMATCH(PT)
C     UPDATE PRICE(NODE)
  218 VALUE           = VALUE- PRICE(NODE)
      PRICE(NODE) = -INF/2
      VALUE           = VALUE + PRICE(NODE)
      GO TO 250
C
C     OLD SUBSET WAS SET1
  220 IF  (NSET.NE.SET0)  GO TO 222
C     NEW SET IS SET0
      CALL OPEN(NODE)
      IF  (PRICE(NODE).LT.-TOLER)  CALL INCRSE(NODE)
      VALUE = VALUE - PRICE(NODE)
      PRICE(NODE) = 0
      GO TO 250
C
C     NEW SET IS SET2
  222 IF  (NSET.NE.SET2)  GO TO 224
      IF  (PRICE(NODE).LE.+TOLER)  GO TO 250
      CALL OPEN(NODE)
      VALUE = VALUE - PRICE(NODE)
      PRICE(NODE)  = -INF/2
      VALUE = VALUE + PRICE(NODE)
      GO TO 250
C
C     NEW SET IS SET3
  224 IF  (PRICE(NODE).LT.-TOLER)  CALL INCRSE(NODE)
      GO TO 250
C
C     OLD SET WAS SET2
C     NEW SET IS SET1 OK
  230 IF  ( NSET.EQ.SET1 ) GO TO 250
C     NEW SET IS SET0 OR SET3
      CALL OPEN(NODE)
      CALL INCRSE(NODE)
      GO TO 250
C
C     OLD SET WAS SET3
```

```
  240 IF  (SET(NODE).EQ.SET0.AND.PRICE(NODE).LE.TOLER)  GO TO 250
      IF  (SET(NODE).EQ.SET1.AND.MATE(NODE).GE.NULL)  GO TO 250
      IF  (SET(NODE).EQ.SET2.AND.PRICE(NODE).LE.TOLER.AND.MATE(NODE).
     &                GE.NULL)  GO TO 250
C
C     UNMATCH ALL MATES OF NODE
  242 PT = MATE(NODE)
      IF (PT) 244, 248, 246
  244       EDGE = MATE(-PT)
            CALL NMATCH(EDGE)
            GO TO 242
  246 CALL NMATCH(PT)
C     UPDATE PRICE(NODE)
  248 VALUE = VALUE - PRICE(NODE)
      PRICE(NODE) = -INF/2
      IF ( SET(NODE).EQ.SET0 ) PRICE(NODE) = 0
      VALUE = VALUE + PRICE(NODE)
C
  250 CONTINUE
      IF (PRINT) WRITE(6,7) (STACK(K),SET(STACK(K)),K=1,K2)
    7 FORMAT(//' NODE SET CHANGE (NODE,NEWSET):'/10(I10,I2))
C
C     EDGE ELIMINATION
  300 IF (K3.LT.1) GO TO 400
      DO 310 K = 1, K3
      IF (TRACE) CALL DUMP
C     GET EDGE
      READ(5,FREE,END=999)  EDGE
      IF (TRACE.OR.OPTION.EQ.4) WRITE(6,8) EDGE
    8 FORMAT(' EXCLUDE EDGE :          ',I5)
      STACK(K)     = EDGE
      EDGE = EDGE + L2
      IF (EDGE.LT.M0.OR.EDGE.GT.M2.OR.INDEX(EDGE).EQ.NULL) CALL ERRO
     &                R(7)
C     UPDATE POINTERS
      CALL NMATCH(EDGE)
      IND            = INDEX(EDGE)
      LIST(IND)      = NULL
      INDEX(EDGE)    = NULL
      M              = M - 1
C     CHECK LINK/MATE POSITIONS
      DO 310 PT = P0, P2
            IF (MATE(PT).EQ.EDGE.OR.MATE(PT).EQ.M9-EDGE) MATE(PT) = N
     &                ULL
  310 CONTINUE
      IF (PRINT) WRITE(6,9) (STACK(K),HEAD(STACK(K)),HEAD(M9-STACK(K
     &                )),PRICE(STACK(K)),K=1,K3)
    9 FORMAT(//' EDGE EXCLUSION (EDGE,NODE1,NODE2,COST):'/5(I9,2I3,F
     &                6.1))
C
C     NODE ELIMINATION
  400 IF (K4.LT.1) GO TO 500
      DO 420 K = 1, K4
      IF (TRACE) CALL DUMP
C     GET NODE
      READ(5,FREE,END=999) NODE
      IF (TRACE.OR.OPTION.EQ.4) WRITE(6,11) NODE
   11 FORMAT(' EXCLUDE NODE :          ',I5)
      STACK(K) = NODE
      IF (NODE.LT.N0.OR.NODE.GT.N2.OR.INDEX(NODE).EQ.NULL) CALL ERRO
```

```
     &              R(6)
      CALL OPEN(NODE)
      DO 410 IND = M0, M1
           EDGE = LIST(IND)
           IF (HEAD(EDGE).NE.NODE.AND.HEAD(M9-EDGE).NE.NODE) GO TO 4
     &              10
           IF (TRACE) WRITE(6,8) EDGE
           CALL NMATCH(EDGE)
           M            = M - 1
           INDEX(EDGE) = NULL
           LIST(IND)   = NULL
      DO 410 PT = P0, P2
           IF (MATE(PT).EQ.EDGE.OR.MATE(PT).EQ.M9-EDGE) MATE(PT) = N
     &              ULL
  410      CONTINUE
      VALUE          = VALUE - PRICE(NODE)
      IND            = INDEX(NODE)
      LIST(IND)      = NULL
      INDEX(NODE)    = NULL
  420 N              = N - 1
      IF (PRINT) WRITE(6,12) (STACK(K),SET(STACK(K)),K=1,K4)
   12 FORMAT(//' NODE EXCLUSION (NODE,SET):'/10(I10,I2))
C
C     NODE INTRODUCTION
  500 IF (K5.LT.1) GO TO 600
      DO 510 K = 1, K5
      IF (TRACE) CALL DUMP
C     GET NODE AND SUBSET
      READ(5,FREE,END=999)    NODE, NSET
      IF (TRACE.OR.OPTION.EQ.4) WRITE(6,13) NODE, NSET
   13 FORMAT(' INCLUDE NODE :          ',I4,I2)
      STACK(K)       = NODE
      IF (NODE.LT.N0.OR.NODE.GT.N2.OR.INDEX(NODE).NE.NULL) CALL ERRO
     &              R(6)
      IF (NSET.LT.0.OR.NSET.GT.3) CALL ERROR(3)
      IF (N1.GE.N2) CALL ERROR(2)
C     UPDATE POINTERS
      N              = N  + 1
      N1             = N1 + 1
      INDEX(NODE)    = N1
      LIST(N1)       = NODE
      SET(NODE)      = NSET
      LABLO(NODE)    = UNLABD
      PSEUDO(NODE)   = NODE
      PRED(NODE)     = NULL
      APEX(NODE)     = NODE
      MPRICE(NODE)   = 0
      IF (NSET.EQ.SET1) MPRICE(NODE) = INF
      MATE(NODE)     = NULL
      BCI(NODE)      = NODE
      IF (NSET.EQ.SET1) BCI(NODE) = L2
      PRICE(NODE)    = 0
      IF (SET(NODE).EQ.SET1.OR.SET(NODE).EQ.SET2) PRICE(NODE)=-INF/2
      VALUE          = VALUE + PRICE(NODE)
      LINK(NODE)     = NULL
  510 HEAD(NODE)     = NULL
      IF (PRINT) WRITE(6,14) (STACK(K),SET(STACK(K)),K=1,K5)
   14 FORMAT(//' NODE INCLUSION (NODE,SET):'/10(I10,I2))
C
C     EDGE INTRODUCTION
```

```
  600 IF (K6.LT.1) GO TO 700
      DO 610 K = 1, K6
      IF (TRACE) CALL DUMP
C     GET EDGE, NODES, AND EDGE COST
      READ(5,FREE, END=999)  EDGE, NODE1, NODE2, NPRICE
      IF (TRACE.OR.OPTION.EQ.4) WRITE(6,15)  EDGE,NODE1,NODE2,NPRICE
   15 FORMAT(' INCLUDE EDGE :           ',3I5,F7.1)
      STACK(K) = EDGE
      EDGE = EDGE + L2
      IF (EDGE .LT.M0.OR.EDGE .GT.M2.OR.INDEX(EDGE ).NE.NULL)  CALL E
     &              RROR(7)
      IF (NODE1.LT.N0.OR.NODE1.GT.N2.OR.INDEX(NODE1).EQ.NULL)  CALL E
     &              RROR(6)
      IF (NODE2.LT.N0.OR.NODE2.GT.N2.OR.INDEX(NODE2).EQ.NULL)  CALL E
     &              RROR(6)
      IF (NPRICE.GT.INF.OR.NPRICE.LT.-INF)                     CALL E
     &              RROR(9)
      IF (M1.GE.M2) CALL ERROR(4)
  610 CALL INCLUD(EDGE,NODE1,NODE2,NPRICE)
      IF (PRINT) WRITE(6,16)  (STACK(K),HEAD(STACK(K)),HEAD(M9-STACK(
     &              K)),PRICE(STACK(K)),K=1,K6)
   16 FORMAT(//' EDGE INCLUSION (EDGE,NODE1,NODE2,COST):'/5(I9,2I3,F
     &              6.1))
C
C     RE-INITIALIZE VARIABLES
  700 ROOT$ = 0
      NODE$ = 0
      EDGE$ = 0
      SHRK$ = 0
      UNSH$ = 0
      AUGM$ = 0
      DUAL$ = 0
      LABL$ = 0
      RMAT$ = 0
      HSCAN = 0
      TSCAN = 0
C
C     UPDATE LIST OF VALID NODE/EDGES
C     PLANT ALTERNATING FOREST
      I  = N0
      N1 = N
      DO 720 IND = N0, N1
  710             NODE = LIST(I)
                  I    = I + 1
             IF (NODE.EQ.NULL) GO TO 710
             LIST(IND)     = NODE
             INDEX(NODE)   = IND
             PNODE         = PSEUDO(NODE)
             LABLO(PNODE)  = UNLABL
             HEAD(PNODE)   = NULL
             PRED(PNODE)   = NULL
             TREE(PNODE)   = NULL
             IF (MATE(NODE).NE.NULL) GO TO 720
             IF (SET(NODE).EQ.SET0)  GO TO 720
             IF (SET(NODE).EQ.SET2.AND.PRICE(NODE).GE.-TOLER)  GO TO 72
     &              0
             IF (NODE.NE.APEX(PNODE)) GO TO 720
             IF (SET(NODE).EQ.SET3.AND.PRICE(NODE).GE.MPRICE(NODE)-TOL
     &              ER) GO TO 720
C            EXPOSED NODE
```

```
          LABLO(PNODE)  = OUTER
          HEAD(PNODE)   = NODE
          PRED(PNODE)   = NULL
          LINK(PNODE)   = NULL
          ROOT$         = ROOT$ + 1
          HSCAN         = HSCAN + 1
          IF (HSCAN.GT.N9) HSCAN = 1
          NSCAN(HSCAN) = PNODE
  720     CONTINUE
      I  = M0
      M1 = M0 + M - 1
      DO 740 IND = M0, M1
  730            EDGE1 = LIST(I)
                 I     = I + 1
          IF (EDGE1.EQ.NULL) GO TO 730
          LIST(IND)     = EDGE1
  740     INDEX(EDGE1) = IND
      IF (TRACE) CALL DUMP
      LABL$ = ROOT$
      IF (PRINT) WRITE(6,17) ROOT$
   17 FORMAT(/' ALTERNATING FOREST PLANTED WITH',I4,' ROOTS')
      DELTA = 0
      IF (OPTION.EQ.6) CALL DUMP
      CALL TIME(1,0,TIME1)
      IF (OPTION.NE.-ROOT$) RETURN
      TRACE = .TRUE.
      CALL DUMP
      RETURN
C
C     ERROR MESSAGE
  999 CALL ERROR(1)
 1000 STOP
      END
```

```
      SUBROUTINE INCLUD(EDGE1,NODE1,NODE2,NPRICE)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C
C     /INCLUDE EDGE1 IN THE NETWORK/
C     NODE1 -- EDGE1 --> NODE2
C     PRICE(EDGE1) = NPRICE
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
      LOGICAL FREE(1) /'*'/
      REAL COST, NPRICE
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) EDGE1, NODE1, NODE2, NPRICE
    1 FORMAT('  #TRACE# INCLUD (',3(I4,','),F6.1,')')
C
C     UPDATE VARIABLES
      M            = M  + 1
      M1           = M1 + 1
      LIST(M1)     = EDGE1
      INDEX(EDGE1) = M1
      EDGE2        = M9 - EDGE1
      HEAD(EDGE1)  = NODE1
      HEAD(EDGE2)  = NODE2
      PRICE(EDGE1) = NPRICE
      COST         = PRICE(EDGE1) - PRICE(NODE1) - PRICE(NODE2)
      IF (COST.GE.0) RETURN
C
C     EDGE IS OUT-OF-KILTER
C     TRY TO DECREASE THE PRICE OF NODE1
      CALL OPEN(NODE1)
      CALL DECRSE(EDGE1,COST)
      IF (COST.GE.0) RETURN
C     TRY TO DECREASE THE PRICE OF NODE2
      CALL OPEN(NODE2)
      CALL DECRSE(EDGE2,COST)
      IF (COST.LT.-TOLER) CALL MATCH(EDGE1)
      RETURN
```

```
C
  999 CALL ERROR(3)
      RETURN
      END
```

```
      SUBROUTINE DECRSE(EDGE,COST)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /DECREASE PRICE(NODE) UNTIL COST = ZERO /
C     NODE -- EDGE -->
C
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABEL(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
      REAL COST
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) EDGE, COST
    1 FORMAT('  #TRACE# DECRSE (',I4,',',F6.1,')')
C
C     GET NODE
      NODE = HEAD(M9-EDGE)
C     NODE IN SET0 CANNOT HAVE ITS PRICE DECREASED
      IF (SET(NODE).EQ.SET0) RETURN
C     NODE IN SET3 WITH NULL PRICE CANNOT HAVE ITS PRICE DECREASED
      IF (SET(NODE).EQ.SET3.AND.PRICE(NODE).LT.TOLER) RETURN
C     DECREASE NODE PRICE
      PRICE(NODE) = PRICE(NODE) + COST
      VALUE       = VALUE + COST
      COST        = 0
      IF (SET(NODE).NE.SET3.OR.PRICE(NODE).GE.-TOLER) RETURN
C             EDGE OF A-
              COST        = PRICE(NODE)
              PRICE(NODE) = 0
              VALUE       = VALUE - COST
              RETURN
      END
```

```
      SUBROUTINE INCRSE(NODE)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /INCREASE PRICE(NODE) UP TO ZERO/
C     CHECK EDGES INCIDENT WITH NODE
C
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150),MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
      REAL COST
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) NODE
    1 FORMAT('  #TRACE# INCRSE (',I4,' )')
C
C     INCREASE PRICE(NODE) TO ZERO
      VALUE       = VALUE - PRICE(NODE)
      PRICE(NODE) = 0
C
C     CHECK EDGES INCIDENT WITH NODE
      DO 10 IND = M0, M1
            EDGE  = LIST(IND)
            NODE1 = HEAD(M9-EDGE)
            NODE2 = HEAD(EDGE)
            IF (NODE.NE.NODE1.AND.NODE.NE.NODE2) GO TO 10
            COST = PRICE(EDGE) - PRICE(NODE1) - PRICE(NODE2)
            IF (COST.GE.-TOLER) GO TO 10
C                 NEGATIVE COST
                  NODE0 = NODE1 + NODE2 - NODE
                  CALL OPEN(NODE0)
                  CALL DECRSE(EDGE,COST)
                  IF (COST.LT.-TOLER) CALL MATCH(EDGE)
   10       CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OPEN(NODE)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C
C     MINIMUM COST 1-MATCHING/COVERING PROBLEM
C     THE UNIVERSITY OF MICHIGAN
C     WRITTEN BY CLOVIS PERIN
C     AUGUST 1980
C
C     /OPEN PSEUDO(NODE)/
C     UNSHRINK PSEUDO(NODE) OR UNMATCH MATE(NODE)
C
      IMPLICIT INTEGER (A-Z)
      REAL VALUE,DELTA,MCOST,TOLER,INF,PRICE,MPRICE
      LOGICAL TRACE,PRINT
      COMMON /NODES/ SET(100),LABLO(150),PSEUDO(150),APEX(150),PRED(
     &               150),TREE(150,,MPRICE(150),BCI(150)
      COMMON /EDGES/ HEAD(6150),PRICE(3150),LIST(3150),INDEX(3150)
      COMMON /LISTS/ NSCAN(101),INSP(3100),NORIG(100),LINK(1500),MAT
     &               E(1500)
      COMMON /PNTRS/ HSCAN,TSCAN,HINSP,HORIG,HINK
      COMMON /PARMS/ N,N0,N1,N2,N9,L,L0,L1,L2,M,M0,M1,M2,M9,P0,P2
      COMMON /VARBS/ VALUE,DELTA,MCOST,OPTION,TRACE,PRINT
      COMMON /STATS/ ROOT$,NODE$,EDGE$,SHRK$,UNSH$,AUGM$,DUAL$,RMAT$
     &               ,LABL$,LINK$,TIME1,TIME2,TIME3
      COMMON /CONTS/ TOLER,INF,OUTER,INNER,UNLABD,SET0,SET1,SET2,SET
     &               3,NULL
      COMMON /PRNTS/ STACK(3000),PACK(100)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (TRACE) WRITE(6,1) NODE
    1 FORMAT('  #TRACE# OPEN    (',I4,')')
C
C     BRANCH IF NODE IS NONCURRENT
      IF (PSEUDO(NODE).NE.NODE) GO TO 10
C           NODE IS CURRENT
            IF (SET(NODE).EQ.SET0) RETURN
            IF (SET(NODE).EQ.SET3.AND.PRICE(NODE).EQ.0) RETURN
C           UNMATCH MATE(NODE)
            EDGE = MATE(NODE)
            CALL NMATCH(EDGE)
            RETURN
C
C     UNSHRINK PSEUDO(NODE)
   10 PNODE = PSEUDO(NODE)
      EDGE  = MATE(APEX(PNODE))
C     UNMATCH MATE(NODE)
      CALL NMATCH(EDGE)
      IF (PNODE.EQ.NODE) RETURN
      IF (TRACE) WRITE(6,2) PNODE
    2 FORMAT('  #TRACE# OPEN PSEUDO :       ',I4)
C
C     REDUCE NODE PRICES
      HORIG = 0
      CALL ORIG(PNODE)
      DO 20 H = 1, HORIG
            NODE2 = NORIG(H)
            IF (NODE2.GT.N2) CALL ORIG(NODE2)
            IF (NODE2.LE.N2) PRICE(NODE2) = PRICE(NODE2) - PRICE(PNOD
```

```
      &                E)
   20       CONTINUE
      VALUE = VALUE - PRICE(PNODE)
C
C     UPDATE MATES, APICES, HEADS, AND PREDS
      BASE   = MATE(PNODE)
      CALL STPSDO(BASE)
      LABLO(BASE) = UNLABD
      BLSM = HEAD(BASE)
   30       EDGE1          = PRED(BLSM)
            EDGE2          = M9 - EDGE1
            NODE2          = HEAD(EDGE1)
            NODE1          = HEAD(EDGE2)
            MATE(NODE1)    = EDGE1
            MATE(NODE2)    = EDGE2
            BLSM1          = HEAD(BLSM)
            CALL STPSDO(BLSM)
            CALL STPSDO(BLSM1)
            LABLO(BLSM)    = UNLABD
            LABLO(BLSM1)   = UNLABD
            CALL STAPEX(NODE1)
            CALL STAPEX(NODE2)
            HEAD(BLSM)     = 0
            PRED(BLSM)     = 0
            BLSM           = HEAD(BLSM1)
            HEAD(BLSM1)    = 0
            PRED(BLSM1)    = 0
      IF ( BLSM .NE. BASE ) GO TO 30
      HEAD(BASE) = 0
      PRED(BASE) = 0
C
C     UPDATE PSEUDONODE LIST
      IND                      = INDEX(PNODE)
      LIST(IND)                = LIST(L1)
      LIST(L1)                 = PNODE
      INDEX(LIST(IND))         = IND
      INDEX(PNODE)             = L1
      L1                       = L1 - 1
      GO TO 10
C
      END
```