MATHEMATICA – REVUE D'ANALYSE NUMÉRIQUE
ET DE THÉORIE DE L'APPROXIMATION
L'ANALYSE NUMÉRIQUE ET LA THÉORIE DE L'APPROXIMATION
Tome 14, N° 2, 1985, pp. 117–121

ON FINDING THE ELEMENTARY PATHS OF A DIGRAPH

DĂNUȚ MARCU
(București)

Abstract. In this paper we suggest a FORTRAN program for finding all the elementary paths between any two specified nodes of a digraph.

Introduction. One of the most important concepts of a graph theory, and one which appears often in areas superficially unrelated to graphs, is that of path. In some situations, it is necessary to be able to generate a complete list of all the elementary paths of a digraph (directed graph). This may, for example, be the case when the "best" path needs to be chosen, but the criterion to be used for deciding what path is "best" is very delicate (one could decide on the best path to choose using also criteria which are either difficult to incorporate directly into an algorithm or which are subjective in nature) so that direct optimization (not involving complete enumeration) is impractical. In other situations, the manipulations could be simplified by first generating all the elementary paths.

The algebraic methods which have appeared in the literature [2, 3, 4, 9] for determining elementary paths are theoretically interesting but are to loose to be of value for arbitrary graphs encountered in practice, they being unable to deal with problems of more than a few tens of vertices because of the large amounts of both computer memory and time (e.g., see the program LATINE from [4]).

A slight modification of these methods (e.g., see [1]) reduces both the storage and time requirements by large factors. However, even with these modifications, a computer program [3] written in PL/I (to take advantage of both the string manipulation and variable storage allocation facilities offered by this language) on an IBM 360/65 computer with a memory of 120,000 bits can be used only for graphs with at most 20 nodes and an average nod degree of more than 4. Moreover, even for the above-mentioned size graphs, the method used virtually all memory locations.

More successful are the enumerative techniques (e.g., see [1, 8]) which do not have large computer storage requirements but whose time requirement still increases exponentially with the number of nodes in the graph.

In this note, we give a FORTRAN program which generates all the elementary paths between any two specified nodes of a digraph, using the nodes' number and the adjacency matrix.

Description of the method. Let $D = (V, E)$ be a *digraph* [1] (directed graph) without loops (a *loop* is an arc whose initial and final nodes are the same) having V ($|V| = n$) the set of *nodes* (vertices), E the set of *arcs* (edges), and $A = (a_{ij})$, $i, j = 1, 2, \ldots, n$ the *adjacency matrix*, where

$$a_{ij} = \begin{cases} 1, \text{ if } (v_i, v_j) \in E, \\ 0, \text{ otherwise.} \end{cases}$$

A *path* in a digraph is any sequence of arcs where the final node of one is the initial node of the next one. An *elementary* path is a path which does not use the same node more that once.

Fora digraph we consider the *path matrix* $P = (p_{ij})$, $i, j = 1, 2, \ldots, n$, where

$$p_{ij} = \begin{cases} 1, \text{ if there exists a path from } v_i \text{ to } v_j. \\ 0, \text{ otherwise} \end{cases}$$

and let v_i, v_j two distinct nodes of D such that $p_{ij} \neq 0$. Obviously, (v_k, v_j) is the last arc of a path from v_i to v_j if and only if

(1) $$p_{ik}\, a_{kj} = 1.$$

Thus, if the indices k for which (1) holds are $k_1, k_2, \ldots, k_t$ then (v_{k_s}, v_j), $s = 1, 2, \ldots, t$ are the last arcs of the paths from v_i to v_j. For every one of these indices we repeat the procedure, i.e., we find the last arcs of the form (w, v_{k_s}), belonging to the paths from v_i to v_{k_s}, by taking v_{k_s} in the role of v_j, and so on.

If (1) does not hold for any $k \neq i, j$, then the arc (v_i, v_j) is the single elementary path from v_i to v_j.

In this way, all the elementary paths from v_i to v_j can be generated (the method described above is summarized in the FORTRAN program below).

Let NOD1 and NOD2 be two distinct nodes of a digraph (with N nodes and MA its adjacency matrix). Following the driver program below, all the elementary paths from NOD1 to NOD2 are generated and listed (the subroutine PATH computes the path matrix MD, according to the algorithm described in [7]).

```
C     DRIVER PROGRAM
C
* DEFINE FILE*1 = 1, *2=3
      DIMENSION MA (6,6),MD(6,6), MS(6,6), IK(6), NS(6), NSL(6)
      READ (1,1) N, NOD1, NOD2
    1 FORMAT(3I2)
      READ(1,2) ((MA(I,J), I=1,N),J=1,N)
    2 FORMAT(36I1)
      GENERATION OF THE ELEMENTARY PATHS FROM NOD1
      TO NOD2
      CALL PATH(N,MA,MD)
      CALL PMS (N,MA,MD,MS,IK,NOD1,NOD2)
```



```
      CALL GEP(N,MS,NS,NSL,NOD1,NOD2)
      STOP
      END

      SUBROUTINE PATH (N,MA,MD)
      DIMENSION MA(N,N),MD(N,N)
      DO 1 I=1,N
      DO 1 J=1,N
    1 MD(I,J)=MA(I,J)
      DO 2 J=1,N
      DO 2 I=1,N
      IF(MD(I,J).NE.1) GO TO 2
      DO 3 K=1,N
      IF(MD(I,K).EQ.1) GO TO 4
      IF(MD(J,K).EQ.1) GO TO 4
      MD(I,K)=0
    3 CONTINUE
    2 CONTINUE
      RETURN
    4 MD(I,K)=1
      GO TO 3
      END

      SUBROUTINE PMS(N,MA,MD,MS,IK,NOD1,NOD2)
      DIMENSION MA(N,N),MD(N,N),MS(N,N),IK(N)
      DO 1 L=1,N
    1 IK(L)=0
      DO 2 L=1,N
      DO 2 K=1,N
    2 MS(L,K)=0
      K=0
      DO 3 L=1,N
      IF(MA(L,NOD2).NE.1) GO TO 3
      IF(MD(NOD1,L).NE.1) GO TO 4
    5 K=K+1
      MS(NOD2,K)=L
      GO TO 3
    4 IF(NOD1.EQ.L) GO TO 5
    3 CONTINUE
    8 DO 6 L=1,N
      IF(IK(L).EQ.1) GO TO 6
      IF(MS(L,1).EQ.0) GO TO 6
      K=1
   12 IF(MS(L,K).NE.0) GO TO 7
      IK(L)=1
      GO TO 8
    7 IT=0
      DO 9 NJK=1,N
      IF(MA(NJK,MS(L,K)).NE.1) GO TO 9
      IF(MD(NOD1,NJK).NE.1) GO TO 10
```

```
 11 IT=IT+1
    MS(MS(L,K),IT)=NJK
    GO TO 9
 10 IF(NOD1.EQ.NJK) GO TO 11
  9 CONTINUE
    K=K+1
    GO TO 12
  6 CONTINUE
    RETURN
    END

    SUBROUTINE GEP(N,MS,NS,NSL,NOD1, NOD2)
    DIMENSION MS(N,N),NS,(N),NSL(N)
    IT=1
    NS(1)=NOD2
 13 NNS=2
    NJK=MS(NOD2,IT)
  4 N1=NNS-1
    DO 1 L=1, N1
    IF(NS(L).EQ.NJK) GO TO 2
  1 CONTINUE
    NS(NNS)=NJK
    K=1
 12 IF(MS(NJK,K).EQ.O) GO TO 3
 10 NJK=MS(NJK,K)
    NNS=NNS+1
    GO TO 4
  3 IF(NS(NNS).NE.NOD1) GO TO 9
    DO 5 L=1,NNS
  5 NSL(L)=NS(NNS-L+1)
    WRITE(3,6) (NSL(L),L=1, NNS)
  6 FORMAT(' ',/ / ,6I4)
  9 NNS=NNS-1
    IF(NNS.EQ.1) GO TO 7
    NJK=NS(NNS)
    L=1
 11 IF(MS(NJK,L).NE.NS(NNS+1)) GO TO 8
    K=L+1
    IF(MS(NJK,K).EQ.O) GO TO 3
    GO TO 10
  8 L=L+1
    GO TO 11
  2 NNS=N1
    NJK=NS(NNS)
    K=K+1
    GO TO 12
  7 IT=IT+1
    IF(MS(NOD2,IT).NE.O) GO TO 13
    RETURN
    END
```


Conclusions. Considerable experiments (on an IBM 360/40 and a FELIX C-512 computers) were conducted to investigate the efficiency (the efficiency of the program can be measured in terms of the computational times required to obtain the solutions) of the proposed method. Based on this computational experience, it may be said that the above FORTRAN program can be successfully used for large-sized diagraphs, generating all the elementary paths in a reasonably computer memory and computational time (e.g., in the complete digraph K_6^*, between every pair of nodes there exist 65 elementary paths, and our program has generated and listed these paths in 14 seconds on a FELIX C-512 computer).

The technique used in the above program is similar to that one used in [6] for finding all the elementary circuits of a digraph.

Finally, we mention that for the calculus of elementary paths' number between two specified nodes of a digraph, the result of [5] can be used to write a computer program.

REFERENCES

[1] Christofides N., *Graph theory — An algorithmic approach*, Academic Press, New York, London, San Francisco, 1975.

[2] Danielson G. H., *On finding the simple paths and circuits in a graph*, IEEE Trans., CT-15, 1968.

[3] Dhawan V., *Hamiltonian circuits and related problems in graph theory*, M. Sc. Report, Imperial College, London, 1969.

[4] Kaufmann A. et Coster D., *Exercices de combinatorique avec solutions* (tome II), Dunod, Paris, 1970.

[5] Marcu D., *Note on the elementary paths of a digraph*, Bull. Univ. Braşov, C-23, 1981.

[6] Marcu D., *Algorithms and Fortran programs for determining circuits and condensated graph of a finite oriented graph* (in Romanian), RTTc., **6**, 1983.

[7] Marcu D., *Finding the essential arcs in a digraph* (in Romanian). St. cerc. calcul economic şi cibernetică economică, **2**, 1975.

[8] Roberts S. M. and Flores B., *Systematic generation of Hamiltonian circuits*, Comm. of ACM, **9**, 1966.

[9] Yan S. S., *Generation of all Hamiltonian circuits, paths and centres of a graph and related problems*, IEEE Trans., CT-14, 1967.

Received, 25.II.1984

Faculty of Mathematics,
University of Bucharest,
Academiei Str. 14,
70109 — Bucharest,
ROMANIA