MATHEMATICA – REVUE D'ANALYSE NUMÉRIQUE
ET DE THÉORIE DE L'APPROXIMATION
L'ANALYSE NUMÉRIQUE ET LA THÉORIE DE L'APPROXIMATION
Tome 20, Nos 1–2, 1991, pp. 59–63

ON FINDING THE ELEMENTARY PATHS AND CIRCUITS OF A DIGRAPH

DĂNUȚ MARCU
(Bucharest)

In this paper, for a finite digraph, we suggest a very simple FORTRAN 77 program, for finding (optionally) all the elementary paths (or circuits) of a given length and having a specified starting vertex.

Introduction. One of the most important concepts of graph theory, and one which appears often in areas superficially unrelated to graphs, is that of a path (or circuit). In some situations, it is necessary to be able to generate a complete list of all the elementary paths (or circuits) of a digraph. This may, for example, be the case when the "best" path (or circuit) needs to be chosen, but the criterion to be used for deciding what path (or circuit) is "best" is very delicate (one could decide on the best path (or circuit) 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 (or circuits).

In a number of industries, especially the chemical and pharmaceutical industries, the following basic scheduling problem arises: a number (say n) of items is to be manufactured using a single processing facility or reaction vessel.

The facility (vessel) may or may not have to be reset (cleaned), after item p_i has been manufactured, (but before production of p_j is started), depending on the item combination (p_i, p_j). The cost of resetting the facility is constant regardless of the item, p_i that has just been produced or the item p_j that is to follow, and, of course, there is no cost incurred if no resetting of the facility is required (this problem can arise in two ways: either the resetting costs are in reality independent of the items or, alternatively, detailed cost data are not available and an average constant value is taken as an approximation).

Suppose that these n items are to be manufactured in a continuous cyclic manner, so that, after the production of the last of the n items, the manufacture of the first item in the fixed cycle is started again.

The problem then arises as to whether a cyclic production sequence for p_i, $i = 1, 2, \ldots, n$, can be found which requires no resetting of the facility. The answer to this question depends on whether a digraph, whose vertices represent the items and where the existence of an arc (p_i, p_j) implies that p_j can follow the production of item p_i on the facility without resetting, possesses a hamiltonian circuit or not.

Thus, a method for finding whether a digraph contains a hamiltonian circuit (or not) has direct applications in problems of sequencing or scheduling of operations. Equally important, however, is the use of such a method as a basic step in algorithms for the solution of other, seemingly unrelated, graph theory problems.

In this paper, we will be dealing with the following problem: given a gneral digraph, find all the elementary paths (or circuits) having a specified length and a starting vertex, if one or more such paths (or circuits) exist and we will give a general FORTRAN 77 program for the solution of the above mentioned problem.

Description of the method. Let $D = (V, E)$ be a digraph [1] (directed graph), having $V(|V| = n)$ the set of vertices (nodes), 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 vertex of one is the initial vertex of the next one. An elementary path is a path which does not use the same vertex more than once. A circuit is a path in which the initial vertex coincides with the final vertex.

The length of the path (or circuit) is the number of arcs appearing in the path (or circuit). An elementary path of length $n - 1$ is known as a hamiltonian path. Similarly, an elementary circuit which passes through all the n vertices of a digraph, i.e., of length n, is a hamiltonian circuit (of course, not all digraphs have a hamiltonian path or circuit).

The algebraic methods which have appeared in the literature [2–5] for determining elementary paths (or circuits), are theoretically interesting but are] too loose to be of value for arbitrary digraphs 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 [3]) reduces both the storage and time reguirements 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 bytes can be used only for digraphs with at most 20 vertices and an average vertex degree of more than 4. Moreover, even for the above mentioned size digraphs, the method used virtually all memory locations.

In this note, we give a very simple FORTRAN 77 program, for finding (optionally) all the elementary paths (or circuits) of a given length and having a specified starting vertex.

In the sequel, we denote:

N	= the vertices' number of the digraph,
MA	= the adjacency matrix of the digraph,
NODS	= the specified starting vertex of a path (or circuit),
LD	= the given length ($\geqslant 2$) of path (or circuit), starting from NODS,
NC	$= \begin{cases} 0, & \text{if we find the paths of length LD, starting from NODS,} \\ 1, & \text{if we find the circuits of length LD, starting from NODS.} \end{cases}$

The idea of the algorithm is to construct all the possible vertex sets P, of cardinality LD, such that the vertices in P represent a path from NODS. Initially, P consists of NODS. If x is the last-entered vertex from P, then a new vertex y is added to P iff $y \notin P$ and $(x, y) \in E$. Two possibilities now exist which will prevent any vertex being entered into P at some stage. Either:

(a) no vertex can be added to P

or

(b) the cardinality of P is LD.

In case (b), if NC $= 0$, P is printed and the output is followed by backtracking; if NC $= 1$, either:

(c) arc $(x$, NODS$)$ exists in D and a circuit of desired length is therefore found

or

(d) arc $(x$, NODS$)$ does not exist and no circuit of desired length can be obtained.

In case (c), the circuit consisting in the vertices of P and the arc $(x$, NODS$)$ is printed and the output is followed by backtracking.

In cases (a) and (d), backtracking must occur.

Backtracking involves the removal of the last-entered vertex from P, to produce the set $P := P - \{x\}$ and the addition into P of a vertex $z \neq x$, such that $(x', z) \in E$, where x' is the last vertex of $P - \{x\}$. If no vertex z exists, a further backtracking step is taken and so on.

The end of the search occurs when the set P consists only of the vertex NODS and no vertex exists for adding into P, so that a backtracking step would leave P empty.

The paths (or circuits) found up to that time are then all the paths (or circuits) of length LD that exist in the digraph. The method described above is summarized in the FORTRAN 77 program below, where the set P is piled in the vector NCC of maximum N components.

The matrix $MS(I, J)$, $I, J = 1,2,\ldots,N$ is defined as follows:

$$MS(I, J) = \begin{cases} 0, & \text{if } d^+(x_I) < J, \\ K, & \text{if } (x_I, x_K) \in E \end{cases}$$

```
      SUBROUTINE PCGL (N, MA, MS, NCC, NODS, LD, NC,
      NCLD)
      DIMENSION MA (N, N), MS (N, N), NCC (N)
      DO 1 I = 1, N
      DO 1 J = 1, N
1     MS (I, J) = 0
      DO 2 I = 1, N
      K = 0
      DO 2 J = 1, N
      IF (MA(I, J), EQ. O) GO TO 2
      K = K + 1
      MS (I, K) = J
2     CONTINUE
      IF (NC. EQ. 1) GO TO 3
      LD = LD + 1
3     NCLD = 0
      L = 1
      NN = 1
      NCC(L) = NODS
8     IF (L. EQ. LD) GO TO 4
      K = 1
7     NA = MS(NCC(L), K)
      IF (NA.EQ.O) GO TO 5
      DO 6 I = ,1 L
      IF (NA. NE. NCC (I)) GO TO 6
      K = K + 1
      GO TO 7
6     CONTINUE
14    L = L + 1
      NCC(L) = NA
      GO TO 8
5     L = L – 1
      IF (L. EQ. O) RETURN
      IF (L. NE. 1) GO TO 9
      NN = NN + 1
      K = NN
      GO TO 7
9     K = 1
11    NA = MS (NCC (L). K)
      IF (NA. EQ. O) GO TO 5
      IF (NA. LT. NCC (L + 1)) GO TO 13
      IF (NA. NE. NCC (L + 1)) GO TO 10
13    K = K + 1
      GO TO 11
```

```
10    DO 12 I = 1, L
      IF (NA. NE. NCC (I)) GO TO 12
      GO TO 13
12    CONTINUE
      GO TO 14
4     IF (NC. EQ. O) GO TO 15
      IF (MA (NCC(L), NODS). NE. 1) GO TO 5
      PRINT 16, (NCC (I), I = 1, L), NODS
16    FORMAT (//, 1X, 4I4, I4)
18    NCLD = NCLD + 1
       GO TO 5
15    PRINT 17, (NCC (I), I = 1, L)
17    FORMAT (//, 1X, 4I4)
      GO TO 18
      END
```

where $d^+(x)$ denotes the outdegree of the vertex x.

In the subroutine PCGL (Paths or Circuits of Given Length), NCLD denotes the number of all the elementary paths (or circuits) of length LD and having NODS as a starting vertex.

Conclusions. Considerable experimentations, on a PDP-11 computer, were conducted to investigate the efficiency (the efficiency of the program can be measured in terms of the computational time required to obtain the solution) of the proposed method. Based on this computational experience, it may be said that the above program can be successfully used for large-sized digraphs, generating the desired paths (or circuits) in a reasonably computer memory and computational time.

REFERENCES

1. Berge, C., *Graphes et hypergraphes*, Dunod, Paris, 1970.
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 D. Coster, *Exercices de combinatorique avec solutions* (tome II), Dunod, Paris, 1970.
5. Yau, S. S., *Generation of all Hamiltonian circuits, paths and centers of a graph and related problems*. IEEE Trans., CT-14, 1967.

Received 15.III.1990

Str. Pasului 3, Sect. 2,
70241 — Bucharest,
Romania