THE GATED QUEUE NODE: PROPERTIES OF A NEW ELEMENT FOR STOCHASTIC QUEUEING NETWORKS

Nancy Blom
University of California, Davis, CA

Yoichi Akiba
Business Division, Cosumnes River College, Sacramanto, CA

and

W. W. Happ
University of Guanajuato, Salamanka, Mexico

ABSTRACT

Deficiencies of queueing networks to model multiserver single queue systems are examined and alternatives to remedy them are explored. The Gated Queue is defined as a new primitive which overcomes these deficiencies. It is a storage node whose release is controlled by an outside mechanism. Implementation was demonstrated by programming and testing it as part of the GERTS III Q program resulting in properties which cannot be synthesized from existing elements. A compatible set of modular logic modules are defined using the gated queue node and are examined analytically and tested. An example showing the ease of modelling and simulating using the new GERTS GQ is included.

GERTS GQ: A USER-ORIENTED NETWORK PROGRAM

Modelling complex systems from the standpoint of network techniques has been studied in various contexts. Flowgraph theory, PERT (Project Evaluation Review Technique) CPM (Critical Path Method), network flows, decision trees, all have in common graphical analysis.

GERT (Graphical Evaluation and Review Technique)[1] is also an analytical method for the formulation and analysis of queueing networks. The system, once represented in the form of a "Q" network, can then be simulated by the GERTS GQ (the S here stands for simulation) computer package. This FORTRAN based computer program is a modification of GERTS III Q which is presently available at more than twenty universities as well as at numerous private facilities.[2,3,4]

GERTS GQ networks permit a wide class of queueing systems to be simulated, providing a useful tool for modelling large complex queueing networks. Programmers, engineers, and nonprogrammers—in short, all who need to conduct simulation studies from a graphical or network point of view, and who are not concerned with detailed programming can benefit from this technique.[5,6]

The ease and accessibility of solutions possible with digital simulation of queueing networks provides a new and improved approach response to operational engineering requirements.

THE TERMINOLOGY USED IN GERTS

The network is a "next event simulation" network, that is, time is advanced from event to event. All events and event transitions are recorded in a continually updated event file. Some concepts commonly understood in GERTS are:

- Stochastic—Describes random phenomena which display predictable long-run regularities.
- GERTS network—A graphical representation of the project as a network. Each activity in the project is shown as a branch, each event as a node.
- Nodes—Decision-making elements functionally related by branches.
- Realization—Designates the actual occurrence of the event or activity represented by a designated node or branch during performance of the project.
- Release—The realization, exactly once, of an activity along a branch emanating from a node upon realization of that node.
- Network modification—Structural alterations made by automatically scheduled switching actions while simulation is actually in progress.
- Activity numbers—Identify those activities whose completion will trigger network modifications.
- Removal—Cancellation of all in-progress activities already scheduled into a node (i.e., still progressing along an entering branch) immediately upon realization of the node.

A node in GERTS is a decision point in a network representation. Items travel along a "branch" with a particular time distribution until they reach a node. Many items may travel along the same branch at the same time. That is, units of information or items are processed through nodes and distributed as entities without label, the label results only from the network topology.

Function and Structure

Nodes

Of key importance is the graphical representation of a system. Figure 1 shows a graphical symbolism used to depict the functional characteristics of nodes in the GERTS system.

Figure 1

A: Denotes the number of releases needed to realize the node the first time.

B: Denotes the number of releases needed to realize the node on succeeding occurrences.

#: Denotes the unique number associated with each node.

When the node is a special node called a Q-node, A and B have a slightly different meaning. In that case we have:

A: The number originally in the queue.

B: The maximum number allowed in the quque.

Branches

A directed branch, representing an activity and/or an information transfer, has associated with it one node from which it emanates and one node at which it terminates.

The graphical representation, without showing the nodes involved, is a directed line together with five associated parameters, as shown in Figure 2.

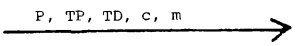


Figure 2

- "P" is the probability that the particular branch eminating from Q-node will be taken. This probability is specified when the branch is defined.
- "TP" and "TD" define the time to traverse the branch, i.e., to complete the activity represented by the branch.
- "TP," an integer value, points to a particular line in a table which specifies minimum time, maximum time standard deviation, and mean. "TD," the statistical distribution, an integer from 1 (one) to 9 (nine), specifies one of nine distributions available (see Figure 3).
- The counter type "c" and network modification "m" may or may not be specified for a particular branch.

t_D	Distribution
1.	Constant
2.	Normal
3.	Uniform
4.	Erlang
5.	Lognormal
6.	Poisson
7.	Beta
8.	Gamma
9.	Beta (fitted to three parameters as in PERT)

Figure 3

Single Server Queue

The major restriction associated with an ordinary Q-node is that only one service activity can emanate from a Q-node, i.e., there can be only one output branch when the output is deterministic. Two or more probabilistic outputs are considered to represent the same server.

Thus, the Queue node in GERTS III Q models is a single-server processor. When an entity leaves the Queue, another entity can not leave until the first entity arrives at the next node. This implies the "processing" time, the time to traverse the branch, must be greater than zero, otherwise the Queue will not hold any entities. A special array within the FORTRAN simulation program designates whether the "processor" is busy or not. No other entity may traverse the branch while the processor is busy. In practice this Queueing node provides incorrect results when incorporating this type of Queue into a network with more than one server. The gated Q, which overcomes this deficiency will be defined and examined separately.

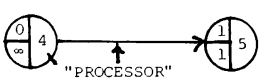


Figure 4

THE MODULAR NETWORK APPROACH TO SIMULATION

In order to give an engineer or nonprogrammer greater flexibility in modelling a system, Happ, Akiba and Dabaghian[7] developed a modular approach to GERTS III Q. That is, they devised from GERTS primitives a standard set of modules which could be used as an intermediary step from state diagram to nodal diagram, thus giving a more reliable model of the real world system. The "HAD" modular approach entails combining network primitives into a standard set of modules for greater flexibility in modelling and simulation.[8]

As an example of the modular concept Figure 5 shows the block symbolism used by Happ, Akiba and Dabaghian for the Initial Finite Pulse Generator. This module's function is to emit a pre-selected finite number of pulses or items. The nodal diagram which this symbol represents is shown in Figure 6. Network modification triggered by realization of the "Counter" (node #3) truncates the otherwise perpetually repeated sequence of pulses discharged by node #2.

Table 1 denotes the tested modules which presently make up the libray of "HAD" modules.

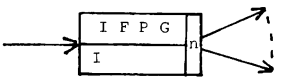


Figure 5

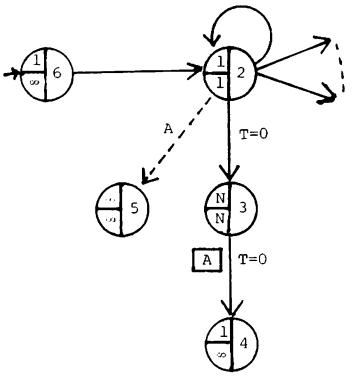


Figure 6

Deficiencies Found in GERTS III Q

With the conception, implementation and testing of the HAD modules came the knowledge of the deficiencies of the queueing system under GERTS III Q.[9] The program worked well for networks with single-server, single-queue constructs, but distributing an element to one of several queues was not possible without a complicated and essentially inaccurate representation, using holding patterns and/or network modification.

The Distributor Module. A distributor module which was proposed makes use of the Q-node's capability to balk an entity to another node when its processor is busy. Figure 7 shows the nodal diagram.

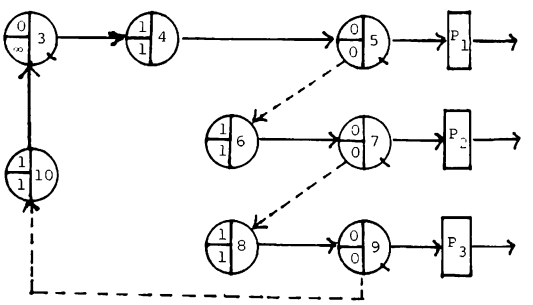


Figure 7

- Zero storage is specified in the Q nodes #5, #7, and #9.
- When processors are busy, entities are diverted through balk nodes.
- When all the processors are busy entities are circulated back to Q node #3 through node #10.

However, the branch emanating from a Q node (the processor) must be traversed with a time greater than zero, otherwise no queue will result. Thus there is a holding time from node #3 to node #4 which introduces inaccuracies. Also, entities are circulated continuously through the loop while all the processors are busy.

In order to eliminate the holding loop, it is desirable to have a gate in the Q node which is capable of stopping all incoming entities, storing them and then releasing them on command from another part of the system.

Table 1

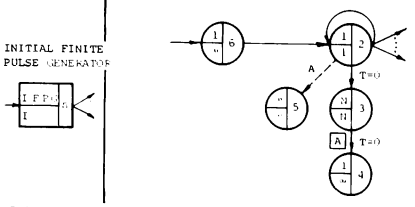
The library of "HAD" modules

PROLIFERATIVE

SIMPLE REPETITIVE PULSE GENERATOR

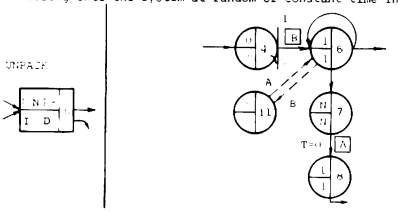
Simple Repetitive Pulse Generator: generates transactions at constant or random time intervals throughout the simulation run.

INITIAL FINITE PULSE GENERATOR



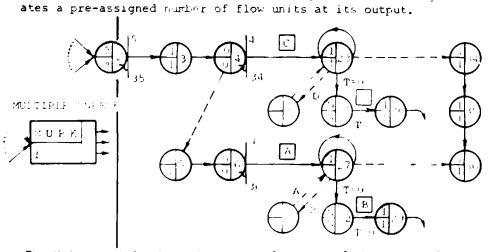
Initial Finite Pulse Train Generator: creates a sequence, consisting of a pre-assigned finite number of transactions, for entering into the system at random or constant time intervals.

UNPACK



UNPACK: divides the units of an entity flow into a larger number of subunits. A single flow unit arriving at this module generates a pre-assigned number of flow units at its output.

MULTIPLE UNPACK

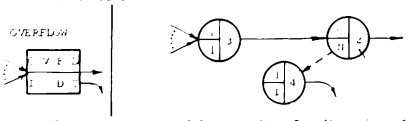


The Multiple Unpack combines the features of the UNPK and DIST modules. A single entity enters the module and searches for a server which is available to unpack it.

DELAY

DELAY: delays or holds up transactions at a particular point in the network.

OVERFLOW



OVerFLow: a queueing module providing for diversion of transactions to alternate locations whenever the queue is full.

STORAGE

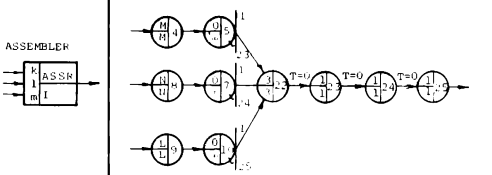
STORAGE: incoming transactions are accumulated ("stockpiled") and are withdrawn, individually or in groups, only as service facilities outside the module become available.

CONSOLIDATIVE

PACK

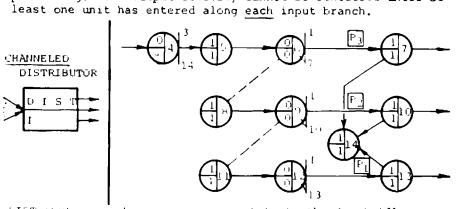
PACK: consolidates transactions; entering transactions can accumulate in the module until a specified capacity N is attained.

ASSEMBLER

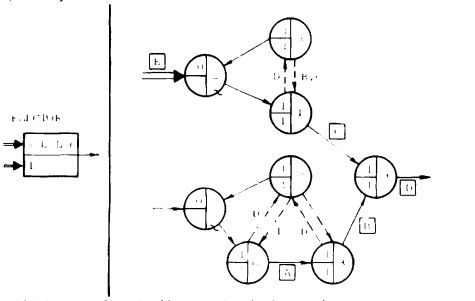


ASSembleR: combines a pre-defined set of entities for future processing. The output activity cannot be scheduled until at least one unit has entered along each input branch.

CHANNELED DISTRIBUTOR

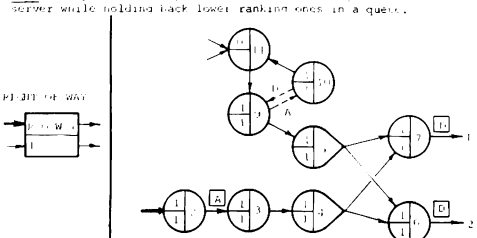


DISTributor: routes an incoming entity to the first idle server encountered in a sequential search of the processors being supplied by the module.



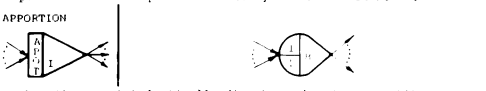
Collator: preferentially reroutes higher-ranking entities to a server while holding back lower ranking ones in a queue.

RIGHT OF WAY



Right of Way: preferentially controls entity flow through a traffic intersection with a high (H) and a low (L) priority entry path and with two probabilistically selected exit paths.

APPORTION



APportion: probabilistically directs each entering entity to one of its output branches.

The Assembler Module. A holding mechanism which entails network modification was used to construct a module which assembles various elements and holds them until the proper "mix" is achieved. The holding mechanism, Figure 8, entails network modification. This powerful feature in GERTS III Q for restructuring the network configuration was needed by the assembler module in order to function properly. The queue holds entities while a single entity traverses the branch from the queue to the next node. When an entity travels from node #6 to node #16, the "A" switch is activated, node #7 replaces node #5, and entities are trapped between node #4 and node #7. The holding continues until node #16 is realized. Then switch "D" is activated and the network returns to its original configuration. Erroneous results are obtained not only for the queue length in holding patterns but also make simple processes become relatively complex, thus making an improved mechanism mandatory. Needed was a simple model without holding patterns as well as network modification adequate to hold an entity in the queue until the signal for its release was given.

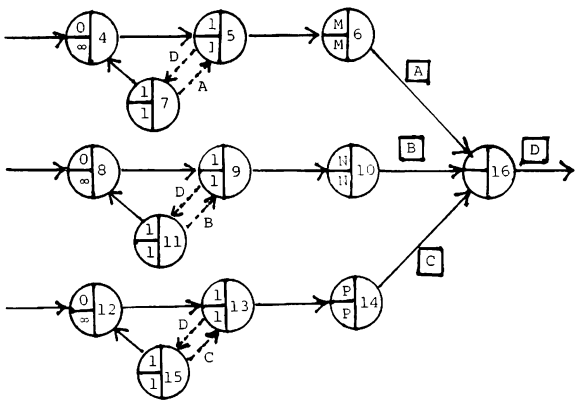


Figure 8

THE GATED-Q: A NEW ELEMENT

The gated queue node constitutes a new primitive, and is designed to hold entities until a signal from another part of the network arrives. Its function differs from the conventional queue node with respect to the ability of the queue node to hold entities until a signal arrives. Two essential characteristics of the gated queue result:

- The event immediately subsequent to the queue node is not the signalling event and,
- The queue and the signalling event are not necessarily connected except in the user's conception of the data input.

The gated Q node is characterized by:

- The number of items initially in the queue,
- The maximum number of items allowed in the queue,
- The node to which an item would "balk" or be diverted if it arrived when the queue was already full,
- The event node which opens the gate (the signal node)
- The number of items to let out of queue before closing the gate initially.

The gate may be specified as open or closed initially. Open is specified by a number "N" larger than zero. The gate closes after N items leave the queue; from then on each time an item arrives at the signal node one item may leave the queue and the gate closes again. When an item arrives at the signal node, the gate will open for one item, then close. Each arrival at the signal node permits one more item to leave the queue. If a new item arrives at the Q node when the gate is open it will proceed. If the gate is closed it is placed into the queue. Just as for the conventional queue when the maximum number is reached, newly arriving items must either balk to another node or from the system entirely if no balk node has been specified.

The New Distributor Module Using G-Q

The new distributor module is designed to route an incoming entity to the first idle server encountered in a fixed sequential search of the processor being supplied by the module. To illustrate the function of the module we shall follow the flow of a series of entities (Figure 9).

- The entity enters through the gated queue, node #4. The pulse passes through node #6, another gated queue, and the gate closes after the pulse has entered the processor. The queue is gated until the switch is later activated.
- A second pulse arrives, is balked to node #8, passes through Node #9, where the gate closes.
- When a singla node for a gated queue is realized the gate opens and lets out one pulse.
- A common node which has no affect on the program is set up, node #14. This node is used as the signal node for the gated queue, node #4.

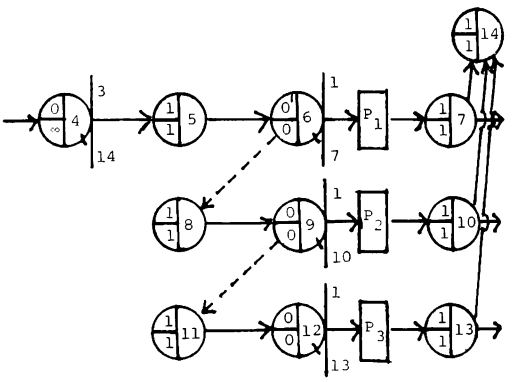


Figure 9

The New Assembler Module with G-Q

In Figure 10 the new assembler module is shown. It does not rely on holding patterns with their inherent network modification. One entity from each section is sent to node #22, closing the gate for that section until node #22 is realized.

The new assembler is simple in design and provides a true queue length. Waiting items stay in the queues instead of circulating continually.

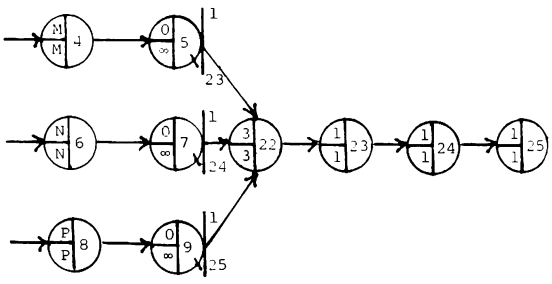


Figure 10

APPLICATION: A TYPICAL QUARRYING OPERATION

To illustrate planning considerations for the excavation and transport of raw materials a typical aggregate quarrying system utilizing one backhoe-excavator, several trucks, and one loader has been modelled. The system comprises four basic operations:[10]

- the backhow excavates aggregates which are stockpiled prior to the initiation of any loading operation,
- in addition to stockpiled aggregated, a loader and an empty truck must be available. Simultaneous availability of all three of these elements permits the loading operation to begin,
- once loaded, a truck transports aggregates from the excavation site to the plant site where they are unloaded into the main feeder of the rock plant, and
- the empty truck returns to the excavation site before it will again be available for further loading.

This system was simulated for an eight-hour shift; all operations still in progress at the end of an eight-hour shift were terminated. A state diagram summarizing this system is shown in Figure 12. The modular network is shown in Figure 13.

The nodal representation of this system is shown in Figure 14.

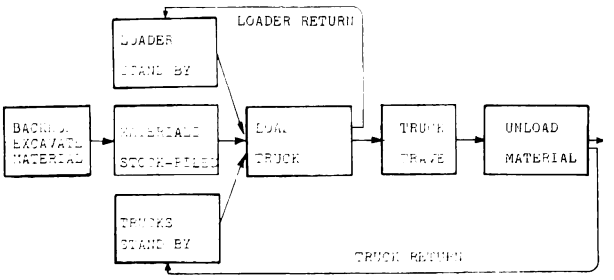


Figure 12

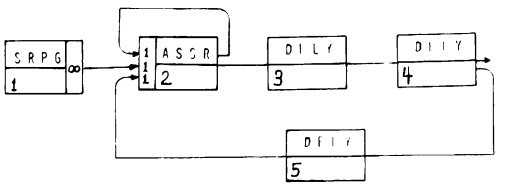


Figure 13

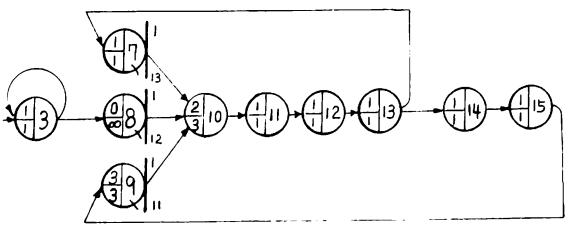


Figure 14

Simulation Significance

Several simulation runs were executed by the computer; results showing how the efficiency of the system varies with the number of trucks used.

Simulation permits the details of this dependence to be explored. Figure 15, a summary of results from several computer runs, shows how the number of trucks available affects the tonnage hauled from pit to plant. When too few trucks are available they become a bottleneck for the system. Conversely, as the number of trucks increases to five or more, the total tonnage hauled reaches an asymptotic value of 3200 tons per shift indicating that a new bottleneck has occurred elsewhere in the sysem.

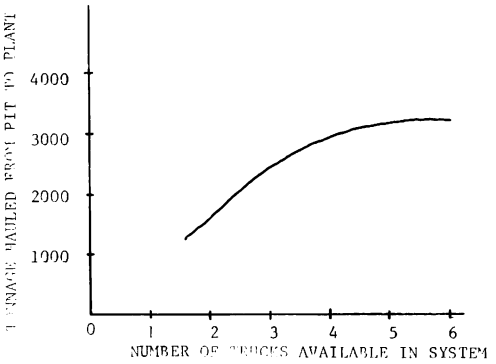


Figure 15

Simulation permitted the main interarrival time for trucks reaching the aggregate plant and the mean waiting time for truck loading and get-ready at the loading station to be examined. Table 2 shows that with four trucks in the system, the mean truck interarrival time is reduced to 5 minutes and the mean waiting time for truck loading and get-ready to about 2 minutes. With five trucks in the system, these times are improved. This improvement, however, is not as significant as that observed when three trucks were increased to four trucks.

Mean waiting time was also reduced although not significantly. For five trucks or more in the system these times became asymptotic to minimum values.

For this particular system four trucks are the optimal economic choice. Although five trucks will give better "performance" the increased output cannot offset the added costs. Simulation greatly facilitates sensitive decisions such as this.

Table 2

Simulation results:
number or trucks available in system

Number of trucks		3	4	5	6
Tonnage hauled per shift	(tons)	2400	2910	3210	3120
Truck inter-arrival time	μ(minutes)=	5.996	4.929	4.460	4.6017
	σ =	3.367	1.953	2.012	2.1286
Waiting time for truck loading	μ(minutes)=	3.034	1.834	1.471	1.5599
	σ =	3.190	1.547	1.515	1.6959
Material stockpiled	μ(truck-loads)=	14.55	0.229	0.055	0.0209
Trucks stand by	μ =	0.045	0.206	0.763	1.7828

CONCLUSION

The library of modules (Table 1) has been developed utilizing GERTS GQ. The development of the modular network modelling approach is motivated by the need for a clearly defined interface between the practicing engineer and computer readable simulation model.

The gated-queue node is a new element, developed from GERTS III Q. It was needed in order to model complex queueing situations, such as a single-queue multiserver process. The result, GERTS GQ, is a significant improvement in accuracy and responsiveness.

BIBLIOGRAPHY

1. Pritsker, A. A. B. and Happ, W. W., "GERT: Part I - Fundamentals," J. Ind. Eng., Vol. XVII, No. 5, 1966, pp. 267-274.

2. Pritsker, A. A. B. and Burgess, R. R., "The GERT Simulation Programs: GERTS III, GERTS III Q, GERTS III C, and GERTS III R." NASA/ERC Contract NAS-12-2113, Departmental Report, Virginia Polytechnic Institute, Dept. of Industrial Engineering, 1970.

3. Whitehouse, G. E., Systems Analysis and Design Using Network Techniques, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973.

4. Pritsker, A. A. B., "The Status of GERT," in H. Lombaers (ed.), Project Planning by Network Analysis, North-Holland Publishing Co., Amsterdam-London, pp. 147-153, 1970.

5. Akiba, Y., Blom, N., Duran, G., and Dabaghian, L., "GERTS GQ Simulator for Systems Simulations," Report to NSF Grant GY-11496, School of Engineering, California State University, Sacramento, CA, November 1974.

6. Akiba, Y., "Decision and Commodity Flow for Cargo Facilities," Proceedings, 3rd Annual Greater Los Angeles Area Transportation Symposium, Octiber 3, 1974.

7. Dabaghian, L., Akiba, Y., and Happ, W. W., "Network Modules to Simulate Quantized Entity Flow," Proceedings, Joint Automatic Control Conf., Austin, TX, June 19, 1974.

8. Akiba, Y., Dabaghian, L. and Happ, W. W., "Validation of Component and System Performance in Modular Queueing Networks," Proceedings, Seventh Asilomar Conference on Circuits, Systems, and Computers, Monterey, CA, November 1973.

9. Akiba, Y., "A Modular Technique for Synthesis of Activity Networks: Development and Applications," Master's Thesis, School of Engineering, CSUS, Sacramento, CA, May 1975.

10. Halpin, D. W. and Happ, W. W., "Network Simulation of Construction Operations," Proceedings, Third International Congress on Project Planning by Network Techniques, pp. 222-232, Stockholm, March 1972.