[6] R. É. Halbach, "The noninvasive measurement of limb blood flow by means of nuclear magnetic resonance," Ph.D. dissertation, Marquette University, Milwaukee, WI, May 1977.
[7] R. E. Halbach, J. H. Battocletti, A. Sances, Jr., R. L. Bowman, and V. Kudravcev, "Cylindrical crossed-coil NMR limb blood flowmeter," *Rev. Sci. Instrum.*, vol. 50, pp. 428-434, 1979.
[8] J. H. Battocletti, J. H. Linehan, S. J. Larson, A. Sances, Jr., R. L. Bowman, V. Kudravcev, W. K. Genthe, R. E. Halbach and S. M. Evans, "Analysis of a nuclear magnetic resonance blood flowmeter for pulsatile flow," *IEEE Trans. Biomed. Eng.*, vol. BME-19, pp. 403-407, Nov. 1972. (Note: Figs. 5 and 6 are interchanged.)
[9] S. X. Salles-Cunha, "Noninvasive techniques in the evaluation of the peripheral circulation," Ph.D. dissertation, Marquette University, Milwaukee, WI, Aug. 1978.
[10] J. P. Woodcock, "Plethysmography," *Biomed. Eng.* vol. 9, pp. 406-409, p. 417, 1974.
[11] C. B. Anderson, E. E. Etheredge, H. R. Harter, J. E. Codd, R. J. Graff, and W. T. Newton, "Blood flow measurements in arteriovenous dialysis fistulas," *Surgery*, vol. 81, pp. 459-461, 1977.

Generation of Directed Circuits in a Directed Graph

PRADIP K. SRIMANI

Abstract—**A new matrix algorithm has been presented to generate all directed circuits in a given directed graph. The method is based on multiplication of the modified adjacency matrix of the graph by itself according to some modified rules of matrix multiplication.**

I. Introduction

Enumeration of all possible directed circuits in a given directed graph constitutes an important problem in the broad arena of linear graph theory, and this information of all circuits becomes quite essential in various other applications. This problem has already been discussed by various authors [1]-[7]. In the present letter, we propose a new matrix algorithm to enumerate all the directed circuits in a diagraph by doing a single matrix multiplication where the rules of matrix multiplication are slightly modified. The proposed algorithm is, in principle, similar to the revised matrix algorithm [8] to enumerate the shortest paths in a digraph, and it appears to enjoy some computational advantage over the existing algorithms. We define a modified adjacency matrix (MAM) of a directed graph G with n vertices to be a n by n matrix $[A_{ij}]$, where the (i, j)th entry is v_j iff the vertex v_i is connected to vertex v_j by a directed edge from v_i to v_j and zero otherwise. For convenience of description we say that a vertex v_i is less than a vertex v_j if $i < j$. The directed circuits are generated as closed vertex strings at different diagonal entries of the product matrix obtained by multiplying the MAM of the given graph G with itself according to the modified rules of multiplication.

II. The Algorithm and Its Proof

We describe the algorithm in steps as below.

Step 1: The vertices of the given graph G are arbitrarily named as $v_1, v_2, \cdots, v_n$ and the modified adjacency matrix $\lfloor A_{ij} \rfloor$ of the graph G is constructed.

Step 2: The MAM $\lfloor A_{ij} \rfloor$ is then multiplied by itself to generate the matrix $\lfloor C_{ij} \rfloor$ according to the following modified rules of matrix multiplication.

i) The (i, j)th entry of the product matrix $[C_{ij}]$ is calculated, as usual, by multiplying the ith row with the jth column of the matrix $\lfloor A_{ij} \rfloor$, using the relation $C_{ik} = \sum_j (A_{ij} * A_{jk}) + A_{ik}$, where • denotes product operation. Evidently the entries of the matrix $\lfloor C_{ij} \rfloor$ are, in general, in the sum of products form where the product of two vertices v_i and v_j is defined to be their concatenation $v_i v_j$. During the formation of the product terms, the entry belonging to the multiplicand row should precede that in the multiplier column, since such ordering in the concatenation process is necessary for proper generation of the circuits. The element 0 as an entry of the matrix $\lfloor A_{ij} \rfloor$ has the property that $0 + P_k = P_k$ and $0P_k = 0$, for any P_k where P_k stands for, in general, any sum of products of vertices.

ii) An entry of the product matrix, once calculated, immediately replaces the corresponding entry in the two matrices $\lfloor A_{ij} \rfloor$ which are being multiplied to form the product matrix before the next entry of the product matrix is calculated. Evidently, in such a process, the order of calculation plays a very significant role as the entry calculated last uses entries calculated earlier. Here the product matrix is calculated starting from the top row and from left to right along a row.

iii) To calculate the off diagonal entries of the product matrix, if a vertex string is generated such that a particular vertex v_a occurs twice in the string, the vertex string is compressed by deleting the vertices occuring between two v_a's and also one of the v_a's. This is done to eliminate the presence of directed circuits in the off diagonal entries. If in any entry of the product matrix, identical product terms are generated more than once, only one is retained eliminating the duplicates.

Step 3: All the vertex strings (product terms) generated in the diagonal entries of the resulting product matrix are listed eliminating any duplicates that may be present and this gives the list of all directed circuits of G.

We now state and prove two theorems to establish that the aforementioned algorithm actually generates all the directed circuits of the given graph G.

Theorem 1: The diagonal entries $C_{11}, C_{22}, \cdots, C_{kk}$ together of the product matrix $\lfloor C_{ij} \rfloor$ for $k = 1, 2, \cdots, n$ contain at least all the directed circuits constituted of only the vertices less than or equal to v_k.

Proof: The theorem is obviously true for $k = 1$, since if there is a self-loop with the vertex v_1, it is indicated by the presence of the entry v_1 in the MAM $[A_{ij}]$ and the (1, 1) entry in the product matrix will also contain v_1, of course, along with other vertex strings. Let $k = 2$. The number of possible directed circuits with only two vertices v_1 and v_2 is one and that is represented by (2-1-2). The entry C_{22} is calculated by multiplying the second row and second column of modified $[A_{ij}]$ (since before calculation of C_{22}, C_{12} and C_{21} were calculated and they sequentially replaced A_{12} and A_{21} in $[A_{ij}]$ as and when calculated). The second row contains the information (1-2), i.e., whether vertex v_1 can reach vertex v_2 and the second column contains the information (2-1), i.e., whether vertex v_2 can reach v_1. It should be emphasized that the concerned row and column contain other informations as well, but we are concerned with the informations about vertices less than or equal to v_k. Thus C_{22} must contain the circuit (2-1-2) if it at all exists in the graph G. Thus the theorem is proved for $k = 2$. Let $k = 3$. Before calculation of the entry C_{33}, the entries $C_{12}, C_{13}, C_{21}, C_{23}, C_{31}, C_{32}$ of the product matrix will be generated (and will replace the corresponding entries of $\lfloor A_{ij} \rfloor$) and they will contain the information about the possible interconnections of the vertices v_1, v_2, v_3 as follows, along with other informations which are omitted here, $C_{12} \to (1-2)$; $C_{13} \to (1-3) + (1-2-3)$; $C_{21} \to (2-1)$; $C_{23} \to (2-3) + (2-1-3)$; $C_{31} \to (3-1) + (3-2-1)$; $C_{32} \to (3-2) + (3-1-2)$. Hence C_{33} will contain the informations as $C_{33} \to (3-1-3) + (3-2-3) + (3-1-2-3) + (3-2-1-3) +$ other informations. Thus C_{33} along with C_{22} contain all possible circuits only with the vertices less than or equal to v_3, if at all they are present in G. Hence the theorem is proved for $k = 3$. This reasoning can be extended in a straightforward way for all other values of and let us assume that the theorem is true for $k = m$. To prove it for $k = m + 1$, we note that the entries $C_{m,m+1}$ and $C_{m+1,1}, C_{m+1,2}, \cdots, C_{m+1,m}$ of the product matrix will be computed before $C_{m+1,m+1}$, and they will use the previously computed entries (with row and column specifications $\leq m$) of the matrix. Thus $C_{m+1,m+1}$ will necessarily contain information about the circuits containing v_{m+1} and vertices less than v_{m+1} together with other possible informations. That is the diagonal entries $\{C_{ii}, 1 \leq i \leq m + 1\}$ together contain information about circuits with vertices less than or equal to v_{m+1}. Hence the theorem.

Theorem 2: The algorithm generates all the directed circuits of the given graph.

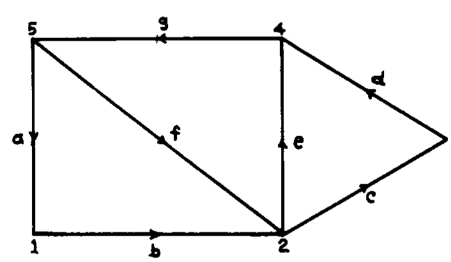


Fig. 1. The graph G.

Manuscript received October 16, 1978; revised March 30, 1979.
The author is with Electronics Unit, Indian Statistical Institute, 203 Barrackpore Trunk Road, Calcutta-700035, India.

Proof: By Theorem 1, we get that for $k = n$, where n is the number of vertices in graph G, the diagonal entries $C_{11}, C_{22}, \cdots, C_{nn}$ together of the product matrix developed through the execution of the algorithm must contain all possible circuits with the vertices of the graph less than or equal to v_n. Since there is no vertex in the graph greater than v_n, the theorem is proved.

Obviously the computational complexity of the proposed algorithm is expected to be exponential in nature since in each of the entries of the product matrix (in the form of sum of product terms) the length of a product term will be n and number of such terms will be larger than n in the worst case and each entry uses in its computation the previously computed entries of the matrix.

We will now illustrate the algorithm by considering an example graph G as shown in Fig. 1. The graph G has 5 vertices designated as 1, 2, 3, 4, 5 and seven edges designated as a, b, c, d, e, f, g. The MAM $[A_{ij}]$ of the graph G is shown in the adjoining diagram. This matrix is multiplied by itself according to the modified rules of Step 2. The final product matrix $[C_{ij}]$ is generated as shown.

$$[A_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 4 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 1 & 2 & 0 & 0 & 0 \end{bmatrix} \end{array}.$$

Collecting all the product terms in the diagonal entries of $[C_{ij}]$, we get

$$[C_{ij}] = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 2 & 23 & (24 + 234) & (245 + 2345) \\ 2 & 0 & 0 & 3 & (4 + 34) & (45 + 345) \\ 3 & 0 & 0 & 0 & 4 & 5 \\ 4 & 51 & (512 + 52) & (5123 + 523) & (5124 + 51234 + 524 + 5234) & 5 \\ 5 & 1 & (12 + 2) & (123 + 23) & (124 + 1234 + 24 + 234) & (1245 + 12345 + 245 + 2345) \end{array}$$

the four directed circuits of G as $\{(12451), (123451), (2452), (23452)\}$ or in terms of the edges of the graph as $\{bega, bcdga, egf, cdgf\}$.

References

[1] N. Deo, *Graph Theory with Applications to Engineering and Computer Science.* Englewood Cliffs, NJ.: Prentice-Hall, 1974.

[2] F. Harary, R. Z. Norman, and D. Cartwright, *Structural Models: An Introduction to the Theory of Directed Graphs.* New York: Wiley 1965.

[3] T. Kamae, "A systematic method of finding all directed circuits and enumerating all directed paths," *IEEE Trans. Circuit Theory*, vol. CT-14, pp. 166-171, June 1967.

[4] T. C. Gleason, "The number of paths and cycles in a digraph," *Psychometrika*, vol. 31, pp. 179-199, June 1966.

[5] S. S. Yau, "Generation of all Hamiltonian circuits, paths, and centres of a graph and related problems," *IEEE Trans. Circuit Theory*, vol. CT-14, pp. 79-81, Mar. 1967.

[6] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM J. Comput.*, vol. 4, no. 1, pp. 77-84, 1975.

[7] P. Mateti and N. Deo, "On algorithms for enumerating all circuits of a graph," *SIAM J. Comput.*, vol. 5, no. 1, pp. 90-99, 1976.

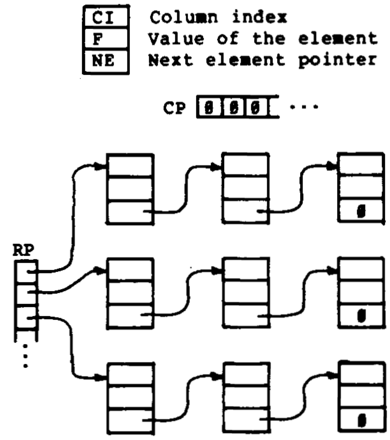
[8] T. C. Hu, "Revised matrix algorithms for shortest paths," *SIAM J. Appl. Math.*, vol. 15, no. 1, pp. 207-218, Jan. 1967.

A Note on Sorting Sparse Matrices

F. L. ALVARADO

***Abstract*—The problem of sorting all the rows of a sparse matrix according to increasing or decreasing column indices is considered. An algorithm for doing the sort in order τ operations (where τ is the number of nonzeroes in the matrix) is given.**

Manuscript received March 6, 1979.
The author is with the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, WI 53706.



(a)

```
      .
      .
      .
      CALL RSORT (N,RP,CP,CI,NE)
      CALL RSORT (N,CP,RP,CI,NE)
      .
      .
      .
```

(b)

```
      SUBROUTINE RSORT (N,H1,H2,CI,NE)
      IMPLICIT INTEGER(A-Z)
      DIMENSION H1(1),H2(1),CI(1),NE(1)
      DO 10 I=1,N
10    H2(I)=0
      I=N
20    P1=H1(I)
30      IF(P1.EQ.0) GO TO 40
        P2=NE(P1)
        J=CI(P1)
        CI(P1)=I
        NE(P1)=H2(J)
        H2(J)=P1
        P1=P2
        GO TO 30
40    I=I-1
      IF(I.GT.0) GO TO 20
      RETURN
      END
```

(c)

Fig. 1. A Fortran algorithm for simultaneous radix sort of a space matrix stored by row linked lists. (a) The data structure. (b) The calling statements. Notice differences in parameter lists. (c) The subroutine RSORT.

Sparse matrices arising in connection with electrical networks are usually stored by means of row singly linked lists [1], [2]. There is often a need for sorting these linked lists according to column indices. This is frequently desirable for output purposes. Certain sparse matrix algorithms also require that the row linked lists be sorted according to increasing or decreasing column indices. Furthermore, the renumbering of rows and columns for sparsity preservation purposes [1] often forces a resorting of the rows. It has generally been accepted that an insertion sort is most desirable. Let r_i be the number of nonzeroes in the ith row. Assume a random initial permutation of the ith row. The expected number of comparisons required to sort the ith row by insertion sort is known to be of order n^2 [3]. For small numbers, the actual value is quite small, but it increases rapidly with r_i:

r_i	1	2	3	4	5	10	100	1000
comparisons	0	1	2.67	4.92	7.39	29.6	2570	2.5×10^5

Let τ be the number of nonzeroes in the sparse matrix, that is:

$$\tau = \sum_{i=l}^{n} r_i .$$

A matrix can be said to be sparse if the number of nonzeroes is less than n^2. To be more specific, a class of matrices can be said to be