Paper 1-2

AN ALGORITHM TO CONSTRUCT A MINIMUM DIRECTED SPANNING TREE IN A DIRECTED NETWORK

FREDERICK BOCK

IIT Research Institute, Chicago, Illinois, U.S.A.

ABSTRACT

The problem this algorithm solves is: Given a network composed of nodes and directed links, each link having a numerical cost, and given the identity of a node selected as origin, find (if one exists) a subset of links with minimum total cost that forms a directed tree connecting the origin to all other nodes. The algorithm exploits the relationships between the primal and dual mathematical programming formulations of the problem to achieve an efficient solution procedure. If the given cost data are integers, then all calculations are in integers. The method is similar in several respects to the Hungarian method for the classical assignment problem. Affinities exist between this and various other network optimization problems, e.g., finding a minimum spanning tree in an undirected network (a special case of the present problem), the shortest path problem, the traveling salesman problem. The algorithm has been programmed in the ALGOL language and run on a digital computer. Areas of potential application include the design of economical transportation and communication systems and the planning of strategic moves with multiple objectives in a space-time framework.

1 The Problem

A verbal statement of the problem this algorithm solves is: Given a network composed of a set of nodes and a set of directed links, each link having a specified initial node, final node, and nonnegative numerical value, and given the identity of a node selected as origin, find a subset of the links that provides a path connecting the origin to each of the other nodes of the network and has the property that the sum of the link values is a minimum. Such a structure will be called a minimum directed spanning tree. The matrix of link values may be symmetric or asymmetric and may be complete or incomplete. For a particular origin in a particular network a spanning directed tree may or may not exist. An alternative goal, if no directed tree from the origin spans the entire set of nodes, can be to construct a tree that includes as many nodes as possible and has a minimum sum of link values.

Using mathematical programming notation, this primal problem and its corresponding dual can be formulated as follows in terms of initial data, constraints, and objective functions. Let S be the set of positive integers 1, ..., n for specified n, let S_k be a proper subset of S having n_k elements, and let Q be a set of indices of all the proper subsets of S. Then the initial data for an example of the primal or dual problem are the given nonnegative integers

$$\begin{array}{ll} n & \\ c_{ij} & i,j \in S, \\ j_0 & 1 \leqq j_0 \leqq n. \end{array}$$

Primal Problem

Find Boolean numbers

$$x_{ij} \qquad i,j \in S \qquad x_{ij} \in \{0, 1\}.$$

Satisfy

$$\sum_{i \in S} x_{ij} \geqq 1 \qquad j \in S, \qquad j \neq j_0, \tag{1}$$

$$\sum_{i,j \in S_k} x_{ij} \leqq n_k - 1 = b_k \qquad k \in Q. \tag{2}$$

Minimize

$$z = \sum_{i,j \in S} c_{ij} x_{ij}. \tag{3}$$

Dual Problem

Find nonnegative integers

$$u_j \qquad j \in S, \quad j \neq j_0,$$
$$u_k \qquad k \in Q.$$

Satisfy

$$u_j - \sum_{k \in Q} a_{ijk} u_k \leqq c_{ij} \qquad i, j \in S. \tag{4}$$

Maximize

$$w = \sum_{\substack{j \in S \\ j \neq j_0}} u_j - \sum_{k \in Q} b_k u_k. \tag{5}$$

The indices i and j denote the initial and final nodes, respectively, of the directed links ij of a given network. Equivalently, i and j are row and column indices of a square matrix. The c_{ij} are the link values. The origin node is denoted by j_0.

The primal activity variables x_{ij} have the value 1 if the corresponding link is in the solution set and 0 if not. Primal constraints (1) require that at least one link must lead to every node except the origin, while primal constraints (2) require that no subset of links forms a circuit. The primal objective function (3) expresses the quantity, z, to be minimized, i.e., the sum of the values of the links in the solution set.

In the dual problem the dual activity variables u_j correspond to primal constraints (1) and the dual activity variables u_k correspond to the primal constraints (2). The dual constraints (4) require that a linear combination of the values of the dual activity variables applying to each link ij of the given network does not exceed the link value c_{ij}. The coefficients a_{ijk} have the value 1 if both $i \in S_k$ and $j \in S_k$, and have the value 0 otherwise. The dual objective function (5) expresses the quantity, w, to be maximized subject to the dual constraints.

To convert inequalities to equalities and to provide a measure of compliance (infeasibility, slack, or exact satisfaction) for each of the primal and dual constraints of the problem at any stage of the solution process, subtract primal compliance variables y_j from the left sides of the primal column constraints (1), add y_k to the primal submatrix constraints (2), and add dual compliance variables v_{ij} to the left sides of the dual constraints (4). These are called compliance variables in contrast to the primal and dual activity variables, x_{ij}, u_j, and u_k, of the inequality constraints. If a compliance variable has a positive value, then the constraint in which it appears is more than satisfied or is "slack." If its value is negative, then the constraint is not satisfied, i.e., there is infeasibility with respect to that constraint. If its value is 0, then the constraint is just satisfied and may be called limiting.

The optimum solution has been attained when all primal and dual variables, measuring both activity levels and degrees of compliance, have nonnegative values and the cross products of the corresponding primal and dual variables, i.e., $v_{ij}x_{ij}$, u_jy_j, and u_ky_k are zero. Under these conditions the primal and dual objective variables, z and w, have the same value.

2 The Algorithm

Algorithm 1, stated in the appendix in the ALGOL language [1], exploits the primal-dual relationships in the solution process. The algorithm has been compiled on a digital computer and tested on a number of examples ranging in size up to 42 nodes.

The notation of Algorithm 1 is as follows:

Scalars

EX	Number identifying the example to be solved.
N	The number of rows and columns in the data matrix of the example (number of nodes in the given network).
I	Row index (index of the initial nodes of directed links).
J	Column index (index of the final nodes of directed links).
JO	Index of the column of the matrix corresponding to the node of origin.
I1, J1, I2, J2	Indices of particular cells of the matrix that are candidates for placement or transfer of a star or bar.
K	Index of the highest-numbered column so far treated in the solution process.
H, H1	Particular values of SPAN [J] (see below).
SS	The number of submatrices so far encountered that are spanned by critical elements.
M	An arbitrarily large integer indicating infeasibility of an element having that value.
⊙	Zero

DU	The largest increment that can be made in values of the dual variables U1 [J] while preserving dual feasibility.
Z	The value of the solution, i.e., $\sum c_{ij}x_{ij}$.

Vectors

U1 [J]	Dual activity variables applying to the primal constraints (1) requiring that a link of the solution set lead to each node other than the origin.
ISTAR [J]	Index of the row of a starred cell in column J of the matrix.
IBAR [J]	Index of the row of a barred cell corresponding to a starred cell in column J of the matrix.
JBAR [J]	Index of the column of a barred cell corresponding to a starred cell in column J of the matrix.
SPAN [J]	Index of the successive submatrix constraints applied during the solution process; the index of the latest such constraint is given for each column intersected by the submatrix.

Matrix

C [I, J]	The given matrix of link values.

The method of constructing the minimum directed spanning tree that is implemented in Algorithm 1 is analogous in several respects to one version of the Hungarian method for constructing the minimum assignment [2, 3].

The basic theorems on which the Hungarian method for the assignment problem and the method presented here for the directed-spanning-tree problem rest can be stated in parallel for the purpose of comparison as follows.

FUNDAMENTAL THEOREMS UNDERLYING ALGORITHMS FOR THE ASSIGNMENT AND DIRECTED-SPANNING-TREE PROBLEMS

Given a square matrix with elements that are either zero or nonzero

MAXIMUM ASSIGNMENT
MINIMUM LINE COVER

The maximum number of zero elements such that no two are in the same row or the same column is the same as the minimum number of lines (rows and/or columns) that contain all the zeros.

MAXIMUM TREE
MINIMUM SUBMATRIX COVER

The maximum number of zero elements such that no subset forms a loop is one less than the order of the minimum square submatrix that contains all the zeros.

A proof for the second theorem, upon which Algorithm 1 is based, consists in the necessity of occurrence of at least one loop in any directed network composed of n different nodes and n different links.

Characteristics common to the two algorithms include the following.

1) One of these two possibilities is realized in a finite number of steps: (a) a subnetwork providing an optimum solution is constructed; (b) the nonexistence of a subnetwork satisfying all primal constraints is demonstrated.

2) The computations are in integers throughout.

3) The methods are dual feasible, i.e., the dual constraints are always satisfied.

4) The methods are primal feasible with respect to one set of primal constraints—the row constraints in the case of the assignment problem and the submatrix constraints in the case of the directed-spanning-tree problem.

5) The construction of the solution subnetwork is carried forward starting with the examination of column 1 of the C matrix and proceeding column by column. The column constraints for the first K columns are always satisfied before proceeding to column $K + 1$.

6) The dual variables are utilized to restrict the number of matrix elements considered for inclusion in the solution subnetwork. In the first place, elements ij are never actively considered unless the dual constraints are actually limiting, i.e., $v_{ij} = 0$. Further, the search for a primal improvement is restricted to certain of these limiting elements, called *critical* elements. At any stage the critical elements are marked by figuratively placing a star or a bar in the corresponding cell of the matrix. A star is equivalent to the condition $x_{ij} = 1$, absence of a star to the condition $x_{ij} = 0$. A bar indicates that x_{ij} can potentially take on the value 1 in a sequence of transfers of stars to bars and bars to stars. Together, the sequences of possible transfers form a forest of directed transfer trees. In each transfer tree barred and starred elements alternate along any path.

The barred elements can be further classified on the basis of their location in a transfer tree:

Sink	Has a preceding but not a succeeding star
Source	Has a succeeding but not a preceding star
Join	Has both a preceding and a succeding star
Source-Sink	Has neither a preceding or a succeeding star

Finding a primal improvement is then equivalent to finding a path through a transfer tree from a source to the sink, since by changing bars to stars and stars to bars in such a path the total number of stars is increased by one.

The directed-spanning-tree algorithm differs from the assignment algorithm in the nature of the constraints: both have column constraints but in addition the former has submatrix constraints while the latter has row constraints. The critical elements of the directed-spanning-tree algorithm emerge in the search for a set of stars, one in each column except the origin column, with the property that no subset of starred elements forms a circuit. New limiting elements are generated by a dual step in which the columnwise complement of the smallest submatrix that (1) is intersected by column K, and (2) contains an isolated portion of the directed spanning tree at the current stage of construction, is searched to find the smallest $v_{ij} = DU$. Such a submatrix is called a critical submatrix. $DU \geqq 0$ is then added to u_j for each column intersecting that submatrix and also becomes the value of u_k for the submatrix itself. The first limiting element found in the columnwise complement of the submatrix is treated as a new critical element and is barred. If it is a source or a source-sink with respect to the transfer tree, it is starred and interchanges of stars and bars in the path through the transfer tree are made, after which the next column of the matrix, if any, is examined. On the other hand, if the new critical element is in column K and is not a source-sink, then it becomes a sink in a transfer tree. The third possibility is that the new critical element is not in column K and is not a source; in this case it enters a transfer tree as a join.

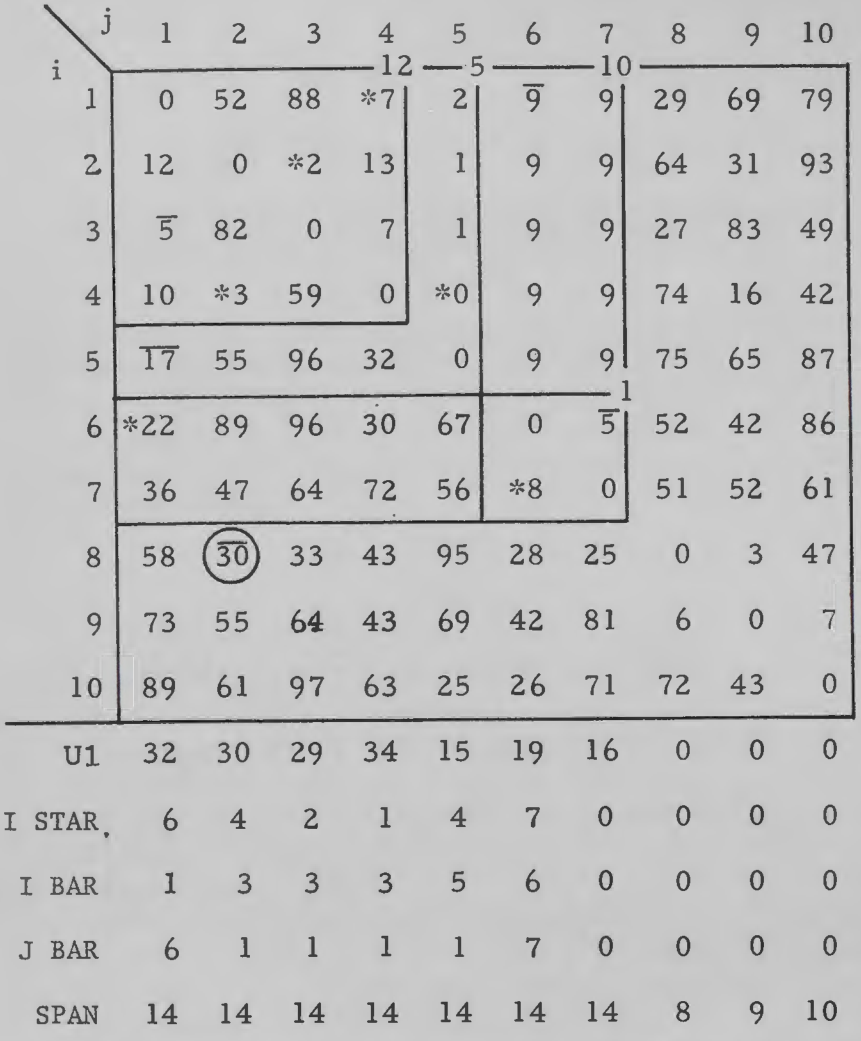
The computational power of the method comes from the fact that only a small number of critical submatrices out of the large number *a priori* are ever actually encountered. The maximum number of critical submatrices of size $n_k > 1$ is $n - 2$. There can also be, trivially, a maximum of n critical submatrices of size $n_k = 1$. These are the elements on the main diagonal of the matrix and correspond to links in which the initial and final nodes are identical. In Algorithm 1 these diagonal elements are eliminated from consideration by initially setting SPAN [J] = J for J = 1, ..., N. Every pair of critical submatrices is either disjoint or nested, i.e., no pair of critical submatrices is partially overlapping. Furthermore, every critical submatrix remains critical throughout the computation, and the value of u_k for a critical submatrix, once assigned, is never changed. In exact correspondence to each critical submatrix is one barred critical element.

Critical elements occur in a transfer tree according to these rules: (1) An arrow leads from a barred element to the starred element, if any, in the same column (forces compliance with column constraints). (2) An arrow leads from a starred element to the barred element that (a) is contained in the same critical submatrix, if any, but that (b) is not contained in any smaller critical submatrix (forces compliance with submatrix constraints).

3 An Example

An asymmetrical network example with $n = 10$ nodes is shown in Fig. 1. The numbers in the rows and columns of the body of the matrix are the c_{ij} values. Node $j_0 = 10$ is the origin. An intermediate stage in the construction of the minimum directed spanning tree is presented numerically in Fig. 1 and graphically in Fig. 2. Four circuits formed by critical elements have been encountered up to this point as indicated by the four outlined submatrices lying on the main diagonal of the matrix. Each such submatrix contains one bar not contained in a smaller critical submatrix. (The barred elements of Fig. 1 are represented as dashed arrows in Fig. 2). In Fig. 1 the submatrix dual quantities u_k appear at the upper right corners of the critical submatrices. These quantities are not explicitly provided by Algorithm 1. At the computational stage shown in Fig. 1 the circled element (8, 2) has just been shown to be limiting in the columnwise complement of the submatrix encompassing nodes 1 through 7, and this element has been barred as the new critical element of the matrix.

The transfer tree formed by the critical elements of Fig. 1 (and Fig. 2) is shown in Fig. 4. There is an alternation of stars and bars at successive levels of the tree, as there must be to provide a chain of possible transfers. The three types of barred elements in Fig. 4—sink, join, source—are so labeled. A sink is always in column K; here $K = 7$. The source in Fig. 4 is the circled element in Fig. 1, the newly found critical element (8, 2). Starting with ele-



i \ j	1	2	3	4	5	6	7	8	9	10
				12	5		10			
1	0	52	88	*7	2	$\bar{9}$	9	29	69	79
2	12	0	*2	13	1	9	9	64	31	93
3	$\bar{5}$	82	0	7	1	9	9	27	83	49
4	10	*3	59	0	*0	9	9	74	16	42
5	$\overline{17}$	55	96	32	0	9	9	75	65	87
							1			
6	*22	89	96	30	67	0	$\bar{5}$	52	42	86
7	36	47	64	72	56	*8	0	51	52	61
8	58	$\textcircled{\overline{30}}$	33	43	95	28	25	0	3	47
9	73	55	64	43	69	42	81	6	0	7
10	89	61	97	63	25	26	71	72	43	0
U1	32	30	29	34	15	19	16	0	0	0
I STAR	6	4	2	1	4	7	0	0	0	0
I BAR	1	3	3	3	5	6	0	0	0	0
J BAR	6	1	1	1	1	7	0	0	0	0
SPAN	14	14	14	14	14	14	14	8	9	10

Fig. 1. Intermediate stage in the construction of a minimum directed spanning tree in an asymmetrical 10-node network by means of Algorithm 1. Node 10 is the origin.

ment (8, 2) as source, a chain of transfers can be made along the unique path through the tree leading to element (6, 7) as sink. To accomplish the transfer, every bar in the path is replaced by a star and every star in the path is replaced by a bar. There is thus a net increase of one star; in other words,

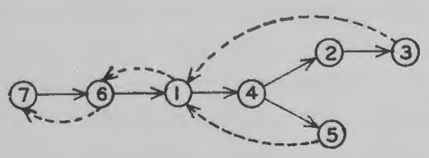


Fig. 2

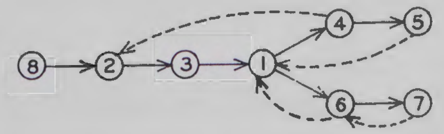


Fig. 3

Fig. 2. The subnetwork corresponding to the starred links (solid arrows) and the alternative barred links (dashed arrows) interconnecting nodes 1 through 7 in Fig. 1.

Fig. 3. The subnetwork corresponding to the starred and barred links of Fig. 1 after the transfer of bars and stars from source to sink through the transfer tree as indicated in Fig. 4 has been carried through.

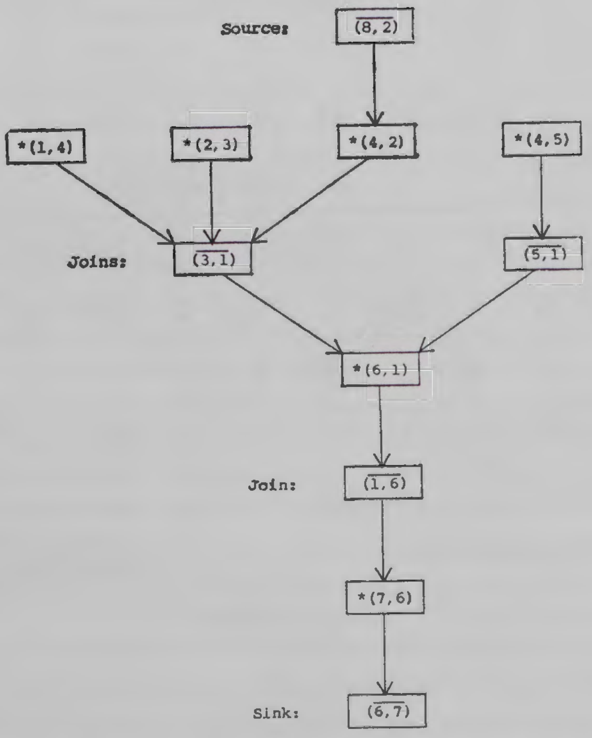


Fig. 4. The transfer tree implied by the pattern of starred and barred links in Fig. 1.

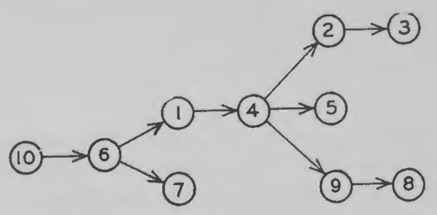


Fig. 5. Minimum directed spanning tree of the example of Fig. 1. The sum of link values is $z = 87$.

a step has been taken in the direction of primal feasibility while preserving dual feasibility. The stage in the construction of the minimum directed spanning tree reached as a result of this step is shown in Fig. 3.

In the present example two additional steps are required to complete the solution. The complete minimum directed spanning tree with a value $z = 87$ is shown in Fig. 5.

4 Discussion

The problem of finding the minimum directed spanning tree in a directed network is a generalization of the problem of finding a minimum spanning tree in an undirected network [4, 5]. The latter problem is equivalent to the former problem with a symmetrical matrix of link values. A particularly simple algorithm solves the undirected case since a shortest link connecting a subset of the nodes of the network to its complement will be in the minimum spanning tree.

The traveling salesman problem for an asymmetrical distance matrix can be formulated by combining the constraints of the assignment problem (eliminate forks in the tour) and the constraints of the present problem (eliminate sub-loops in the tour) while visiting every city.

Areas of potential application of the present algorithm include the design of economical transportation and communication systems and particularly, because of the inherent asymmetry of time, the planning of strategic or tactical moves with multiple objectives in a space-time framework.

References

1. P. Naor (Ed.), Revised Report on the Algorithmic Language ALGOL 60, *Communications of the Association for Computing Machinery*, **6**, 1–17 (1963).
2. H. W. Kuhn, The Hungarian Method for the Assignment Problem, *Naval Research Logistics Quarterly*, **2**, 83–97 (1955).
3. H. W. Kuhn, Variants of the Hungarian Method for Assignment Problems, *ibid.*, **3**, 253–58 (1956).
4. J. B. Kruskal Jr., On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem, *Proc. Am. Math. Soc.*, **7**, 48–50 (1956).
5. R. C. Prim, Shortest Connection Networks and Some Generalizations, *Bell System Technical Journal*, **36**, 1389–1401 (1957).

APPENDIX

ALGORITHM 1. MINIMUM DIRECTED SPANNING TREE

```
BEGIN    INTEGER EX, N, I, J, J⊙, I1, J1, I2, J2, K, H, H1, SS, M,
         DU, Z;
         INTEGER ARRAY U1, I STAR, I BAR, J BAR,
         SPAN[1 : 5⊙], C[1 : 5⊙, 1 : 5⊙];
         PROCEDURE INPUT; PROCEDURE OUTPUT;
L1:      INPUT (EX, N, J⊙, M);
         FOR I := 1 STEP 1 UNTIL N DO FOR J := 1 STEP
         1 UNTIL N DO INPUT (C[I, J]);
         GO TO L97;
L2:      FOR J := 1 STEP 1 UNTIL N DO
             BEGIN U1[J] := I STAR[J] := I BAR[J] := J
                      BAR[J] := ⊙;
                      SPAN[J] := J
             END;
         SS := N; K := Z := ⊙;
L3:      IF K = N THEN GO TO L99; K := K + 1;
         IF K = J⊙ THEN GO TO L3;
COMMENT STATEMENTS L4 THROUGH L5 FIND A LEAST-COST
         (CANDIDATE) LINK IN THE COLUMNWISE COM-
         PLEMENT OF THE LARGEST SPANNED SUBMATRIX
         INTERSECTED BY COLUMN K;
L4:      DU := M; H := SPAN[K];
         FOR J := 1 STEP 1 UNTIL K DO IF SPAN[J] = H THEN
             FOR I := 1 STEP 1 UNTIL N DO
                 IF SPAN[I] ≠ H ∧ C[I, J] < M ∧ C[I, J]
                    − U1[J] < DU THEN
                    BEGIN DU := C[I, J] − U1[J];
                             I1 := I; J1 := J
                 END;
         IF DU = M THEN GO TO L98;
```

```
L5:          FOR J := 1 STEP 1 UNTIL K DO IF SPAN[J] = H THEN
                  U1[J] := U1[J] + DU;

COMMENT STATEMENTS L6 THROUGH L8 TRACE BACKWARD
             IN THE STARRED PARTIAL TREE LEADING TO THE
             CANDIDATE LINK UNTIL A CIRCUIT OR THE ROOT
             OF THE PARTIAL TREE IS FOUND;

L6:          J := I1;

L7:          IF SPAN[J] = H THEN
                BEGIN SS := SS + 1;
                        FOR J := 1 STEP 1 UNTIL K DO
                              IF SPAN[J] = H ∨ SPAN[J] < ⊙
                              THEN SPAN[J] := SS;

                        GO TO L4
                END;

L8:          IF I STAR[J] > ⊙ THEN
                BEGIN IF SPAN[J] > ⊙ THEN
                           BEGIN H1 := SPAN[J];
                                          FOR J2 := 1 STEP 1 UNTIL
                                                K DO
                                                IF SPAN[J2] = H1 THEN
                                                   SPAN[J2] :=
                                                   −SPAN[J2]
                           END;
                        IF I BAR[J] = ⊙ THEN
                           BEGIN I BAR[J] := I1; J BAR[J] := J1
                           END;
                        J := I STAR[J]; GO TO L7
                END;

COMMENT STATEMENTS L9 THROUGH L11 STAR THE CANDI-
             DATE LINK, MAKE NECESSARY TRANSFERS OF
             STARS AND BARS, AND UPDATE REFERENCES OF
             STARS TO BARS;
```

```
L9:       FOR J := 1 STEP 1 UNTIL K DO
              IF SPAN[J] < ⊙ THEN SPAN[J] := −SPAN[J];
          I2 := J2 := ⊙;
L1⊙:      FOR J := 1 STEP 1 UNTIL K DO
              IF I BAR[J] = I1 ∧ J BAR[J] = J1 THEN
                  BEGIN I BAR[J] := I2; J BAR[J] := J2
                  END;
          I := I BAR[J1]; J := J BAR[J1];
          I BAR[J1] := I2; J BAR[J1] := J2;
          I2 := I STAR[J1]; I STAR[J1] := I1;
          IF I2 = ⊙ THEN GO TO L3;
          J2 := J1; I1 := I; J1 := J;
L11:      GO TO L1⊙;
COMMENT   THE FOLLOWING STATEMENTS INDICATE WHETH-
          ER OR NOT THERE IS A FEASIBLE SOLUTION FOR
          THE DATA PROVIDED AND GIVE THE VALUES OF
          SPECIFIED VARIABLES AT THE POINT EITHER THAT
          INFEASIBILITY IS DETECTED OR THAT AN OPTI-
          MUM SOLUTION IS ATTAINED;
L97:      OUTPUT ('ALGORITHM 1, MINIMUM DIRECTED
          SPANNING TREE, EXAMPLE', EX, ',', N, 'NODES,
          ORIGIN NODE', J⊙); GO TO L2;
L98:      OUTPUT ('INFEASIBLE'); Z := M; GO TO L1⊙⊙;
L99:      FOR J := 1 STEP 1 UNTIL N DO IF J ≠ J⊙ THEN
              Z := Z + C[I STAR[J], J];
          OUTPUT ('OPTIMUM SOLUTION');
L1⊙⊙:     OUTPUT ('Z =', Z);
          OUTPUT ('J:'); FOR J := 1 STEP 1 UNTIL N DO
             OUTPUT(J);
          OUTPUT('U1[J]:'); FOR J := 1 STEP 1 UNTIL N DO
             OUTPUT(U1[J]);
          OUTPUT('I STAR[J]:'); FOR J := 1 STEP 1 UNTIL N
             DO OUTPUT(I STAR[J]);
          OUTPUT('I BAR[J]:'); FOR J := 1 STEP 1 UNTIL N
             DO OUTPUT(I BAR[J]);
```

```
            OUTPUT('J BAR[J]:'); FOR J := 1 STEP 1 UNTIL N
              DO OUTPUT(J BAR[J]);
            OUTPUT('SPAN[J]:'); FOR J := 1 STEP 1 UNTIL N DO
              OUTPUT(SPAN[J])
END ALGORITHM 1
```