01 Jan 1974

A Review of Directed Graphs as Applied to Computors

Paul D. Stigall
Missouri University of Science and Technology, tigall@mst.edu

Ömür Tasar

Recommended Citation

P. D. Stigall and Ö. Tasar, "A Review of Directed Graphs as Applied to Computors," *Computer*, vol. 7, no. 10, pp. 39 - 47, Institute of Electrical and Electronics Engineers; Computer Society, Jan 1974.
The definitive version is available at https://doi.org/10.1109/MC.1974.6323332

Special Tutorial

A Review of Directed Graphs as Applied to Computors

Paul D. Stigall and Ömür Tasar
University of Missouri-Rolla

Introduction

The complexity of digital computers and their large scale use have led some researchers to investigate tools not commonly used. In recent years, applications of graph theory to computers as well as other fields of study have given fruitful results and have attracted more and more scientists. The attempt here will be to review previous accomplishments on a fundamental level and to stimulate the reader to investigate an area where valuable work is being performed.

A great advantage for anyone who works on graphs is to be able to transfer his problems to a computer. The graphs are represented by matrices that can be easily handled in computer programs. Useful programs, which are applicable to many engineering problems, are documented by Henley and Williams.[1]

The theory relevant to the study of graphs is rigorously developed.[2,3] Because its applications are many, it is worth mentioning a few interesting papers. A FORTRAN recognizer, which is itself a FORTRAN program, has been modeled by a graph by Gonzalez and Ramamoorthy.[4] This graph is reduced by the techniques described in the following sections. The information obtained from the reduced graph is useful in determining the suitability of a program for parallel processing. Another paper presents a discussion of the techniques for optimal scheduling of tasks in a multiprocessor system.[5] Given a set of computational tasks and the relationship between them, a graph is formed. The algorithm developed finds a schedule for tasks for which the total execution time and the number of processors required are minimal. Bruno and Altman[6] have modeled the control structure of an asynchronous digital system. Basic control modules are formed to perform single control functions. The obtained graph is used to find the necessary and sufficient conditions for a class of well-formed control networks.

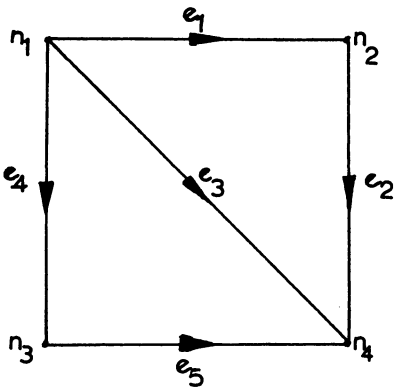
The discussion of graphs in this paper will not span the classical use of graphs, such as state diagrams which model finite-state sequential machines.[7]

Basic Definitions

A graph is simply a mathematical model of a system. It exhibits a relation or the absence of a relation among the elements of a set. The terms "point," "vertex," and "node" are frequently referred to as the elements of this set. A relation between these elements is usually called "line," "branch," "link," or "edge." In this paper, interest is concentrated on directed graphs, where the edges must be directed, and the terms "node" and "edge" will be used.

What is to be coordinated with nodes and edges is a matter for the problem in question. In the case presented here, a node may possibly represent a register, a flip flop, a gate, or a unit of the computer. An edge may represent a connection between two registers, or if it has a value associated, it may reflect a property of the system such as speed. Figure 1 shows a simple directed graph.

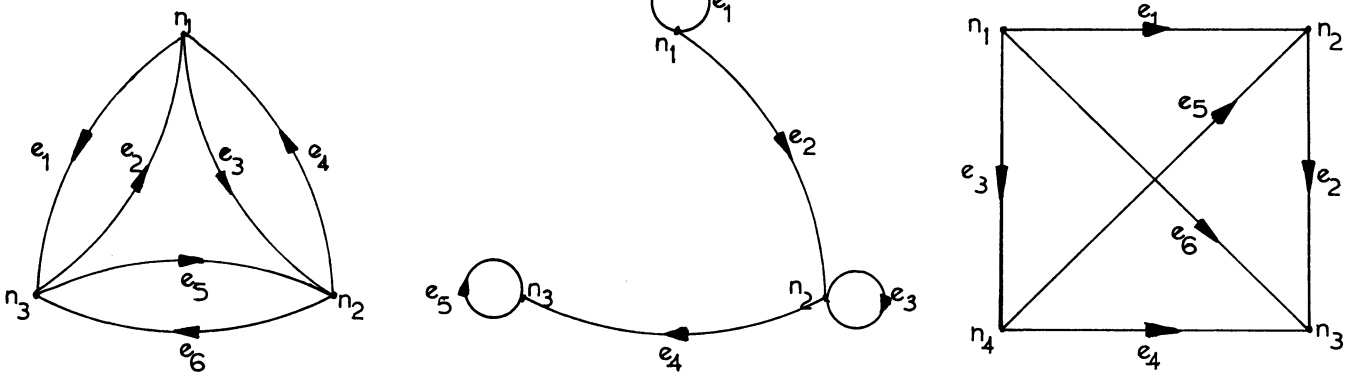
Graphs may have properties such as symmetry, reflexiveness, and completeness. A graph is *symmetric* if every node satisfies the following condition: existence of an edge from node (a) to node (b) implies an edge directed



N = set of nodes = n_1, n_2, n_3, n_4

E = set of edges = e_1, e_2, e_3, e_4, e_5

Figure 1. A Directed Graph



a. Symmetric Graph

b. Reflexive Graph

c. Complete Graph

Figure 2. Properties of Directed Graphs

from node (b) to node (a). A graph is *reflexive* if every node has a loop on itself. A graph is *complete* if every pair of nodes is connected in at least one direction. These properties are reflected in Figure 2.

Types of Directed Graphs

The following set of directed graphs are not used throughout the rest of the paper but are included here for the sake of completeness.

A *net* is a directed graph consisting of a set of nodes and edges. The set of nodes is finite and not empty. The set of edges is finite.[3] Hence a single node can constitute a trivial net. An example of a net is shown in Figure 3a. *Petri nets*, named after their inventor, C. A. Petri, are used to represent systems in which both static and dynamic conditions can exist simultaneously.[8] J. L. Bear introduced them as graph models for parallel computation in a survey in which multiprocessing was studied.[9]

A *relation* is a directed graph which satisfies the above conditions but does not have any parallel edges. Two edges connecting the same two nodes are not considered parallel if they are oppositely directed. In Figure 3b, a relation obtained from Figure 3a is shown.

A *digraph* is an *irreflexive* relation.[3] Namely, it is a directed graph or a net having no parallel edges and loops. The theoretical studies and matrix representation of digraphs will be discussed in detail below. Figure 3c illustrates a digraph reduced from Figure 3b. *Acyclic* directed graphs form a special class of digraphs where no two nodes are mutually reachable.[3,9]

A *network* is a relation in which the edges are assigned values.[3] If all the values on the edges are one, the graph is still a relation. In this sense, relations form a subset of networks. Systems dealing with frequencies, probabilities, and cost analysis can easily be represented by networks. A corresponding network of the relation in Figure 3b is shown in Figure 3d. In *flow* networks, edges are interpreted to represent capacities of a flow, such as signals, cars, people, oil, and trade items. Then maximum flow considerations become important. Techniques and algorithms are developed to find a maximum flow in a network between any two nodes.[2,3] Network flows are formulated as linear programs. Maximum flow in relation to linear programming is discussed by T. C. Hu.[10]

Matrix Representation

Matric representation is a practical tool with which one can work on graphs. It allows algebraic manipulation and use of computer programming so that large dimensioned matrices, hence large graphs that have many nodes and edges, can be analyzed.

In forming the matrix, one row and one column for each node in the graph are assigned. Unless the assignment varies, a square matrix is generated. Depending on what is intended for the matrix, it is equally valuable to assign rows and columns to edges, or rows to nodes and columns to

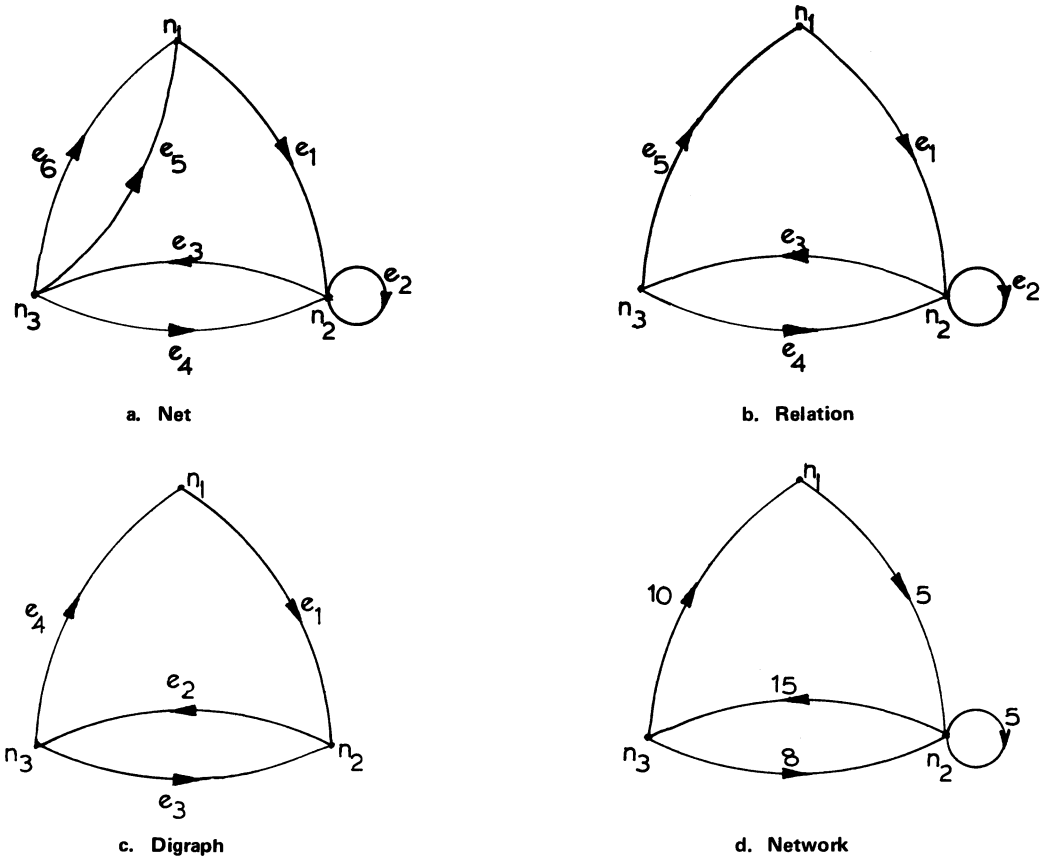


Figure 3. Types of Directed Graphs

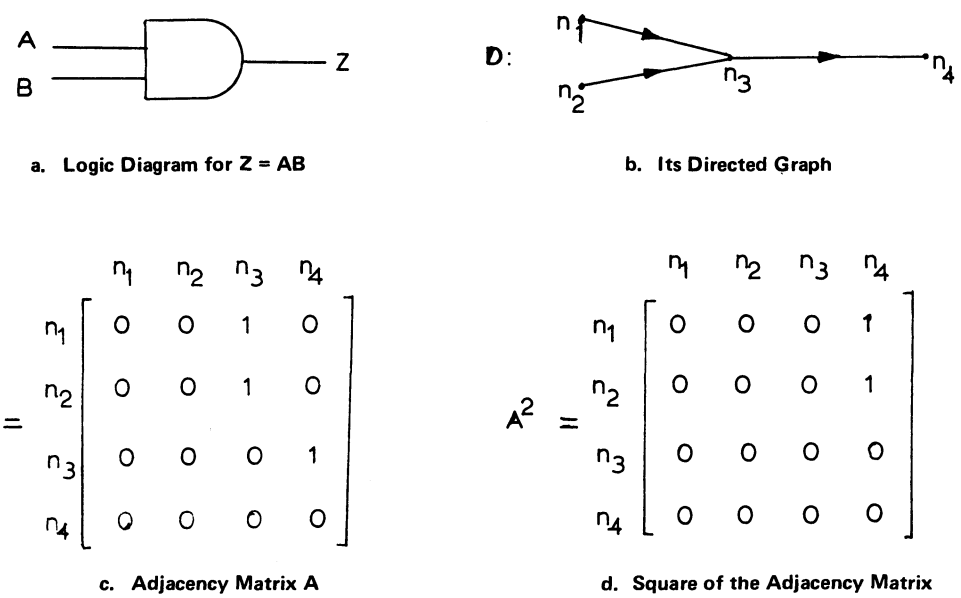


Figure 4. Adjacency Information of D Representing a Logic Diagram

edges. However, the case with nodes is discussed here, and it should be kept in mind that a similar approach may be taken for other cases.

The Adjacency Matrix The *adjacency* or connectivity matrix A has extensive use and is defined as follows: if the element a_{ij} of the matrix A is one, it indicates that there is an edge from node (i) to node (j); if a_{ij} equals zero, the graph does not contain an edge from node (i) to node (j).[3,11-12] In other words, the nonzero elements of A show how many paths of length one, i.e., directed edges, exist between the corresponding nodes. Similarly, the elements of A^2, where A^2 is obtained by regular matrix multiplication of A itself, indicate the number of possible paths of length two. The idea can be extended to find the number of possible paths of length n. These are illustrated in Figure 4. For example, there exists one possible path of length two from n_1 to n_4.

The row sum of node (i) of A is called the *outdegree* of node (i), and the column sum of node (i) of A is called the *indegree.* These figures give the total number of edges going out of and into the node, respectively.

The adjacency matrix has been effectively applied to computer areas. Ramamoorthy and Chang[11] have based an algorithm on the adjacency matrix to segment a large system into smaller subsystems with the purpose of diagnosing the system in parallel. Russel and Kime[12] used the adjacency matrix as well as indegree and outdegree concepts in fault diagnosis of combinational networks. Kleir and Ramamoorthy[13] have used the adjacency matrix in optimization strategies for microprograms. It has also been used in the structural theory aspects of machine diagnosis.[14]

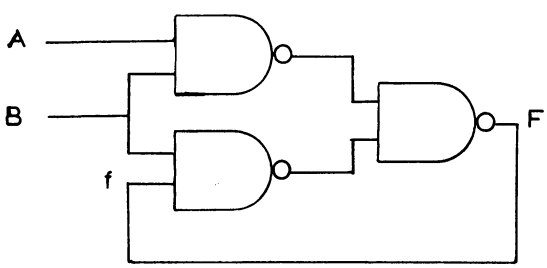
The Reachability Matrix The *reachability* matrix, R, gives useful information about the behavior of a graph. The element r_{ij} equals one if node (i) reaches node (j) over any path regardless of its length and if i equals j. If r_{ij} equals zero, it indicates that there is no possible path whereby node (j) can be reached from node (i).[3,11-12] The transpose of R, namely R^T, represents a graph in which the directions are reversed with respect to the original graph.

The elementwise product, $Q = R \times R^T$, is obviously a symmetric matrix. The nonzero elements, q_{ij}, of this matrix indicate that nodes (i) and (j) are mutually reachable. This information is useful for the purpose of reducing graphs if necessary. One reduction technique requires the selection of the nodes whose columns are equal. These are the nodes which are reachable from the same set of nodes and which reach the same set of nodes. Hence, they can be combined into one node. The set of nodes combined into one new node is called a *strong component.* If all columns of Q happen to be equal, the graph is said to be *strongly connected,* in which case every node is reachable from every other node. An example of the points illustrated above is given in Figure 5.

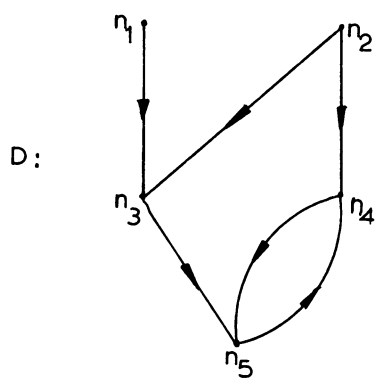
Hence, the graph has four strong components, $\{n_1\}$, $\{n_2\}$, $\{n_3\}$, and $\{n_4, n_5\}$, which are called N_1, N_2, N_3, and N_4, respectively. To find the reduced graph with nodes N_1, N_2, N_3, and N_4, the new edges must be found. The rule is as follows: there exists an edge from N_i to N_j if there is a path from any node in N_i to any node in N_j. The reduced graph is called a *condensation* of D and is shown in Figure 6.

The reachability matrix has been applied to computer related problems.[12,14-15] It provides a powerful reduction technique. Most of the algorithms in fault diagnosis are derived for reduced graphs.[16]

The Connectedness Matrix A valuable measure for classifying graphs is connectedness. Earlier, a strongly connected graph and its strong components were defined. To explore the other possibilities, the following types of graphs can be briefly described with respect to connected-



a. Logic Diagram for F = AB + Bf



b. Its Directed Graph

$$R = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{array} \begin{array}{c} \begin{array}{ccccc} n_1 & n_2 & n_3 & n_4 & n_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

c. Its Reachability Matrix R

$$Q = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{array} \begin{array}{c} \begin{array}{ccccc} n_1 & n_2 & n_3 & n_4 & n_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

d. $Q = R \times R^T$

Figure 5. Reachability Information of D Representing a Logic Diagram

ness. A *disconnected* graph implies that there exists at least one node or a set of strongly connected nodes that neither reaches nor is reached from any other node. In a *strictly weak* graph, there exists a sequence of edges between any two nodes; however, the directions are not continuous. A *strictly strong* graph has a directed path between any two nodes. The example in Figure 7 clarifies these descriptions.

It is an easy matter to deduce the idea of connectedness from the reachability information. Hence, a connectedness matrix, C, can be defined, which will indicate the above attributes, given two nodes. Let us say that 0, 1, 2, and 3 represent the four kinds of connectedness: disconnected, strictly weak, strictly strong, and strongly connected, respectively. A new matrix, J, is defined as follows:

$$j_{kl} = \begin{cases} 0 \text{ if node (k) and node (1) are disconnected} \\ 1 \text{ otherwise} \end{cases}$$

Thus, the matrix representation for connectedness is simply $C = R + R^T + J$. For instance, in Figure 7a, because j_{14}, r_{14}, and r_{41} are zero, c_{14} equals zero implying a disconnected graph.

In case the graph is composed of disconnected subgraphs, $D_1, D_2, \ldots D_n$, the connectedness matrix for each subgraph can be found, $C(D_1), C(D_2), \ldots C(D_n)$. The connectedness matrix of the whole graph will have the forms shown in Figure 8.

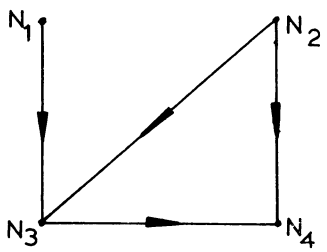
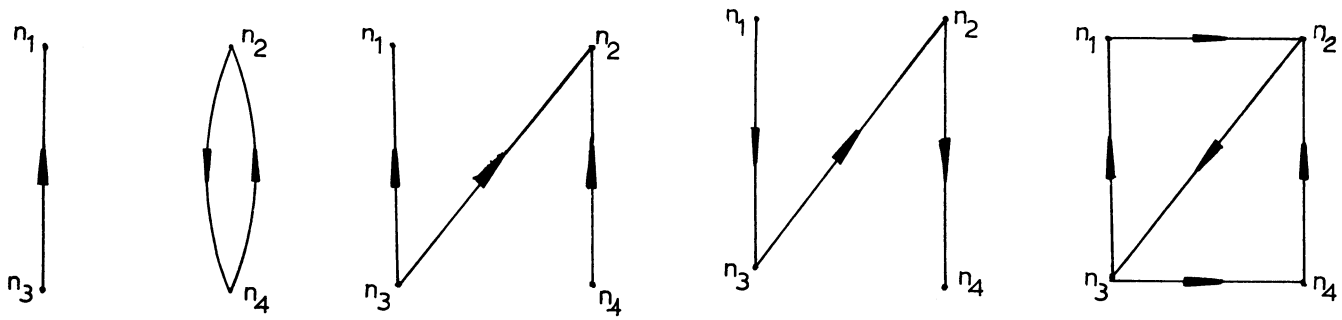


Figure 6. The Reduced Graph of Figure 5b.

The Value Matrix For a network, the adjacency matrix has scalar entries rather than ones and zeros. To distinguish the two matrices, the adjacency matrix for a network is called the *value* matrix, M. If the values on the edges are associated with probability or cost, the matrix is called a *probability* matrix, P, or a *cost* matrix, G, respectively. In cases where the outdegree of each node on a probability network is one, and the probabilities assigned to the edges are time-dependent, this particular type of graph is

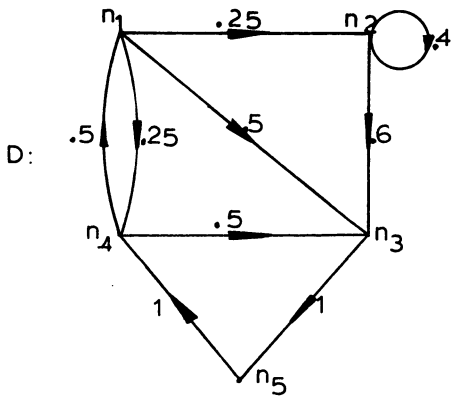


a. Disconnected Graph b. Strictly Weak Graph c. Strictly Strong Graph d. Strongly Connected Graph

Figure 7. Types of Graphs with Respect to Connectedness

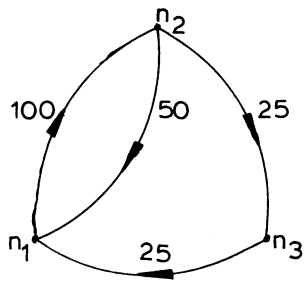
$$C = \begin{bmatrix} C(D_1) & 0 & 0 & \cdot & \cdot & 0 \\ 0 & C(D_2) & 0 & \cdot & \cdot & 0 \\ 0 & 0 & C(D_3) & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & C(D_n) \end{bmatrix}$$

Figure 8. Connectedness Matrix of a Disconnected Graph



$$P = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{array} \begin{array}{c} \begin{array}{ccccc} n_1 & n_2 & n_3 & n_4 & n_5 \end{array} \\ \begin{bmatrix} 0 & .25 & .5 & .25 & 0 \\ 0 & .4 & .6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ .5 & 0 & .5 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Figure 9. A Probability Network and Its Probability Matrix



$$G = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \end{array} \begin{array}{c} \begin{array}{ccc} n_1 & n_2 & n_3 \end{array} \\ \begin{bmatrix} 0 & 100 & \infty \\ 50 & 0 & 25 \\ 25 & \infty & 0 \end{bmatrix} \end{array}$$

Figure 10. A Cost Network and Its Cost Matrix

occasionally called a *markov chain.* M. A. Breuer modeled the statistics of intermittent faults in digital circuit by a first order Markov model.[17] The cost of going from node (i) to node (i + 1) is infinite, if there is no edge from node (i) to node (i + 1). Examples are shown in Figures 9 and 10.

Description of the Basic Theorems At this point, it is interesting to investigate whether the matrices discussed relate to each other in any manner. The matrix R is expected to be a function of the powers of A, because the elements of A^n indicate the possible number of paths of length n, and the reachability question asks whether any path exists between two given nodes. The procedure then must be to take powers of A until the longest path has been searched and to transform the total number of possibilities to a "one" to indicate reachability. The latter function is labeled U and defined as follows: U(a) = 1, where a is any number other than zero, and U(0) = 0.

After this stage, the reachability matrix will not change. The above descriptions can be summarized with a theorem:

Theorem 1: $R_n = U(I + A + A^2 + \ldots + A^n)$ in which I is the identify matrix and defines reachability over length n. If $R_n = R_{n+1}$, then $R = R_n$.[3]

The cost of going from one node to another over a specified length is of great importance. One certainly would try to find the minimum cost path, which is called *cost geosdesic.* When taking powers of G, the matrix multiplication is performed by using the modified multiplication "X" and the modified addition "+" operations defined as follows: a X b = a + b and c + d = min(c,d).[3]

Obviously, the arithmetic does not add all possible paths of a certain length but rather finds the cost of different paths of the same length and chooses the minimum value. Let us find g^2_{11} of G^2, i.e., the cost of going from node (1) to itself over a path of length two by referring to Figure 10.

$$\begin{aligned} g^2_{11} &= g_{11} \times g_{11} + g_{12} \times g_{21} + g_{13} \times g_{31} \\ &= \min(g_{11} + g_{11}, g_{12} + g_{21}, g_{13} + g_{31}) \\ &= \min(0, 150, \infty) \\ &= 0. \end{aligned}$$

Staying at node (1) is assumed to have no cost, hence that path is chosen.

Theorem 2: There is a positive integer for which $G^n = G^{n+1}$.[3]

After reaching this condition, there is no need to take higher powers of G, and G^n is called the total cost matrix.

In general, if the edges of the network indicate only zeros and ones, a *distance* matrix is formed, and the modified arithmetic yields the minimum distance.

PL/1 programs for finding the R and C matrices, given the A matrix, and for calculating total cost matrix are documented and available from the authors.

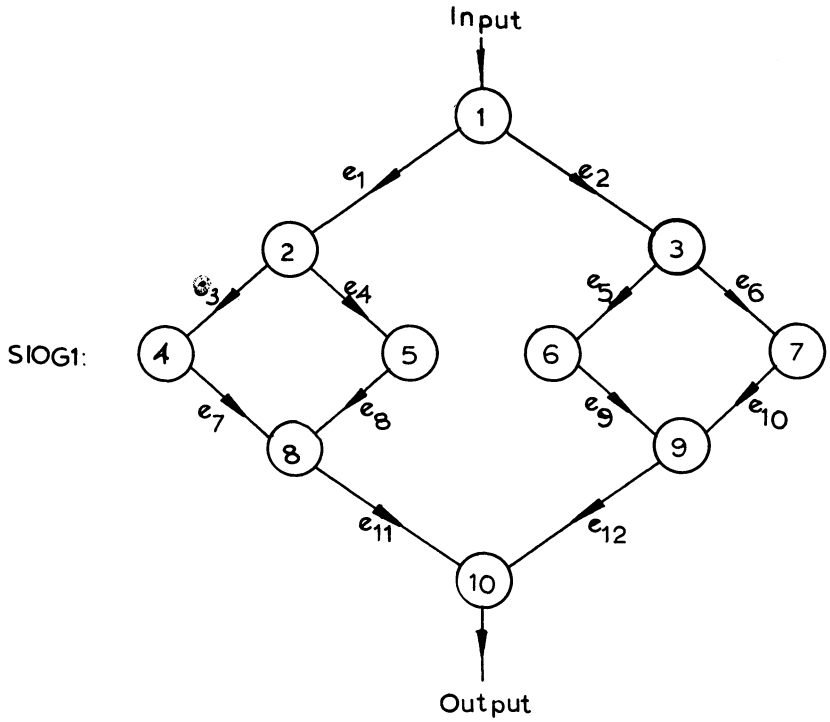


Figure 11. A Single Input Single Output Graph, SIOG1

Fault Diagnosis Using Graph Theory

A fault is a malfunction in the system. It is of vital importance whether a fault exists or not. Testing the machine for a fault if it exists is called a diagnosis. Fault diagnosis in digital computers is one of the major areas where efficient methods have to be established. In recent years, applications of graph theory to this area have been promising. An approach called a test point method finds spots on the system where test points can be located.[14] Some points serve as inputs, and the result can be detected at other points. The blocking gate method is another approach in which the minimum number of edges to be blocked are found to diagnose the system.[16] The latter method is discussed in detail below.

The blocking gate method assumes that there exists only one fault at a time and that it is not self-correcting. Also no faults can cancel each other during diagnosis. One way to diagnose the system is to insert blocking gates on every edge in the graph. The idea can be extended to conduct the diagnosis more efficiently.

First, the system is modeled with a directed graph. Then the graph is reduced so that it does not contain any strongly connected components. Finally, the graph is transformed into a *single input single output* graph (SIOG) to enable the method to work with one input and one output. This is easily accomplished. If the graph has more than one input, an input node (i) is entered to the graph that fans out to all the necessary inputs. If there is more than one output, they will lead to an extra output node (o). One can start with the example of the SIOG shown in Figure 11 to illustrate the steps in the blocking gate method.

At this point, a reduced SIOG exists and it is ready for the introduction of the pattern for finding the locations of the minimum number of blocking gates. The range of node (k) is the set of nodes on the directed path from node (i) to node (o), when node (k) is deleted. The node range matrix, NR, of a SIOG is a square matrix with rows and columns corresponding to the nodes of the graph. The NR matrix of the graph SIOG1 is given in Figure 12. The k'th row of the matrix has elements of ones for the nodes in the range of node (k). Otherwise, the elements are zero. The set of nodes, whose columns are equal, constitutes the partition of maximum distinguishability, MD. Referring to Figure 12, the MD set for the graph SIOG1 is found as follows:

$$MD = \{\overline{1,10}; \overline{2,8}; \overline{3,9}; \bar{4}; \bar{5}; \bar{6}; \bar{7}\}.$$

Every edge of the graph is to be tested so that the set of minimum number of edges is found. The blocking gates have to be located on this set of edges to distinguish the particular distinguishability class. Another matrix is used to perform what is mentioned above. The matrix, ER, has columns corresponding to the partitions of maximum distinguishability; its rows correspond to the edges of the graph. If the edges are ordered with consideration for the cost of blocking gates to be located on them, then the resultant sets of edges obtained from the matrix, ER, can be compared with the set with the minimum cost can be chosen. Figure 13 illustrates the ER matrix of the graph SIOG1. It is assumed that the cost of building blocking gates on the edges increases as the output node is approached.

The edge range of an edge (k) is defined as the set of nodes on a directed path from node (i) to node (o), when edge (k) is blocked. The element er_{k1} on the k'th row of the matrix ER would be one if the partition 1 includes the nodes within the range of edge (k). Otherwise, the elements

$$NR = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \left[\begin{array}{cccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

Figure 12. The NR Matrix of the Graph SIOG1

are zero. After constructing the matrix, equal rows are found. These rows form the sets of a new partition, named E. The equal rows of the ER matrix form the following set:

$$E = \{\overline{1,11}; \overline{2,12}; \overline{3,7}; \overline{4,8}; \overline{5,9}; \overline{6,10}\}.$$

The elements of E are to distinguish the partition of MD. Remamoorthy and Mayeda[16] discussed the method theoretically and proved that the set E distinguishes the set MD.

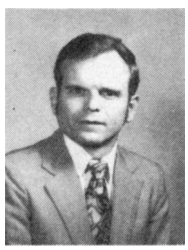
The matrix ER is reduced to another matrix EQ when the equal rows are deleted. The small numbered edges name the rows of EQ, because they imply less cost. To complete the example, if blocking gates are placed on edges 1, 2, 3, 4, 5, and 6 during design stages, the system will be able to diagnose faults within nodes 4, 5, 6, 7, and sets of nodes $\{1,10\}, \{2,8\}, \{3,9\}$.

Conclusion

The developments in graph theory are of current interest in computer research. Because many books and papers are being published on the subject, the aim of this paper has been to present to the reader the fundamentals of graph theory so as to acquaint him with the terminology and commonly used techniques. Today most of the work on graph theory is original and to some degree is separated into independent groups; however, a great deal of work needs to be done to unite all the ideas and applications into a consistent unified approach. ■

Acknowledgment

This work was supported in part by National Science Foundation Grant GJ-32596.



Paul D. Stigall is an Assistant Professor of Electrical Engineering and Computer Science at the University of Missouri-Rolla. At Rolla since 1970, he has been teaching and doing research in both electrical engineering and computer science.

He accumulated three years industrial experience with McDonnell Douglas in research and design of special purpose digital computers, digital aircraft simulators, and digital radar simulation. Other experience includes summer employment with the Navy Electronics Laboratory and Collins Radio Company. Also, he was a Graduate Assistant and Instructor at the University of Wyoming from 1963 to 1968.

He received the BS degree in electrical engineering at the University of Missouri-Rolla in 1962, and the MS and PhD in electrical engineering from the University of Wyoming in 1965 and 1968, respectively.

Dr. Stigall is author of several technical publications and principal investigator on several research grants. He is a member of IEEE, the Computer Society, ACM, Eta Kappa Nu, Sigma Tau, Sigma Xi, and a Registered Professional Engineer in Missouri.

Omür Tasar has been a research assistant in the Department of Electrical Engineering at the University of Missouri-Rolla since January 1973. Before coming to the United States for her graduate work in August 1972, Mrs. Tasar was an electrical engineer with the Turkish Radio and Television Corporation and a teaching assistant in the School of Engineering at Robert College, Istanbul, Turkey.

She received the BS degree with honors in electrical engineering from Robert College in June 1971 and is currently working towards the MS degree in electrical engineering at the University of Missouri-Rolla. She is a member of American Field Service and the IEEE Computer Society.

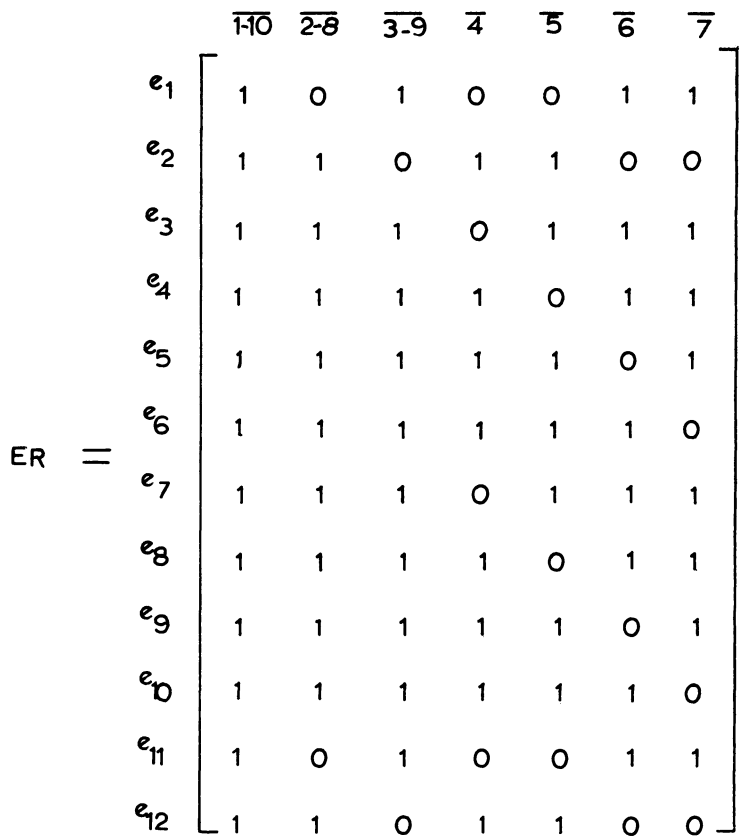


Figure 13. The ER Matrix of the Graph SIOG1

References

1. E. J. Henley and R. A. Williams, *Graph Theory in Modern Engineering,* New York, Academic Press, 1973.

2. H. Frank and I. T. Frish, *Communication, Transmission, and Transportation Networks,* Reading, Massachusetts, Addison-Wesley, 1971.

3. F. Harary, R. Z. Norman, and D. Cartwright, *Structural Models: An Introduction to the Theory of Directed Graphs,* New York: John Wiley & Sons, Inc., 1965.

4. M. J. Gonzalez, Jr., and C. V. Ramamoorthy, "Program Suitability for Parallel Processing," *IEEE Transactions on Computers,* Vol. C-20, June 1971, pp. 647-655.

5. C. V. Ramamoorthy, K. M. Chandy, and M. J. Gonzalez, Jrs., "Optimal Scheduling Strategies in a Multiprocessor System," *IEEE Transactions on Computers,* Vol. C-21, February 1972, pp. 137-147.

6. J. Bruno and S. M. Altman, "A Theory of Asynchronous Control Networks," *IEEE Transactions on Computers,* Vol. C-20, June 1971, pp. 629-639.

7. H. C. Torng, *Switching Circuits,* Reading, Mass., Addison-Wesley, 1972.

8. R. E. Miller, "A Comparison of Some Theoretical Models of Parallel Computation," *IEEE Transactions on Computers,* Vol. C-22, pp. 710-717, August 1973.

9. J. L. Bear, "A Survey of Some Theoretical Aspects of Multiprocessing," *ACM Computing Surveys,* Vol. 5, No. 1, March 1973.

10. T. C. Hu, *Integer Programming and Network Flows,* Reading, Mass., Addison-Wesley, 1969.

11. C. V. Ramamoorthy and L. C. Chang, "System Segmentation for the Parallel Diagnosis of Computers," *IEEE Transactions on Computers,* Vol. C-20, March 1971, pp. 261-270.

12. J. D. Russel and C. R. Kime, "Structural Factors in the Fault Diagnosis of Combinational Networks," *IEEE Transactions on Computers,* Vol. C-20, November 1971, pp. 1276-1285.

13. R. L. Kleir and C. V. Ramamoorthy, "Optimization Strategies for Microprograms," *IEEE Transactions on Computers,* Vol. C-20, July 1971, pp. 783-794.

14. C. V. Ramamoorthy, "A Structural Theory of Machine Diagnosis," in *AFIPS Conference Proceedings,* Vol. 30, 1967 SJCC, Washington, D.C.: Thompson, 1967, pp. 743-756.

15. P. D. Stigall, "A Data-Flow Structure for Dynamic Microprogrammed Computers," *IEEE Computer Society Repository*, R 72-239.

16. C. V. Ramamoorthy and W. Mayeda, "Computer Diagnosis Using Blocking Gate Approach," *IEEE Transactions on Computers,* Vol. C-20, November 1971, pp. 1294-1300.

17. M. A. Breuer, "Testing for Intermittent Faults in Digital Circuits," *IEEE Transactions on Computers,* Vol. C-22, March 1973, pp. 241-246.