Order Number 9500727

Analytic solution of stochastic project management networks by the method of polygonal approximation

Lawrence, Frederick Peter, Ph.D.

Arizona State University, 1994

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

ANALYTIC SOLUTION OF STOCHASTIC PROJECT MANAGEMENT NETWORKS BY THE METHOD OF POLYGONAL APPROXIMATION

by

Frederick Peter Lawrence

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

ARIZONA STATE UNIVERSITY

August 1994

ANALYTIC SOLUTION OF STOCHASTIC PROJECT MANAGEMENT NETWORKS BY THE METHOD OF POLYGONAL APPROXIMATION

by

Frederick Peter Lawrence

has been approved

July 1994

APPROVED:

Jeffrey K Cochran ,Co-Chairperson

Dwayne A. Rollier ,Co-Chairperson

Philip M. Wolfe

Supervisory Committee

ACCEPTED:

D. A. Rollier

Department Chairperson

Jerry R. Thomas

Dean, Graduate College

ABSTRACT

A method for the linear polynomial (polygonal) approximation of continuous activity resource consumption (duration) distributions of stochastic project management networks is developed, derived from the spline approximations used in numerical differentiation and integration. It is the first new method for network approximation and reduction to be advanced since discretization, which was the basis for all previously developed algorithms. The method is successfully mated with three network reduction approaches - arc duplication, sequential approximation, and a heuristic for identifying the K most critical paths - to form the members of a new family of Polygonal Approximation and Reduction Techniques (PART). The development of a PART algorithm using "independent multiple arcs" (dual arcs) is the first successful implementation of an arc-duplication reduction method. Collectively, PART algorithms constitute an analytic reduction capability which is operative across the entire range of project management networks.

Algorithm validation is conducted within a design-of-experiments framework, which has not previously been employed in this context. Compared to other existing discretization-based methods, PART algorithms are demonstrated to be as accurate or more accurate in the characterization of the throughput distribution function. Accuracy is observed to be a function of network size, as driven by the number of activities much more strongly than the number of nodes, how great a challenge the activity distribution functions present to series-parallel reduction operations based on the polygonal approximation, and the number of partition classes. PART algorithms are shown to execute an order of magnitude faster than their competitors. Polygonal approximation and associated PART algorithms represent a new and innovative concept in the analytic arsenal aimed at stochastic project management networks. They have the potential to put the power of network management into the hands of anyone in possession of a desktop computing capability. In this research, they demonstrate their worthiness for continued development.

ACKNOWLEDGEMENTS

I am especially grateful for the guidance of and encouragement from my committee co-chairs, Dr. Jeffery K. Cochran and Dr. Dwayne A. Rollier. I thank my committee member, Dr. Philip M. Wolfe, for his advice and support during this research.

I also thank Jerry A. Fergeson and Patallen H. Shortell for their assistance in the development of portions of the FORTRAN computer code which implements the polygonal approximation and reduction technique, and John D. Miskulin for his assistance in the use of the SAS System for statistical analyses.

Finally, my appreciation goes to Faith for her patience and understanding.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

GLOSSARY

Activity resource consumption. The amount of a specified resource required to complete an activity, or event, in an activity network; the resource consumption required to transverse an arc between two nodes in an activity network. When the resource is time, the time required to complete an activity is referred to as either the activity completion time or the arc passage time.

Autocorrelation. A phenomenon usually observed in time-series data; the effect of the correlation (association) of a piece of data at time (t) with another piece of data at time (t+k).

Convolution. The additive combination of two or more probability distributions.

Equivalent arc. An arc resulting from the combination of two or more arcs in either series or parallel to a single arc.

Event. A well-defined occurrence in time by which all preceding (predecessor) activities have terminated and at which all subsequent (successor) activities are initiated (see Node).

False steady state. The apparent but erroneous appearance of a steady state (see Steady state).

Network/Activity network (AN). A graphical representation of the activities and the precedence relationships among the activities required to complete a project.

- Acyclic, directed/All-AND node network. An activity network with only AND nodes (see AND node).

- Deterministic activity network (DAN). An acyclic, directed activity network with only constant (single, fixed value) activity resource consumption distributions.

- Generalized activity network (GAN). An activity network with both stochastic (probabilistic) activity resource consumption distributions and probabilistic branching (see Probabilistic node).

- Mixed distribution network. An activity network with activity resource consumption distributions not limited to a single type of frequency/probability distribution.

- Probabilistic activity network (PAN). An acyclic, directed activity network with stochastic (probabilistic) activity resource consumption distributions.

- Single distribution network. An activity network with activity resource consumption distributions limited to a single type of frequency/probability distribution.

Node. A point in a network at which all predecessor activities terminate and all successor activities are initiated.

- AND node. A node:
  - realized only when all its predecessor activities terminating at the node are completed, and
  - all of whose successor activities emanating from the node are initiated when the node is realized.
- Probabilistic node. A node:
  - realized when only one (or more) of its predecessor activities is (are) completed (probabilistic input), and/or
  - one (or more) of whose successor activities is (are) initiated when the node is realized in accordance with specified probabilities (probabilistic output).

- Realized node. A node, all of whose required predecessor activities have been completed.

- Sink node. The last node in the network, representing the completion of the project.

- Source node. The first node in the network, representing the start of the project.

Piecewise polygonal approximation. The approximation of a curvilinear function by a piecewise polygonal function, defined on intervals within its domain by straight line segments.

Parallel arc set. A multiple arc set which connects two nodes, A and B, such that all arcs emanating from node A terminate at node B.

Separable network. A network which can be decomposed into subnetworks, each of which is either a parallel arc set or a series arc set; a network which can be reduced by successive applications of series/parallel reduction operations to an equivalent network with a single activity between source and sink nodes.

Series arc set. A multiple arc set in which only a single arc connects two successive nodes.

Steady State. A condition attained in a simulation when the process exhibits a stationary behavior as characterized by its parameters.

Stochastic (probabilistic) activity resource consumption. An activity resource consumption which behaves as a random variable, with a specified frequency/probability distribution, as opposed to a deterministic activity resource consumption whose value is constant.

CHAPTER 1
INTRODUCTION

Every project, whether large or small, requires a plan, a schedule, and a system of control. The plan specifies the objectives of the project and outlines the actions required to reach those objectives. The schedule translates the plan into specific assignments, indicating when each activity must be completed. The system of control assures that the assignments are carried out on schedule to reach the desired objectives (American Society of Tool and Manufacturing Engineers, 1967). One method of accomplishing these functions is through the use of an activity/project management network. The most common concept of a network display is that of a graphical representation of the relationships among various events, the occurrences in time at which activities start and are completed, involving directed arcs representing the activities and nodes representing the events (Moder and Phillips, 1964). A typical activity network is shown in Figure 1.

1.1 Role of Networks in Project Management

The principal benefit derived from network use is an increase in the manager's ability to control a project, because the network displays all the component interrelationships involved in the accomplishment of the project. As projects become larger and more complex, traditional management techniques tend to overload the manager with information. Network analysis is an attempt to ease this information overload and enable the manager to exercise some real control over the events for which he is responsible. Network analysis allows him to manage events, rather than permitting events to manage him. By defining specific priorities and utilizing resource consumption analysis and resource leveling before the project is placed into motion, the manager is able to see and comprehend intricate interrelationships that traditional project management techniques will

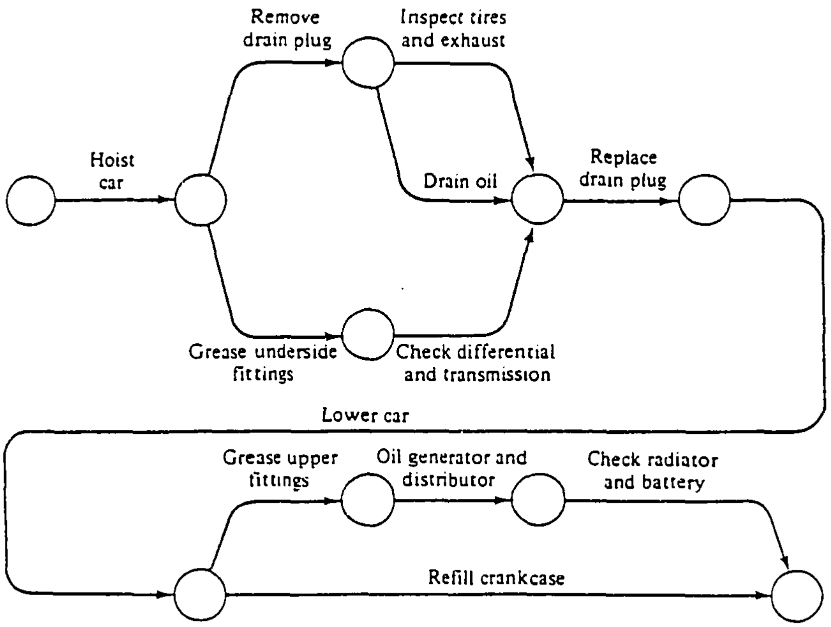

Remove drain plug
Inspect tires and exhaust
Hoist car
Drain oil
Replace drain plug
Grease underside fittings
Check differential and transmission
Lower car
Grease upper fittings
Oil generator and distributor
Check radiator and battery
Refill crankcase

Figure 1. A typical activity network.
[Adapted from Whitehouse (1973)]

not allow (Battersby, 1970). Idealistically, every project could be accomplished by network management. In reality, this is not the case, since there have been many problems associated with the use of networks (Welsh, 1965). This research is concerned with two of these problems:

(1) the determination of the activity resource consumption distribution through an activity/project management network (the throughput distribution), and

(2) the development of information which will be useful to the managers of projects whose networks have large numbers of nodes and activities without the determination of the throughput distribution.

Such resources typically consumed by activities include, principally, time, and also money, materials, personnel, and equipment.

1.2 Types of Networks

It is the concept of precedence among events, occurring naturally because of technological and other considerations, that distinguishes activity networks from other established models for the planning, scheduling, and controlling of activities, such as the Gantt chart and its varieties. An activity network is just a graphical representation of the activities and the precedence relationships among the activities which are required to complete a project. To accomplish each activity in a network requires the consumption of some amount(s) of one or more resources. In the development of network analysis techniques, the networks first studied were those whose activities all consume a single resource, usually thought of as time. Later, multiple resource consumption networks were studied as a separate problem class, usually modeled with the principal resource not subject to a capacity constraint and the other resources capacity-constrained.

If the principal resource of an activity/project management network is time, the precedence relationships among the activities determine which activities must be completed before other activities can be started; alternatively stated, the precedence relationships determine the order of occurrence of the network's events, the points in time at which all predecessor activities have been completed and, consequently, all successor activities may be started. No activity is permitted to loop back in an activity/project management network, since an activity cannot start at a later time and end at an earlier time; the network is said to be acyclic. Since each activity moves forward in time as it is being accomplished, it may be represented as a directed arc connecting its starting node and ending node (activity-on-arc representation). Hence, activity/project management networks are said to be acyclic, directed networks.

One of the first models for single resource networks, the Critical Path Method (CPM), assumes that the amounts of the resource consumed by all the network's activities are known deterministically, while another of the early models, the Program Evaluation and

Review Technique (PERT), assumes that the amounts of resource consumption are only known in a probabilistic sense. Later network analysis work expanded this original characterization into several directions and fundamentally novel approaches, to the point that it is now more convenient and precise to abandon the CPM-PERT categorization, while retaining the dichotomy between deterministic and stochastic (probabilistic) network models. Elmaghraby (1977) refers to the former as deterministic activity networks (DANs) and the latter as probabilistic activity networks (PANs).

All the activities in an activity/project management network must be completed in order for the project to be completed; that is to say, when each node in the network is realized upon the completion of all its predecessor activities, all its successor activities must be started. Since starting all of a node's successor activities corresponds to the AND logical operator, acyclic, directed networks are also called all-AND node networks. When this restriction is relaxed and probabilistic branching is permitted at the network nodes, Elmaghraby (1977) refers to such networks as general activity networks (GANs). The Graphical Evaluation and Review Technique (GERTS) was developed for the analysis of GANs.

1.3 Analysis of Stochastic Networks

Three approaches have been undertaken for the solution of the problem of determining the throughput distribution of a stochastic activity/program management network:

(1) exact analytic solution,

(2) Monte Carlo simulation, and

(3) numerical approximation and reduction.

If the network is deterministic (a DAN), the problem is trivial; CPM provides the exact throughput. If the network is stochastic (a PAN), the problem is a more challenging one. Attempts to develop exact analytic solutions to stochastic activity networks have suggested infeasibilities. Monte Carlo simulation has been successfully implemented in commercial software packages, but has limited practical application because of the lengthy run times required for moderate-to-large size networks. Mainframe computers or workstations are needed to support network simulation software. Most recently, numerical approximation and reduction has emerged as an attractive solution approach for small-to-moderate size networks. A theoretical reduction process based on arc duplication has been developed (Martin, 1965; Dodin, 1985b and 1985c) but not practically implemented. The only successful implementations of these techniques have involved the discretization of continuous activity resource consumption distributions (Dodin, 1980 and 1985a; Hagstrom, 1990). To achieve acceptable accuracies in the description of the throughput distribution, densely packed discretizations of the activity resource consumption distributions must be employed, which result in high computer memory and run time requirements, even for moderate size networks. While these implementations can be supported on mainframe computers and workstations, they are incompatible with the current generation of personal computers.

When networks have large numbers of nodes and activities, the problem of how to develop information useful to project managers is no small task. Because of their high computer memory and run time requirements, determining network throughput distributions by either simulation or numerical approximation and reduction techniques is unattractive, and often unrealistic. Schonberger (1981) advised that the "complex, delay-prone segments" of a network should be the focus of managerial attention. Activity criticality and path criticality indices have been suggested as quantitative tools to identify the paths most likely to become critical in a network To estimate path criticality indices, simulation can be used to determine the relative frequency with which each path in a network is critical. However, in addition to long simulation run times for moderate-to-large networks, the task of uniquely identifying and recording which path is critical in each repetition of a simulation is now added. Ragsdale (1989) has concluded that this quickly becomes impractical for problems of "realistic size." Dodin (1984) suggested a procedure for estimating the rankings of the path criticality indices for the K most critical paths in a network based on stochastic dominance and demonstrated it using numerical approximation and reduction with discretization of continuous activity resource consumption distributions. The high computer memory and run time requirements associated with discretization have limited the application of Dodin's method to mainframe computers and workstations.

1.4 Objectives of the Research

Numerical approximation and reduction techniques have recently emerged as attractive analytic tools to develop useful information for project managers from stochastic activity/project management networks. Although the work of Dodin (1980 and 1985a) and Hagstrom (1990) has shown that these techniques have the potential for widespread application in the project management arena, there has been a lack of research on the design of powerful, efficient techniques whose computational requirements are within the operating ranges of the computing capabilities of the vast majority of potential users.

The first objective of this research is to develop a new, novel, more capable, and more efficient numerical approximation method for continuous activity resource consumption distributions which, when combined with network reduction methods, could form algorithms whose accuracy, speed, and range of performance exceed those of existing procedures and whose computational requirements are within the operating envelopes of current desktop computing capabilities. The second objective is to develop three such network approximation and reduction algorithms, based on the new numerical approximation method combined with each of the leading network reduction methods: arc duplication, sequential approximation, and K most critical paths. Collectively these algorithms, which approximate:

(1) the throughput distribution for small-to-moderate size networks, and

(2) the K most critical paths for large size networks and the associated activity and node criticality indices,

constitute an analytic reduction capability which is operative across the entire range of activity/project management networks. The new numerical approximation method and its three associated algorithms are validated in performance tests against "strongly randomized" networks generated in accordance with an experimental design. Comparisons of the performance of these algorithms with the reported performance of competing network approximation and reduction procedures are also presented.

1.5 Outline of the Research

In this chapter, activity/project management networks were introduced, and the problems of providing useful information from the networks to project managers were discussed. Different approaches to network analysis were also briefly described. The remainder of this research is organized as follows:

Chapter 2 reviews the different approaches to the analysis of activity/project management networks.

Chapter 3 develops the new numerical approximation method for continuous activity resource consumption distributions and the algorithms which combine the new method with the three leading network reduction methods.

Chapter 4 validates the new method and its three associated network approximation and reduction algorithms.

Chapter 5 summarizes this research and gives recommendations for further research in this area.

CHAPTER 2

LITERATURE REVIEW

2.1 Deterministic Networks

Prior to the recent development of techniques for determining throughput activity resource consumption distributions, network analysis necessitated the introduction of simplifying assumptions. One early, common assumption is that all activity resource consumption functions are constant. This assumption is often stated in terms of time as the principal resource of interest: all activity completion times are fixed (Martin, 1965). This assumption is not always valid, since an activity resource consumption function often behaves as a random variable. The assumption of constant activity resource consumption functions reduces the practical utility of project management networks by failing to consider the consequences due to their stochastic nature.

However, when all activity resource consumption functions are constant, or are assumed constant, the network is deterministic (a DAN) and can easily be reduced by the Critical Path Method (CPM). CPM was developed within the construction industry where previous experience in similar work can be used to predict time durations and cost within tight ranges (Whitehouse, 1973). There is no universally acceptable description of CPM because many practitioners have developed their own versions of cost models based on activity networks; of these, the most well-known is probably the system developed by the General Electric Company. MacCrimmon and Ryavec (1964), Battersby (1967), Whitehouse (1973), Elmaghraby (1977) et al. give complete discussions.

Since all activity resource consumption functions are constant, the length of each path through the network from the source node (start node) to the sink node (termination node) is just the sum of the resource consumption constants of the activities on the path. The path with the greatest, or largest, length is called the critical path; any increase in any of the activity durations on the critical path will result in a corresponding increase in the total time required to complete the project. The length of the critical path(s) of a deterministic project management network is the total project time. The earliest time node i can be realized, $t_i(E)$, and the latest time node i can be released and the project completed on time, $t_i(L)$, define the slack time, s_i, of the node and the activity floats (slacks) of the activity a_{ij} between node i and node j and of length t_{ij}:

Slack time of node (event) i: $s_i = t_i(L) - t_i(E)$

Total float of activity a_{ij}: $TF = t_j(L) - t_i(E) - t_{ij}$

Safety float of activity a_{ij}: $SF = t_j(L) - t_i(L) - t_{ij}$

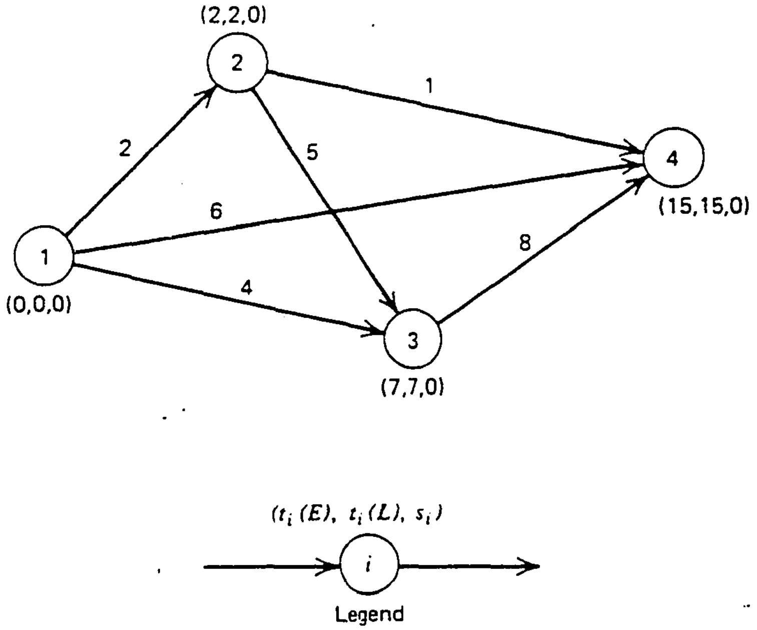
Free float of activity a_{ij}: $FF = t_j(E) - t_i(E) - t_{ij}$

Interference float of activity a_{ij}: $IF = \max[0, t_j(E) - t_i(L) - t_{ij}]$

A simple CPM network and its node slacks and activity floats are shown in Figure 2. The project completion time for this network is 15. Three activities possess positive float even though all the nodes have zero slack; for all activities, all four types of float are equal in this example.

2.2 Stochastic Networks

The first attempt to overcome the limited utility of constant activity resource consumption functions was the development of the Project Evaluation and Review Technique (PERT), in which all activity resource consumption functions are assumed to follow a single type of probability distribution and all source-to-sink paths through the network are assumed independent. There are three types of distributions used most frequently with PERT networks: beta, uniform, and normal. The beta distribution is the



Arc	Float
(1,2) (2,3) (3,4)	0
(1,3)	3
(1,4)	9
(2,4)	12

Activity floats S_{ij}^k.

Figure 2. A CPM network with node slacks and activity floats.
[Adapted from Elmaghraby (1977)]

most common single distribution used with PERT (Whitehouse, 1973). PERT uses three time estimates for each activity:

a, the optimistic time, which should have only a very low probability of occurring;

m, the most likely time, the mode of the beta distribution; and

b, the pessimistic time, which also should have only a very low probability of occurring.

PERT then assumes that these three estimates can be used to describe a beta distribution for each activity duration, as shown in Figure 3. The expected time for an activity, t_e, the standard deviation, σ_{t_e}, and the variance, V_e, are then computed as:

$$t_e = \frac{a + 4m + b}{6}$$

$$\sigma_{t_e} = \frac{b - a}{6}$$

$$V_L = \left(\frac{b-a}{6}\right)^2$$

A simple PERT network and its activity parameters (a,m,b) are shown in Figure 4. To permit finding the mean, μ, and the variance, σ^2, of the earliest, T_E, and latest, T_L, node (event) times, PERT assumes that the individual activity durations are statistically independent. Then, the mean and variance of T_E and T_L are calculated under the further assumption that the path to a given node with the largest expected time will always dominate all the other paths to that node. Slack at a node is similarly defined as in CPM:

$$\text{Slack at a node} = \mu_{T_L} - \mu_{T_E}$$

The critical path is the path(s) with the longest expected time through the network. All nodes (events) which have zero slack appear on a critical path. The calculations for the PERT network in Figure 4 are shown in Table 1.

Once the expected earliest time of a node (event), μ_{T_E}, and its standard deviation, σ_{T_E}, have been determined, probability theory can be used to calculate the chances of meeting a specific scheduled time, T_s, for the node. By the central limit theorem, the earliest completion time of a node is assumed to be normally distributed with mean μ_{T_E} and standard deviation σ_{T_E}. The probability of meeting the desired schedule time T_s is the area under the normal curve to the left of T_s, as shown in Figure 5. The random variable

$$Z = \frac{T_s - \mu_{T_E}}{\sigma_{T_E}}$$

is distributed as a standard normal, and hence the desired probability can be obtained directly from a standard normal table. Similarly, the probability of the project completion time falling between a lower value, T_l, and an upper value, T_u, can be computed.

The assumptions of a single type of distribution and independent source-to-sink paths have historically produced inaccuracies. MacCrimmon and Ryavec (1964) conducted an analytic study of the PERT assumptions and reported a Monte Carlo simulation of one

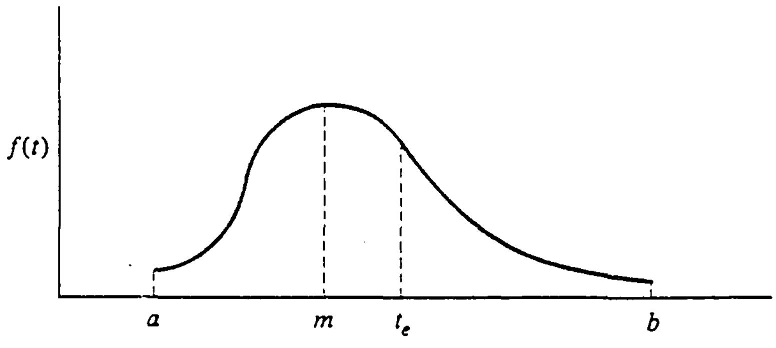


Figure 3. A PERT beta distribution.
[Adapted from Whitehouse (1973)]

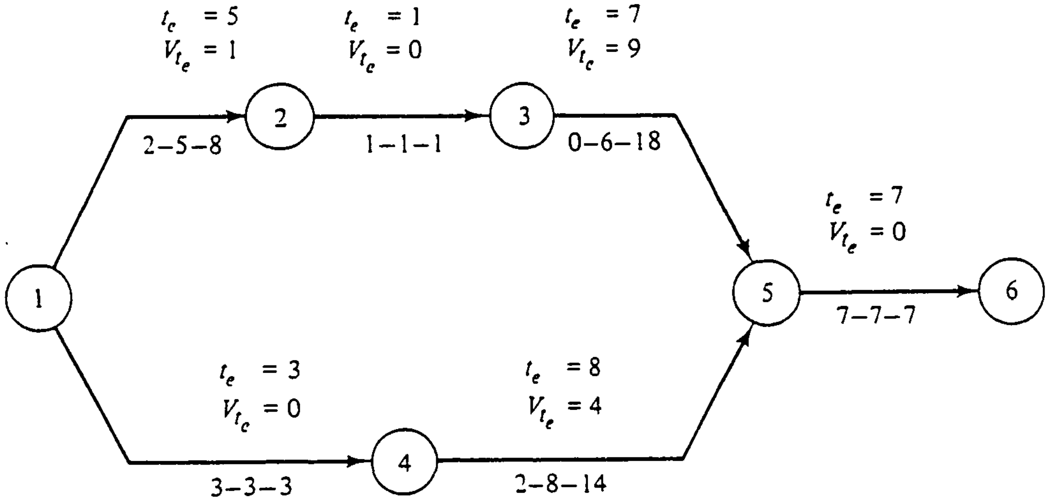


Figure 4. A PERT network.
[Adapted from Whitehouse (1973)]

Table 1. PERT Calculations.
[Adapted from Whitehouse (1973)]

	Earliest Time (T_E)		*Latest Time* (T_L)		
Event No.	*Expected Value* μ_{T_E}	*Variance* $\sigma^2_{T_E}$	*Expected Value* μ_{T_L}	*Variance* $\sigma^2_{T_L}$	*Slack*
1	0	0	0	10	0
2	5	1	5	9	0
3	6	1	6	9	0
4	3	0	5	4	2
5	13	10	13	0	0
6	20	10	20	0	0

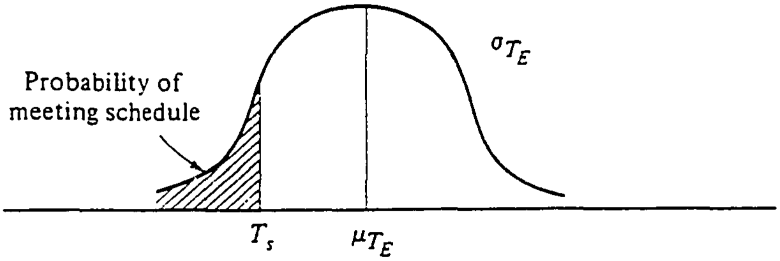


Figure 5. Calculation of the probability of meeting schedule.
[Adapted from Whitehouse (1973)]

selected PERT network using a beta distribution which evidenced a 34% error in estimation of the first moment (mean). This is typical of the errors experienced with PERT.

When the PERT assumption of a single type of distribution of activity resource consumption (duration) is relaxed and activities are permitted to take on arbitrary distributions, a project management network becomes a general stochastic project management network. Stochastic project management networks constitute a subclass of the class of stochastic networks. Stochastic networks have the following properties:

> 1. Each network consists of nodes denoting logical operations and transmittances.
> 2. A transmittance has associated with it a probability that the activity represented by the network will be performed.
> 3. Other parameters describe the activities which the transmittances represent. These parameters are usually additive, such as time or cost.
> 4. A realization of a network is a particular set of transmittances and nodes which describes the network for one experiment
> 5. If the time associated with a transmittance is a random variable, then a realization also implies that a fixed time has been selected for each transmittance.
>
> (Whitehouse, 1973)

Various approaches to the solution of stochastic networks have been investigated. Eisner(1962) suggested the use of logical elements in PERT-type networks. Elmaghraby (1964, 1966) developed a notation for a multiparameter branch network and Eisner's logical elements, and developed an algebra and coined the term generalized activity networks (GANs) to describe such networks. His algebra is limited to branches with constant times. Huggins (1957) and Howard (1960, 1964) employed flowgraphs to represent and analyze probabilistic systems. This early work led Pritsker, Happ, and Whitehouse (1966) to introduce the graphical evaluation and review technique (GERT), a procedure which combines the disciplines of flowgraph theory, moment generating functions, and PERT to obtain solutions to stochastic problems. In GERT networks, nodes are constructed from the combination of an input side and an output side. Three input sides are possible: EXCLUSIVE-OR, INCLUSIVE-OR, and AND; and two output sides: DETERMINISTIC and PROBABILISTIC. Project management (PERT-type) networks are stochastic (GERT-type) networks with all nodes of the AND-DETERMINISTIC combination, and as such are often referred to as stochastic (probabilistic) all-AND node networks. Using principles of flowgraph theory and moment generating functions, Pritsker and Happ (1966) demonstrated that the subclass of GERT networks consisting of all EXCLUSIVE-OR node networks can be exactly solved analytically.

2.3 Attempts at Analytic Solution of Stochastic Networks

2.3.1 Exact Solutions

Pritsker and Happ's solution of the GERT networks subclass of all EXCLUSIVE-OR node networks sparked efforts across a variety of fronts to find exact analytic solutions for other subclasses of GERT networks, in particular the stochastic all-AND node, project management networks. These attempts have either proven infeasible or met with only limited success. Pritsker (1966) suggested the use of Laplace transforms, which involve integrations in the complex plane. Although Whitehouse (1973) suggested several avenues by which the difficulties in manipulating Laplace transforms may be overcome, this approach has been abandoned, primarily because of the inability to develop a workable computer implementation. Their suggestion of the use of the Mellin transform instead has not been pursued, probably for the same reason. Frank (1969) prescribed a theoretical procedure for analytically determining the minimal throughput distribution which involves characteristic functions and requires transform inversions, but it has never been successfully implemented.

Charnes, Cooper and Thompson (1964) suggested that an exact expression of the probability density function of a network's throughput activity resource consumption could be obtained by deriving density functions conditioned on independent arcs which are common to more than one source-to-sink path. The first implementation of this idea was provided by Martin (1965), who presented an algorithm for determining the exact probability density function of throughput consumption for one specified subclass of acyclic, directed (all-AND node) networks involving only uniformly distributed activity resource consumptions (times). Hartley and Wortham (1966) performed analysis on PERT networks in a method conceptually similar to Martin's, but their technique was limited to networks that could be reduced to an equivalent arc between source node and sink node by only series, parallel or Wheatstone Bridge reduction operations. Ringer (1966) extended the Hartley-Wortham approach by introducing procedures to reduce double Wheatstone Bridge and criss-cross subnetworks. Additionally, he suggested an approach for reducing "general PERT" networks (PANs); however, Sigal (1977) illustrated an inconsistency for one subnetwork class. Burt (1969) concluded analytic reduction of an arbitrary PAN to be intractable. Hopfinger and Steinhardt (1975, 1976) developed a theoretical, backward dynamic programming-like algorithm for the exact solution of a PAN, but it has not been operationalized because of difficulties in evaluating associated multiple integrals. In the late 1970s, the search for exact analytic solutions of stochastic project management networks was abandoned.

2.3.2 Approximation Solutions

The difficulty in developing any exact analytic solution of stochastic project management networks led investigators to resort to either approximating or bounding the activity resource consumption throughput distribution function. With Garman, Burt (1971) proposed a conditional Monte Carlo simulation approach, which Garman (1972) further developed. Their approach was based on an expression for the throughput distribution function which is conditioned on the common arcs; the expression is evaluated with Monte Carlo simulation by sampling on the common arcs. In an investigation of the stochastic shortest route problem, Sigal (1977) determined the minimum number of activities which must be conditioned upon with uniformly directed cut sets. In the calculation of path optimality indices, he overcame difficulties in evaluating associated multiple integrals only through the use of discretization approximations, after a manner discussed by Haber (in Hopfinger and Steinhardt, 1976). Elmaghraby (1977) critically reviewed these procedures. Sigal, Pritsker, and Solberg (1979) developed a conditional Monte Carlo simulation procedure similar to that of Burt and Garman. They used the path independent arcs of the maximum directed cutset to replace the common arcs in Burt and Garman's procedure. O'Conners (1981) developed an analytic expression for the throughput distribution by conditioning on most of the common arcs, then using complete enumeration to evaluate the simplified expression; however, his procedure works only for networks all of whose activities have durations that have discrete densities.

All of these procedures are conceptually sound; but, with the exceptional of crude Monte Carlo simulation (VanSlyke, 1968), they are not practical. They are of limited value for all but small networks, because applying any of these procedures requires:

> 1. The identification of all the paths in the network, which can be a burdensome task.
> 2. Writing an analytic expression for the distribution function by conditioning on the common arcs (in the case of Burt and Garman), or on the arcs which are not in the maximum directed cutset (in the case of Sigal et al.), or on most of the common arcs (in the case of O'Conners). It is not easy to develop such an expression for most non-trivial stochastic networks.
> 3. The analytical expression developed in 2.[above] is evaluated using complete enumeration (in the case of O'Conners), or using Monte Carlo sampling over the arcs conditioned upon.
>
> (Dodin, 1985a)

The difficulty in 3. [above] stems from the large number of arcs used in the conditional probability expression. The ratio of common (or arcs not in the maximum directed cutset) to the total number of arcs in the network approaches one as the network size increases. In Dodin (1985a), this ratio was reported greater than 0.9 for all strongly randomly generated networks with only 10 nodes and 15 arcs.

Dodin (1980, 1985a) developed the first practical procedure for approximating the throughput distribution based on the method of sequential approximation. An extension of the procedure determines the k most critical paths in a PERT-type network (Dodin, 1984). He also developed a theoretical reduction process based on arc duplication which converts an irreducible (nonseparable) network into an "equivalent" separable network, but has not provided a practical implementation (1985b, 1985c). The process extends Martin's (1965) concept of "dual arcs" and relies on the existence of Garman's *a* and *b* activities of irreducible networks, but Garman's work (1972) is not credited. Hagstrom (1988) presented results for computational complexity for two problems associated with PERT networks whose activities have discrete time-cost distributions: computing a [single] value of the cumulative distribution function of project duration/cost, and computing the mean of this distribution. The former is #P-complete, the latter is at least as hard, and neither can be computed in time polynomial in the number of points in the range of the project duration/cost unless P=NP. She also developed a computationally demanding algorithm based on ordered recursive conditioning of the task durations; the technique is similar to pivoting (factoring, or backtracking) in reliability computations (Hagstrom, 1990). Both Dodin's and Hagstrom's methods require the discretization of all continuous distributions.

2.4 Elements of Stochastic Network Reduction

In graph theoretic terms, a stochastic project management network is a directed, connected, acyclic graph, $G = (N, A)$, composed of a set of n nodes, N, and a set of m arcs (activities), A, with one source node (starting node) and one sink node (terminal node). The nodes are numbered such that an arrow leads from a smaller numbered node to a larger one. By convention, the source node is node 1 and the sink node is node n. Every chain between the source node and the sink node is a path directed from node 1 to node n. (Martin, 1965; Horowitz and Sahni, 1976).

A subnetwork, (N_1, A_1), of a network, (N, A), is a connected network such that $N_1 \subset N$ and $A_1 \subset A$. A subnetwork of a directed, acyclic network is also a directed, acyclic network. Two subnetworks are said to be connected in series if the sink node of one subnetwork is the source node of the other. Two or more subnetworks are connected in parallel if they all have the same source node and sink node. A series subnetwork, (N, A), is a directed, acyclic network with the following properties:

1. Exactly one node, $S_s \in N$, has exactly one arc, $a_1 \in A$, leaving it, while, if any arcs enter S_s, they are not members of A. S_s is the source node of the series subnetwork.
2. Exactly one node, $T_s \in N$, has exactly one arc, $a_m \in A$, entering it, while, if any arcs leave T_s, they are not members of A. T_s is the sink node of the series subnetwork.
3. Every other node, $n \in N$, is entered by exactly one arc, $a_i \in A$, and has exactly one arc leaving it, $a_j \in A$.

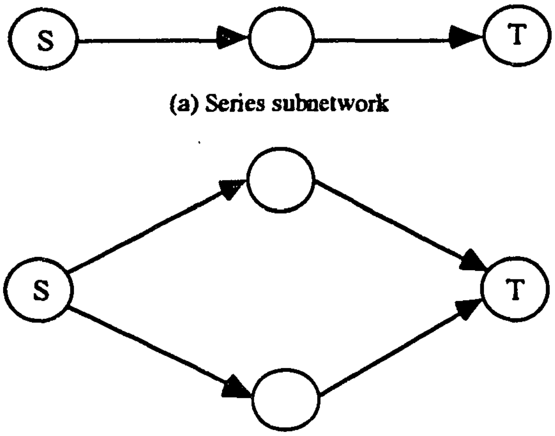
(Martin, 1965)

A parallel subnetwork, (N, A), is a directed, acyclic network with the following properties:

1. Exactly one node, $S_p \in N$, has a set of more than one arc, $\{a_1, \dots, a_k\} \subset A$, leaving it, while, if any arc enters S_p, it is not a member of A. S_p is the source node of the parallel subnetwork.
2. Exactly one node, $T_p \in N$, has a set of more than one arc, $\{a_i, \dots, a_m\} \subset A$, entering it, while, if any arcs leave T_p, they are not members of A. T_p is the sink node of the parallel subnetwork.
3. S_p and T_p are connected by two or more series subnetworks, all having the common source node S_p and all having the common sink node T_p.

(Martin, 1965)

A series subnetwork has a single path from source node to sink node, while a parallel subnetwork has two or more nonintersecting paths from source node to sink node. A series subnetwork is shown in Figure 6 (a); a parallel subnetwork is shown in Figure 6 (b). Using a recursive definition, a series-parallel subnetwork is a series subnetwork, a parallel subnetwork, or two series-parallel networks connected in series or in parallel.



(a) Series subnetwork

(b) Parallel subnetwork

Figure 6. Subnetworks.

It is readily seen that a series-parallel network is a directed, acyclic network with a source node and a sink node. Moreover, every directed, acyclic network (N, A) contains a maximal series-parallel subnetwork (N_1, A_1), in the sense that, for every path, π, contained in (N, A) but not in (N_1, A_1), addition of π to (N_1, A_1) results in a network that does not have the series-parallel property. Thus, every directed, acyclic network may be regarded as consisting of a series-parallel kernel together with cross-connecting subnetworks. (Martin, 1965)

2.4.1 Series-Parallel Reduction Operations

Series subnetworks and parallel subnetworks can be reduced to equivalent arcs between their source nodes and their sink nodes through the use of the mathematical operators, convolution and maximum (multiplication), respectively. For simplicity of further discussion, suppose that the activity resource consumed by a project management network is time.

<u>Series Reduction Operation</u>

Let t_{ij} be the time through arc (activity) a_{ij} of a series subnetwork, with density function $f_{ij}(\cdot)$. The time through a series subnetwork consisting of n arcs is

$$T_n = t_{ij_1} + \ldots + t_{ij_n}$$

The series reduction operation transforms a series subnetwork into a single equivalent arc with passage time T_n and density function $g_n(\cdot)$ by means of successive applications of the convolution operator. Let $g_k(\cdot)$ be the density function of

$$T_k = t_{ij_1} + \ldots + t_{ij_k} \qquad k = 1, \ldots, n$$

Then

$$g_1(t) = f_1(t) \text{ and } g_k(t) = \int_{-\infty}^{\infty} f_k(\tau) g_{k-1}(t - \tau) d\tau \qquad k = 2, \ldots, n.$$

Parallel Reduction Operation

The operation of parallel reduction transforms a parallel subnetwork consisting of n paths to a single equivalent arc with passage time T_n and density function $g_n(\cdot)$. The series subnetworks constituting each path are first transformed, by means of series reduction operations, to single equivalent arcs, the i th such arc having passage time t_i and (cumulative) distribution function $F_i(\cdot)$. Then the time through the parallel subnetwork is

$$T_n = \max[t_1, \ldots, t_n]$$

with distribution function

$$G_n(t) = P[T_n \leq t] = \prod_{i=1}^{n} F_i(t)$$

and density function

$$g_n(t) = G_n'(t) = \frac{d}{dt} G_n(t).$$

2.4.2 Reducibility

If, in an acyclic, directed network, the conditions of either a series reduction operation or a parallel reduction operation or both exist, then the network is termed ***reducible (separable)***; otherwise, it is termed *irreducible (nonseparable)*. If it is reducible to the single equivalent activity $(1, N)$ between the source node and the sink node, then it is termed *completely reducible (completely separable)*. If it is completely reducible, then its throughput distribution can be developed by the successive applications of the series-parallel reduction operations which will reduce the network to the equivalent activity $(1, N)$. Difficulty in determining the throughput distribution arises when the network is irreducible. Three different methods have been developed to reduce irreducible networks: "independent multiple arcs" (dual arcs), sequential approximation, and ordered recursive conditioning. These are discussed in Section 2.5.

2.4.3 Size and Complexity

The size of the network (N,A), composed of n nodes and m arc (activities), is (n,m). There is no general agreement in the literature, however, as to how to quantify how difficult the network is to reduce, if it is irreducible. We propose the following definition of complexity, motivated by the concept of cardinality from mathematics.

To motivate our definition, it is useful to consider a network (N, A) in two alternate representations: first, as a tree diagram; second, as an adjacency matrix. We illustrate both representations with the example network shown in Figure 7.

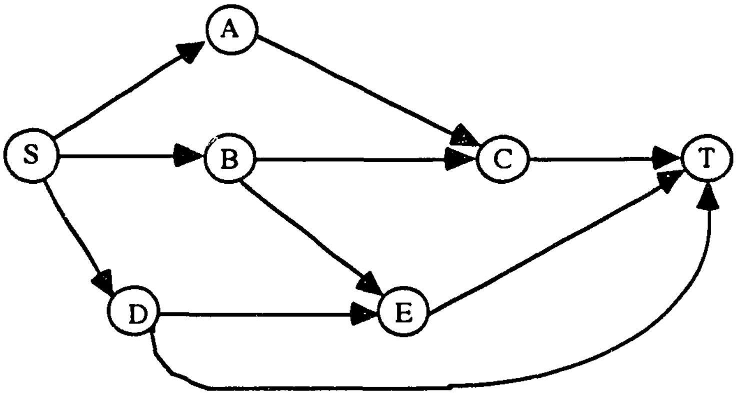


Figure 7. Network with cross-connections.
[Adapted from Martin (1965)]

S is the source node and T is the sink (terminal) node. One can depict the network as a tree in the following manner. Let the network source, S, be the only node at the topmost level of the tree, and let the next level consist of all nodes, N_j, such that $a_{s_j} \in A$, i.e., each of these nodes branches from S in the tree. Then, if node N_i is in the (k-1)st level of the tree, the kth level of the tree includes all nodes, N_j, such that $a_{ij} \in A$. All such nodes, N_j, branch from N_i in the tree. The tree for our example network is shown in Figure 8. When two (or more) of the nodes labeled N_j branch from a common node, N_i, at some level higher than that immediately preceding, then N_j is the terminal node for a cross-connection between two (or more) paths emerging from N_i. This is illustrated by nodes C and E in Figure 9. When two (or more) of the nodes labeled $N_j, N_k, ...$ branch from a common node, N_i, then N_i is the starting node for a cross-connection between two (or more) paths emerging from N_i. This is illustrated by nodes B and D in Figure 10.

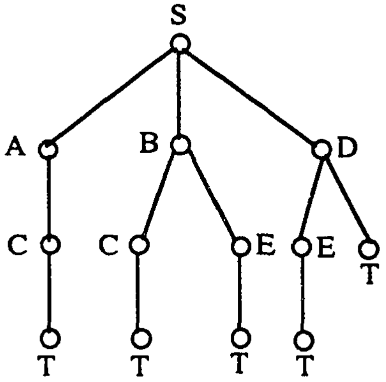


Figure 8. Tree diagram.

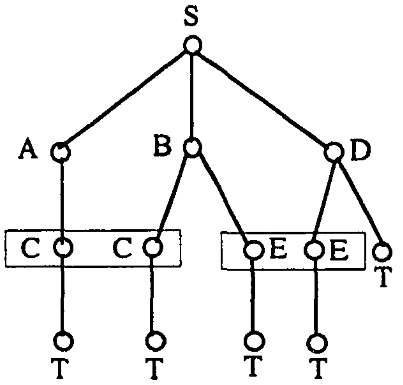


Figure 9. Tree diagram with cross-connections (1).

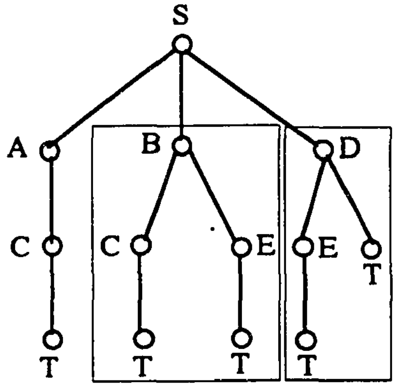


Figure 10. Tree diagram with cross-connections (2).

	S	A	B	C	D	E	T
S	X	1	1		1		
A		X		1			
B			X	1		1	
C				X			1
D					X	1	1
E						X	1
T							X

Figure 11. Adjacency matrix.

Now depict the network as an adjacency matrix M as shown in Figure 11,where $M_{ij} = 1$ if and only if the activity a_{ij} is in the network, i.e., $a_{ij} \in A$. The ***in-degree*** of a node is defined as the column sum of M for the node; the ***out-degree*** is the row sum.

We may define the *complexity* of the network (N, A) to be the 2-tuple $(c, \aleph_c)$, where c is the number of terminating cross-connections in the network and $\aleph_c$ is the cardinality of these cross-connections, i.e., the maximum number of in-degrees among these c cross-connections. The number of terminating cross-connections c is the number of columns in the adjacency matrix M whose column sums are ≥2, excluding the terminal node column, and $\aleph_c$ is the maximum of these c column sums. In the example, the $(c, \aleph_c)$-complexity of the network is (2,2).

Alternatively, we may define the *complexity* of the network (N, A) to be the 2-tuple $(c', \aleph_c')$, where c' is the number of starting cross-connections in the network and $\aleph_c'$ is the cardinality of these cross-connections, i.e., the maximum number of out-degrees among these c' cross-connections. The number of starting cross-connections c' is the number of rows in the adjacency matrix M whose row sums are ≥2, excluding the source node row, and $\aleph_c'$ is the maximum of these c' row sums. In the example, the $(c', \aleph_c')$ -complexity of the network is also (2,2).

2.4.4 Approximation of Activity Resource Consumption Distributions

The two (of the three) numerical approximation and reduction techniques which have been successfully implemented for stochastic project management networks, sequential approximation and ordered recursive conditioning, both require as the first step the discretization of all continuous activity resource consumption (arc passage time) distributions in the network. The discretization is done by approximating each continuous distribution function by a discrete function represented by the set of ordered pairs $F_{ij}(t) = \{(t_k, p(t_k))\}$. Dodin (1980) investigated three approximation procedures:

The $C(ij)$ Method

Let the cardinality $CF(a_{ij})$, the number of ordered pairs in the discretization of the continuous discretization of a_{ij}, be $CF(a_{ij}) = C(ij)$. Then $F_{ij}(t)$ has $2C(ij)$ unknowns: $C(ij)$ realizations, and $C(ij)$ corresponding probabilities. The first $2C(ij)$ moments of the continuous distribution can be used to construct the following system of $2C(ij)$ nonlinear equations:

$$\sum_{k=1}^{C(\cdot)} t_k^n p(t_k) = \int_{-\infty}^{\infty} t^n dF(t) = E(T_{ij}^n) \qquad \text{for } n = 0,1,2,\ldots,2C(ij)-1$$

where $E(T_{ij}^n)$ is the n^{th} moment of $F_{ij}(t)$. Methods known to solve systems of non-linear equations, Gaussian quadrature (Hamming, 1962) and Brown's method (IMSL, 1991), fail to solve this system due to the difference between $E(T_{ij}^0)$ and $E(T_{ij}^{2C-1})$ which can be quite large. Dodin reported that neither method succeeded for $C(ij) > 8$. This difficulty led to obtaining $F_{ij}(t)$ by either assuming that the masses $\{(p(t_k)\}$ or the realizations $\{t_k\}$ are given.

Equal Distances Method

Based on the activity distribution, the minimum and maximum realization values, a and b, can be determined. Then, by the use of an appropriate spacing Δ, depending on the desired accuracy of the discretization, the range $(b-a)$ can be subdivided into equal intervals:

$$t_k = a + \Delta(k-1) \text{ for } k = 1,\ldots,C(ij)+1 \text{ where } C(ij) = \left\lceil \frac{b-a}{\Delta} \right\rceil$$

Dodin (1980) suggested that a and b be determined such that:

$$\mathrm{P}(T < a) = \mathrm{P}(T > b) = \text{small value} \approx 0.005$$

The corresponding probabilities are determined according to:

$$p(t_k) = \int_{t_k-\Delta/2}^{t_k+\Delta/2} dF(t) \qquad \text{for each } k = 2,\ldots,C(ij)-1$$

$$\text{and } p(t_1) = p(a) = \int_{-\infty}^{a+\Delta/2} dF(t) \text{ and } p(t_{C(ij)}) = p(b) = \int_{b-\Delta/2}^{\infty} dF(t)$$

For a small Δ (large $C(ij)$),the determination of the probability can be approximated by $p(t_k) = \Delta f(t_k)$ for each $k = 1,\ldots,C(ij)$ where t_k is the center of the k^{th} interval, i.e., $t_k = a + \Delta(k-1)$ and $f(t) = dF(t)$. This approach treats all points in the domain of the random variable in a uniform fashion, i.e., the domain is partitioned into equal distances, which makes this discretization suitable for use with the Fast Fourier Transformation method for convolutions. This discretization method is convenient for some distributions such as the uniform and the triangular, and some other distributions whose skewness or peaks are not very acute. If sharp peaks are present, such as in the case of the exponential distribution with large scale parameter, or the normal distribution with a small standard deviation, then very small values of Δ must be used to minimize approximation errors, resulting in densely packed discretization points across the domain of the distribution. This drawback prompted consideration of using equal probabilities.

<u>Equal Probabilities Method</u>

Again, the minimum and maximum realization values, a and b, can be determined, as in the first method. Then $F_{ij}(t)$ is determined according to:

$$\Delta = p(t_k) = \frac{1}{C(ij)} \qquad \text{for a given } C(ij)$$

and

$$t_k = F^{-1}(\sum_{j=1}^{k} p(t_j) - \Delta / 2)$$

using the continuous distribution function $F(t)$. This method is suitable for all continuous distributions. However, it is not easy, and/or it can be time consuming, to invert $F(t)$ for some distributions.

Consequently, Dodin (1980, 1985a) employed a hybrid discretization approach in his implementation of sequential approximation. He limited the use of the equal probabilities method to the exponential distribution; the equal distances method was used for the uniform, triangular, normal, gamma, and beta distributions. Table 2 presents computational experience with this hybrid approach on a UNIVAC 1100/80.

Table 2. CPU Time Required for Discretization.
[Adapted from Dodin (1985)]

Distribution	CPU time of one activity (1×10^{-4} second)	CPU time of 100 activities with different continuous df's (in seconds)
Uniform	2	0.02
Triangular	6	0.06
Normal	28	0.28
Exponential	12	0.12
Gamma	116	1.16
Beta	380	3.80

Dodin (1985a) reported that the accuracy of the approximation of the throughput activity resource consumption distribution using sequential approximation is enhanced if the error in discretizing the continuous activity distributions is controlled, by increasing the cardinality of the set $F_{ij}(t)$. He based this conclusion on observations of the effect of increasing the number of discrete approximation points from 20 to 30.

Hagstrom (1990) restricted her implementation of ordered recursive conditioning to networks with discrete, independent probability distributions for task durations. If the method is applied to networks with continuous distributions, those distributions must be discretized; however, she did not specifically address how that should be done.

2.5 Stochastic Network Reduction Techniques

2.5.1 "Independent Multiple Arcs" (Dual Arcs)

Martin (1965) was the first to suggest a method for reducing irreducible networks:

> If, on the other hand, two of the nodes labelled N_j branch from a common node, N_i, at some higher level [in a tree diagram of the network] than the immediately preceding, then N_j is the terminal node for a cross-connection between two paths emerging from N_i. ... The cross-connection can be removed by duplicating each path of the tree from N_i to N_j that cannot be immediately collapsed by means of the series reduction operation. New nodes and arcs are created as necessary in the network. Each newly created arc corresponds to an arc on one of the original paths from N_i to N_j; the arcs of such a pair are called *dual.* The members of a pair of dual arcs both represent the same random variable and are labelled so as to indicate this correspondence. Dual arcs are, of course, not independent of one another.

Although his "independent multiple arcs" (dual arcs) method is conceptually sound, he was unable to develop the concept to the point that he could effect a practical implementation.

While refining a conditioned sampling approach to simulation of stochastic networks which he had earlier developed with Burt, Garman (1972) proved a theorem which is key to the implementation of the "independent multiple arcs" (dual arcs) method:

> Theorem 1. Any series-parallel reduced network which is not trivial (i.e., does not consist of only one activity) will possess (1) at least one activity *a* such that *a* has more than one successor while each of its successors has only *a* as a predecessor; and (2) at least one activity *b* such that *b* has more than one predecessor while each of its predecessors has only *b* as a successor.

Dodin (1985b, 1985c) completed the theoretical development of the "independent multiple arcs" (dual arcs) method. The following notation was used in his development:

a_{ij} = arc $(i, j) \in A$: starting at node i and ending in node j where i less than j, it is the only arc connecting node i to node j in the network.

T_i = random variable , the realization times of node i.

$A(i)$ = $\{j \in N:(i, j) \in A\}$, the set of nodes succeeding node i.

$B(i)$ = $\{k \in N:(k,i) \in A\}$, the set of nodes preceding node i.

$P(j)$ = $\{k \in N:k$ less than i and k connects to i by a path$\}$, i.e. the set of nodes that precede node i and connect to node i by an arc or path.

$D(i)$ = $\{j \in N:j$ greater than i and i connects to j by an arc or path$\}$, i.e. the descendents of node i.

Starting with a stochastic network, it is possible to determine if it is completely reducible, reducible, or irreducible. Pivotal to this determination is the ***interdictive graph*** (IG) shown in Figure 12.

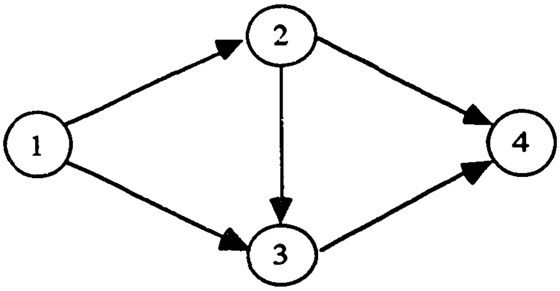


Figure 12. The interdictive graph.
[Adapted from Dodin (1985b)]

The interdictive graph is irreducible and shares with all irreducible networks the following properties:

1. The number of nodes $N \geq 4$ and of arcs $A \geq 5$.
2. For each $i \neq 1$ or N, $|A(i)| + |B(i)| \geq 3$; hence, there are no arcs in series.
3. Either $|A(1)| = 1$, in which case $|B(2)| = |A(1)| = 1$ and $|A(2)| \geq 2$; or $|A(1)| \geq 2$. Therefore, without loss of generality, it can be assumed that in an irreducible network $|A(1)| \geq 2$.
4. Either $|B(N)| = 1$, in which case $|A(N-1)| = |B(N)| = 1$ and $|B(N-1)| \geq 2$; or $|B(N)| \geq 2$. Similarly, it can be assumed that $|B(N)| \geq 2$.
5. There exists a smallest numbered node $j \neq 1,2$ or N and a node $i \in P(j)$, such that there are two independent paths (no arcs in common) connecting i to j.

Dodin (1985b) proved the following characterization of network reducibility:

> Theorem 2: An activity network is not completely reducible if and only if it contains the interdictive graph.

Completely reducible networks can be reduced to the equivalent activity $(1,N)$ in a fixed number of series-parallel reduction [convolution and maximum (multiplication)] operations; Dodin (1985b) proved this result:

> Proposition 1: A completely reducible network, $G(N,A)$, can be reduced to the activity $(1,N)$ in $N-2$ convolution and $A-N+1$ multiplication operations.

The properties of the interdictive graph and Theorem 2 combine to the following steps, which, when applied to any network, identify all possible series-parallel reduction operations and the corresponding irreducible network, if the network is not completely reducible:

1. Calculate $|A(i)|$ and $|B(i)|$, the in-degree and out-degree, for all nodes $i = 1,\dots,N$ in the network $G(N,A)$.
2. If $|A(i)| + |B(i)| \geq 3$ for all $i \neq 1$ or N, then the network is irreducible; stop. If, on the other hand, $|A(i)| + |B(i)| = 2$ for at least one $i \neq 1$ or N, then the network is reducible, since a convolution is possible, since any two nodes of the network are connected by at most one arc. If the convolution is carried out, it might give rise to further series-parallel reduction operations.
3. Successively scan the network to identify and then carry out all possible series-parallel reduction operations, i.e., effect the reduction $G(N,A) \rightarrow G(N',A')$ where $N' < N$ and $A' \leq A$, as shown in the flowchart in Figure 13.

If $A' = 1$, the network is completely reducible and the reduction procedure terminates with the distribution function of T_N, which is equal to the distribution function of the equivalent activity $(1,N)$. If $A' \neq 1$, then $A' \geq 5$ and $N' \geq 4$, and $G(N',A')$ is the corresponding irreducible network $G(N,A)$. The approximate distribution function of T_N is then obtained through further reduction based on the "independent arc multiplication" (arc duplication) operation.

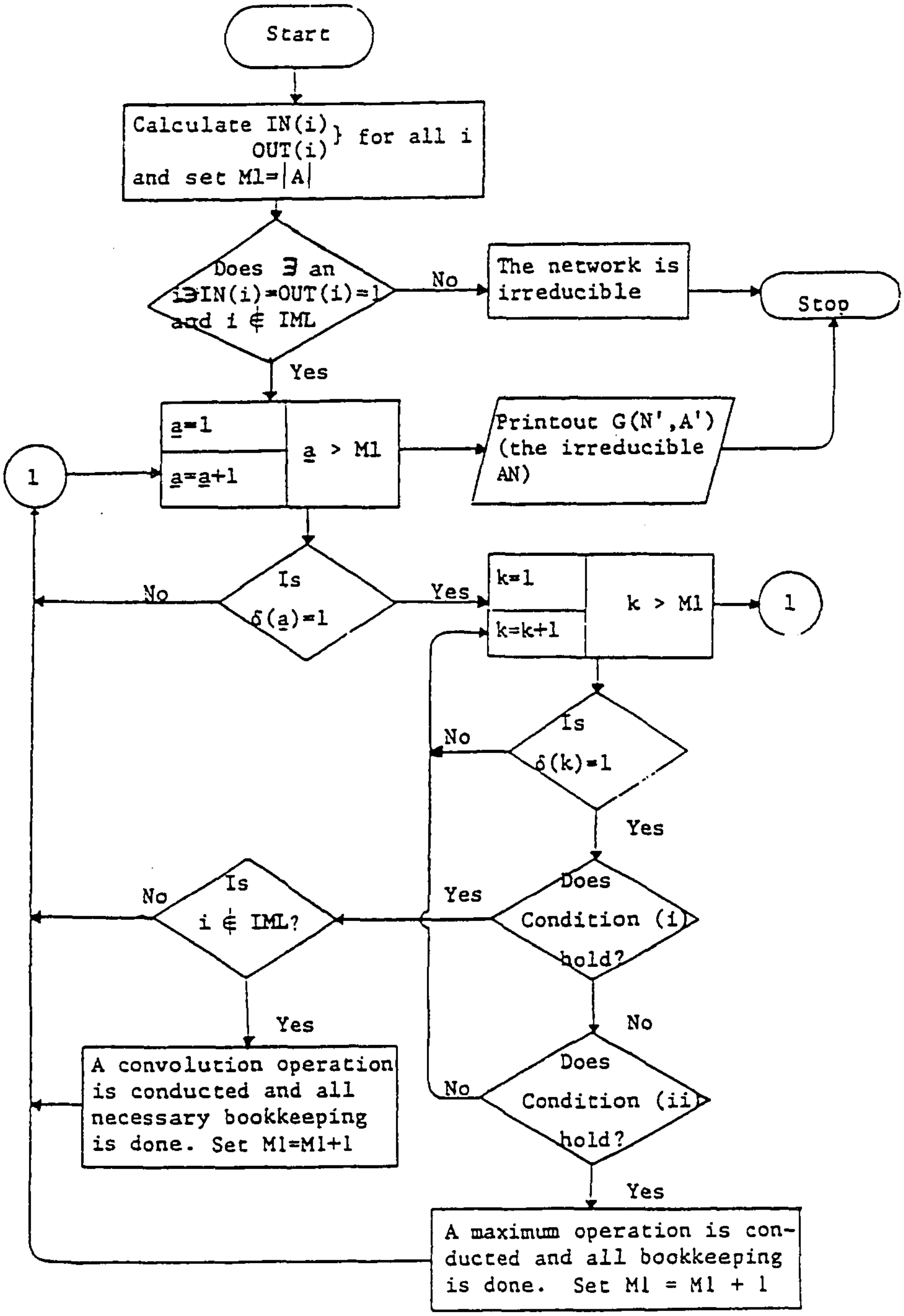


Figure 13. An algorithm for determining the irreducible network.
[Adapted from Dodin (1980)]

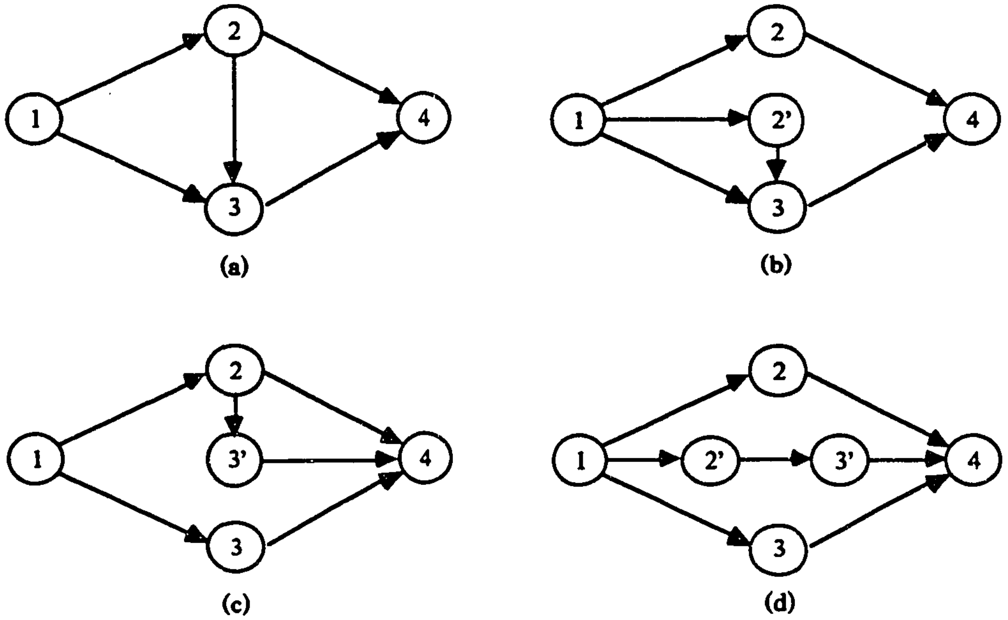


Figure 14. The interdictive graph and all its completely reducible forms. [Adapted from Dodin (1985b)]

An irreducible network can be made completely reducible by duplicating one or more of its arcs. The interdictive graph (Figure 14 (a)), becomes completely reducible if arc (1,2), or arc (3,4), or both are "independently multiplied" (duplicated) in the fashion shown in Figures 14 (b) - (c), respectively. In Figure 14 (b), the paths $\pi_1 = 1 \to 2 \to 4$ and $\pi_2 = 1 \to 2' \to 3 \to 4$ of the interdictive graph become independent of each other. Similarly, in Figure 14 (c), path $\pi_2 = 1 \to 2 \to 3' \to 4$ becomes independent of path $\pi_3 = 1 \to 3 \to 4$, while in Figure 14 (d) the three paths of the interdictive graph become independent of each other. Hence, the "independent multiplication" (duplication) operation means the splitting of an arc along with its node or start node, but not both, into two; the new arc has the same duration and distribution function as the original arc. For example, in Figure 14 (b), the durations of the arcs (1,2) and (1,2') are independent and identically distributed random variables. Such a split may cause the independence of one of the paths (or subpaths) passing through the "independently multiplied" (duplicated) arc from the remaining paths passing through the arc; also, such a split gives rise to a convolution operation which might lead to further reductions in the original irreducible network. Thus, the difference between the network after "independently multiplying" (duplicating) an arc and the original irreducible network is the increase of arcs by one (the split of the "independently multiplied" (duplicated)) arc), the increase of the nodes by one (the duplication of the start or end node of the "independently multiplied" (duplicated) arc), and the loss of some of the dependency between the paths containing the "independently multiplied" (duplicated) arc. Furthermore, if, in the process of "independent multiplication" (duplication), none of the original paths of the networks is lost, and no new paths are gained, then the throughput (realization time) of the final event, T_N, in the network does not change. But, the loss of the dependency between some of the network paths or subpaths may alter the distribution function of T_N. Martin (1965) applied a similar concept in his procedure.

The distribution function obtained by the reduction process bounds the exact distribution function of T_N; such a bound depends on the lost dependency, which depends on the number of "independent multiplication" (duplications) made and the criticality of the "independently multiplied" (duplicated) arcs. If no "independent multiplications" (duplications) are made, then the distribution function obtained by the reduction process is equal to the exact distribution function of T_N. If, on the other hand, the network is not completely reducible, then the distribution function obtained by the reduction process bounds the exact distribution function of T_N from below, if T_N is the duration of the longest path, or from above, if T_N is the duration of the shortest path (Esary, Proschan and Walkup, 1967; Dodin, 1985c).

Moder and Phillips (1964) proved that the CPM approximation based on activity duration means consistently underestimates the mean throughput of stochastic networks. Hence, if T_N is the duration of the longest path, when "independent multiple arcs" reduction is employed, this relationship holds:

$$E((T_N)_{\text{CPM}}) \leq E(T_N) \leq E((T_N)_{\text{"independent multiple arcs"}})$$

To choose the arcs to be "independently multiplied" (duplicated) to transform an irreducible network into a completely reducible network, the arcs of an irreducible stochastic network are divided into two disjoint sets: arcs which can be "independently multiplied" (duplicated), and arcs which cannot. The arcs to be "independently multiplied" are chosen from the first set.

Duplicable Arcs

The objective of "independently multiplying" (duplicating) an arc is to facilitate evaluation of $F(t)$ by reducing the dependency among the paths in the path set P. Dependency is caused by the common arcs; hence, "independent multiplication" (duplication) is limited to the common arcs. However, not every common arc can be "independently multiplied" (duplicated). Only common arcs which bring the original irreducible network closer to its completely reducible form without deleting or adding paths to the network are eligible for "independent multiplication" (duplication). These two objectives are fulfilled by choosing the arcs to be "independently multiplied" (duplicated) from the set of common arcs where each arc in the set has one or both of the following properties:

1. It is the only incident arc on its end node, and the end node has an out-degree of two or more, as shown in Part (a) of Figure 15, i.e., if $a_{ij} \in A'$ is such an arc, then $|B(j)| = 1$ and $|A(j)| \geq 2$. At least one arc with this property always exists in the irreducible network, (e.g. arc (1,2) in any irreducible network). If arc a_{ij} with this property is duplicated, then an arc $a_{ij'}$ having the same starting node is created; it terminates in the newly created node, denoted by j', which is connected to the smallest

numbered node succeeding node j via the first arc emanating from node j. Therefore, $|B(j')| = |A(j')| = 1$, which gives rise to a convolution operation, as shown in Part (b) of Figure 15.

2. It is the only arc incident from its start node, the latter having in-degree of two or more, as shown in Part (c) of Figure 15, i.e., if $a_{ij} \in A'$ is such an arc, then $|B(i)| \geq 2$ and $|A(i)| = 1$. An arc with this property always exists in any irreducible network; it is the mirror image of an arc with the first property, e.g. arc $(N'-1, N')$ is such an arc. In this case an arc $a_{i'j}$ with the same end node is created; it starts in the newly created node i', which is connected to the largest numbered node in the bunch $B(i)$ via the largest numbered arc incident into node i. Thus a convolution operation is created; this duplication is shown in Part (d) of Figure 15.

The above two properties are the same as the a and b activities of Theorem 1 (Garman, 1972), but Dodin (1985b) does not credit Garman for these results. A duplicable arc may have both of the above properties, as shown in Part (e) of Figure 15. This may be the case for some intermediate arcs in the irreducible network. In this case, such an arc is treated as an arc with the first property, as is shown in Part (f) of Figure 15, or as an arc with the second property, as shown in Part (g) of Figure 15. The arcs which possess the above two properties form the duplicable set.

"Independently multiplying" (duplicating) any other arc in the network, a non-common arc or a common arc that does not have either of the above two properties, adds more paths to the network and does not bring the network any closer to its completely reducible form. For example, "independently multiplying" (duplicating) any of the common arcs in the interdictive graph, say arc (2,4), results in the network shown in Figure 16, which has four paths and is still irreducible.

The preceding definition of the "independent multiplication" (duplication) concept is based on duplicating arcs; hence, arcs are characterized into duplicable and non-duplicable

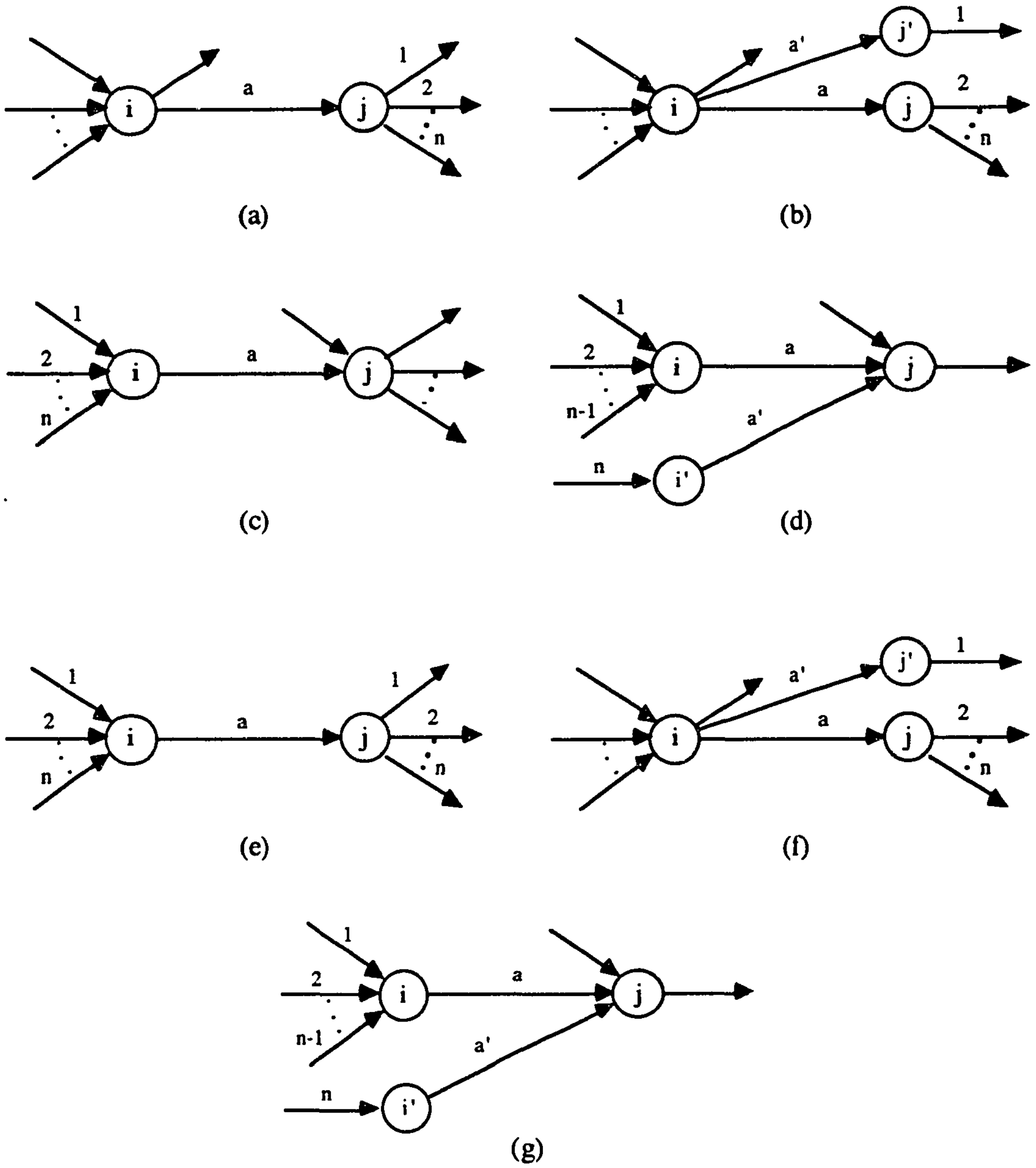


Figure 15. Forms of duplicable arcs.
[Adapted from Dodin (1985b)]

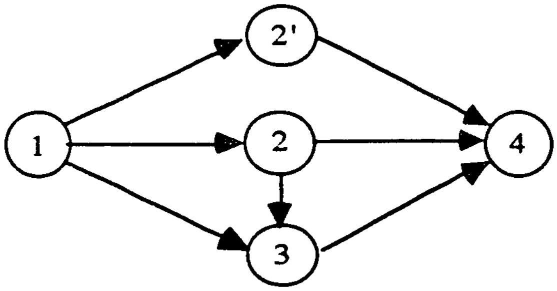


Figure 16. Duplicating arc (2,4) in the interdictive graph.
[Adapted from Dodin (1985b)]

sets. A duplicated arc indicates which node is to duplicated; it is the end node if the arc has Property 1, and the start node if the arc has Property 2, and either node, but not both nodes, if the arc has both properties. The concept of duplication could just as easily be based on duplicating nodes. In this case, the nodes of the network can be partitioned into duplicable and non-duplicable nodes. The duplicable nodes are of two types: the first type is like the end node of an arc with Property 1, and the second type is like the start node of an arc with Property 2; such nodes always exist in an irreducible network. The duplication of a node of the first type requires the "independent multiplication" (duplication) of the only arc incident into it, as shown in Part (b) of Figure 15, while the duplication of a node of the second type requires the "independent multiplication" (duplication) of the only arc emanating from it, as shown in Part (d) of Figure 15. There are no nodes in any network which have the properties of both types.

The Reduction Process

An irreducible network can be reduced to the single equivalent arc $(1,N)$ by "independently multiplying" (duplicating) arcs with Property 1 only, or arcs with Property 2 only, or a combination of both. This implies that the "independently multiplied" (duplicated) arcs are not unique, and that there are many sequences to choose from. Hence, an "independent multiple arcs" approximation reduction method could be based on "independently multiplying" (duplicating) the first available arc with Property 1, or the first available arc with Property 2, or a hybrid process, where the next arc to be "independently multiplied" (duplicated) is selected based on a decision rule or objective function.

Efficiency and Complexity

The interdictive graph can be reduced to the equivalent activity (1,4) by "independently multiplying" (duplicating) only one activity having either Property 1 or Property 2. This observation gives an upper bound on the number of arcs to be "independently multiplied" (duplicated). Dodin (1985b) proved these results:

> Proposition 2: The number of duplications necessary to reduce an irreducible network to the equivalent arc $(1, N)$ is less than or equal to the total number of interdictive graphs in the irreducible network.
>
> Theorem 3: Given an irreducible network $G(N, A)$, then the ["independent multiple arcs" (duplication) approximation] reduction process can reduce it to the equivalent arc $(1, N)$ in L duplicates, where $L \leq A - N - |A(1)| + 2$.
>
> Corollary: The number of [parallel-reduction] multiplication operations required to reduce any network $G(N, A)$ to the arc $(1, N)$ is independent of the reducibility of the network, and it is a constant equal to $A - N + 1$.

Proposition 1, Theorem 2 and the Corollary together give the complexity of an "independent multiple arc" (dual arc) approximation reduction process. From the Corollary, the process performs $A - N + 1$ parallel reduction (maximum) operations, and from Proposition 1 and Theorem 2, $L + N - 2$ series reduction (convolution) operations. If C is the complexity of the series-parallel reduction operations, the reduction process complexity is $O(CA)$. However, $A \leq N(N-1)/2$ for all acyclic, directed networks. Therefore, the reduction process complexity is $O(CN^2)$.

An “independent multiple arc” (dual arc) approximation reduction process will not necessarily duplicate the minimum number of arcs to reduce the network to the equivalent arc $(1,N)$. It may be possible to reduce the number of duplications by duplicating the duplicable arc with the highest path index first, where the path index of an arc is the number of paths containing the arc. Augmenting a reduction process by this rule still does not guarantee that the augmented procedure duplicated the minimum number of arcs; Dodin cites a counterexample. Reducing a network by “independently multiplying” (duplicating) the minimum number of arcs is still an unsolved question, whose solution may require the identification of all the direct and embedded interdictive graphs in the irreducible network. There is no efficient solution technique for this identification problem, and, moreover, the problem appears to be NP-complete. (Dodin, 1985b)

2.5.2 Sequential Approximation

Dodin (1980, 1985a) developed a second method for reducing irreducible networks, sequential approximation, which is based on the determination of the throughput distribution function $F(t)$ by constructing the distribution functions through the nodes of the network in sequential order. Sequential approximation can be applied to any stochastic network with $N > 2$ and $|A| > 1$, reducible or irreducible. An approximation reduction process based on sequential approximation enters the sequential approximation step with the irreducible network $G(N',A')$, which is reached in an analogous manner to the “independent multiple arcs” (dual arcs) approximation reduction methods, i.e., by successively scanning the network to identify and then carrying out all possible series-parallel reduction operations so as effect the reduction $G(N,A) \rightarrow G(N',A')$. Sequential approximation starts at node 1, which has the discrete distribution function $F(1) = \{(0,1)\}$, then proceeds sequentially to approximate the distribution function of the throughputs (durations, or realization times) of the next nodes in increasing order, ending with the sink node, N.

The following notation was used in his development:

a_{ij} = arc $(i,j)\in A$: starting at node i and ending in node j where i less than j, it is the only arc connecting node i to node j in the network.

$B(j)$ = the set of arcs ending at node j.

n_j = number of arcs ending at node j, the in - degree of node j.

m_j = number of arcs emanating at node j, the out - degree of node j.

Y_{ij} = random variable, the duration of activity a_{ij}.

T_j = random variable, the realization times of node j.

The distribution function of T_j for all $j\in N'$ is approximated using the following procedure, which is depicted in the flowchart in Figure 17:

1. Without loss of generality, assume the sequential approximation is at node $j\in N'$, then $m_j > 1$ or $n_j > 1$, i.e., $m_j + n_j \geq 3$, as shown in Figure 18. Determine the set $B(j)$, which is the set of activities ending at node j, and rank the activities in an ascending order of their starting nodes, as shown in Figure 2.
2. For each activity $a_{ij}\in B(j)$, convolute $F_{ij}(y)$ with $F_i(t)$, i.e., the distribution function of Y_{ij} with the distribution function of T_i. Denote this convolution by $\overline{F}(i)$.

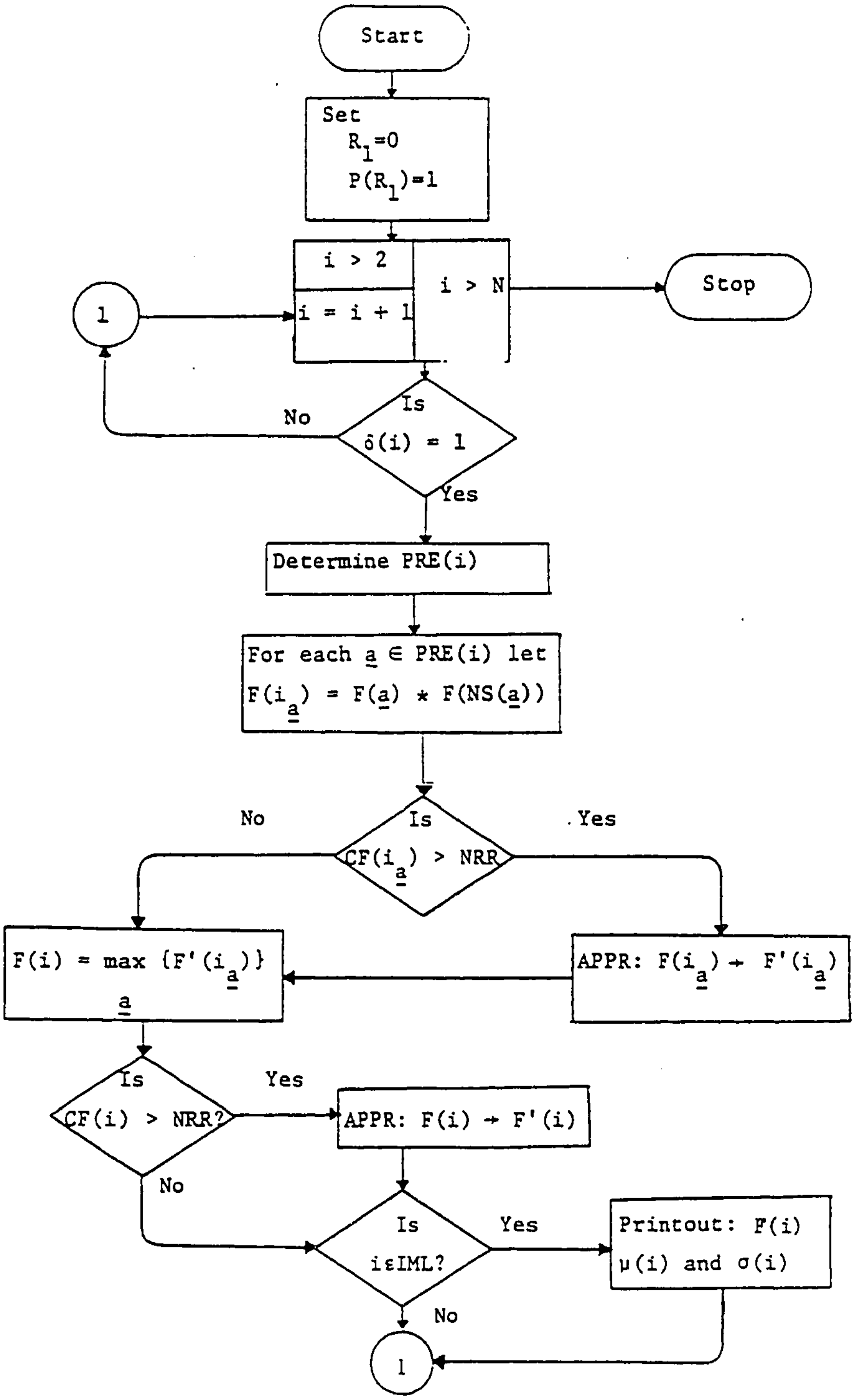

Start
Set
$R_1=0$
$P(R_1)=1$
i > 2
i = i + 1
i > N
Stop
1
No
Is
$\delta(i) = 1$
Yes
Determine PRE(i)
For each $\underline{a} \in$ PRE(i) let
$F(i_{\underline{a}}) = F(\underline{a}) * F(NS(\underline{a}))$
No
Is
$CF(i_{\underline{a}}) > NRR$
Yes
$F(i) = \max_{\underline{a}} \{F'(i_{\underline{a}})\}$
APPR: $F(i_{\underline{a}}) \rightarrow F'(i_{\underline{a}})$
Is
CF(i) > NRR?
Yes
APPR: $F(i) \rightarrow F'(i)$
No
Is
$i \varepsilon IML$?
Yes
Printout: F(i)
$\mu(i)$ and $\sigma(i)$
No
1

Figure 17. Sequential approximation flowchart.
[Adapted from Dodin (1980)]

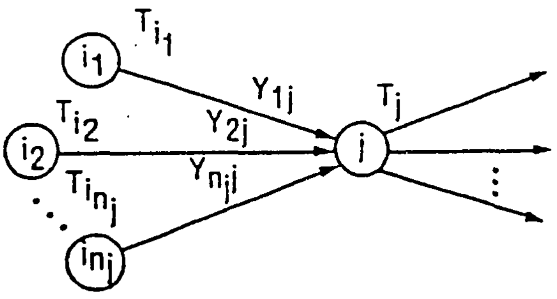


Figure 18. A node in an irreducible network.
[Adapted from Dodin (1985c)]

3. $F_j(t) = P(T_j \le t) = \max\{\bar{F}(i)$ for all $i \in N'$ such that $a_{ij} \in$ $B(j)\}$

The parallel-reduction operator may be performed sequentially to avoid unexpected escalation in storage requirements [when continuous distribution functions are discretized]. For example, let:

$$F(i_1, i_2) = \max\{\bar{F}(i_1), \bar{F}(i_2)\},$$

i.e., for any value of $t > 0$,

$$F(i_1, i_2) = P(\max\{T_{i_1} + Y_{i_1 j}, T_{i_2} + Y_{i_2 j}\} \le t)$$
$$= P(T_{i_1} + Y_{i_1 j} \le t)P(T_{i_2} + Y_{i_2 j} \le t).$$

Then,

$$F(i_2, i_3) = \max\{F(i_1, i_2), \bar{F}(i_3)\},$$

and so on, until $F_j(t)$ is finally obtained where:

$$F_j(t) = F(i_{n-1}, i_n) = \max\{F(i_{n-2}, i_{n-1}), \bar{F}(n)\},$$

$k > j \in N'$, until finally, node N is reached, and $F_N(t)$ is approximated.

The underlying assumption behind the sequential approximation method is reflected in Step 3; it is the independence of the random variables $\{T_i, i \in N'\}$. In any irreducible network, the T_i's may not, usually are not, independent. If the assumption of independence is the only source of error, which is the case only if at the outset all the activities have discrete distribution functions and there is no re-discretization of intermediate convolution or maximum (multiplication) operations to control storage requirements (in Steps 2 and 3), then the approximate distribution function, $F_N(t)$, obtained through the sequential approximation method, bounds the exact distribution function of T_N from below. Kleindorfer (1971) was the first to use this assumption of independence to bound the exact distribution function of T_N from below.

Complexity

The complexity of sequential approximation is a linear function of the number of series-reduction (convolution) and parallel-reduction (maximum) operations. In an irreducible network with A arcs and N nodes, Step 2 of the sequential approximation method requires $A - m_1$ convolutions, and Step 3 requires $n_j - 1$ maximum operations for each $j \neq 1$ or 2. The total number of maximum operations is

$$\sum_{j=3}^{N}(n_j - 1) = A - N + 1.$$

Therefore, the complexity of sequential approximation is $O(CA)$ where C is the complexity of the series-parallel reduction operations. If continuous activity distribution functions are discretized, then C is a function of the cardinality of the discrete approximations. If the cardinality of the discrete distributions is less than or equal to k, then the complexity of the series-parallel reduction operations is at worst $O(k^2)$.

To the complexity of sequential approximation must be added the complexity of the reduction of an original network, $G(N,A)$, to its irreducible network, $G(N',A')$. Completely reducible networks can be reduced to the single, equivalent activity $(1,N)$ in a fixed number of series-parallel operations. To reduce the network to $(1,N)$ requires that $N-2$ nodes and $A-1$ arcs have to be suppressed, but a series-reduction (convolution) operation is necessary to reduce the network by one arc and one node. Therefore, $N-2$ convolution operations and $A-N+1$ parallel-reduction (maximum) operations are required, and the complexity of the reduction process is at worst $O(CA)$ where C is the complexity of the series-parallel reduction operations, exactly like the complexity of sequential approximation. Consequently, the overall complexity of the sequential approximation and reduction method is $O(CA)$.

Efficiency

It is difficult to determine the accuracy of an approximate throughput distribution function, $F_N(t)$, obtained with the sequential approximation and reduction method, since the exact throughput distribution function is generally not known for a stochastic network. Also, it is not possible to compare the accuracy of this method with the accuracy of other methods; for one, with the exception of Monte Carlo simulation, they cannot be applied to large networks; for another, there are no reported computational results for most other methods, with the exception of ordered recursive conditioning (Hagstrom, 1990). The latter is discussed in Section 2.5.3. Dodin (1980, 1985a) measured the accuracy of $F_N(t)$ by its closeness to the corresponding distribution function obtained by extensive crude Monte Carlo simulation, $F_n'(t)$, where n is the simulation sample size (number of simulation replications). [Note that Dodin's nonstandard notation, $F_n'(t)$, refers to the approximated distribution function obtained with Monte Carlo simulation, not the probability density function corresponding to $F_n(t)$.] The distribution function $F_n'(t)$ was used as a surrogate for the exact, but unknown, distribution function, $F(t) = P(T_N \leq t)$, since it is a known result (Kleindorfer, 1971) that $F_n'(t)$ converges to the exact distribution function $F(t)$ as $n \rightarrow \infty$. The closeness between $F_N(t)$ and $F_n'(t)$ was measured using these performance measures:

1. Maximum of the absolute values of the deviations (MADV) between the two distribution functions, $F_N(t)$ and $F_n'(t)$. MADV is the Kolmogorov-Smirnov test statistic.
2. Average of the absolute values of the deviations (AADV).
3. Average values for T_N obtained from both distribution functions, $F_N(t)$ and $F_n'(t)$.
4. Standard deviation of T_N obtained from both distribution functions.

Dodin (1980, 1985a) discretized all continuous activity duration (passage time) distributions and generated "strongly randomized networks." He found that variations in the measures of performance depended upon the structure and size of the network, the distributions of the activity durations, the sample size (number of replications) in the Monte Carlo simulation, and the accuracy of the discretization of the continuous distributions.

The approximated distribution function, $F_n'(t)$, was obtained using crude Monte Carlo simulation for each network tested. During each simulation replication, the simulation model assigns a random value, t_{ij}, to the duration of each activity a_{ij} in the original, unreduced stochastic network, using the inverse transformation method. Then, CPM is applied to the network to determine a realization for T_N. These two steps are repeated for a large number of times, n, to insure that $F_n'(t)$ is a very close approximation to the exact distribution function of T_N. Grouped data analysis is then used to generate $F_n'(t)$ as a discrete cumulative distribution function of the simulation realizations for T_N. VanSlyke (1968) and Elmaghraby (1977) discuss the sensitivity of the accuracy of simulation of PERT-type networks to simulation sample size. Based on their observations, Dodin (1980, 1985a) chose a simulation sample size of 1000 replications for his analysis.

To achieve a more convincing test of accuracy, each network tested was a "strongly random network," generated at random from the space of all connected, acyclic, directed graphs with the required numbers of nodes and activities. The network generator used was Dodin's (1993) modified version of the generator originally developed by Herroelen and Caestecker (1979). Random network generation is discussed in Section 2.7.2.

Activity duration distribution functions were assigned from a set of seven distributions: uniform, triangular, normal, exponential, gamma, beta, and discrete. The parameters of each distribution were set to those of a representative member of that distribution family at the beginning of the analysis. Although they could have been varied during the analysis, they were not, as Dodin (1980, 1985a) held that the choice of distributions and their parameters is of no particular significance except as it reflects the general applicability of the approximating procedure.

Dodin (1980, 1985a) reported results for two experimental combinations: different activity distribution functions with a fixed network, and different size networks with a fixed activity distribution function for each activity. Table 3 shows how the distribution type affects the performance measures for a randomly generated network with 10 nodes and 15 activities. Table 4 reflects the effect of network size; the uniform distribution was applied to each of the eight networks.

Table 3. Performance Measures for Various Distribution Functions.
[Adapted from Dodin (1980, 1985a)]

Distribution Type	Comparison of the Approximate DF with that of MCS					
	Average		Standard Deviation			
	Approx.	MCS	Approx.	MCS	MADV	AADV
Uniform	27.27	27.20	4.868	5.316	0.0426	0.0115
Triangular	29.13	29.21	3.925	4.255	0.0513	0.0180
Normal	40.29	40.29	4.059	4.180	0.0585	0.0206
Exponential	12.50	12.49	3.511	3.899	0.0328	0.0099
Gamma	16.85	16.61	3.504	3.401	0.0436	0.0122
Beta	36.96	37.16	3.879	4.194	0.0772	0.0275
Discrete	18.50	18.51	2.055	2.076	0.0083	0.0016
All	28.46	28.07	3.808	4.005	0.0274	0.0070

Table 4. Effect of Network Size on the Performance Measures.
[Adapted from Dodin (1980, 1985a)]

No. Nodes	No. Arcs	Comparison of the Approximate DF with that of MCS					
		Average		Standard Deviation			
		Approx.	MCS	Approx.	MCS	MADV	AADV
10	15	27.27	27.20	4.868	5.316	0.0426	0.0115
20	40	47.37	46.47	6.733	7.625	0.0557	0.0162
30	50	52.30	51.89	6.251	7.139	0.0525	0.0169
40	60	58.98	57.71	6.962	8.174	0.0633	0.0182
40	80	56.06	55.14	5.952	6.553	0.0303	0.0115
50	75	62.54	62.58	7.698	8.224	0.0651	0.0259
50	100	67.38	65.56	6.182	7.752	0.0880	0.0263
60	150	82.82	80.05	7.155	9.074	0.1082	0.0306

In the eight test networks considered in Table 3, the approximate mean values of the throughput (project completion time) were very close to the simulation means. The maximum deviation between the two means was less than 0.43. The approximate standard deviations were less than the simulation standard deviations, which implies that the approximated distribution functions have less variation than the distribution functions obtained by simulation. The graphs of the density functions obtained by both procedures supported the general forms obtained by crude Monte Carlo simulation by VanSlyke (1968). MADV and AADV varied by distribution type. MADV was always less than 0.08, and A'ADV less than 0.03; although Kolmogorov-Smirnov goodness of fit tests could have been performed on the MADVs, none was reported. The smallest values for both were obtained for the network for which all activity distribution functions were discrete, an expected result, since there were no errors of discretization for this network. The second smallest values were obtained for the network for which the activity distribution functions were of mixed types, including some discrete distributions. The accuracy of approximation is enhanced by having more accurate discretization. This was the case with the network with exponential activity distribution functions; the exponential distribution was approximated on thirty discrete points, whereas each of the other continuous distributions was approximated on only twenty points.

As the simulation sample size increased, the sampled distribution, $F_n'(t)$, converged to the approximate distribution, $F_N(t)$. In particular, the simulation mean increased in the direction of the approximate mean and the MADV was halved for an increase in the simulation sample size from 100 to 1000 replications.

The eight networks in Table 4 were generated at random; the uniform distribution was applied to each. MADV and AADV increased as network size increased. Dodin (1985a) suggested that this was due to his having used a constant number of simulation replications for all the networks tested, since larger networks require larger simulation sample sizes to maintain approximation accuracy. The MADVs, the most sensitive measure, were taken on at values within the first 30% of the distribution functions. These deviations tend to have negative values in the first half of distribution functions and positive values in the second half. As discretization errors decrease and the simulation sample size increases, these deviations converge to zero. Dodin (1985a) suggested that this observation increases the applicability of sequential approximation and reduction, since the right hand side of the throughput distribution is of more interest to project management than the left hand side.

CPU time requirements of the method, excluding simulation, consisted of discretization time, time for series-parallel reduction operations to reduce the original network to its irreducible form, and time sequential approximation time. Initial discretizations required less than 0.04 CPU seconds on the average on a UNIVAC 1100/80. CPU time requirements of the method were minimal when compared to the time requirements of simulation. Dodin (1985a) reported the CPU times in Table 5. Simulation time depended on the distribution type; simulation time doubled for continuous distributions other than the uniform because of inverse transformations. However, distribution type did not affect the CPU time of the method beyond the discretization step. The CPU times of the method in Table 5 confirm the method's complexity. The CPU time of the seventh network was less than the CPU times of any of the other networks of the same size; the seventh network had discrete activity distributions on four points, while the activity durations of the other networks were approximated on at least twenty points. The CPU times of the networks in the lower half of the table increased as the number of activities increased.

For the constant simulation sample size of 1000 replications, simulation times were larger by a factor of three than the CPU times of the method. Dodin (1985a) speculated that this factor might double if distributions other than the uniform were considered. He further observed that if the simulation sample size were increased to achieve greater accuracy of the sampled distribution, the simulation time would increase correspondingly.

Table 5. CPU Time of Sequential Approximation and Reduction Method and Simulation. [Adapted from Dodin (1985a)]

	Network size		CPU time in seconds	
Distribution Type	No. Nodes	No. Arcs	Approx. Method	MCS Sample size = 1000
Uniform	10	15	1.665	4.31
Triangular	10	15	1.673	7.25
Normal	10	15	1.680	6.43
Exponential	10	15	1.678	7.33
Gamma	10	15	1.695	7.58
Beta	10	15	1.766	7.94
Discrete	10	15	0.994	9.34
All	10	15	1.623	8.52
Uniform	20	40	4.448	12.02
Uniform	30	50	5.335	15.58
Uniform	40	60	5.974	17.94
Uniform	40	80	7.919	24.34
Uniform	50	75	7.499	23.54
Uniform	50	100	10.867	35.80
Uniform	60	150	15.822	51.90

Based on his analysis of the sequential approximation and reduction method, Dodin (1985a) concluded:

1. The method is at its best if the activities have discrete distributions to start with. The accuracy of approximation can be increased by reducing the errors of discretization.
2. The approximate mean and standard deviation of the project completion time are very close to the "true" mean and standard deviation. The approximate mean may not bound the true mean from above, since $F_N(t)$ does not necessarily bound the actual distribution function of the project completion time from below.
3. In comparison with the distribution function obtained by Monte Carlo simulation, the sampled distribution converges toward the distribution function as the sample size increases. This demonstrates that the distribution function of the project completion time is not sensitive to the assumption of independence of the paths in the network on one hand, and on the other hand, it also indicates that the distribution function obtained by the method is very close to the actual distribution.

4. The maximum value of the absolute deviation, MADV, is located in the domain of the lower 30% of the distribution; this observation increases the applicability of the approximate distribution function since the realizations of interest to management are those in the upper half of the distribution function.
5. The processing time requirements of the method are minimal if compared with the CPU time requirements of simulation. The CPU time requirements for the method are always less than sixteen seconds for any activity network of size $(N, A) \leq (60,150)$ on a UNIVAC 1100/ 80.

(Dodin, 1985a)

2.5.3 Ordered Recursive Conditioning

Hagstrom (1990) developed a third method for reducing irreducible networks, an algorithm which is based on ordered recursive conditioning on the activity durations. This is a common technique in reliability computations, where it is also referred to as pivoting, factoring, or backtracking. The network must have discrete activity duration distributions, or continuous distributions must be discretized. The algorithm uses a representation of the network reduction problem similar in concept to Mirchandani's (1976) *emergency equivalent network* for stochastic shortest-path problems. Each activity has several states, each one corresponding to a realization of the length of time the activity requires. This is represented in the project network by assigning each activity as many arcs as it has states.

Such a representation is shown in Figure 19. The underlying project network is a standard PERT model, which consists of eleven activities. Each activity requires one of two equally likely durations of time to be completed; hence, in the figure it is represented as having two possible states.

Conditioning algorithms enumerate substructures of the system under consideration. The objective of ordering the conditioning is to minimize the size of the enumeration: the structures enumerated here are the candidates for longest-path arborescence of the network. The recursive conditioning is based on choosing which state of which activity will

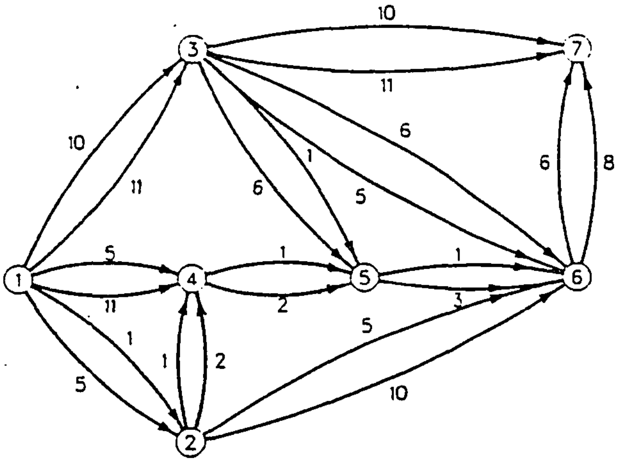


Figure 19. Activity-duration-states representation of a PERT network.
[Adapted from Hagstrom (1990)]

determine the longest path into a particular node. Hagstrom's (1990) version of the algorithm computes moments of the probability distribution of project duration.

The algorithm is stated in terms of a recursive function PIVOTS. At a call of depth k to PIVOTS, the lengths of the longest paths to nodes 1 through k have been fixed. PIVOTS then conditions on which of the arcs directed into node $k+1$ determines the longest path into that node. It then calls itself for each of these cases and proceeds recursively until it reaches a depth corresponding to the number of nodes in the network; at this level, it returns the selected powers of the length of the critical path. When a level k call is finished, PIVOTS returns the moments of the critical path length conditioned on the distances to nodes 1 through k. Hagstrom (1990) proved that PIVOTS correctly computes the moments of the project duration.

Another version of the algorithm computes values of the cumulative distribution function of the project duration. Instead of accumulating expectations, the alternate version accumulates conditional probabilities. Both versions have the flexibility to process several distributions of activity durations at once. As long as the distributions have the same ranges, the order of processing is not changed and the computation requirements are not greatly increased.

Computational Experience

The algorithm was coded and compiled in Pascal and run on an IBM3081 operating under CMS. Hagstrom (1990) ran cases from the literature as well as randomly generated cases. Characteristics of the cases and computation times are shown in Table 6. Input, preprocessing, and output times were excluded; except for small networks, these require an insignificant amount of total computation time. No analysis has been made of computer storage requirements. The algorithm keeps only one copy of the network, and the amount of data in the recursion stack is minimal, so computation time requirements are much more likely to be limiting than storage requirements, even for less space-efficient implementations.

Table 6. Computational Time Requirements for Selected Cases. [Adapted from Hagstrom (1990)]

	No	Effective m-degrees																			CPU
Case	vertices	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	microseconds
TEST5A	5	3	5	7	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	12960
TEST6A	6	3	5	7	5	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	35364
TEST7A	7	3	3	5	9	3	9	1	1	1	1	1	1	1	1	1	1	1	1	1	81381
TEST8A	8	3	5	5	7	3	7	13	1	1	1	1	1	1	1	1	1	1	1	1	741210
TEST9A	9	3	5	3	9	5	11	5	11	1	1	1	1	1	1	1	1	1	1	1	2284369
TEST10A	10	3	3	5	3	7	11	9	9	15	1	1	1	1	1	1	1	1	1	1	5215133
TEST11A	11	3	3	3	5	5	5	7	9	15	9	1	1	1	1	1	1	1	1	1	17879858
TEST12A	12	3	3	3	3	5	7	7	5	7	9	11	1	1	1	1	1	1	1	1	75729868
COMPLET6	6	3	5	7	9	11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	76053
COMPLET7	7	3	5	7	9	11	13	1	1	1	1	1	1	1	1	1	1	1	1	1	153603
COMPLET8	8	3	5	7	9	11	13	15	1	1	1	1	1	1	1	1	1	1	1	1	576241
COMPLET9	9	3	5	7	9	11	13	15	17	1	1	1	1	1	1	1	1	1	1	1	1378182
COMPLET10	10	3	5	7	9	11	13	15	17	19	1	1	1	1	1	1	1	1	1	1	12075520
SERIES10	10	3	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1	1	1	1	910608
ELM4FIG4	4	3	3	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3533
SHOGANI	6	5	5	9	9	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	186719
VANSLYKE	9	3	3	5	3	3	7	1	7	1	1	1	1	1	1	1	1	1	1	1	339900
PRITSKER	9	3	5	3	3	7	3	5	5	1	1	1	1	1	1	1	1	1	1	1	846504
STRIPDWN	20	2	2	2	3	2	3	3	2	2	4	4	4	3	3	3	3	4	4	4	281713828

TEST5A through TEST12A were randomly generated cases where an attempt was made to keep the average number of tasks directed into each node at two. TEST5A and COMPLET6 through COMPLE10 were complete, acyclic digraphs with randomly generated activity duration distributions. SERIES10 is a network in which the only activities are of the form $(i, i+1)$. For all these cases, the probability distribution of activity time was generated by randomly selecting a mean time between 0 and 10, randomly selecting a distance from the mean between 0 and the mean, and then constructing a three-point distribution with probability 0.6 of being the mean and probability 0.2 each for being at the selected distance above and below the mean.

The remaining cases were taken from the literature. VANSLYKE is a discrete version of Figure 10 from VanSlyke (1963); 0.1 probability was assigned to both optimistic and pessimistic activity completion times and 0.8 probability assigned to the most likely time. ELM4FIG4 is Figure 4.4 from Elmaghraby (1977). SHOGAN1 is Example 1 from Shogan (1977). The data for PRITSKER were taken from Table 6.1 in Pritsker and Kiviat (1969). STRIPDWN is Kleindorfer's (1975) example with each activity restricted to taking either its smallest or largest possible length. Table 6 specifies the number of nodes for each case and the effective in-degree:

$$\text{in-degree of node } (k+1) - \text{number of activities incident to node } (k+1) + 1$$

for each node from node 2 to the sink node. A "1" in the table indicates a null entry. Confirmation of an upper bound developed (on the work required to sort the arcs in the in-directed adjacency list of any node) appears to require more large examples than Hagstrom could run.

Analysis of Computation Requirements

Hagstrom (1988) demonstrated that, unless P=NP, we cannot hope to obtain a polynomial algorithm for the reduction of a stochastic project network. The ordered-recursive-conditioning algorithm is not a polynomial algorithm. Its computation time requirements were determined by analyzing the recursion tree associated with applying the algorithm to a particular instance of a stochastic project network. Hagstrom (1990) proved this result:

> Theorem 4: Let H be an upper bound on the work required to sort arcs in the in-directed adjacency list of any node in a project network. Let α be the number of nodes for which all in-directed activities have deterministic durations. Let Π be the product of the effective in-degrees. Then an upper bound for the work required in applying PIVOTS is a constant times $H(\alpha + 2)\Pi$.

In a complete project network with n nodes and k states per activity, the computation requirements of this algorithm would be $O(H \cdot k^n \cdot n!)$. In contrast, computation requirements for Martin's (1965) algorithm would be $O(Q \cdot k^{n^2})$ where Q represents the time required to compute the project duration distribution in a series-parallel (completely reducible) network. Both H and Q are polynomial functions of n and k. Hence, this algorithm has a better worst-case performance than does Martin's algorithm.

Improving On The Algorithm

Hagstrom (1990) observed that the rapid growth of computation time with the size of the network limits the use of this algorithm to relatively small networks. She considered how the algorithm might be improved in order to increase the size of the network that can be handled. If improvements are restricted to the framework of the algorithm as originally developed, the possibility for improvement appears to be limited to fine-tuning the generation of the algorithm's recursion tree, so that a few more branches can be removed, or an improved method of internal sorting. Such improvements may make the algorithm more competitive with Martin's (1965) for sparse networks. However, neither of these improvements will significantly reduce the size of the recursion tree, and the size of the tree is the major determinant of the computation time.

Hagstrom (1990) considered going outside the framework of the algorithm to see if other approaches might yield a significant reduction in computation time. She concluded that the two most obvious approaches do not appear promising. Exhaustive enumeration of some combinatorial substructure which appears within the original problem's structure seems to be required in algorithms for #P-complete problems. The amount of work grows polynomially in the number of this substructure, and, unfortunately the number of these substructures can grow exponentially with the size of the original problem structure. Since the PERT problem is #P-complete, we expect that we will have to enumerate some structure whose number grows exponentially in the size of the network. PIVOTS enumerates almost all the spanning arborescences of the network. The question then arises as to whether there is something better to enumerate. The two most obvious candidates were minimal source-to-sink paths and minimal source-to-sink cuts. However, complexity analysis indicates that we will not succeed by enumerating minimal source-to-sink paths unless P=NP. Although the evidence is weaker, it seems unlikely that we can devise an algorithm that computes characteristics of the distribution of project duration without enumerating all possible states of the project, which cannot be done in time polynomial per cut. Thus, enumerating minimal source-to-sink cuts does not appear to be a promising approach.

Hagstrom (1990) concluded that if we wish to look for algorithms that significantly improve on PIVOTS, we should think of enumerating some combinatorial structure in the network other than minimal source-to-sink paths or cuts, or spanning arborescences. We must expect that in some cases the number of instances of this structure will be exponentially more than the number of minimal paths. Similarly, it seems likely that in other cases the number of instances of this structure will be exponentially more than the number of minimal cuts. In order to improve on PIVOTS, the number of instances of this structure should be less than the number of spanning arborescences.

Comparisons

This ordered-recursive-conditioning algorithm, which requires discrete activity duration distributions, seems to be the best presently available for exact computation on dense PERT-type networks, although it is outperformed by Martin's (1965) method on sparse networks. It may be difficult to find an algorithm with better worst-case performance. However, since the rapid growth in computation time with the size of the network limits the use of the algorithm to relatively small networks, Hagstrom (1990) recommended approximation reduction methods when the computation budget/computational capability is limited.

2.6 Simulation

Monte Carlo simulation was first proposed as a reduction method for stochastic networks by VanSlyke (1963). Early network simulation programs were written in FORTRAN and GASP (Pritsker and Kiviat, 1969). Following the introduction of GERT (Pritsker and Happ, 1966; Pritsker and Whitehouse, 1966), Pritsker developed a family of GERT simulation (GERTS) tools - GERTS III, GERTS IIIC, GERTS IIIR, Q-GERTS - to accommodate cost and resource information and queues in stochastic networks (Whitehouse, 1973; Pritsker, 1978). Burt and Garman (1971) developed conditional sampling in an effort to reduce the computer run time requirements of network simulations. Today, stochastic networks can be simulated using either general purpose simulation languages, such as SLAM (Pritsker, 1986), or specially written software, such as STARC (Badiru, 1991a and 1991b).

Simulation is an effective approach to approximating the throughput distribution of a stochastic network, especially if the network has mixed types of activity duration distributions. Employment of simulation requires the user to exercise a simulation model through sufficient replications to insure that steady state conditions have been attained with respect to the characterization of the throughput distribution, usually as evidenced by stability in the first two moments (mean and variance). Multiple simulation replications are required, since in each replication a different value may be assigned to each activity's duration based on its designated probability distribution, leading to a different realization of the network throughput. As networks become larger and more complex, increased numbers of replications are required in order to properly characterize the steady state behavior of throughput distributions.

2.6.1 Limitations of Simulation

Historically, simulation has enjoyed somewhat limited use in both the commercial and government sectors. This has been attributed primarily to four factors (Moore and Clayton, 1976; Wiest and Levy, 1977; Schonberger, 1981):

1. Lack of technical knowledge about network simulation. For example, no generally acceptable objective criteria have been developed for defining the attainment of a steady state. It is possible, due to the complex behavioral character of stochastic networks with large numbers of nodes and the fact that autocorrelation problems can result from the computer generation of random numbers, that a false steady state may be interpreted as a true condition.
2. Degree of specialized quantitative knowledge required to obtain and interpret simulation results. Due to a lack of this specialized knowledge, the majority of potential program management users are frequently uncomfortable making decisions based primarily on simulation results. In some cases, the manager may even be restricted in his usage of simulation results unless he can obtain a degree of expertise either personally or through an analyst.
3. Analyst's frequent inability to alleviate the manager's feelings of uneasiness. Only an approximated confidence interval on the mean throughput can be constructed from simulation results. Some simulation programs can display crude frequency histograms. However, large variances may exist in the determination of the upper and lower bounds of the throughput distribution. These variances degrade the ability of the manager to employ simulation results in planning accurately for resource consumption. Greater accuracy is available from simulation at the expense of increased

numbers of replications. In the cases of larger and more complex networks, however, the cost effectiveness of more accurate information can very rapidly become questionable.

4. Simulation results are not worth their cost, since work tends to expand to fill the time allocated to it (Parkinson's Law); thus, even a "true" simulated project completion date would likely be exceeded as workers and managers aim toward this new, lengthened deadline.

False Steady State

There are two generally accepted types of simulations, terminating and steady-state:

1. A *terminating simulation* is one for which the desired measures of system performance are defined relative to the interval of simulated time $[0, T_E]$, where T_E is the instant in the simulation when a specified event E occurs. T_E may be a random variable. The event E is specified before the simulation begins.
2. A *steady-state simulation* is one for which the measures of performance are defined as limits as the length of the simulation goes to infinity. Since there is no natural event E to terminate the simulation, the length of one simulation is made large enough to get "good" estimates of the measurement quantities of interest. Alternatively, the length of the simulation could be determined by cost considerations. At the point in time when the *transient distribution* of a measurement quantity of interest is essentially no longer changing with increases in the number of simulation runs, we say, intuitively, that the process being measured is in "steady-state." Thus, steady state does not mean that the actual values of the measurement quantity of interest in a single realization (run) of the simulation become constant after some point in time, but that the distribution of the measurement quantity of interest becomes invariant. (Law and Kelton, 1982)

An activity network simulation program performs a simulation of a network by advancing time from event to event. In simulation parlance this is termed a *next-event simulation.* There are three events associated with the simulation of an activity network: the start of the simulation, the end of an activity, and the completion of a simulation run of the network. The start event causes all source nodes to be realized and schedules the activities emanating from the source nodes according to the output type of the source node, either deterministic or probabilistic. In the former case, all activities emanating from the node are scheduled; in the latter, only one of the activities emanating from the node is scheduled. "Scheduling an activity" means that an "end of activity" event is caused to occur (placed on the event calendar) at some future point in time. The simulation then proceeds from event to event until the condition is obtained which indicates that the simulation of the network is completed: the realization of a specified number of sink nodes. The process is then repeated for a specified number of simulations (replications). (Whitehouse, 1973).

The simulation analyst specifies at the beginning of a network simulation experiment the number of simulations (replications) to be made with the simulation model. Each replication produces a simulation sample value of a performance measure of interest. These simulation sample values are accumulated to build a "picture" of the distribution of the performance measure. The picture is often presented as either a frequency histogram of the simulation sample values or as a set of computed sample statistics, usually the mean and standard deviation (or variance) and possibly some higher order moments, or both. The accuracy of the picture is clearly a function of the number of replications conducted.

Hence, in terms of output analysis, activity network simulations are a composite type of simulation: each replication of the network model is terminated when the specified number of sink nodes are realized, so each replication is a terminating simulation. The network model is exercised for a specified number of replications which has been chosen sufficiently large so that the accumulation of simulation sample values results in a reasonably-to-very accurate picture of the distribution of a performance measure. The objective is to accumulate sufficient simulation sample information so that the picture of the distribution of the performance measure does not measurably change with increases in the number of replications, that is to say, so that the distribution of the performance measure becomes invariant; so the simulation experiment is a steady-state simulation.

Steady state, then, is the condition attained in a simulation when the process exhibits a stationary behavior as characterized by its parameters;*false steady state* is the apparent but erroneous appearance of a steady state (Pritsker, 1986). In an activity network simulation, a false steady state occurs when the picture of the distribution of a performance measure built from the accumulated simulation sample values is inaccurate, i.e., when this picture has not yet achieved invariance with respect to increases in the number of replications.

For project management networks, the performance measure of interest is typically the throughput distribution. Because of Central Limit Theorem results, the throughput distribution for most activity networks with even only modest numbers of nodes and activities is normal or near-normal, regardless of the types of distribution functions describing the activity durations. Consequently, the shape of the simulation-developed picture of the distribution, such as displayed in a frequency histogram, cannot be looked to as an indicator that an apparent steady state is actually false. The danger in a false steady state lies in the simulation sample statistics being inaccurate estimators of the throughput distribution's parameters, particularly the mean and standard deviation (or variance), since these parameters are of greatest utility to managers in understanding and controlling project behavior. Consequently, the trick in project management network simulation is specify a sufficiently large number of replications to insure that the simulation-developed picture of the throughput activity resource consumption distribution is accurate, i.e., to insure that the simulation achieves a true steady state with respect to the throughput distribution. Clearly, the greater the number of replications, the lesser the risk of mistaking a false steady state for a true one. However, for large activity networks, large numbers of replications mean high run times and costs, so there is definite trade-off between the risk of a false steady state and the price to be paid to conduct the simulation experiment.

Project management networks are PERT-type networks, i.e., they are acyclic, directed networks. As such, the risk of a false steady state in the simulation of a project management network is, generally speaking, less - some might argue considerably less - than the risk of a false steady state in the simulation of a generalized activity network (GAN), i.e., an activity network with both stochastic (probabilistic) activity resource consumption distributions and probabilistic branching (Elmaghraby, 1977). The presence of probabilistic branching in GANs increases the risk of simulation false steady states over that for acyclic, directed networks. This is easily understood through the following simplistic example. Consider a GAN with a node with two emanating activities: one activity has a very high duration when the activity is taken, but a very low probability of being taken; the other activity is just the reverse, i.e., has a very low duration when the activity is taken, but a very high probability of being taken. The expected number of replications before the first activity is taken will be quite large, and during most simulation experiments the second activity will be repeatedly taken before the first activity is taken for the first time. A performance measure such as average network duration, will be initially estimated too low by the corresponding simulation sample statistic, then too high when the statistic spikes the first time the first activity is taken, etc. A large number of replications will be required before the transient distribution of the performance measure becomes invariant, i.e., before the simulation reaches steady-state. If the transient distribution of the performance measure is sampled too early, it will appear to have stabilized at a lower level than the true value of the average resource consumption; this is a false steady state. Fortunately, project management networks do not evidence such pathologic behavior. A false steady state in the simulation of a project management network will usually occur only when the network has a large number of nodes and activities with stochastic distribution functions with high variabilities. A large number of replications will be required before the throughput distribution achieves steady-state. However, if the distribution is sampled too early, a false steady state may be observed.

2.6.2 Current Status of Network Simulation

The focus of recent research on stochastic networks has closely paralleled the recommendations of Schonberger (1981) in two primary areas: the distribution of project completion times, and attempts to identify the paths and activities most likely to become critical (Ragsdale, 1989). Simulation is involved in one approach in each of these areas.

"Intelligent" Simulation

Standard PERT techniques tend to result in overly optimistic estimates of project completion times. Early attempts to deal with this problem were based on Monte Carlo simulation techniques and spurred the development of specialized networking languages such as GERTS and SLAM. Schonberger (1981) noted that it can be very costly and time consuming to use these tools. Cook and Jennings (1979) showed that accurate approximations to Monte Carlo simulation results can be achieved less expensively using "intelligent" simulation methods. Their study analyzed three heuristics which can be used to estimate a network's distribution of completion times/costs. Essentially, each heuristic attempts to identify paths in the network which have little or no chance of becoming critical. These paths are then discarded, simplifying the network and reducing the simulation problem. They gave empirical evidence indicating that PERT estimates of completion times may be anywhere from 8 to 30 times less accurate than their best "intelligent" simulation method. Most recently, there has apparently been no further investigation of "intelligent" simulation methods applied to stochastic networks, a situation which Ragsdale (1989) has found disappointing.

Identifying Critical Paths and Activities

Identifying the paths most likely to become critical is important not only in estimating the completion time distribution but also, and perhaps more importantly, for planning and control purposes. Criticality indices attempt to provide a quantitative tool that identifies the "complex, delay-prone segments" of a network which Schonberger (1981) advised should be the focus of managerial attention. The criticality index of a path is defined as the probability that the path's duration exceeds the duration of all other paths in the network. Knowing the criticality index of each path in the network would provide valuable information as to which paths are most critical. Similarly, if one particular activity appeared in many of the most critical paths, the activity itself may be considered critical. The criticality index of an activity is then defined as the sum of all path criticality indices of paths in which the activity occurs. To estimate path criticality indices, simulation is a tool which can be used to determine the relative frequency with which each path is critical. In addition to long simulation run times, the task of uniquely identifying and recording which path is critical in each repetition of the simulation is now added. Ragsdale (1989) concluded that this quickly becomes impractical for problems of "realistic size." Dodin (1984) suggested a procedure for estimating the rankings of the path criticality indices for the K most critical paths in the network. Rather than simulating the entire network repeatedly, Dodin suggested using simulation to determine the K "stochastically dominating" paths entering each successive node in the network. The end result of this process is a list of the K most dominating paths entering the sink node - or the K most critical paths. These critical paths, ordered by their simulated expected length, approximate the rankings of their criticality indices. (Dodin also developed analytic methods to approximate activity and node criticality indices and to identify the K most stochastically dominating paths; these are discussed in Sections 2.8.2 and 3.4.2.)

2.6.3 Recent Simulation Approaches to Network Analysis

Badiru (1991a and 1991b) presents simulation as a useful analytical tool for project network analysis and a powerful tool for evaluating many of the decision parameters involved in project management. To illustrate the effectiveness of computer simulation for project planning, he and G. E. Whitehouse co-developed the PC-supportable STARC (stochastic time and resource constraints) and STARC 2.0 computer programs, which simulate project networks and perform "what if" analysis of projects with stochastic activity times and resource constraints. STATGRAPHICS software is used in conjunction with STARC to illustrate some of the post-simulation statistical analyses that can be conducted. STARC's capabilities are a subset of SLAM's, and STARC's only advantages over SLAM are its customized output and simplicity of model construction and input.

Avramidis, Bauer, and Wilson (1991) investigated several procedures for using path control variables to improve the accuracy of simulation-based point and confidence interval estimators of the mean completion time of a stochastic network. Because each path control variate is the duration of the corresponding directed path from source to sink, a vector of selected path controls has both a known mean and covariance matrix. They incorporated this information into estimation procedures for both normal and nonnormal responses. Their experimental results show that although large improvements in accuracy can be achieved with some of these procedures, the confidence interval estimators for normal responses may suffer loss of coverage probability in some applications.

Simao and Powell (1992a, 1992b) introduced a discrete-time approach for simulating stochastic, transient networks of bulk queues that often arise in consolidation networks. They focused on the simulation of stochastic, transient networks that involve the consolidation of customers on vehicles, originally motivated by the development of a computationally efficient method for simulating the performance of less-than-truckload motor carrier networks. They presented the state variables and equations required to model the problem, introduced a set of approximations to produce a computationally tractable algorithm, and described the results of extensive numerical experiments which tested the accuracy of the approximations and the overall efficiency of the procedure. Their work illustrates the use of numerical approximation methods to simplify network analysis, specifically, to simplify the simulation of a restricted class of transportation networks.

2.7 Performance of Network Reduction Techniques

Since we have proposed to develop a new, more capable, and more efficient numerical approximation technique and then to validate its performance by exercising algorithms based on the technique combined with the leading network reduction methods and comparing the results with the reported performance of their competitors, we must have a rigorous strategy for the validation and performance comparison of network approximation and reduction techniques and the tools necessary to implement that strategy. Our performance validation strategy is based on design of experiments; the tools necessary to implement the strategy are the random generation of test networks and network simulation.

2.7.1 Design of Experiments

Rigorous statistical design in computer experiments has become an area of increasing importance in algorithm testing. Empirical testing of an algorithm should begin with the identification of the characteristics of the problem which influence the performance of the algorithm in solving the problem. One then generates test problems with sufficient variations in those characteristics so that any and all peculiarities in the algorithm's performance can be observed and extracted. The behavior of the algorithm should then be explained by statistical methods as well as analyses of the algorithm itself. The framework for the controlled execution of this testing is design of experiments (DOE).

Hung and Divoky (1988) developed a DOE approach for comparing the computational efficiency of five different algorithms to determine the shortest path through a network. First, the control or independent variables that define a network were identified: the number of nodes, the number of arcs, and the network structure, which they described as the incidence function which defines the end node of each arc. Since each arc has a weight, the distribution of arc weights was characterized by another group of independent variables: type of distribution, and parameter(s) of those types of distributions. A 4-factor factorial design was constructed on the independent variables (numbers of nodes and arcs, arc weight distribution type and mean (or variance)). For each cell in the design, a network was generated using a Hamiltonian path and random assignment of left-over arcs to nodes. Fifty different source nodes were randomly selected for the network in each cell, and the five different shortest path algorithms were exercised, starting once from each of the source nodes. The experimental or dependent variable was the performance of each algorithm against the test networks as measured by CPU time; number of iterations and cost of computation and computer storage were considered as performance measures, but rejected in favor of CPU time. The entire process was repeated for 240 different combinations in the factorial design (5 numbers of nodes, 4 numbers of arcs, 3 types of arc weight distributions, and 4 parameter values for each of those distributions). The CPU times required by the algorithms to determine the shortest paths through the test networks not only permitted head-to-head performance comparisons between the algorithms, but also revealed some theretofore unknown properties of some of the algorithms.

Dodin (1985a) employed a similar, but somewhat less rigorous approach in evaluating the performance of the sequential approximation and reduction method. We discussed the efficiency of sequential approximation in Section 2.5.2 [above]. Dodin generated "strongly randomized networks" for a small number of combinations of number of nodes and number of arcs. Seven different types of activity duration distributions were considered. The activities of each of the test networks were assigned either the same type of duration distribution, or a randomly selected type. The parameter(s) of each activity duration distribution were then randomly selected. Each test network was reduced by sequential approximation and approximated by Monte Carlo simulation, and the results compared with four performance measures. Since not all combinations of network size and structure, and activity duration distribution type and parameter(s), were considered, Dodin's performance evaluation of sequential approximation was an incomplete factorial design. While the computational experience of the algorithm at the design points (cells) considered was in agreement with its expected performance based on the theory of its construction, Dodin's testing failed to exercise the algorithm across its operating envelope. Hence, although there has been no reason as yet evidenced to believe that the algorithm does not perform both as expected and desired in reducing stochastic networks, it has not been completely validated. To do so requires a more robust design space than that of Dodin's evaluation.

Dodin (1993) provided further insight into the randomization of parameters which is required in testing the dependency of a network reduction method's efficiency on the parameters of the activity duration distributions. A method that works well when the variances of the activity duration distributions are small may not be efficient for larger variances. Similarly, a method that is efficient when the activity durations are all of one type of distribution, for example, when they are normally distributed, may not be efficient when they are all of another type, for example, when they are exponentially distributed. Therefore, the parameters need to be generated in a manner compatible with the nature of the problem under consideration. Unfortunately, randomizing the parameters cannot be standardized like the structure of the network or the number of arcs; each structure or value of the number of arcs can have an equal probability of being selected, but the parameters can have infinite realizations and can often depend on the nature of the decision problem. Consequently, randomization of the parameters of the activities does not have an "equal probability" implication; rather, it is accomplished in a manner that reflects the variability of the parameters under consideration and the reality of the decision problem.

2.7.2 Random Network Generation

To derive reliable results in the efficiency tests of stochastic network reduction methods, it is desirable, if not mandatory, that the activity network models (size, structure, and parameters) included in the computational experiments be randomly generated

(Elmaghraby and Herroelen, 1980). If the network size and structure are given, then the randomization is limited to the parameters of the activities. This is the case in most of the published literature; the set of representative test problems is either taken from existing problem sets described in the literature, typical of which is the set of 110 test problems designed by Patterson (1984) for resource constrained scheduling, or individual models designed by individual authors. Examples of the latter are the network in Figure 4 in Kleindorfer (1971), which was used by Shogan (1977) and Dodin (1985c), and networks designed and used by Alvarez-Valdes (1988), Christofides, Alvarez-Valdes and Tamarit (1987), Kurtulus and Davis (1987) and Talbot (1982). Demeulemeester, Dodin, and Herroelen (1993) refer to this kind of activity network model, which is limited to the random generation of the parameters of the activities, as "weakly randomized networks." Sometimes all that is required for efficiency testing is a set of weakly randomized networks; that is the case, for example, if one wishes to examine the efficiency of a given method when triangular or diamond-shaped networks are used. When the randomization is extended to include the size and structure of the network, where the structure is selected with equal probability from the space of all feasible structures, then Demeulemeester et al. (1993) refer to it as a "strongly randomized network."

The size of a stochastic network is determined by the number of nodes, N, and the number of activities (arcs), A. A network with N nodes can have from $N-1$ up to $N(N-1)/2$ arcs. An approximation and reduction method that works well for a network with low density (the ratio of arcs to nodes is small) may not be efficient for a dense network. Therefore, networks with various densities must be used in efficiency and comparison testing. As a result, the number of arcs for a given N needs to be generated from the space of all arcs which ranges from $N-1$ to $N(N-1)/2$ arcs. Then, once the size of the network is determined, the structure of the network needs to be generated at random from the space of all networks with the given size.

Demeulemeester et al. (1993) presented an algorithm which generates strongly randomized activity networks, based on the working papers of Herroelen and Caestecker (1979) and Dodin (1980). The following notation was used in their development:

$G(N, A)$ = activity network with N nodes and A arcs.

a_{ij} = arc(ij), the activity starting in node i and ending in node j; the nodes are numbered such that i is less than j, for i ranging from 1 to $N-1$ and j ranging from 2 to N.

$IN(i)$ = in - degree of node i, the number of arcs incident to node i.

$OUT(i)$ = out - degree of node i, the number of arcs emanating from node i.

$\delta(i, j)$ = indicator equaling 1 if there is an arc connecting node i to node j, and 0 otherwise.

L = number of generated arcs, generated either to be deleted or added .

$\lfloor a \rfloor$ = largest integer less than or equal to the real value of a.

$U(a, b)$ = uniform distribution with a lower limit of a and an upper limit of b.

For a given N and A, the objective is to generate $G(N,A)$ from the space of all activity networks with the given N and A. Either of the two following procedures can be used to guarantee the complete randomization of $G(N,A)$: the deletion method (DM) or the addition method (AM). Proofs are given in Demeulemeester et al. (1993).

Deletion Method

The deletion method starts with a completely connected acyclic network, i.e., the upper triangle of the adjacency matrix is filled with ones, or $\delta(i,j) = 1$ for all entries with $i < j$. Therefore, DM starts with a network with $N(N-1)/2$ arcs and then deletes $K = N(N-1)/2 - A$ arcs subject to the following restrictions:

1. The network has one start and one end node.
2. $IN(i) \geq 1$ for all $i = 2,3,\ldots,N$, and
   $OUT(i) \geq 1$ for all $i = 1,2,\ldots,N-1$
3. The generated network $G(N,A)$ is completely randomized, i.e., all feasible networks possessing the count N and A are equally probable.

The DM proceeds as follows, as depicted in the flowchart in Figure 20:

Step 1. Initialization:
Set
$\delta(i,j) = 1$ for all $i = 1,2,\ldots,N-1$,
and $j = i+1,\ldots,N$
$OUT(i) = N - i$ for all $i = 1,2,\ldots,N-1$
$IN(i) = i - 1$ for all $i = 2,\ldots,N$
$L = 0$ and $K = N(N-1)/2 - A$

Step 2. Generating the arc to be deleted:

i. Generate a random value; denote it by $Y \sim U(0,1)$ and let
$j = \lfloor N + 0.5 - \sqrt{N(N-1)Y + 0.25} \rfloor$

ii. If $OUT(j) = 1$ or $IN(k) = 1$ for all $k > j$, then go to Step 2i; otherwise, continue.

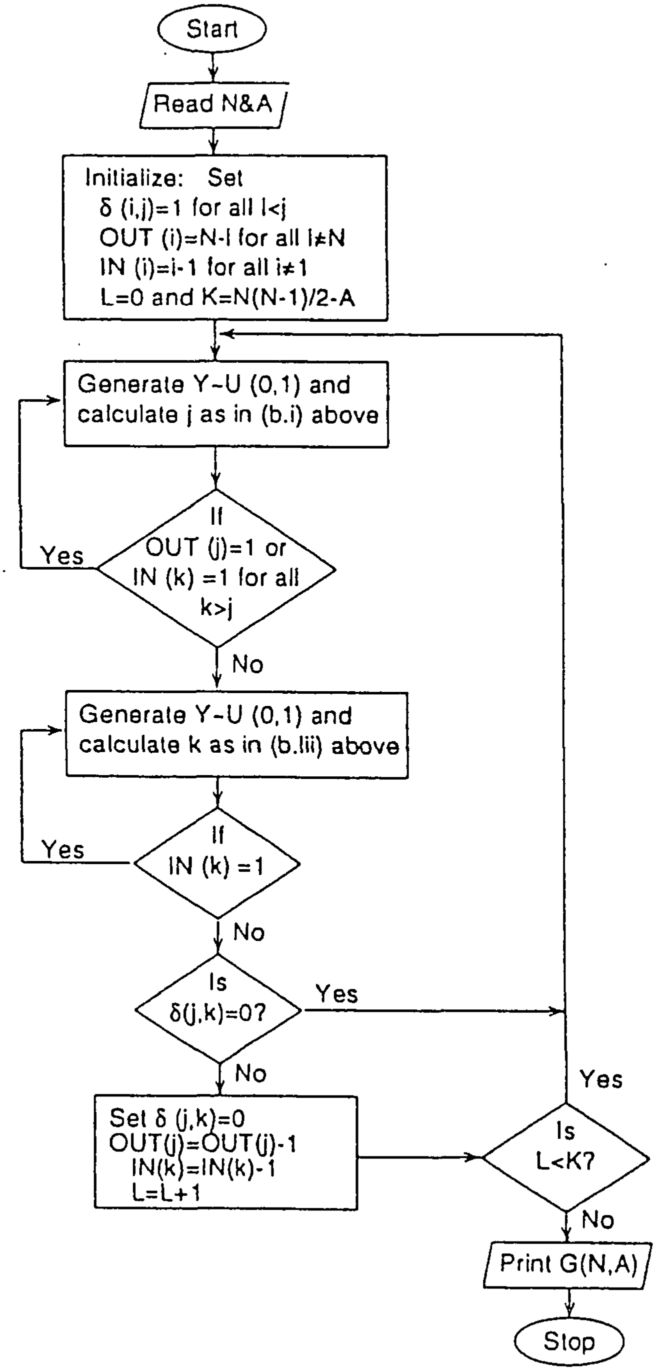


Figure 20. The deletion method.
[Adapted from Demeulemeester et al. (1993)]

iii. Generate another random value $Y \sim U(0,1)$ and let
$k = \lfloor j+1+Y(N-j) \rfloor$

iv. If $IN(k) = 1$, go to Step 2iii; otherwise, the arc (j,k) is a candidate for deletion.

Step 3. Deletion:
If $\delta(j,k) = 0$, go to Step 2i; otherwise, set
$\delta(j,k) = 0$
$OUT(j) = OUT(j) - 1$
$IN(k) = IN(k) - 1$ and
$L = L + 1$

Step 4. Termination:
If $L < K$, go to Step 2i; otherwise, a completely randomized
$G(N,A)$ is generated.

Addition Method

This procedure does the reverse of the DM; it starts with an adjacency matrix with $\delta(j,k) = 0$ for all $i < j$ except $\delta(1,2) = \delta(N-1,N) = 1$, which guarantees one start and one end node. The procedure then proceeds to generate $A-2$ arcs subject to the same three restrictions stated for the DM. The AM deals with two sets of arcs. The first set has the *feasibility arcs* which might be needed to guarantee restriction ii; these may range from zero to $2N-6$ arcs. The second set has the remainder of the arcs, which are referred to as *free arcs*; F denotes the set of free arcs. The AM generates as many as possible of the free arcs. They are generated at random from the set of $N(N-1)/2-2$ arcs. As more free arcs are generated, more nodes become feasible, i.e., they become incident into and/or emanating from nodes. Consequently, the first set of arcs, the feasibility arcs, may decrease and F may increase. If F reaches zero and there are nodes (other than 1 or N) with zero in-degree or zero out-degree, then the AM generates the arcs necessary for feasibility. This step may result in generating more than A arcs; this case could happen in activity networks where $A \leq 2N-4$. The DM is used to delete the extra

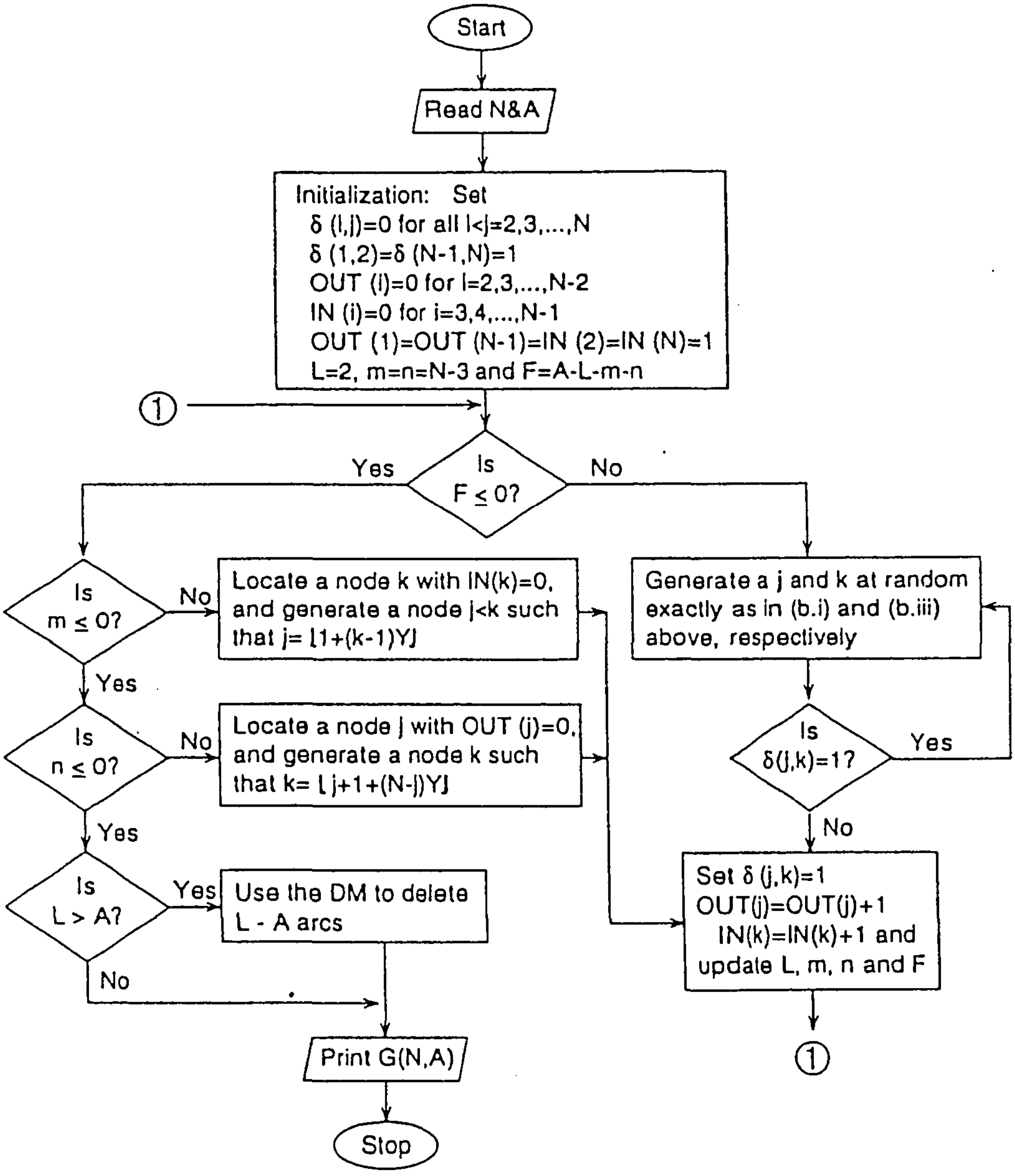

Start
Read N&A
Initialization: Set
δ (I,J)=0 for all I<J=2,3,...,N
δ (1,2)=δ (N-1,N)=1
OUT (I)=0 for I=2,3,...,N-2
IN (i)=0 for i=3,4,...,N-1
OUT (1)=OUT (N-1)=IN (2)=IN (N)=1
L=2, m=n=N-3 and F=A-L-m-n
①
Is F ≤ 0?
Yes
No
Is m ≤ 0?
No
Locate a node k with IN(k)=0, and generate a node j<k such that j= ⌊1+(k-1)Y⌋
Yes
Is n ≤ 0?
No
Locate a node j with OUT (j)=0, and generate a node k such that k= ⌊j+1+(N-j)Y⌋
Yes
Is L > A?
Yes
Use the DM to delete L - A arcs
No
Print G(N,A)
Stop
Generate a j and k at random exactly as in (b.i) and (b.iii) above, respectively
Is δ(j,k)=1?
Yes
No
Set δ (j,k)=1
OUT(j)=OUT(j)+1
IN(k)=IN(k)+1 and
update L, m, n and F
①

Figure 21. The addition method.
[Adapted from Demeulemeester et al. (1993)]

arcs at random. Figure 21 summarizes the steps of the AM.

Clearly, either method can be used to generate a random $G(N,A)$. The DM deletes $N(N-1)/2-A$ arcs while the AM adds $A-2$ arcs. Consequently, using the DM in nondense networks can be time consuming; for example, in an activity network with $N=100$ and $A=150$, using the DM requires the deletion of 4,800 arcs, while the AM requires adding 148 arcs. Therefore, Demeulemeester et al. (1993) recommend the use of the AM for nondense networks and the DM for dense networks. More specifically, they recommend the use of the DM if $A \geq K$, where $K = N(N-1)/4$, and the AM otherwise. This rule is based on the number of calculations required by each method to generate the required network. The quantity K is half the arcs in a completely connected network; therefore, if $A > K$, then one generates the network by deleting fewer than K arcs, which is more efficient than adding $A > K$ arcs.

Generating A Set Of Strongly Random Activity Networks

When a set of strongly generated activity networks is needed to test a network reduction method, this means (1) the generation of a set of (N,A) pairs, where for each pair several network structures are generated using either the DM or the AM, and (2) the generation of corresponding activity parameters. With respect to the number of nodes, N is either specified or drawn randomly from a range depending on the reduction method to be tested. Once N is specified, then A, the number of arcs, is either specified or randomly generated from the range $[N, N(N-1)/2]$. Normally, a number of A values will be used for each N. Considering the number of structures associated with a network size (N,A), then treating all the values of A in the above range in an equally likely manner is not an accurate randomization of A. Table 7 shows the number of feasible network structures for some different combinations of N and A. A more accurate alternative would be to associate a weight with each value of A proportional to the number of structures, $G(N,A)$, satisfying the size (N,A), i.e., determine the distribution of A given N. However, this alternative requires the enumeration of the structures $G(N, A)$ for all values of A in the above range, a task that is very difficult for nontrivial values of N. Consequently, Demeulemeester et al. (1993) developed a heuristic procedure to approximate the distribution of A.

Table 7 shows that for $N = 4$ the probability distribution of A is symmetric. However, for $N > 4$ the number of structures increases sharply and the probability distribution of A given N is not completely symmetric; it is close to being normal with a slight skewness to the right. Figure 22 plots the ranges of A for different values of N. The dotted curve represents the mean of the ranges of the A values. However, Table 7 indicates that the actual mean of A lies below the mean of the ranges. Therefore, the mean of the ranges can be adjusted and used to approximate the mean of A. Set

$$L_A = N - 1 \text{ and } U_A = N(N-1)/2$$

Then the mean of A is approximated by:

$$\mu_a = \frac{(L_A + U_A)}{2} - \frac{(U_A - L_A)^2}{500}$$

and its standard deviation is approximated by:

$$\sigma_A = \frac{(U_A - L_A)}{2.5}$$

Table 7. The Number of Feasible Network Structures for Different Combinations of N and A. [Adapted from Demeulemeester et al. (1993)]

N	A														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1														
3		1	1												
4			1	4	4	1									
5				1	11	33	42	26	8	1					
6					1	26	171	507	840	865	584	262	76	13	1

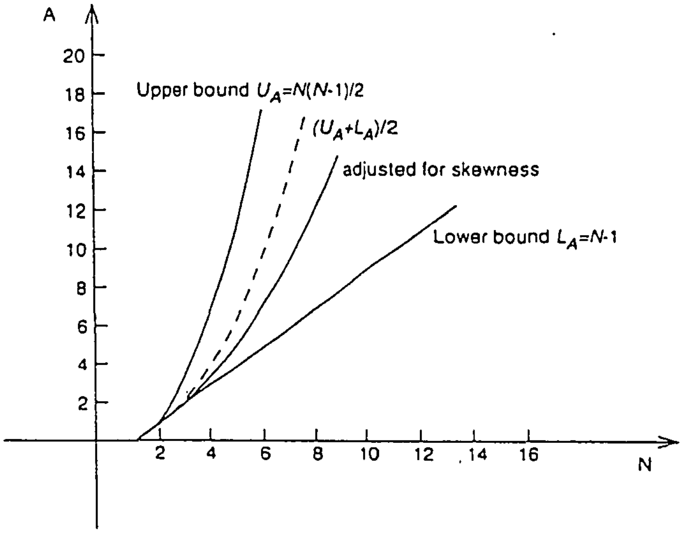


Figure 22. The random variable A as a function of N.
[Adapted from Demeulemeester et al. (1993)]

For a given N, an A value is obtained by drawing a random value from a normal distribution with the above mean and standard deviation. Given the pair (N, A), then either the DM or the AM can be used to generate a random network structure.

Demeulemeester et al. (1993) reported that CPU time requirements for generating strongly random activity networks are negligible. Table 9 shows the CPU time requirements on a VAX 8820 minicomputer for the 25 activity networks specified in Table 8. The activity network sizes given in Table 8 were designed to capture the worst and best case scenarios for the DM and the AM. The first three columns of Table 8 require the use of the AM because $A < N(N-1)/4$. Column 3 has the worst case for the AM because A is close to the border point recommended for use of either the AM or the DM. Column 2 has what is expected to be the best case for the AM because A is close to the value where the AM may not generate more than A arcs, i.e., the DM will not be used within the AM (see Figure 21). Column 1 has a bad case for the AM because there is a higher chance of using the DM to delete the extra arcs generated by the AM. With respect to the DM, there are two cases: (1) the best case, represented by the A values of column 5, where the number of arcs to be deleted is very small, i.e., the network is almost completely connected, and (2) the worst case, represented by column 4, where the arcs to be deleted are very close to the maximum, which is half of the arcs in a completely connected network.

Table 8. Network Sizes Used in the Efficiency Test of the Random Activity Network Generators. [Adapted from Demeulemeester et al. (1993)]

	Addition Method			Deletion Method	
			Close to $N(N\text{-}1)/$ $4-5$	Close to $N(N\text{-}1)/$ $4+5$	Close to $N(N-1)/$ 2
N	$1.5N$	$2N$			
10	15	20	21	25	40
20	30	40	90	100	175
30	45	60	210	220	400
50	75	100	600	620	990
60	90	120	875	900	1500

Table 9. CPU Time Requirements in Seconds for Activity Networks of the Sizes Given in Table 8. [Adapted from Demeulemeester et al. (1993)]

	Addition Method			Deletion Method	
			Close to $N(N-1)/$ $4-5$	Close to $N(N-1)/$ $4+5$	Close to $N(N-1)/$ 2
N	$1.5N$	$2N$			
10	0.012	0.010	0.010	0.012	0.005
20	0.019	0.027	0.051	0.031	0.019
30	0.032	0.043	0.180	0.141	0.027
50	0.043	0.059	0.291	0.280	0.138
60	0.051	0.093	0.426	0.435	0.221

Demeulemeester et al. (1993) concluded from Table 9 that the CPU time requirements do not result in an unacceptable penalty for using the random network generator. The CPU time required for generating any of the 25 networks of Table 8 was less than 0.05 of a second on a VAX 8820. Very dense networks required less time to generate than less dense networks because, using the DM, fewer arcs are generated; compare, for example, columns 5 and 4 of Table 9, where both columns required the DM and networks of column 5 are more dense than those of column 4.

The worst-case scenario for both methods is represented by columns 3 and 4 of Tables 8 and 9. In column 3 the AM was used to generate close to $N(N-1)/4$ arcs, while in column 4 the DM was used to generate also close to $N(N-1)/4$ arcs. First, the CPU time requirements in both cases were compatible, which reaffirmed the rule Demeulemeester et al. (1993) recommended on when to use either method. Second, the CPU time requirements for using either method could not be worse than these two respective cases. As a rule, generating an arc in a network, with a density greater than 1.5, using either method required less than 0.0005 of a second. Generating a network of size (60,1500) required less than 0.30 of a second.

Activity networks with low density, say between $(N-1)/N$ and $(2N-4)/4$, may require more time to generate. This is due to two factors. First, these networks use the AM, and due to the feasibility constraints, more than A arcs may need to be added. In this case, the DM is called to delete the extra arcs. Second, in using the DM to delete the few extra arcs many nodes (start and/or end) may be generated and neglected due to the feasibility issue, i.e., the DM keeps searching for an ordered pair, arc (i, j) with $\delta(i,j) = 1$ in a very sparse network, which can be more time consuming. As the density of the network converges to $(N-1)/N$, the efficiency of both methods decreases. (Demeulemeester et al., 1993)

2.8 Large Network Approaches

When stochastic project management networks are large, reduction methods may require lengthy computer run times and complex network data management/storage. Two approaches have been considered for handling large networks: network decomposition, and the K most critical paths. The idea behind network decomposition is quite simple: break a large network apart into subnetworks which can be separately reduced, then combine the results. As it is unrealistic to expect the network user to effect a workable decomposition, research has focused on aggregation/disaggregation techniques which could be used to identify the subnetworks. This has met with limited success, and the K most critical paths has emerged as the more promising approach.

2.8.1 Network Decomposition

Aggregation/Disaggregation Techniques

Rogers, Plante, Wong, and Evans (1991) surveyed the numerous aggregation and disaggregation techniques which have been developed for optimization applications. Disaggregation analysis is defined as the reverse of aggregation analysis, i.e., where the results of solving a reduced model are used to estimate the solution of the original model or to cluster levels of interest for the model under consideration. The issues of disaggregation analysis are relatively less complex than their antitheses in aggregation analysis. The entity (entities) to disaggregate will naturally be identical to the entity (entities) that are clustered and combined. The level of interest for most problem settings is at the level of the original problem; thus, reversal procedures for similarity measures and cluster procedures are not required. One important disaggregation approach is the method of dissection, which is the antithesis of the method of combination; it is analogous to fixed-weight combination. Aggregation and disaggregation techniques may also allow for a reduction of the computational time required to solve problems of optimality. The entire process of aggregation and disaggregation may be embedded in an iterative procedure to successively aggregate data, solve small problems, and disaggregate data, which are referred to as iterative aggregation-disaggregation (IAD) techniques. The authors discuss a number of IAD techniques, with applications to linear programming (classical transportation problems and extensions; multi-commodity distribution models; production, planning and scheduling), nonlinear programming (transportation problems, production and scheduling, location/allocation methods, and accounting), integer programming, dynamic programming, and Markov decision processes. In the arena of network applications, Hackman and Leachman (1989) are referenced for a study of multiproject planning models involving clustering parallel detailed activities that utilize the same mix of resources. Work flow among aggregates was defined to be continuous time and rate based, leading to a linear programming formulation and hence to an example of aggregate modeling. Of 199 references cited by Rogers et al. (1991), only two by Francis address aggregation/ disaggregation of acyclic, directed networks.

Aggregation/disaggregation techniques in PERT/CPM networks.

Francis (1985) developed an aggregation method based on network flows to solve the capacitated transshipment problem. The method requires an initial partition , N^1, of the original problem network, N^0. The N^1-network problem is an aggregation of the original problem. An iterative procedure uses at the r$^{\text{th}}$ step a basic feasible solution of the N^{r+1}-network problem to "jump start" a basic solution for the N^{r+1}-network problem, where the N^{r+1}-network is obtained from the N^r-network by a refinement of the N^r-partition obtained by splitting one of the partition subnetworks into two subnetworks. When the N^r-partition can be refined no further, it is the original problem network, N^0, and the current basic feasible solution solves that original problem. Francis (1986) applied his method to the analysis of PERT/CPM networks, showing that subprojects, defined as subgraphs of the original network, may be analyzed as separate networks and the earliest and latest event times need not be recalculated. Slack time for each activity of the subproject analysis was shown to be a lower bound for the actual slack time of the activities of the subproject included in the analysis of the original network. Regrettably, Francis' method does not address the problem of how the initial partition, N^1, which is a decomposition of the original network, is to be obtained. About N^1 he wrote:

> We assume that N^1 is determined subjectively and logically for the particular problem context and managerial application at hand. ... It is assumed at Block 2 [of the algorithm] that N^1, the initial node partition set, "makes sense" for the problem context at hand.
>
> (Francis, 1986)

2.8.2 The *K* Most Critical Paths

When networks are large and burdensome to reduce, managerial insight into the projects represented by the networks can be obtained by identifying the paths and activities that are critical to achieving the objectives of the projects. In deterministic activity networks (DANs), in which activity durations are constants, an activity is either critical with a slack time of zero, or noncritical with a slack time greater than zero. However, a noncritical activity with a smaller slack time should receive more attention from project management than a noncritical activity with a larger slack time, because any slippage in the critical path(s), the sequence(s) of activities with zero slacks, might cause an activity to become critical. Therefore, measured against other activities, the criticality of an activity constitutes a relative measure indicating the activity's demand upon project management attention. Hence, critical paths and activity criticalities play an important role in project management.

By contrast, in a stochastic network in which the durations of activities are random variables, the tasks of project management and control are more problematic due to the difficulty in determining the critical (longest) path(s) and criticalities of the activities. Dodin (1984) illustrates the difficulty:

> Let P denote the set of paths in a PERT network, and $Z(\pi_l)$ denote the duration of path $\pi_l \in P$. Then,
>
> $$Z(\pi_l) = \sum_{a_{ij} \in \pi_l} T_{ij},$$
>
> where T_{ij} is the duration of activity $a_{ij} \in A$. The criticality of a path $\pi_l \in P$ is measured by the probability that its duration, $Z(\pi_l)$, is greater than or equal to the duration of every other path. We call this probability the *criticality index* of the path, and denote it by CR. Thus for any $\pi_l \in P$,
>
> $$CR(\pi_l) = \mathrm{P}[Z(\pi_l) \geq Z(\pi_q) \text{ for all } \pi_q \in P]. \qquad (1)$$
>
> The criticality of an activity is measured by the sum of the criticality indices of the paths containing that activity. We call this measure the *criticality index* of the activity, and denote it by CA. Therefore, for an arc $a_{ij} \in A$,
>
> $$CA(a_{ij}) = \sum_{\{\pi_l | a_{ij} \in \pi_l\}} CR(\pi_l).$$
>
> Clearly, identifying the most critical paths and activities, i.e., the paths and activities with the highest criticality indices, requires both identifying all the paths in the set P, which is a burdensome task, and evaluating the criticality index (1) for every path $\pi_l \in P$. which is impossible for most PERT networks.
>
> (Dodin, 1984)

The problem of identifying the K most critical paths in a stochastic network is of both theoretical and practical interest. Identifying the K paths with the highest criticality indices aids in determining the most critical segments in a project, and the most critical activities in a stochastic network (problem 13, chapter 1, Elmaghraby, 1977): if an activity lies on all of the K most critical paths, then it must be among the most critical activities in a project. The durations of the most critical paths in a stochastic network can be used to test the sensitivity of the project completion time, $T_N = \max_{\pi_l \in P}\{Z(\pi_l)\}$. As in DANs, identifying the most critical paths and activities supports the tasks of project scheduling and control.

The K most critical paths in a stochastic network can be identified in one of two ways: by complete enumeration, which is practical only for small-size networks, or by Monte Carlo simulation. In either case, all the paths must be identified and the corresponding

*CR*s calculated, both of which are burdensome tasks; then the paths must be ranked in nonincreasing order of their *CR*s. The problem of calculating *CR*s and *CA*s was first considered by VanSlyke (1963), who used a crude simulation to approximate their values. Martin (1965) defined *CR* and *CA* conceptually without suggesting a practical method to obtain their values. Sigal et al. (1979) suggested the use of conditional Monte Carlo simulation to approximate the *CR*s, and then used them to approximate the *CA*s.

Dodin (1984) developed a heuristic procedure based on stochastic dominance relations among the paths. The heuristic was the first analytical procedure to identify the K most critical paths without using Monte Carlo simulation or identifying all the paths and the corresponding *CR*s. The procedure starts at node 2, where it identifies the single path ending at node 2. Then, it moves sequentially to nodes 3,4,... until it reaches the sink node, N. At each step, if the number of paths ending at the node is greater than K, the procedure has identified and ranked the K most critical paths to this node; otherwise, the procedure has identified and ranked all paths ending at the node.

The following notation was used in his development:

$$IN(j) = \text{in - degree of node } j.$$

$$P_j = \text{set of paths ending at node } j.$$

$$\pi_j^k = k^{\text{th}} \text{ critical path ending at node } j.$$

$$Z(\pi_j^k) = \text{random variable, duration of path.}$$

Path π_j^k *dominates in probability (stochastically dominates)* path π_j^l, symbolically $\pi_j^k \geq \pi_j^l$, if

$$\mathrm{P}[Z(\pi_j^k) \geq Z(\pi_i^l)] \geq \mathrm{P}[Z(\pi_i^l) \geq Z(\pi_j^k)]$$

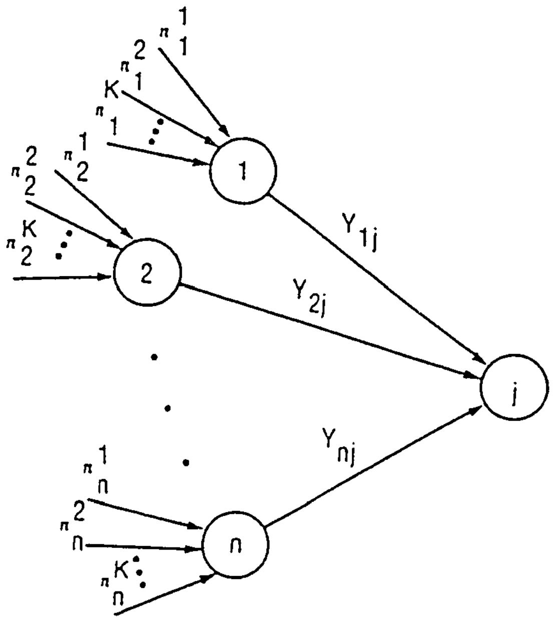


Figure 23. Node j in a PERT network.
[Adapted from Dodin (1984)]

The heuristic assumes that the network has a source node, node 1, and a single sink node, node N, and additionally that each node i $(i \neq 1, N)$ is connected from below and above, i.e., there is at least one path from source to sink containing every node i $(i \neq 1, N)$. The heuristic proceeds as follows:

1. Start at node $j = 1$. i.e., the starting node which has no paths ending at it. Therefore, $\pi_1^k = \varnothing$ and $Z(\pi_1^k) = 0$ with probability one for all $k = 1, 2, \ldots, K$.
2. Sequentially proceed to node $j + 1 \leq N$, identifying the first K critical paths as follows:

   2a. Assume that the process is at node j where $j = 2, 3, \ldots, N$. Suppose there are n arcs ending in node j and for convenience these arcs and their starting nodes are numbered from 1 to n as shown in Figure 23. Furthermore, assume that the K paths ending at node $i < j$ are ordered (as they are identified) such that $l < k$ if $\pi_i^l \geq \pi_i^k$. Hence, we have n ordered sets of paths with each set containing K or fewer paths (Figure 23).

2b. To identify π_j^1, we consider only the first path in each of the n ordered sets. Let $\pi_i^{-1} = \pi_i^1 + a_{ij}$ and $Z(\pi_i^{-1}) = Z(\pi_i^1) + T_{ij}$ for all $i = 1,2,\ldots,n$. Then, $\pi_j^1 = \pi_l^{-1}$ if $\geq \pi_i^{-1}$ for all $i = 1,2,\ldots,n$ and $i \neq l$. In this case, π_l^1 is flagged and π_l^2 will take its place, i.e., let $\pi_l^{-1} = \pi_l^2 + a_{lj}$. The second path π_j^2 is chosen similarly from a set of n paths (the first path in each of the n ordered sets); i.e., $\pi_j^2 = \pi_q^{-1}$ if $\pi_q^{-1} \geq \pi_i^{-1}$ for all $i = 1,2,\ldots,n$ and $i \neq q$. The remaining $K-2$ paths are chosen similarly.

2c. If $\pi_j^k = \pi_i^l + a_{ij}$ for an $i = 1,2,\ldots,n$ and $l = 1,2,\ldots,K$, then $Z(\pi_j^k) = Z(\pi_i^l) + T_{ij}$. The probability density function (pdf) of the random variable $Z(\pi_j^k)$ is obtained by convoluting the pdf's of the random variables $Z(\pi_i^l)$ and T_{ij}.

3. If $j = N$, stop; otherwise, go to Step 2.

The heuristic is depicted in the flowchart in Figure 24.

Complexity

For the case when continuous activity duration distributions are discretized, Dodin (1984) established the complexity of the procedure, which is a function of N, K, and the number of discretization points (realizations), NR. At any node $j \neq 1$, where $n = IN(j) \leq j - 1$, the heuristic must identify K paths. From Step 2b, the first path is chosen from the first n paths $\{\pi_i^1 + a_{ij}$ for $i = 1,2,\ldots,n\}$. Hence, identifying the first path π_j^1 requires n convolutions, where each convolution is of the form $Z(\pi_i^1) + T_{ij}$. Also, from Step 2b, each of the remaining $K-1$ paths requires only one convolution of the same type. Each convolution is $O(NR)^2$, where NR is the maximum number of realizations of either of the discrete random variables $Z(\pi_i^k)$ and T_{ij}. If the number of arithmetic operations in one convolution is C, then the total number of arithmetic operations related to convolutions is:

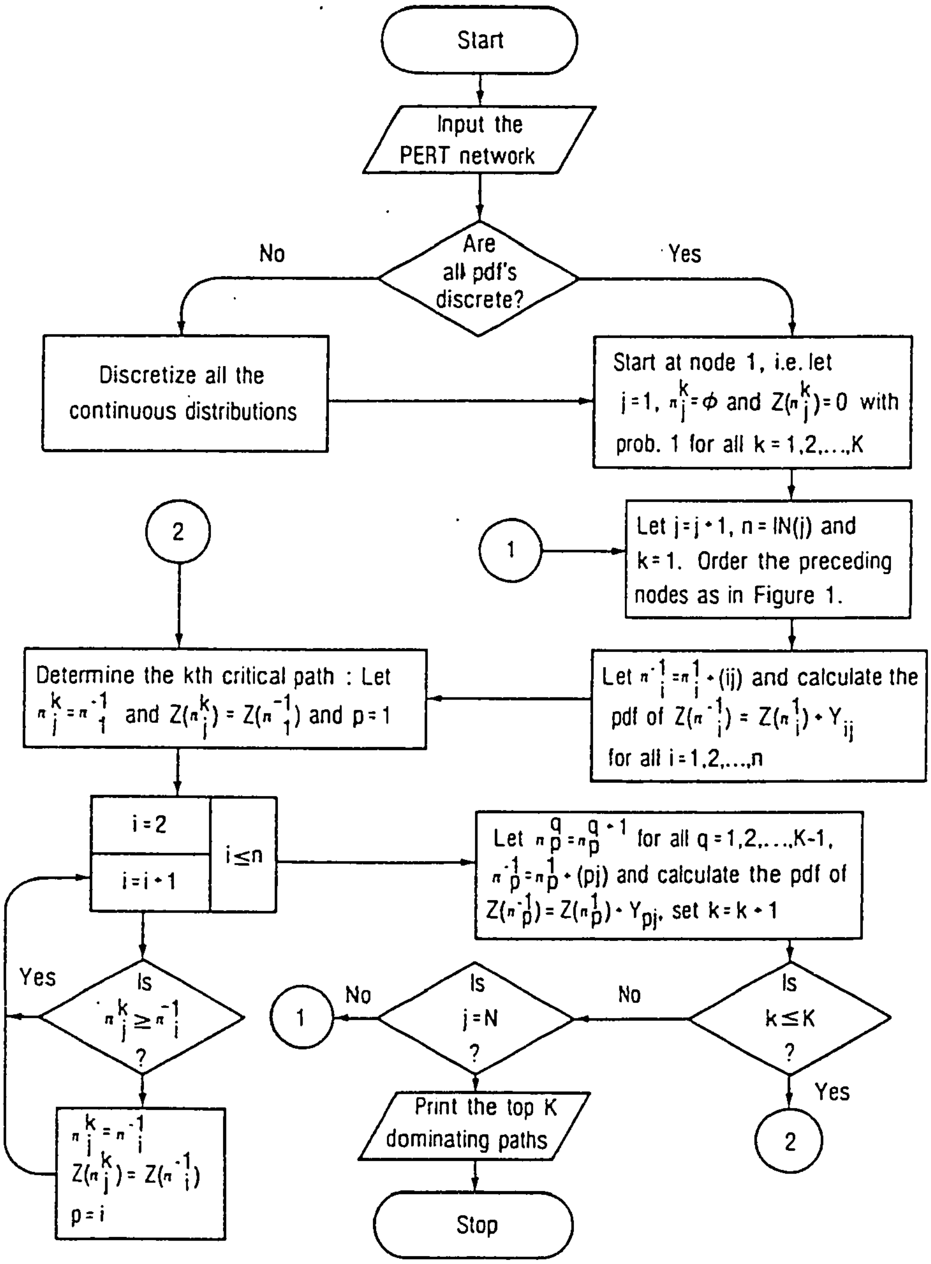

Start
Input the PERT network
Are all pdf's discrete?
No
Yes
Discretize all the continuous distributions
Start at node 1, i.e. let $j=1$, $\pi_j^k=\phi$ and $Z(\pi_j^k)=0$ with prob. 1 for all $k=1,2,\ldots,K$
1
Let $j=j+1$, $n=IN(j)$ and $k=1$. Order the preceding nodes as in Figure 1.
2
Let $\pi_i^{-1}=\pi_i^1+(ij)$ and calculate the pdf of $Z(\pi_i^{-1})=Z(\pi_i^1)+Y_{ij}$ for all $i=1,2,\ldots,n$
Determine the kth critical path : Let $\pi_j^k=\pi_1^{-1}$ and $Z(\pi_j^k)=Z(\pi_1^{-1})$ and $p=1$
i=2
i=i+1
i≤n
Let $\pi_p^q=\pi_p^{q+1}$ for all $q=1,2,\ldots,K-1$, $\pi_p^{-1}=\pi_p^1+(pj)$ and calculate the pdf of $Z(\pi_p^{-1})=Z(\pi_p^1)+Y_{pj}$, set $k=k+1$
Yes
Is $\pi_j^k\geq\pi_i^{-1}$?
$\pi_j^k=\pi_i^{-1}$
$Z(\pi_j^k)=Z(\pi_i^{-1})$
$p=i$
1
No
Is $j=N$?
No
Is $k\leq K$?
Yes
2
Print the top K dominating paths
Stop

Figure 24. A heuristic to identify the K most critical paths.
[Adapted from Dodin (1984)]

$$D \le \sum_{j=2}^{N} C[n+K-1] \le C\sum_{j=2}^{N}(K+j-2)$$

The value D is added to the arithmetic operations involved in the $K(n-1)$ comparisons used to identify the K most critical paths ending in node j, denoted by E. Each path π_j^k for $k = 1,2,\ldots,K$ is chosen from a set of n paths in Step 2b. Each comparison is also $O(NR)^2$, since it requires the evaluation of an expression similar to:

$$\mathrm{P}[Z(\pi_j^k) \ge Z(\pi_i^l)] \ge \mathrm{P}[Z(\pi_i^l) \ge Z(\pi_j^k)]$$

which is the definition of the stochastic dominance relation. Therefore,

$$E \le \sum_{j=2}^{N} CK(n-1) \le CK\sum_{j=2}^{N}(j-2)$$

and the total number of arithmetic operations of the heuristic is:

$$\begin{aligned} D+E &\le C\sum_{j=2}^{N}[(K+j-2)+K(j-2)] \\ &= CK\sum_{j=2}^{N}(j-1) + C\sum_{j=2}^{N}(j-2) \\ &= CKN(N-1)/2 + C(N^2-1)/2 \end{aligned}$$

which is $O(CKN^2)$ where C is $O(NR)^2$.

Stochastic Dominance

For any two paths π_1 and π_2 ending in node $i \le N$, it may be possible that π_1 is more critical than π_2. Hence, from the definition of CR:

$$\mathrm{P}[Z(\pi_1) \ge Z(\pi_q) \text{ for all } \pi_q \in P_i] \ge \mathrm{P}[Z(\pi_2) \ge Z(\pi_q) \text{ for all } \pi_q \in P_i]$$

The difficulty in evaluating this expression led to the introduction of the dominance in probability (stochastic dominance) relation:

$$\mathrm{P}[Z(\pi_1) \ge Z(\pi_2)] \ge \mathrm{P}[Z(\pi_2) \ge Z(\pi_1)]$$

An immediate consequence of the definition of stochastic dominance is this result:

Proposition 3. Let π_1 and π_2 be two paths ending in node i, so that $\pi_1 \geq \pi_2$. Suppose node i is an immediate predecessor of node j. Form the paths π_1' and π_2' by adding the arc a_{ij} to both paths π_1 and π_2, respectively. Then, $\pi_1' \geq \pi_2'$.

(Dodin, 1984)

If π_1, π_2, and π_3 are three paths ending in node i,, with durations $Z(\pi_1), Z(\pi_2)$, and $Z(\pi_3)$, respectively, and $\pi_1 \geq \pi_2$ and $\pi_2 \geq \pi_3$, then it is normally the case that $\pi_1 \geq \pi_3$, i.e., the dominance relation "≥" is transitive. However, there are counterinstances where the dominance relation is intransitive. Dodin (1984) stated that it can be shown that the dominance relation is transitive if the pdf's of the random variables $Z(\pi_l)$ for $l = 1,2$ and 3 belong to the same family of distributions. Further, if the node number i, is large, then the pdf of the random variable $Z(\pi_l)$ for $\pi_l \in P_i$, which is defined by $Z(\pi_l) = \sum_{a_{pq} \in \pi_l} T_{pq}$, can be approximated by a symmetric distribution, usually the normal. Tversky (1969) stated without proof that if the random variables $\{Z(\pi_l), \pi_l \in P_i\}$ are normally distributed, then the relation "≥" is transitive. Dodin (1984) proved the more general result that "≥" is transitive if the pdf's of the random variables $\{Z(\pi_l), \pi_l \in P_i\}$ are symmetric around their respective means.

The transitivity of the relation "≥" can be applied in stochastic networks to identify the first K critical paths. For high-numbered nodes, Dodin's (1984) transitivity results eliminate many paths by dominance in the heuristic procedure. In small-numbered nodes (2, 3, 4 and 5), the transitivity of the relation, which may not hold at these nodes, is not needed: if K is large, then the number of paths ending in these nodes is less than K, and all such paths can be searched and ranked; if, however, K is small and the pdf's of the paths ending in these nodes are neither symmetric nor belong to the same family of distributions, a search is conducted over all the paths ending in any of the nodes. Dodin

(1984) proved this result about the heuristic procedure:

> Theorem 5: For any node in a PERT network in which $2 \le j \le N$, the K paths identified by the heuristic procedure dominate in probability all other paths ending at node j.

From the above result, Dodin (1984) concluded that the question of identifying the highest K critical paths reduces to showing that:

$$\pi_l \ge \pi_k \text{ implies } \mathrm{CR}(\pi_l) \ge CR(\pi_k)$$

The two statements in the implication are positively correlated, as is apparent from their definitions: π_l is the *most dominating* path in a network if

$$\mathrm{P}[Z(\pi_l) \ge Z(\pi_q)] \ge \mathrm{P}[Z(\pi_q) \ge Z(\pi_l)] \qquad \text{for all } \pi_q \in P$$

and π_l is the *most critical* path if

$$\mathrm{P}[Z(\pi_l) \ge Z(\pi_q) \text{ for all } \pi_q \in P] \ge \mathrm{P}[Z(\pi_k) \ge Z(\pi_q) \text{ for all } \pi_k \in P] \qquad \text{for every } \pi_k \in P$$

If the implication were true, then the K paths identified by the heuristic would be the most critical paths, and the heuristic would be an exact procedure. Dodin (1984) demonstrated that, even though the implication is true in most cases, there are counterinstances when it does not hold.

Computational Experience

Dodin (1984) coded and compiled the heuristic in FORTRAN on a UNIVAC 1100/80. As his was the first attempt to solve the problem of identifying the K most critical paths in stochastic networks, it was not possible to compare its accuracy with the accuracy of other procedures. Consequently, he tested its accuracy and efficiency against Monte Carlo simulation.

A Monte Carlo simulation model was developed for stochastic networks whose activity duration distributions are one of seven pdf's (uniform, triangular, normal, exponential, gamma, beta, or discrete). The model assigns a random value (realization) to each arc in each network tested, generated from the activity duration pdf. Then, CPM is used to identify and record the critical path(s). These two steps are repeated for a predetermined number of replications; each time the frequencies of the critical path(s) are updated. Dodin (1984) observed that this Monte Carlo simulation process is possible, but impractical for large and dense networks for the following reasons:

> 1. The large number of paths in the network: The number of paths in an activity network with N nodes is bound[ed] from above by 2^{N-2}, which is the number of paths if the network is completely connected. Therefore, in a dense network with large N, identifying the paths alone constitutes a major task; and taking a large enough sample to obtain a fair estimate of the *CR*s requires an inordinate amount of time.
> 2. Recording the critical paths and their frequencies: Every MCS [Monte Carlo sampling] run (realization) identifies at least one critical path. Such a path has either been identified and recorded in previous realizations, or it is being identified for the first time and must be added to the list of critical paths. Deciding whether this path has been recorded earlier or not requires *comparing* it with all other paths in the list. This task is burdensome; to illustrate the difficulty of such a task, let us consider the realization 5001. Assume that the previous 5000 realizations have identified 1000 different critical paths; i.e., the list of critical paths has length 1000. To decide if the latest critical path has not been previously recorded, it must be compared with some or all of the critical paths on the list. If it is on the list, then its frequency will be increased by one; otherwise, it will be added to the list, which could yield an impractically large list.

These difficulties limit the use of Monte Carlo simulation in identifying the most critical paths in stochastic networks and enhance the need for the heuristic procedure. Because of these difficulties, Dodin (1984) limited the use of simulation to test the accuracy and efficiency of the heuristic to networks of sizes $N \leq 30$ and $A \leq 90$. The heuristic, however, can be applied to large, dense activity networks without encountering the difficulties of simulation.

To achieve a more convincing test of accuracy, each network tested was a "strongly random network," generated at random from the space of all connected, acyclic, directed graphs with the required numbers of nodes and activities. The network generator used was Dodin's (1993) modified version of the generator originally developed by Herroelen and Caestecker (1979). Random network generation is discussed in Section 2.7.2. Continuous activity duration distributions were discretized. Dodin (1984) employed the same hybrid discretization approach he used in testing sequential approximation (Dodin, 1980 and 1985a): he limited the use of the equal probabilities method to the exponential distribution; the equal distances method was used for the uniform, triangular, normal, gamma, and beta distributions. Table 2 [in Section 2.4.4 (above)] presents computational experience with this hybrid approach on a UNIVAC 1100/80. Table 10 presents the parameters of the pdf's used in Dodin's analysis of the accuracy of the heuristic.

Dodin (1984) randomly generated the 14 networks in Table 11 and applied the heuristic to identify the three most stochastically dominating paths. He also simulated the networks to identify the three most critical paths, since it is not possible to identify the exact critical paths for any stochastic networks. In each network, the pdf's characterized in Table 10 were used equally; for example, in the network with $N = 20$ and $A = 70$, the first ten arcs

Table 10. Parameters of the PDF's Used in the Accuracy Analysis.
[Adapted from Dodin (1984)]

Order	Distribution Type	Mean Value or 1st Parameter	Standard Deviation or 2nd Parameter	Minimum Realization (u)	Maximum Realization (v)
1	Uniform	5.0	—	0.00	10.00
2	Triangular	5.0	—	1.00	11.00
3	Normal	8.0	2.0	2.00	14.00
4	Exponential	2.0	—	0.00	15.00
5	Gamma	3.0	1.0	0.00	10.00
6	Beta	3.0	2.0	1.00	11.00
7	Discrete	{(2.0, 0.20),	(3.0, 0.30),	(4.0, 0.30),	(5.0, 0.20)}

Table 11. Performance Measures for Randomly Generated PERT networks. [Adapted from Dodin (1984)]

Problem No.	Size of Activity Network		CPU Time (in seconds)		Order of Paths		
	N	A	Procedure	MCS (sample size = 5000)	Path 1	Path 2	Path 3
1	10	15	0.92	21.55	1	1	1
2	10	20	1.94	30.43	1	1	1
3	10	30	7.80	42.55	1	0	0
4	10	40	11.60	56.25	1	1	0
5	20	30	9.47	46.20	1	1	1
6	20	40	9.92	60.10	1	1	1
7	20	50	15.61	69.60	1	0	0
8	20	60	26.81	85.88	1	1	1
9	20	70	31.55	97.90	1	0	1
10	20	80	39.84	149.38	1	1	0
11	30	45	16.70	70.13	1	1	1
12	30	60	29.26	89.68	1	0	1
13	30	75	33.96	106.35	0	1	0
14	30	90	43.65	155.75	1	1	0

had the uniform pdf, the second ten arcs had the triangular pdf,..., and the last ten arcs had the discrete pdf. The accuracy of the heuristic was measured by the closeness between the set of stochastically dominating paths, identified by the heuristic, and the set of corresponding critical paths, identified by Monte Carlo simulation; the efficiency of the heuristic, by comparison of the CPU times required.

Dodin (1984) reported that the variations in the measures of performance depended on the size and density of the test networks, the activity duration distributions, and the accuracy of discretizing the continuous distributions. Table 11 shows the impact of the size and density of networks. It compares the three stochastically dominating paths with the three most critical paths of Dodin's 14 test networks (problems). An entry of "1" in the last three columns of the table indicates that the K^{th} stochastically dominating path was identical to the corresponding critical path, whereas an entry of "0" indicates that the two paths were not identical, but differed by at least one arc. For example, in the third problem with $N = 10$ and $A = 30$, the first stochastically dominating path was identical to the first critical path, while the second stochastically dominating path was not identical the second path. Table 11 evidences that the first stochastically dominating path matched the first critical path in 13 of the 14 problems reported. Ten out of 14 of the second stochastically dominating paths matched the second critical paths, and eight of the third stochastically dominating paths matched the third critical paths. The first stochastically dominating path in problem no. 13 matched the third critical path, whereas the third stochastically dominating path matched the first critical path; i.e., in problem no. 13, the set of the three most stochastically dominating paths equaled the set of the first three critical paths; however, the order of the paths in both sets did not match. Problems nos. 3 and 7 were similar. In all the paths that did not match, the stochastically dominating path differed from the corresponding critical path in no more than four arcs. Most of the nonmatching paths occurred in networks with arc to node densities higher than two, which Dodin attributed to the large number of paths in dense networks with similar or close durations. In all the networks tested, Dodin reported that the three most critical (or stochastically dominating) paths had many arcs in common, i.e., many of the arcs of the most critical (or stochastically dominating) paths were constituents of the other critical (or stochastically dominating) paths.

Dodin (1984) reported that the CPU time for the heuristic depends on the network structure, arc to node density, and number of discretization points, NR. Column 4 of Table 11 shows the CPU time for the 14 test networks, where every continuous activity duration distribution was approximated by a discrete distribution with ten points (realizations). The table evidences that the CPU time depends on the network structure; this dependence becomes evident when one compares the CPU times required for networks with the same number of arcs, for example, problem nos. 4 and 6. The impact of network density on CPU time can also be observed from column 4 in the sets of networks with 10, 20, or 30 nodes: as the density increases, the CPU time requirements reach the worst case numbers in the discussion of the complexity of the heuristic [above]. For all the networks tested, the CPU time required by the heuristic on a UNIVAC 1100/80 was less than 45 seconds when the duration of every activity had no more than ten discretization points (realizations). The computational requirements of the heuristic were less than the CPU times required by the simulation model, which depends on network size, simulation sample size,. and types of activity duration distributions used. Column 5 of Table 11 shows the simulation times for a simulation sample size of 5,000.

In the development of the complexity of the heuristic [above], Dodin (1984) showed the number of discretization points, *NR*, affected the complexity of both the convolution operation and the dominance relation. Consequently, it affects the CPU time requirements. Table 12 evidences the impact of *NR* on the CPU time of the heuristic: CPU time increases as *NR* increases; but, the accuracy of discretization also increases as *NR* increases. As is the case with most approximations, there is a trade-off between accuracy and cost of execution of the heuristic. The problems in Table 12 were similar: when $NR = 20$, the first three stochastically dominating paths matched the first three critical paths in the first three problems, while in the fourth problem the match was the same as for problem no. 4 in Table 11. When $NR = 5$, the match was less accurate than when $NR = 10$, which is shown in the first four problems of Table 11.

Table 12. Impact of Number of Discretization Points on CPU Time of the Procedure. [Adapted from Dodin (1984)]

Problem No.	Activity Network Size		CPU Time (in seconds)		
	N	*A*	NR = 5	NR = 10	NR = 20
1	10	15	0.09	0.92	3.07
2	10	20	0.31	1.94	6.48
3	10	30	0.469	7.80	25.50
4	10	40	0.968	11.60	40.35

Dodin (1984) demonstrated that the heuristic procedure can be used to identify the most critical activities in stochastic project management networks. First, the heuristic can be used to identify the most critical path. Second, in networks with low densities, the heuristic can be used to identify the second, third,..., and K^{th} critical paths, whereas in dense networks it can be used to approximate the set of the K most critical paths, i.e., without consideration of the path's rank in the set. As implemented to date, the heuristic can be applied to any stochastic network by discretizing all continuous activity duration distributions. When discretization is used, the accuracy of the heuristic is increased by minimizing the errors of discretization, and the CPU time requirements depend on the number of discretization points (realizations) of the activity duration distributions. The best reported performance is less than 45 seconds CPU time for an activity network of size $N \leq 30$ and $A \leq 90$ with no more than ten discretization points for any activity duration.

2.9 Computer Applications

2.9.1 Recent Trends

Increasing competition, both in the US and overseas, conspired with the global economic downturn in the early 1990's to squeeze all sectors of the US economy. While the short-term impact of the fiscal crunch was detrimental to the nation's industrial base, the long-term ramifications may prove to be, quite unexpectedly, quite positive. American industry, forced to streamline processes, slash development cycles, improve cost accounting, and enhance quality, is emerging from the recession as a lean, quality-driven operation. This period of cost-consciousness coincided with a change in the information technology environment which significantly impacted the availability of sophisticated project management cost and control tools. Advances in hardware technology and improvements in software user-friendliness have lowered the entry barriers to the use of these tools. (Rogers, 1993)

Downsized Solutions

While organizations have historically been forced to choose between huge, expensive mainframe and minicomputer-based project management systems that offered the sophistication and functionality to maintain tight control over large projects, and PC-based systems that offered very limited control at a fraction of the cost, 1990's buyers are free from this tradeoff. Today's leading edge applications combine the functionality of graphical user interfaces (GUIs) with the high-end processing capabilities which are typically found in mainframe or minicomputer environments. Organizations can now acquire sophisticated project management capability at a fraction of the cost of "yesterday's 'big iron" systems" (Rogers, 1993). This technological advance has significantly lowered the threshold to implementation of project management tools. Organizations that had considered implementing project management tools, but had been scared away by the size of the financial investment, or those which recognized their importance but opted for less expensive and less powerful systems, are now moving into the new generation of powerful, downsized applications. While in the past many corporations only implemented project management on expensive or "critical" tasks, today there are few corporations which have not become cognizant of the impact to the bottom line of every operation they undertake. The majority are embracing the newly affordable tools and bringing an increasing number of their operations within project management control.

As the price of high-level tools has dropped, they have also become more user-friendly and, consequently, more practical to install at all levels of an organization, even down to the shop floor. GUIs, once a feature of low-level project management tools, now provide intuitive access to much more sophisticated tools. Changing the duration of a project component, once a task that required database entry and adjustment, can now be achieved with a mouse by "pointing and clicking" or "dragging and shrinking;" the impact on total cost is calculated, and required resource-leveling is performed instantly. GUIs are making traditional project managers increasingly more productive and are bringing a new population of users into the discipline.

Enterprise-wide Deployment

The client/server processing architecture, a newer and more sophisticated technology than the GUIs, is increasing in maturity. The enterprise-wide access that it provides has combined with the graphical revolution to widen the domain of project management applications. Client/server was developed in response to the market's demand for a downsized architecture that could harness the capabilities of all sizes of processors - PCs, minicomputers, and mainframes. The architecture exploits the efficiencies of division-of-labor to facilitate enhanced processing throughput. While GUIs are making project management understandable to project rank and file, client/server is allowing the deployment of project management systems to every terminal throughout an organization. Together, the two advances are making "the project management system accessible to the user and the user accessible to the project management system" (Rogers, 1993).

Real-time Processing

As PCs have been introduced, project management has benefited from their real-time processing capabilities, the third element in the GUI, client/server, technological development triad. Budgets can be adjusted, new variables introduced, and activity durations altered, and the effects of these changes can be calculated and displayed immediately. The reliance on mainframe or minicomputer batch-processing-based systems to perform complex, "what-if" scenario modeling and resource-leveling is a thing of the past, and project managers need no longer suffer batch-processing delays of hours and, sometimes, days. In the time-sensitive discipline of project management, such delays aren't acceptable any longer: by the time the full ramifications of a change have been recognized, it may be too late to adjust activity schedules/resource allocations to prevent cost/schedule overruns.

2.9.2 Software Sophistication

While project management has profited from these recent technological advances in hardware performance, processing architectures, and software user-friendliness, and the capabilities available particularly to low-end users have improved considerably, the sophistication of project management software has not kept pace. Today there are numerous project management software packages available from commercial software developers, applications software developers, and management consultants. With the exception of the upper-end, network-based simulators, such as SLAM and SIMAN, these software packages offer only the "traditional" project management analysis tools - CPM, buckets, Gantt, PERT, WBS, OBS, outline, Pareto, constrained resource management/ resource leveling, et al., albeit with highly sophisticated graphical interfaces. Few offer the capability for even basic output analysis. In an industry-wide survey of commercially available project management software packages excluding simulators, Gido (1985) reported 127 sources. Only 27% of these programs make statistical computations, the majority of which are the probabilities of achieving scheduled times for milestone events and network completion, computed with PERT: for each activity, the user must input three duration estimates (optimistic, most likely, and pessimistic), and all activity times are assumed to follow a single probability distribution (Beta). An additional 11% of these programs permit the user to input the three activity duration estimates required by PERT, but surprisingly do not make any probability calculations. None of these programs is capable of characterizing a network's throughput distribution. Recent efforts have focused on packaging basic project management network tools for PC users in the research and small business sectors. Smith (1992) developed a microcomputer-based program in Turbo Pascal which generates an activity-on-node network model in graphical form for a small project with fewer than 50 activities, but it is not capable of reducing the network or performing any other network analysis. Most recently, Duffy (1993) reported on the 15 leading vendors of project management software for windows environments supported on computer network operating systems. Only one offers a statistical analysis capability (probability profiles) more sophisticated than histograms.

2.10 Summary

The traditional methods for the reduction of project management networks are CPM for deterministic networks and PERT for stochastic networks. Attempts to develop exact analytic solutions of stochastic networks proved unsuccessful and were abandoned in the late 1970's in favor of simulation and an emerging interest in approximation solutions. Simulation is effective but may require lengthy run times to insure steady states have been achieved. Stochastic network reduction involves the repeated application of series-parallel reduction operations until a network is either completely reduced to a single, equivalent arc from source node to sink node or to an irreducible core consisting of one or more interdictive graphs. Three approximation reduction methods have been put forward for the reduction of irreducible networks. Under the "independent multiple arcs" (dual arcs) method, duplicate arcs are inserted into an irreducible network to achieve separability; the separable network is then reduced with series-parallel reduction operations. The theory has been developed, but the method has not been implemented. The sequential approximation method steps through the network nodes in order, building the throughput distribution as it goes. Implementation to date has required that continuous activity duration distributions be discretized. To achieve acceptable accuracies of network reduction, the number of discretization points needed may be large, which imposes high computer run-time and storage requirements. The ordered recursive method requires a network arc for each discrete state value of every activity duration distribution, and is correspondingly run-time limited. It has practical utility only for small networks. To evaluate the performance of network reduction techniques, a design of experiments approach should be used; the numbers of nodes and activities, the network structure, and the types and parameters of activity duration distributions should be randomly generated for all test networks. The reduction of large networks may be impractical because of high computer run-time and storage requirements, and a large network decomposition method has not yet been developed. The most critical paths and activities cannot be determined without extensive simulation; however, stochastic dominance can be used to approximate the K most critical paths. Improvements in computer hardware technology, GUIs, and client/server processing architectures have brought high-end user project management capabilities to low-end users, but software packages continue to lack sophistication in output analysis. Other than simulation, there is no stochastic project management network reduction capability available on PCs.

CHAPTER 3

METHODOLOGY

3.1 Characteristics of Analytic Reduction

The focus of this research is the development of a new, novel, more capable, and more efficient numerical approximation method for stochastic activity/project management network reduction. The shortcomings of the first attempt at an exact solution to this problem provide an excellent point of departure for methodology construction.

Martin (1965) developed the concept of a series-parallel algorithm for the exact analytic solution of acyclic, directed networks (Sections 2.4 and 2.5). In theory, the series reduction (convolution) and parallel reduction (maximum) operations would be accomplished analytically, via direct integration and differentiation, so in final form the expression of the throughput distribution would be exact. In order that all the direct integration and differentiation could be carried out on a digital computer, he restricted the concept development to networks, all of whose activity duration distributions had piecewise-defined polynomial densities. Because of the limitations on speed and storage of digital computers of the time, he further restricted the actual computer implementation to separable networks, all of whose activity duration distributions were uniform. He demonstrated the concept by successfully obtaining the exact pdf of throughput time for a simple network with nine nodes and eight arcs and all-uniformly distributed activity times.

As encouraging as Martin's results were, three significant shortcomings prevented his approach from being developed further:

> First, the algorithm was restricted to networks, all of whose activity duration distributions had piecewise-defined polynomial densities. Continuous activity duration distributions could not accommodated, because there was no way to program a digital computer to carry out the direct

integration and differentiation required to accomplish series-parallel reduction operations.

The second shortcoming was the problem of "exploding coefficient storage." All the activity duration distributions had to have piecewise-defined polynomial density functions. If two such distributions, whose piecewise-defined polynomial densities have terms of as-high-a degree as m and n, respectively, are series-reduced, the convolution also has a piecewise-defined polynomial density which can have terms of as-high-a degree as $m + n + 1$. Similarly, if two such distributions, whose piecewise-defined polynomial densities have terms of as-high-a degree as m and n, respectively, are parallel-reduced, the maximum also has a piecewise-defined polynomial density which can have terms of as-high-a degree as $m + n + 1$. (Each cdf can have terms of as high a degree as $m + 1$ and $n + 1$, respectively; the product of the cdf's can have terms of as high a degree as $m + n + 2$; so, the pdf of the maximum, which is the derivative of the product of the cdf's, can have terms of as high a degree as $m + n + 1$.) Martin's simple nine-node, eight-arc example network with all-uniformly distributed activity times had a piecewise-defined polynomial throughput time density with terms of as-high-as the fifth degree; for that particular network structure, the throughput time density could have had terms of as-high-as the ninth degree, had the uniform activity time distributions been defined differently. As soon as it is created, the coefficient of each term in a piecewise-defined polynomial density must be held in a computer storage location because it may potentially be involved in later series-parallel reductions. When larger networks with piecewise-defined polynomial activity duration densities with terms of higher degrees are considered, the computer storage requirements for coefficients quickly explode.

The third shortcoming was the related problem of "proliferating classes." Again, all the activity duration distributions had to have piecewise-defined polynomial density functions. Each such density is piecewise-defined on a partition of some number of classes (cells), c, over its domain of definition; the partition consists of $c+1$ points spread across the domain. If two such densities, piecewise-defined on c_1 and c_2 classes, are series-reduced, the convolution will be piecewise-defined on as many as $(c_1 + 1)(c_2 + 1) - 1 = c_1c_2 + c_1 + c_2$ classes. Similarly, if two such densities, piecewise-defined on c_1 and c_2 classes, are parallel-reduced, the maximum will be piecewise-defined on as many as $c_1 + c_2 + 1$ classes. As soon as it is created, the coefficient of each term of each polynomial piecewise-defined on each of these classes must be held in a computer storage location because it may potentially be involved in later series-parallel reductions. When larger networks with piecewise-defined polynomial activity duration densities defined on even moderate numbers of classes are considered, the computer storage requirements for coefficients again quickly explode.

The second and third shortcomings are overcome if the activity duration densities are maintained as polynomials of a fixed degree, piecewise-defined on a fixed number of classes, since coefficient storage requirements are now controlled. This is accomplished as follows. Whenever an intermediate or final product is formed by a series-parallel reduction operation, the pdf of that product is immediately transformed into polynomials of a fixed degree, piecewise-defined on a fixed number of classes over its domain. If, prior to network reduction, continuous activity duration distributions are similarly transformed into polynomial densities of the same fixed degree, piecewise-defined on the same fixed number of classes over their domains, then the first shortcoming is also overcome.

3.2 Polygonal Approximation

Numerical approximation and reduction techniques for stochastic project management networks are proposed based on the following concept of operations:

> Continuous activity duration distributions are initially approximated by polynomial densities of some predetermined degree which are piecewise-defined on partitions of some predetermined number of classes over their domains. The network is then reduced with one of the operative stochastic network reduction methods: either an implementation of "independent multiple arcs" (dual arcs), or sequential approximation. Whenever a series-parallel reduction operation is required, it is carried out directly on the piecewise-defined polynomial densities which currently approximate the two involved distributions. The intermediate product of the series-parallel reduction operation is then immediately approximated and replaced by a polynomial density of the predetermined degree which is piecewise-defined on a partition of the predetermined number of classes over its domain. When the network has been completely reduced to a single equivalent activity from the source node to the sink node, the throughput distribution has been approximated by a piecewise-defined polynomial density, which can be used directly in post-reduction analysis.

The numerical approximation method for continuous activity duration distributions will be completely described when the degree of the piecewise-defined approximating polynomials, the method of fitting the polynomials, and the number of classes in the partitions of the domains are specified. Approaches developed in numerical analysis for solving similar approximation problems, in particular numerical differentiation and numerical integration (quadrature), give insight into how best to make these specifications.

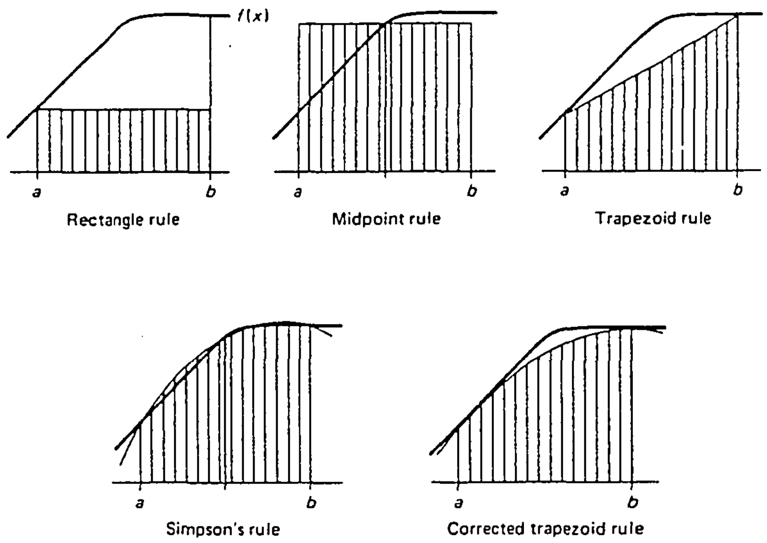


Figure 25. Numerical integration rules based on interpolating polynomials. [Adapted from Conte and deBoor (1972)]

3.2.1 Linear Approximation

When a sufficiently continuous function cannot be easily differentiated or integrated in closed form over a domain of values, the derivative or integral can be approximated numerically. A partition is erected on the domain, and a set of piecewise-defined interpolating polynomials is constructed such that one of the interpolating polynomials approximates the function in each class (cell) of the partition. The interpolating polynomials are differentiated or integrated within their respective classes and the results compiled across the partition. Figure 25 shows zero-, first-, and second-degree interpolating polynomials over a single class for the numerical integration (quadrature) problem. The higher the degree of the interpolating polynomials, the more accurate the numerically approximated derivative or integral, but the more computationally intensive the construction and subsequent differentiation or integration of the polynomials. Since the accuracy of the approximated derivative or integral can also be increased by refining the partition (increasing the number of classes by adding additional interior points), the best strategy is to approximate the function with piecewise-defined interpolating polynomials of "low" degree, and then refine the partition until the approximated derivative or integral has a desired accuracy (Conte and deBoor, 1972).

Now suppose that in a stochastic network reduction, all distributions are approximated by polynomials which are piecewise-defined on partitions of c classes over their domains; further suppose that two distributions are to be convolved by a series-reduction operation, and the intermediate product of the convolution is to be approximated and replaced by polynomials which are also piecewise-defined on a partition of c classes over its domain. The intermediate product of the convolution will be piecewise-defined on a partition over its domain which may have up to $c^2 + 2c$ classes. When the intermediate product is approximated and replaced by a new set of polynomials which are piecewise-defined on a new partition of only c classes over its domain, each of the c classes in the new partition will include all or parts of some of the up-to-$(c^2 + 2c)$ classes of the partition of the intermediate product. If the number of classes in the "old" partition of the intermediate product is "large" (its maximum is $c^2 + 2c$) relative to number of classes in the "new" partition, c, some of the "new" partition classes must necessarily include (all or some of) a considerable number of "old" partition classes. The density of the intermediate product is approximated by a different piecewise-defined polynomial in each class of the "old" partition. Consequently, when that density is approximated and replaced by a polynomial which is piecewise-defined over a class of the "new" partition which overlaps several classes of the "old" partition, the different behavior of the density in each of the overlapped classes of the "old" partition must be accounted for. If an interpolating polynomial were used, then it would have to be interpolated through at least one point in each of the overlapped classes of the "old" partition. If the polynomial were interpolated through k points, it would be of degree $k - 1$. If $k - 1$ is moderate or large (but not small), then this would violate the numerical approximation strategy of interpolating with polynomials of "low" degree.

Since the polynomials which are piecewise-defined on the classes of the "old" partition constitute an approximation of the density of the convolution, numerical values taken on by these polynomials, such as at possible points of interpolation, differ from the true values of the density at those points, although the actual errors aren't known. So, fitting a set of "new" polynomials, which are piecewise-defined on the classes of the "new" partition, is better viewed as a data fitting problem, rather than an interpolation problem, where the data to be "best fit" are the numerical values taken on by the "old" polynomials at selected points in each of the classes of the "new" partition. In the absence of a convincing argument to select a different metric, it is customary to use the least-squares error to measure the accuracy of data fitting. When the least-squares error is minimized, the fitting method is the well-known least-squares approximation or least-squares regression. (Conte and deBoor, 1972)

Now that the method of fitting the piecewise-defined approximating polynomials has been identified as least-squares regression, the degree of the polynomials and the number of classes in the partitions of the domains remain to be specified. If the approximating polynomials are zero or first degree, the fitting method is simple linear regression (SLR); if second or higher degree, multiple linear regression (MLR). As with interpolating polynomials, the higher the degree of regression-fit polynomials, the better approximators they are, but the more computationally intensive their construction is. Since the accuracy of approximation can be increased by both refining the partition and increasing the number of fitting points within each class of the partition, the best strategy is, again, to approximate with regression-fit polynomials of "low" degree and then refine the partition and/or increase the number of fitting points within each class of the partition. The continuous, non-uniform activity duration distributions most commonly encountered in

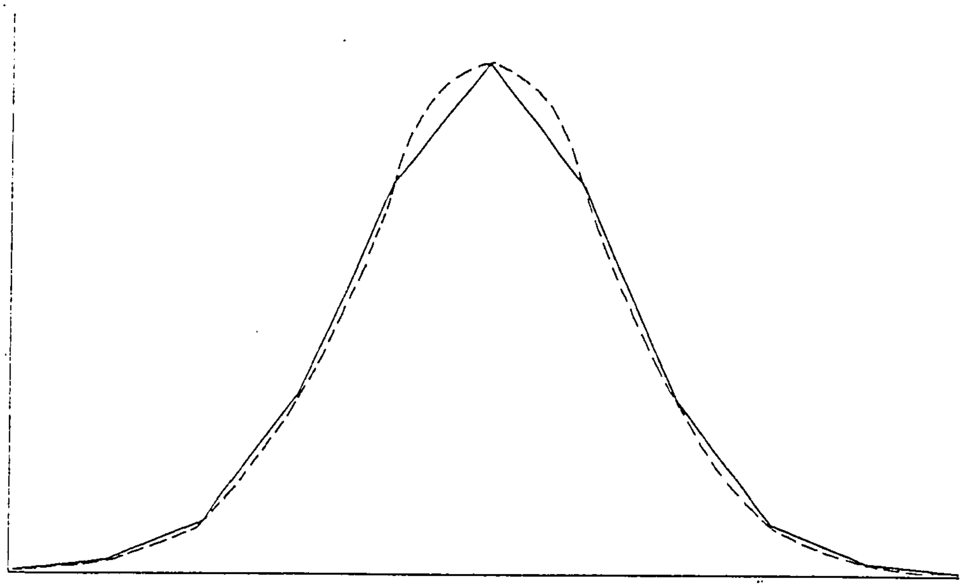


Figure 26. Polygonal (linear polynomial) approximation of a curvilinear function. [Adapted from Fergeson and Shortell (1978)]

stochastic project management networks (triangular, normal, exponential, gamma, and beta) have sufficient curvature that at least first-degree polynomials should be regression-fit to them. Since increasing the degree of the approximating polynomials above first degree would result in increasing the computational requirements of the regression from SLR to MLR, the piecewise-defined, regression-fit approximating polynomials are specified to be of the first degree (linear). When a curvilinear function is approximated by piecewise-defined first-degree polynomials, the approximation is *polygonal.* Polygonal approximation of a curvilinear function (suggestive of a normal pdf) is shown in Figure 26.

Fergeson and Shortell (1978) studied the polygonal approximation of pdf's and conducted a large number of trial and error tests involving the number of classes in the partition, the number of regression fitting points in each class of the partition, the computer run time required, and the accuracy of the piecewise-defined polynomial approximations.

They found the best results are achieved when the number of classes in the partition is ten, the regression fitting points are positioned uniformly across each class, and the number of fitting points is determined by:

$$n = 10 + \text{integer}(\text{class width} * 3)$$

In each class, the approximating polynomial is a line segment of the form:

$$b_0 + b_1 t$$

where the regression coefficients are computed from the standard SLR formulas[1]:

$$b_1 = \frac{\sum t_i y_i - \frac{\sum t_i \sum y_i}{n}}{\sum t_i^2 - \frac{(\sum t_i)^2}{n}}$$

$$b_0 = \frac{\sum y_i}{n} - b_1\left(\frac{\sum t_i}{n}\right)$$

$$y_i = f(t_i), \text{ the pdf evaluated at } t_i$$

To control error build-up from the polygonal approximation, after all ten sets of regression coefficients have been computed, the coefficients are normalized so that the probability under the approximated pdf is one:

$$\text{correction factor } (CF) = 1.0 \,/\, \text{computed probability under approximation}$$

$$b_0 = b_0 * CF \quad \text{and} \quad b_1 = b_1 * CF$$

The proposed numerical approximation and reduction techniques now consist of polygonal (linear polynomial) approximation coupled with one of the operative stochastic network reduction methods. Each technique is referred to as a Polygonal Approximation and Reduction Technique (PART).

[1] Discussions of SLR can be found throughout the probability and statistics literature. For example, Neter and Wasserman (1972) present a thorough treatment.

3.2.2 Series Reduction

In accordance with the PART concept of operations, at all times during the network reduction process, regardless of which of the operative stochastic network reduction methods is being executed, the distributions of all activity durations are maintained as piecewise-defined polygonal approximations on partitions of ten classes over their domains. When two activity duration distributions are convolved through a series-reduction operation, an intermediate product is temporarily formed. This intermediate product is a polynomial of up-to the third degree which is piecewise-defined on a partition of up-to 120 classes. This section discusses the formation of series-reduction products.

Suppose that activities A_1 and A_2 are to be convolved through a series-reduction operation, where the duration distribution of activity A_1 is currently approximated by the polygonal approximation f which is piecewise-defined on a ten-class partition with class boundaries $f_{\text{lower bound}} = bf_1, bf_2, \ldots, bf_{11} = f_{\text{upper bound}}$ over its domain $[f_{\text{lower bound}}, f_{\text{upper bound}}]$, and the duration distribution of activity A_2 is similarly currently approximated by the polygonal approximation g which is piecewise-defined on a ten-class partition with class boundaries $g_{\text{lower bound}} = bg_1, bg_2, \ldots, bg_{11} = g_{\text{upper bound}}$ over its domain $[g_{\text{lower bound}}, g_{\text{upper bound}}]$. The convolution $f \oplus g$ is piecewise-defined on the set of classes whose boundary points are the values $bf_i + bg_j$, where $i,j = 1,2,\ldots,11$, i.e., all the possible combinations of the class boundaries of f and g. The $bf_i + bg_j$'s are sorted and ranked in ascending order; duplicate values are ignored. The may be as few as 21 distinct combinations, and hence 20 classes in the convolution - the case when f and g are defined over the same domain - or as many as 121 distinct combinations, and hence 120 classes in the convolution. The domain of the convolution is:

$$[\min_{i,j}(bf_i + bg_j), \max_{i,j}(bf_i + bg_j)]$$

The general form of the convolution is:

$$(f \oplus g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

Over its i^{th} class, f is defined by:

$$f_i(t) = f_0^i + f_1^i t$$

and over its j^{th} class, g is defined by:

$$g_j(t) = g_0^j + g_1^j t$$

Over its k^{th} class, the convolution is:

$$(f \oplus g)_k = \sum_{\substack{\text{valid } i^{\text{th}} \text{ class of } f \\ \text{and } j^{\text{th}} \text{ class of } g}} \int_{\text{lower limit}}^{\text{upper limit}} (f_0^i + f_1^i \tau)[g_0^j + g_1^j (t - \tau)] d\tau \qquad (1)$$

For the k^{th} class of the convolution, the i^{th} class of f and the j^{th} class of g are valid if either Part (a) or Part (b) of Figure 27 applies, or both.

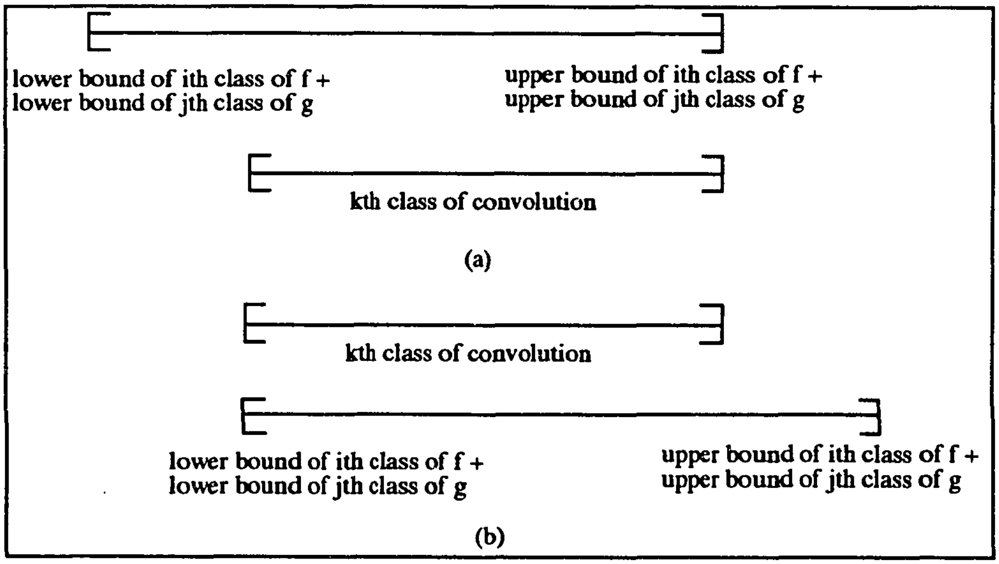

lower bound of ith class of f +
lower bound of jth class of g
upper bound of ith class of f +
upper bound of jth class of g
kth class of convolution
(a)
kth class of convolution
lower bound of ith class of f +
lower bound of jth class of g
upper bound of ith class of f +
upper bound of jth class of g
(b)

Figure 27. Valid classes for convolution.

The following test is recursively performed over all possible combinations of the i^{th} class of f and the j^{th} class of g to identify the valid class combinations for the k^{th} class of the convolution:

$$\begin{bmatrix}\text{lower bound}\\ \text{of } k^{\text{th}} \text{ class of}\\ \text{convolution}\end{bmatrix} \geq \begin{bmatrix}\text{lower bound of } i^{\text{th}} \text{ class}\\ \text{of } f + \text{lower bound of}\\ j^{\text{th}} \text{ class of } g\end{bmatrix}$$

and

$$\begin{bmatrix}\text{upper bound}\\ \text{of } k^{\text{th}} \text{ class of}\\ \text{convolution}\end{bmatrix} \leq \begin{bmatrix}\text{upper bound of } i^{\text{th}} \text{ class}\\ \text{of } f + \text{upper bound of}\\ j^{\text{th}} \text{ class of } g\end{bmatrix}$$

If both conditions hold, then the i^{th} class of f, $[bf_i, bf_{i+1}]$, and the j^{th} class of g, $[bg_j, bg_{j+1}]$, are a valid combination of classes contributing to the k^{th} class of the convolution, $[b_k, b_{k+1}]$. Then:

$$bf_i \leq \tau \leq bf_{i+1} \quad \text{and} \quad bg_j \leq t - \tau \leq bg_{j+1}$$

or:

$$bf_i \leq \tau \leq bf_{i+1} \quad \text{and} \quad t - bg_{j+1} \leq \tau \leq t - bg_j \tag{2}$$

From Equation (2):

$$\max(bf_i, t - bg_{j+1}) \leq \tau \leq \min(bf_{i+1}, t - bg_j) \tag{3}$$

From Equation (3), the limits of integration in Equation (1) are:

$$\begin{aligned}\text{lower limit} &= \max(bf_i, t - bg_{j+1})\\ \text{upper limit} &= \min(bf_{i+1}, t - bg_j)\end{aligned} \tag{4}$$

The conditions of Equation (4) are applied as follows:

$$\begin{aligned}&\text{If } [bf_i \geq (b_{k+1} - bg_{j+1})], \text{ then lower limit } = bf_i;\\ &\quad \text{otherwise, lower limit } = t - bg_{j+1}.\\ &\text{If } [bf_{i+1} \leq (b_k - bg_j)], \text{ then upper limit } = bf_{i+1};\\ &\quad \text{otherwise, upper limit } = t - bg_j.\end{aligned} \tag{5}$$

Then the contribution of the i^{th} class of f and the j^{th} class of g to Equation (1) is:

$$\begin{aligned}&\int_{\text{lower limit}}^{\text{upper limit}}(f_0^i + f_1^i\tau)[g_0^j + g_1^j(t-\tau)]d\tau \\ &= \Big[[f_0^i g_0^j\tau + \tfrac{1}{2}(f_1^i g_0^j - f_0^i g_1^j)\tau^2 - \tfrac{1}{3}f_1^i g_1^j\tau^3] + (f_0^i g_1^j\tau + \tfrac{1}{2}f_1^i g_1^j\tau^2)t\Big]_{\text{lower limit}}^{\text{upper limit}}\end{aligned} \tag{6}$$

If, from Equation (5), a limit of integration in Equation (6) is a constant, $c = bf_i$ or bf_{i+1}, the evaluation at that limit is:

$$[f_0^i g_0^j c + \tfrac{1}{2}(f_1^i g_0^j - f_0^i g_1^j)c^2 - \tfrac{1}{3}f_1^i g_1^j c^3] + (f_0^i g_1^j c + \tfrac{1}{2}f_1^i g_1^j c^2)t - \tfrac{1}{2}f_0^i g_1^j t^2$$

If a limit is of the form $t - c = t - bg_j$ or t - bg_{j+1}, the evaluation at that limit is:

$$\begin{aligned}&[-f_0^i g_0^j c + \tfrac{1}{2}(f_1^i g_0^j - f_0^i g_1^j)c^2 + \tfrac{1}{3}f_1^i g_1^j c^3] + \\ &(f_0^i g_0^j - f_1^i g_0^j c - \tfrac{1}{2}f_1^i g_1^j c^2)t + \tfrac{1}{2}f_1^i g_0^j t^2 + \tfrac{1}{6}f_1^i g_1^j t^3\end{aligned}$$

The contributions from Equation (6) are combined for all valid combinations of classes of f and g to form a piecewise-defined polynomial in the k^{th} class of the convolution. Once the convolution has been formed as this intermediate product, it is immediately replaced by a piecewise-defined polygonal approximation on a partition of ten classes over its domain.

3.2.3 Parallel Reduction

Similarly, when two activity duration distributions are reduced through a parallel-reduction operation, two intermediate products are temporarily formed. These intermediate products are polynomials of up-to the fourth and third degrees, respectively, which are piecewise-defined on a partition of up-to 21 classes. This section discusses the formation of the parallel-reduction products.

Suppose that activities A_1 and A_2 are to be reduced through a parallel-reduction operation, where the duration distribution of activity A_1 is currently approximated by the polygonal approximation f which is piecewise-defined on a ten-class partition with class boundaries $f_{\text{lower bound}} = bf_1, bf_2, \ldots, bf_{11} = f_{\text{upper bound}}$ over its domain $[f_{\text{lower bound}}, f_{\text{upper bound}}]$, and the duration distribution of activity A_2 is similarly currently approximated by the polygonal approximation g which is piecewise-defined on a ten-class partition with class boundaries $g_{\text{lower bound}} = bg_1, bg_2, \ldots, bg_{11} = g_{\text{upper bound}}$ over its domain $[g_{\text{lower bound}}, g_{\text{upper bound}}]$. The distribution of the maximum of f and g is piecewise-defined on the set of classes whose boundary points are the values $\{bf_i\} \cup \{bg_j\}$, where $i,j = 1,2,\ldots,11$, i.e., all the possible class boundaries of f and g. The bf_i's and bg_j's are sorted and ranked in ascending order; duplicate values are ignored. Since the probability is zero that the distribution of the maximum takes on any values less than both $f_{\text{lower bound}}$ and $g_{\text{lower bound}}$, the lower bound of the domain of the maximum is:

$$[\max(f,g)]_{\text{lower bound}} = \max(f_{\text{lower bound}}, g_{\text{lower bound}})$$

All boundary values lower than this lower bound are discarded. The may be as few as 11 distinct boundary points, and hence 10 classes in the parallel-reduction - the case when f and g are defined over the same domain, for example - or as many as 22 distinct boundary points, and hence 21 classes in the parallel-reduction. The general form of the cumulative distribution function (cdf) of the maximum is:

$$F_{\max(f,g)}(t) = F(T) \cdot G(t)$$

where F is the cdf of f and G is the cdf of g. Over its i^{th} class, f is defined by:

$$f_i(t) = f_0^i + f_1^i t$$

and over its j^{th} class, g is defined by:

$$g_j(t) = g_0^j + g_1^j t$$

For the k^{th} class of the maximum, the i^{th} class of f and the j^{th} class of g are valid if:

$$bf_i \le b_k \quad \text{and} \quad b_{k+1} \le bf_{i+1} \qquad \text{and} \qquad bg_j \le b_k \quad \text{and} \quad b_{k+1} \le bg_{j+1}$$

If all four conditions hold, then the i^{th} class of f, $[bf_i, bf_{i+1}]$, and the j^{th} class of g, $[bg_j, bg_{j+1}]$, are valid classes contributing to the k^{th} class of the maximum, $[b_k, b_{k+1}]$. Then the cdf of maximum in the k^{th} class is:

$$
\begin{aligned}
[F_{\max(f,g)}(t)]_k &= \mathrm{P}(\tau \le t \mid t \in [b_k, b_{k+1}]) \\
&= [\mathrm{P}(0 \le \tau_f \le b_k) + \mathrm{P}(b_k \le \tau_f \le t)] \cdot [\mathrm{P}(0 \le \tau_g \le b_k) + \mathrm{P}(b_k \le \tau_g \le t)] \\
&= [F(bf_i) + \int_{bf_i}^{t} f(\tau)d\tau] \cdot [G(bg_j) + \int_{bg_j}^{t} g(\tau)d\tau] \\
&= [F(bf_i) + \int_{bf_i}^{t} (f_0^i + f_1^i\tau)d\tau] \cdot [G(bg_j) + \int_{bg_j}^{t} (g_0^j + g_1^j\tau)d\tau] \\
&= [F(bf_i) + f_0^i(t - bf_i) + \tfrac{1}{2} f_1^i(t^2 - bf_i^{\,2})] \cdot [G(bg_j) + g_0^j(t - bg_j) + \tfrac{1}{2} g_1^j(t^2 - bg_j^{\,2})] \\
&= [(F(bf_i) - f_0^i bf_i - \tfrac{1}{2} f_1^i bf_i^2) + f_0^i t + \tfrac{1}{2} f_1^i t^2] \cdot \\
&\qquad [(G(bg_j) - g_0^j bg_j - \tfrac{1}{2} g_1^j bg_j^2) + g_0^j t + \tfrac{1}{2} g_1^j t^2]
\end{aligned}
$$

To simplify notation, let:

$$
\begin{aligned}
C_1(bf_i) &= F(bf_i) - f_0^i bf_i - \tfrac{1}{2} f_1^i bf_i^2 & \qquad C_1(bg_j) &= G(bg_j) - g_0^j bg_j - \tfrac{1}{2} g_1^j bg_j^2 \\
C_2(bf_i) &= f_0^i & C_2(bg_j) &= g_0^j \\
C_3(bf_i) &= \tfrac{1}{2} f_1^i & C_3(bg_j) &= \tfrac{1}{2} g_1^j
\end{aligned}
\tag{7}
$$

The coefficients in Equation (7) are computed recursively: $C_l(bf_i), C_l(bg_j)$, $l = 1,2,3$ for $i, j = 1, \ldots, 10$. Then the cdf of the maximum in the k^{th} class is:

$$[F_{\max(f,g)}(t)]_k = [C_1(bf_i) + C_2(bf_i)t + C_3(bf_i)t^2]\cdot[C_1(bg_j) + C_2(bg_j)t + C_3(bg_j)t^2]$$

$$= \sum_{s=2}^{6} \sum_{\substack{m+n=s \\ m,n=1,2,3}} C_m(bf_i)C_n(bg_j)t^{m+n-2}$$

and the pdf is:

$$[f_{\max(f,g)}(t)]_k = \frac{\mathrm{d}}{\mathrm{dt}}[F_{\max(f,g)}(t)]_k$$

$$= \sum_{s=3}^{6} \sum_{\substack{m+n=s \\ m,n=1,2,3}} (m+n-2)\; C_m(bf_i)C_n(bg_j)t^{m+n-3} \tag{8}$$

Equation (8) is computed with the coefficients from Equation (7) to form a piecewise-defined polynomial in the k th class of the maximum. Once the pdf of the maximum has been formed as this second intermediate product, it is immediately replaced by a piecewise-defined polygonal approximation on a partition of ten classes over its domain.

3.3 Polygonal Approximation and Reduction Techniques (PART)

The proposed numerical approximation and reduction techniques, Polygonal Approximation and Reduction Techniques (PART), now consist of polygonal (linear polynomial) approximation of continuous activity duration distributions and series-parallel reduction operations coupled with one of the operative stochastic network reduction methods. The candidate methods are an implementation of "independent multiple arcs" (dual arcs) and sequential approximation. Ordered recursive conditioning, which is not a series-parallel reduction-based method, is not a candidate for coupling with polygonal approximation.

3.3.1 "Independent Multiple Arcs" Approximation

There has been no implementation of an "independent multiple arcs" (dual arcs) approximation and reduction method. As discussed in Section 2.5.1 [above], an irreducible network can be reduced to the single equivalent arc $(1,N)$ by "independently multiplying" (duplicating) arcs with Property 1 only, or arcs with Property 2 only, or a combination of both. "Independently multiplied" (duplicated) arcs are not unique, and there may be many sequences of duplicable arcs from which to choose. Hence, an "independent multiple arcs" approximation reduction method could be based on "independently multiplying" (duplicating) the first available arc with Property 1, or the first available arc with Property 2, or a hybrid process, where the next arc to be "independently multiplied" (duplicated) is selected based on a decision rule or objective function.

If a network is not completely reducible, then the distribution function obtained by any "independent multiple arcs" (dual arcs) approximation and reduction process bounds the exact distribution function of T_N, the duration of the longest path, from below, and hence:

$$E(T_N) \leq E((T_N)_{\text{"independent multiple arcs"}}) \qquad (9)$$

Therefore, one criterion for selecting the next arc to be "independently multiplied" (duplicated) is the minimization of the error arising from the approximation of the throughput distribution T_N, as measured by the overestimation of its mean.

Minimum Approximation Error Method

The following "independent multiple arcs" (dual arcs) approximation and reduction method will minimize the approximation error, as measured by the overestimation of the mean of the throughput distribution:

1. Perform series-parallel reduction on the network until no further reduction is possible. If the reduced network is trivial, i.e., has only one equivalent arc from the source node to the sink node, stop: the original network is completely reducible (completely separable), or the reduced network has now been completely reduced.

2. If the reduced network is nontrivial, then it is irreducible (nonseparable). Choose from among the arcs with either Property 1 (Garman's *a* activities) or Property 2 (Garman's *b* activities) the arc whose duration distribution (arc passage time) has the smallest variance.

    a. If there are ties among the variances, pick the Property 1 arc (*a* activity) with the most successors or the Property 2 arc (*b* activity) with the most predecessors.

    b. If there are still ties, pick arbitrarily.

3. "Independently multiply" (duplicate) the selected arc the number of times necessary to achieve reducibility across the arc, i.e., $s-1$ times if the arc has Property 1 (*a* activity) with *s* successors, or $p-1$ times if the arc has Property 2 (*b* activity) with *p* predecessors. Go to Step 1.

The proof follows directly from Theorem 1 (Garman, 1972) and Dodin's (1985b) definition of "duplicable arcs", which guarantee the existence of at least one arc with Property 1 (*a* activity) and one arc with Property 2 (*b* activity) at Step 2. Choosing the arc whose duration distribution has the smallest variance contributes the least to the inflation of the mean of the approximated throughput distribution per the inequality in Equation (9). The tie-breaker in Step 2.a. was suggested by Garman (1972) for conditioned Monte Carlo simulation of stochastic networks, and seems reasonable for extension here. Step 3 assures that the selected arc and its associated successor node (if Property 1) or predecessor node (if Property 2) are "independently multiplied" (duplicated) the minimally sufficient number of times to neutralize the node as a point of irreducibility (nonseparability) in the network.

While this method has the attraction of minimizing the approximation error, it is, on the other hand, one of the most computationally demanding implementations of "independent multiple arcs" (dual arcs) approximation and reduction. At Step 2 , all the arcs with Property 1 (*a* activities) and all the arcs with Property 2 (*b* activities) must be located in the current configuration of the network, and the variances of their duration distributions compared, in order to identify the arc to be "independently multiplied" (duplicated). Not including the calculation of variances, the complexity of this search task is $O(N^2)$. For moderate or large networks, this may be too high a run time penalty to be forced to pay, particularly if the error accumulation in a less computationally demanding implementation, such as a method based on the first arc encountered with Property 1 (a activity) or the first arc encountered with Property 2 (b activity), is not unacceptable.

First Available Arc Methods

"Independent multiple arcs" (dual arcs) approximation and reduction methods based on first available arcs may search the network in either the forward or backward direction for points of irreducibility (nonseparability), i.e., cross-connections. These methods series-parallel reduce all reducible subnetworks as they are encountered. Then, when the last cross-connection has been located and removed, only one reducible subnetwork will remain to be reduced. The following forward search method is based on the first available arc with Property 1 (a activity). The notation follows Dodin (1985b) (Section 2.5.1 [above]).

1. Series-Parallel reductions:

   a. Beginning with the second node, search the nodes through node $N-1$ in ascending order for all nodes i such that $|B(i)| = |A(i)| = 1$. At each of these nodes, series-reduce (convolve) the subnetwork consisting of the single arc terminating at the node and the single arc emanating from the node.

   b. Beginning with the first node, search the nodes through node $N-1$ in ascending order for pairs of arcs both of whose starting nodes are the same node i and both of whose ending nodes are the same node j. Parallel-reduce the subnetworks consisting of each these pairs of arcs with maximum operations. Go to Step 1.a.

c. When no more series-reductions or parallel-reductions are possible:

(1) If there is only a single equivalent arc between the source node and the sink node, the network reduction is complete. Stop.

(2) Otherwise, the network is now irreducible. Go to Step 2.

2. Beginning with the second node, search the nodes through node $N-2$ for the first node i such that $|B(i)| = 1$ and $|A(i)| \geq 2$. This node is the first cross-connection in the network, and the single arc terminating at the node is the first available arc with Property 1 (a activity). "Independently multiply" this arc $|A(i)| - 1$ times to remove the cross-connection. Go to Step 1.

Again, the proof follows directly from Theorem 1 (Garman, 1972) and Dodin's (1985b) definition of "duplicable arcs." A mirror image backward search method would be based on the first available arc with Property 2 (b activity).

Computer Implementation

The computer program for a PART algorithm using "independent multiple arcs" approximation based on the first-available-arc-with-Property 1 method is in Appendix A. Network reduction is controlled by the main program with the forward search method. Continuous activity duration distributions and intermediate products from series-parallel reduction operations are linearized in subroutine LINEAR by polygonal (linear polynomial) approximation (Section 3.2.1 [above]). Series reduction operations are conducted in subroutine SERIES (Section 3.2.2 [above]), and parallel reduction operations are conducted in subroutine PARA (Section 3.2.3 [above]). Subroutine SORT does the class boundary point sorting required for series-parallel reduction operations. Data are input through three files: network structure, activity duration distributions, and program control information. There are eight different output options, including analytic and graphical presentations of the network throughput distribution and summary statistics. Example program input and output are presented in Chapter 4. The validation of this algorithm is discussed in Section 3.5, and its performance is discussed in Chapter 4.

3.3.2 Sequential Approximation

Dodin (1980, 1985a) implemented sequential approximation using discretization of continuous activity duration distributions (Section 2.5.2 [above]). If discretization is replaced by polygonal (linear polynomial) approximation, a sequential approximation algorithm should be more efficient and require less computer storage for network reduction, because polygonal approximations take up less array space and can be manipulated faster than discretizations of continuous activity duration distributions which are sufficiently dense to yield comparable output accuracies.

The computer program for a PART algorithm using sequential approximation is in Appendix B. The algorithm follows the flowchart in Figure 17, except that discretization is replaced by polygonal approximation. Network reduction is controlled by the main program with the sequential approximation method (Section 2.5.2 [above]). Continuous activity duration distributions and intermediate products from series-parallel reduction operations are linearized in subroutine LINEAR by polygonal approximation. Series reduction operations are conducted in subroutine SERIES, and parallel reduction operations are conducted in subroutine PARA. Subroutine SORT does the class boundary point sorting required for series-parallel reduction operations. Data are input through three files: network structure, activity duration distributions, and program control information. Any node can be designated on an Output Critical List for output report of the throughput distribution through that node. There are eight different output options, including analytic and graphical presentations of the node and network throughput distributions and summary statistics. Example program input and output are presented in Chapter 4. The validation of this algorithm is discussed in Section 3.5, and its performance is discussed in Chapter 4.

3.4 Large Network Approaches

3.4.1 Network Decomposition

One approach to handling large networks is network decomposition: breaking the network apart into subnetworks that can be separately reduced, and then combining the results. For most networks, it is not immediately or intuitively clear whether such a decomposition is possible, much less how to effect a decomposition when one is possible. Research focused on aggregation/disaggregation techniques as a means to identify subnetworks in decompositions of large networks has met with limited success (Section 2.8.1 [above]).

Even with the power of present-day computing systems, the problem of decomposing a large network may be intractible, unless the network is completely reducible. If a large network is not completely reducible, it can only be partially reduced through the repeated application of series-parallel reduction operations until an equivalent irreducible network is reached. By Theorem 2, the irreducible network contains one or more interdictive graphs; these may be either direct, i.e., exactly as shown in Figure 12, or embedded, i.e., where the nodes involved in the cross-connection are not adjacent to each other in the network, but are connected by arcs in such a manner as to achieve the effect of Figure 12. The interdictive graph, which is the minimal-form irreducible network, cannot be decomposed. Consequently, any decomposition of a large network, which is not completely reducible, must be such that each interdictive graph contained in the network is contained within one of the subnetworks of the decomposition. Hence, the problem of decomposing a large network includes the problem of identifying all the interdictive graphs, either direct or embedded, which are contained in the network. There is no efficient solution technique for this identification problem; moreover, the problem appears to be NP-complete (Dodin, 1985b). For this reason, the *K* most critical paths has emerged as the more promising, alternate approach for large networks.

3.4.2 The *K* Most Stochastically Dominating Paths

Dodin (1984) implemented a heuristic for identifying the *K* most stochastically dominating paths using discretization of continuous activity duration distributions (Section 2.8.2 [above]). If discretization is replaced by polygonal (linear polynomial) approximation, a *K*-most-stochastically-dominating-paths algorithm should be more efficient and require less computer storage for network reduction, again because polygonal approximations take up less array space and can be manipulated faster than discretizations of continuous activity duration distributions which are sufficiently dense to yield comparable output accuracies.

The computer program for a PART algorithm for large network approximation and reduction is in Appendix C. The algorithm computes all activity criticality, normalized activity criticality, node criticality, and normalized node criticality indices for a network, and identifies the *K* most stochastically dominating paths. Computation of criticality indices is controlled by the main program with the method discussed below. The *K* most stochastically dominating paths are identified in subroutine DOMPTH, which follows the flowchart in Figure 17, except that discretization is replaced by polygonal approximation. Subroutine COMPAR computes probabilities of the form $P(E_1 \geq E_2)$. Continuous activity duration distributions and intermediate products from series-parallel reduction operations are linearized in subroutine LINEAR by polygonal approximation (Section 3.2.1 [above]). Series reduction operations are conducted in subroutine SERIES (Section 3.2.2 [above]), and parallel reduction operations are conducted in subroutine PARA (Section 3.2.3 [above]). Subroutine SORT does the class boundary point sorting required for series-parallel reduction operations. Data are input through three files: network structure, activity duration distributions, and program control information. Example input and output from this program are presented in Chapter 4. The validation of this algorithm is discussed in Section 3.5, and its performance is discussed in Chapter 4.

Computation of Criticality Indices

In a network (N, A) with path set P, the criticality index CR of a path $\pi_l \in P$ is the probability that the duration $Z(\pi_l)$ of the path is greater than or equal to the duration of every other path:

$$CR(\pi_l) = \mathrm{P}[Z(\pi_l) \geq Z(\pi_q) \text{ for all } \pi_q \in P]$$

and the criticality index of an arc $a_{ij} \in A$ is defined as the sum of the criticality indices of all paths π_l containing the activity:

$$CA(a_{ij}) = \sum_{\{\pi_l | a_{ij} \in \pi_l\}} CR(\pi_l)$$

(Sections 2.6.2 and 2.8.2 [above]). To directly compute path criticality indices, and hence activity criticality indices, requires the complete enumeration and storage of all paths through the network, which can be computer run-time and memory prohibitive for large networks. Alternatively, activity criticality indices can be computed indirectly by exploiting the observation that:

$$CA(a_{ij}) = \sum_{\{\pi_l | a_{ij} \in \pi_l\}} CR(\pi_l)$$

$$= \sum_{\{\pi_l | a_{ij} \in \pi_l\}} \mathrm{P}[Z(\pi_l) \geq Z(\pi_q) \text{ for all } \pi_q \in P]$$

$= \mathrm{P}($duration of all paths which include $a_{ij} \geq$ duration of all other paths)

$= \mathrm{P}($duration of all paths which include $a_{ij} \geq$ duration of the max of all other paths)

The following procedure, adapted from Dodin (unpublished computer code), computes $CA(a_{ij})$ as the latter probability:

1. For each node i, $i = 2,\ldots,N$, recursively determine the *forward* distribution through the node, as:

   $$\max[(\text{duration of a predecessor arc to node } i) \oplus (\text{forward distribution through that arc's starting node})]$$

2. For each node i, $i = N-1,\ldots,1$, recursively determine the *backward* distribution from the node, as:

   $$\max[(\text{duration of a successor arc to node } i) \oplus (\text{backward distribution from that arc's ending node})]$$

From this point on, the procedure is backward recursive. Assume nodes N through $i+1$ have been processed and that node i, $i \geq 2$, is next to be processed.

3. At node i:

   $$\begin{aligned}&\text{number of paths to be considered}\\ &= \text{number of paths considered at node } (i+1)\\ &\quad + |B(i)| - |A(i)|\end{aligned}$$

   and all paths which were considered at node $(i+1)$ will again be considered at node i, except those paths whose predecessor arcs, as defined at node $(i+1)$, start at node i. Bring the distributions of those paths forward from node $(i+1)$ to node i.

4. For each predecessor arc of node i, determine:

   $$\begin{aligned}&\text{duration of all paths which include that predecessor arc}\\ &= (\text{forward distribution through the start node of that predecessor arc})\\ &\quad \oplus (\text{duration of that predecessor arc})\\ &\quad \oplus (\text{backward distribution from node } i)\end{aligned}$$

5. Determine:

   max [durations of all paths considered at node $(i+1)$, except those paths whose predecessor arcs, as defined at node $(i+1)$, start at node i]

6. For each j^{th} predecessor arc of node i, determine:

   max[max of durations of all paths considered at node $(i+1)$, except those paths whose predecessor arcs, as defined at node $(i+1)$, start at node i; and, duration of all paths which include the kth predecessor of node i, $k \neq j$]

   = duration of all paths which do not include the jth predecessor arc of node i

7. Then, for each j^{th} predecessor arc of node i, determine:

$$\begin{aligned} &P(\text{duration of all paths which include the } j\text{th predecessor arc of node } i \geq \\ &\quad \text{duration of all paths which do not include the } j\text{th predecessor arc of node } i\,) \\ &= \text{criticality index of the } j\text{th predecessor arc of node } i \\ &= CA(j\text{th predecessor arc of node } i\,) \end{aligned} \tag{10}$$

The activity criticality indices CA of all the activities in the network are recursively determined by this procedure by Equation (10), as the activity criticality indices of the j^{th} predecessor arcs of node i, $i = N, \ldots, 2$. The criticality indices of the nodes are then computed from the CA's:

$$\text{criticality index of node } i = \sum_j CA(j\text{th successor activity of node } i) \qquad i = 1, \ldots, N-1$$

$$\text{criticality index of node } N = \sum_j CA(j\text{th predecessor activity of node } N)$$

The normalized activity criticality and node criticality indices follow as:

$$\text{normalized activity criticality index} = \frac{\text{activity criticality index}}{\text{node criticality index of node } N}$$

$$\text{normalized node criticality index} = \frac{\text{node criticality index}}{\text{node criticality index of node } N}$$

Computation of Probabilities of the Form P $(E_1 \geq E_2)$

During the computation of the activity criticality indices and the identification of the K most stochastically dominating paths, it is necessary to compute probabilities of the form $\mathrm{P}(E_1 \geq E_2)$ where E_1 and E_2 are path events, such as in Equation (10). Subroutine COMPAR computes these probabilities, using the following approach. Suppose that E_1 is described by the random variable (duration) t_1 whose distribution is currently approximated by the polygonal approximation f which is piecewise-defined on a ten-class partition with class boundaries $f_{\text{lower bound}} = bf_1, bf_2, \ldots, bf_{11} = f_{\text{upper bound}}$ over its domain $[f_{\text{lower bound}}, f_{\text{upper bound}}]$, and that E_2 is described by the random variable t_2 whose distribution is similarly currently approximated by the polygonal approximation g which is piecewise-defined on a ten-class partition with class boundaries $g_{\text{lower bound}} = bg_1, bg_2, \ldots, bg_{11} = g_{\text{upper bound}}$ over its domain $[g_{\text{lower bound}}, g_{\text{upper bound}}]$. Over its i^{th} class, f is defined by:

$$f_i(t_1) = f_0^i + f_1^i t_1$$

and over its j^{th} class, g is defined by:

$$g_j(t_2) = g_0^j + g_1^j t_2$$

The joint distribution of t_1 and t_2 is defined on the ten-class by ten-class rectangle, $\{(t_1, t_2) \mid f_{\text{lower}} \leq t_1 \leq f_{\text{upper}}, g_{\text{lower}} \leq t_2 \leq g_{\text{upper}}\}$ with class boundaries (bf_i, bg_j), $i, j = 1, \ldots, 11$. $\mathrm{P}(E_1 \geq E_2) = \mathrm{P}(t_1 \geq t_2)$ is the accumulated probability under the joint pdf of t_1 and t_2 over the shaded area in the rectangle below the line $t_1 = t_2$ as shown in Figure 28.

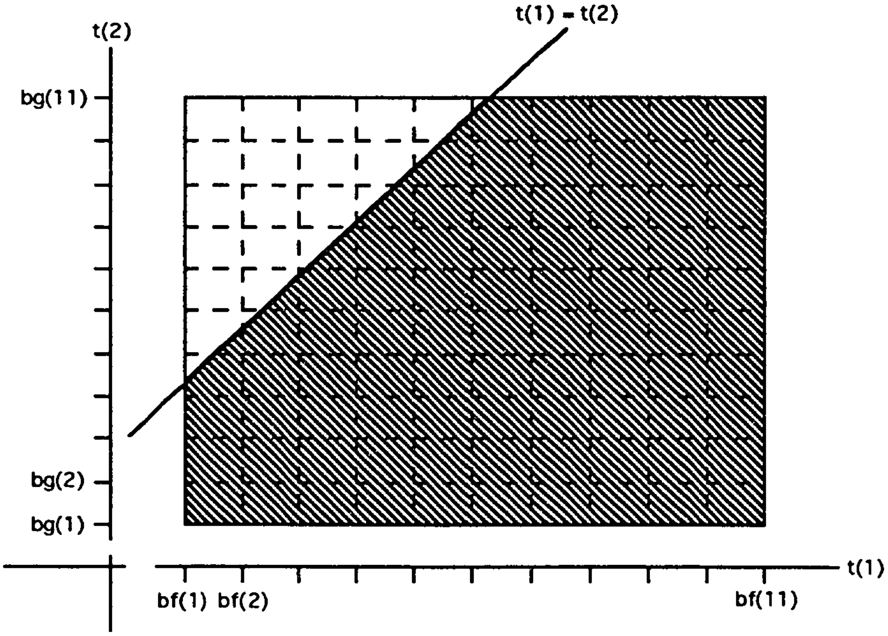


Figure 28. Geometry of computing P($E_1 \geq E_2$).

E_1 and E_2 are assumed to be independent, though they clearly are not if they have any common activities. Under the assumption of independence, the pdf of the joint distribution of t_1 and t_2 is:

$$[f,g](t_1,t_2) = f(t_1)\cdot g(t_2)$$

The joint distribution function obtained under the assumption of independence bounds the exact joint distribution of t_1 and t_2. The bound depends on the lost dependency, which depends on the number of common activities between E_1 and E_2 and the criticality of these activities. If E_1 and E_2 have no common activities, then the joint distribution obtained under the assumption of independence is equal to the exact joint distribution of t_1 and t_2. If, on the other hand, E_1 and E_2 have one or more common activities, then the joint distribution obtained under the assumption of independence bounds the exact joint distribution from below, analogous to the situation with the "independent multiple arcs" reduction method (Esary et al., 1967; Dodin, 1985c).

$P(E_1 \geq E_2) = P(t_1 \geq t_2)$ is obtained by computing the contribution from each of the 100 rectangular classes in the domain of the approximated joint distribution and then totaling those contributions. The contribution from each class is the probability under the joint pdf of t_1 and t_2 over the area within the class below the line $t_1 = t_2$. There are six cases, as shown in Figure 29. If the class lies entirely above the line $t_1 = t_2$ [Part (a) of Figure 29], the contribution from the class is zero. If the class lies entirely below the line $t_1 = t_2$ [Part (b) of Figure 29], the contribution from the class is the probability under the joint pdf of t_1 and t_2 over the entire class. If the class straddles the line $t_1 = t_2$ [Parts (c) - (f) of Figure 29], the contribution from the class is the probability under the joint pdf of t_1 and t_2 over the portion of the class which lies below the line $t_1 = t_2$. The geometry of the straddle affects the limits of integration as shown below. The 100 classes are considered in succession. Each class is first tested to determine which of the six cases applies to the class, using a set of recognition criteria, which are inequality relationships among the class boundaries arising from the geometries shown in Figure 29. Then the contribution to $P(E_1 \geq E_2)$ is computed as the double integral of the joint pdf of t_1 and t_2 over the portion of the class which lies below the line $t_1 = t_2$. The recognition criteria, integration formulations, and intermediate coefficients for the (i, j) class are as follows:

Intermediate coefficients:

$$C_0 = \left[f_0^i(bf_{i+1} - bg_{j+1}) + \frac{f_1^i}{2}(bf_{i+1}{}^2 - bg_{j+1}{}^2)\right]\left[g_0^j(bg_{j+1} - bg_j) + \frac{g_1^j}{2}(bg_{j+1}{}^2 - bg_j{}^2)\right]$$

$$C_1 = -\left[f_o^i(g_0^j bg_j + \frac{g_1^j}{2} bg_j^2)\right] \qquad C_2 = \left[\frac{f_0^i g_0^j}{2} - \frac{f_1^i}{2}(g_0^j bg_j + \frac{g_1^j}{2} bg_j{}^2)\right]$$

$$C_3 = \left[\frac{f_1^i g_0^j}{3} + \frac{f_0^i g_1^j}{6}\right] \qquad C_4 = \left[\frac{f_1^i g_1^j}{8}\right]$$

Case 1. [Part (a) of Figure 29]:

Recognition criteria: $bf_{i+1} \leq bg_j$

Contribution from class:

$[\mathrm{P}(E_1 \geq E_2)]_{\mathrm{class}} = 0$

Case 2. [Part (b) of Figure 29]:

Recognition criteria: $bg_{j+1} \leq bf_i$

Contribution from class:

$$[\mathrm{P}(E_1 \geq E_2)]_{\mathrm{class}}$$

$$= \int_{bf_i}^{bf_{i+1}} \int_{bg_j}^{bg_{j+1}} (f_0^i + f_1^i t_1)(g_0^j + g_1^j t_2)\, dt_2 dt_1$$

$$= \left[f_0^i (bf_{i+1} - bf_i) + \frac{f_1^i}{2}(bf_{i+1}{}^2 - bf_i{}^2) \right] \left[g_0^j (bg_{j+1} - bg_j) + \frac{g_1^j}{2}(bg_{j+1}{}^2 - bg_j{}^2) \right]$$

Case 3. [Part (c) of Figure 29]:

Recognition criteria: $bg_j < bf_i < bg_{j+1} < bf_{i+1}$

Contribution from class:

$$[\mathrm{P}(E_1 \geq E_2)]_{\mathrm{class}}$$

$$= \int_{bf_i}^{bg_{j+1}} \int_{bg_j}^{t_1} (f_0^i + f_1^i t_1)(g_0^j + g_1^j t_2)\, dt_2 dt_1 + \int_{bg_{j+1}}^{bf_{i+1}} \int_{bg_j}^{bg_{j+1}} (f_0^i + f_1^i t_1)(g_0^j + g_1^j t_2)\, dt_2 dt_1$$

$$= C_0 + C_1(bg_{j+1} - bf_i) + C_2(bg_{j+1}^2 - bf_i^2) + C_3(bg_{j+1}^3 - bf_i^3) + C_4(bg_{j+1}^4 - bf_i^4)$$

Case 4. [Part (d) of Figure 29]:

Recognition criteria: $bf_i \le bg_j < bg_{j+1} < bf_{i+1}$

Contribution from class:

$$[\mathrm{P}(E_1 \ge E_2)]_{\mathrm{class}}$$

$$= \int_{bg_j}^{bg_{j+1}} \int_{bg_j}^{t_1} (f_0^i + f_1^i t_1)(g_0^j + g_1^j t_2)\,dt_2 dt_1 + \int_{bg_{j+1}}^{bf_{i+1}} \int_{bg_j}^{bg_{j+1}} (f_0^i + f_1^i t_1)(g_0^j + g_1^j t_2)\,dt_2 dt_1$$

$$= C_0 + C_1(bg_{j+1} - bg_j) + C_2(bg_{j+1}^2 - bg_j{}^2) + C_3(bg_{j+1}^3 - bg_j{}^3) + C_4(bg_{j+1}^4 - bg_j{}^4)$$

Case 5. [Part (e) of Figure 29]:

Recognition criteria: $bf_i \le bg_j < bf_{i+1} \le bg_{j+1}$

Contribution from class:

$$[\mathrm{P}(E_1 \ge E_2)]_{\mathrm{class}}$$

$$= \int_{bg_j}^{bf_{i+1}} \int_{bg_j}^{t_1} (f_0^i + f_1^i t_1)(g_0^j + g_1^j t_2)\,dt_2 dt_1$$

$$= C_1(bf_{i+1} - bg_j) + C_2(bf_{i+1}{}^2 - bg_j{}^2) + C_3(bf_{i+1}{}^3 - bg_j{}^3) + C_4(bf_{i+1}{}^4 - bg_j{}^4)$$

Case 6. [Part (f) of Figure 29]:

Recognition criteria: $bg_j < bf_i < bf_{i+1} < bg_{j+1}$

Contribution from class:

$$[\mathrm{P}(E_1 \ge E_2)]_{\mathrm{class}}$$

$$= \int_{bf_i}^{bf_{i+1}} \int_{bg_j}^{t_1} (f_0^i + f_1^i t_1)(g_0^j + g_1^j t_2)\,dt_2 dt_1$$

$$= C_1(bf_{i+1} - bf_i) + C_2(bf_{i+1}{}^2 - bf_i{}^2) + C_3(bf_{i+1}{}^3 - bf_i{}^3) + C_4(bf_{i+1}{}^4 - bf_i{}^4)$$

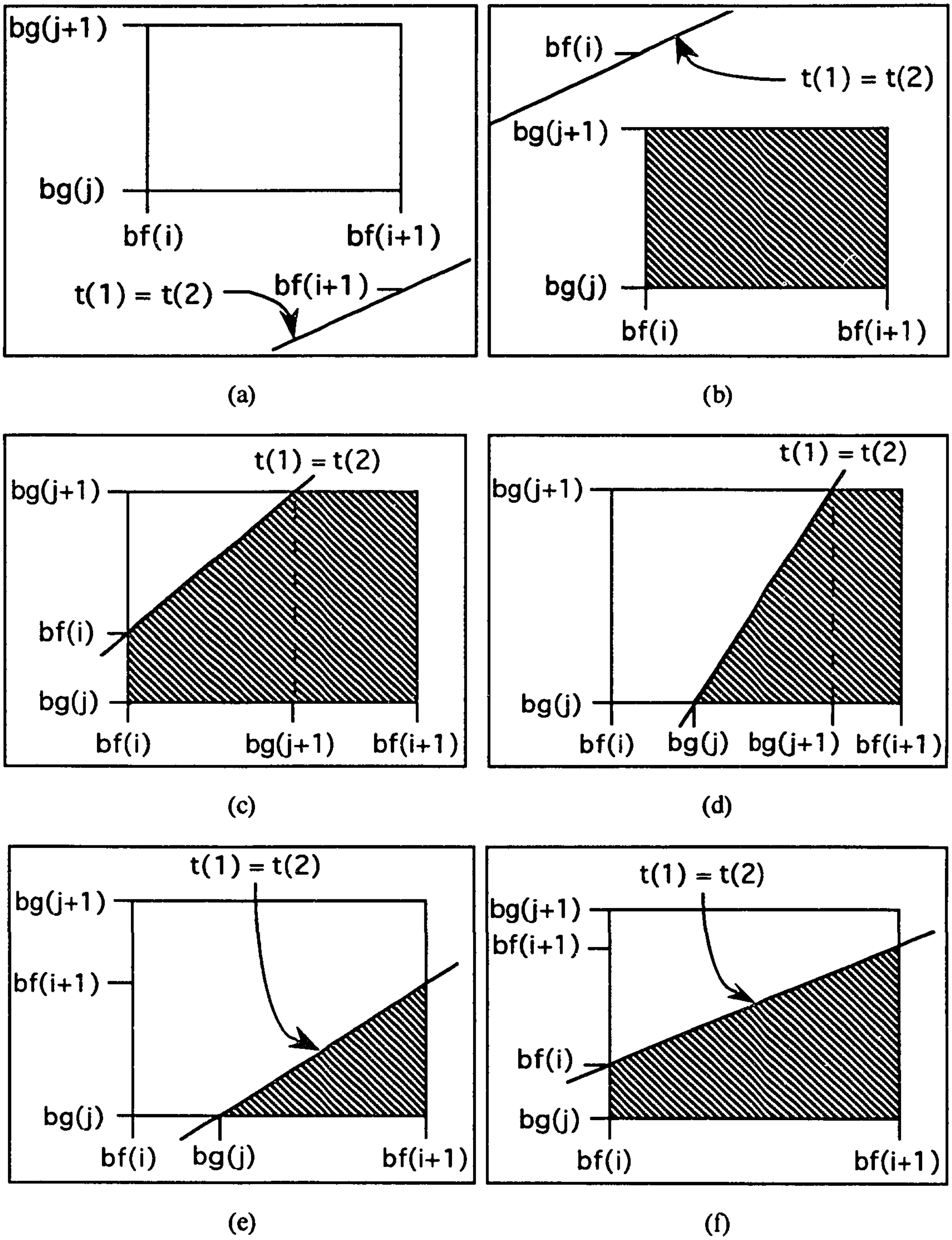


Figure 29. Contributions of joint distribution classes to P($E_1 \geq E_2$).

To control error build-up from the polygonal approximation, after $\mathrm{P}(E_1 \geq E_2)$ has been computed, the value is normalized so that the probability under the approximated joint pdf is one:

correction factor $(CF) = 1.0$ / computed probability under approximated joint pdf

$$CF = 1.0 \Big/ \left[\sum_{\text{all classes}} \int_{bf_i}^{bf_{i+1}} \int_{bg_j}^{bg_{j+1}} (f_0^i + f_1^i t_1)(g_0^j + g_1^j t_2) dt_2 dt_1 \right]$$

$$= 1.0 \Big/ \left\{ \sum_{\text{all classes}} \left[f_0^i (bf_{i+1} - bf_i) + \frac{f_1^i}{2} (bf_{i+1}{}^2 - bf_i{}^2) \right] \left[g_0^j (bg_{j+1} - bg_j) + \frac{g_1^j}{2} (bg_{j+1}{}^2 - bg_j{}^2) \right] \right\}$$

$$\mathrm{P}(E_1 \geq E_2) = \left[\mathrm{P}(E_1 \geq E_2) \right] * CF.$$

3.5 Validation Approach

Three PART algorithms, based on polygonal (linear polynomial) approximation, have been developed: a PART algorithm using "independent multiple arcs" approximation based on the first-available-arc-with-Property 1 method, a PART algorithm using sequential approximation, and a PART algorithm for identifying the *K* most stochastically dominating paths for large network approximation and reduction. To validate the polygonal approximation and reduction technique for stochastic project management networks, these algorithms must be empirically tested, and their performance during the testing must be carefully observed and evaluated. The framework for the controlled execution of this testing is design of experiments (DOE). Since activity duration distribution parameters can take on an infinite number of values, traditional experimental designs, such as factorials, cannot be employed. Rather, an effective experimental design must be specified which forces the algorithms to demonstrate performance under challenging conditions.

3.5.1 Design of Experiments

The DOE approach to evaluating the performance of computer algorithms begins with the identification of the characteristics of the problem which influence the performance of the algorithms in solving the project. Test problems are generated with sufficient variations in these characteristics so that any sensitivity in the performance of the algorithms to the characteristics can be detected and explained, and metrics are specified so that how closely the algorithms approximate the solutions to the test networks can be measured and assessed. The solutions to the test problems must be known or obtainable; otherwise, surrogates are generated independent of the algorithms.

For stochastic project management networks, these characteristics are: the number of nodes, the number of activities (arcs), the network structure, and the distributions of activity durations and their parameters. "Strongly randomized [test] networks" (Dodin, 1985a) are obtained when, at the minimum, the network structure (for a given number of nodes, N, and a given number of activities, A), the distributions of activity durations, and their parameters are randomly determined. Since stochastic project management networks cannot be analytically solved - the throughput distribution determined for small and moderate-sized networks, and the K most stochastically dominating paths identified for large networks - the results of extensive simulations serve as surrogates for the network solutions. The closeness of the algorithms' approximations to the simulation results, taken as the "true" solutions, is measured by goodness-of-fit tests, in the case of the throughput distributions, and, for large networks, by the numbers of activities which match between the K most stochastically dominating paths and the K most critical paths. Since not all combinations of network size (numbers of nodes and activities) and structure, and distributions of activity durations and their parameters, can be considered, even within a set of ranges of values of these characteristics, a traditional experimental design, such as a factorial design, cannot be employed. If maximum randomness is the design criterion, then, at best, network size and structure can be randomly selected from predetermined ranges of values of N and A, and distributions of activity duration can be randomly selected from a set of available distribution types and their parameters randomly selected from ranges of values of those parameters. A better criterion, however, is how great a challenge a randomly constructed test network presents to the algorithms charged with its solution. A pair of activity duration distributions with "high" variances, i.e., whose standard deviations are high percentages of their means (high coefficients of variation), is the most difficult convolution for a numerical approximation-based series-reduction operator, while a pair of maximally overlapping activity duration distributions is the most difficult for a numerical approximation-based parallel-reduction operator to reduce. An effective experimental design for algorithm testing, then, is to:

> specify representative sets of activity duration distributions which contain pairs (or higher multiples) of distributions with "high" variances and pairs (or higher multiples) of maximally overlapping distributions;
>
> then, for each set of distributions, construct a set of test networks by randomly selecting network size and structure from predetermined ranges of values of N and A and all activity duration distributions from that set of distributions.

If the sets of test networks are sufficiently large, some of the test networks will contain combinations of activity duration distributions which are quite challenging for the approximation-reduction algorithms to resolve with accuracy, while others of the test networks will be less challenging for the algorithms. Algorithm performance should be uniform across the sets of distributions and test networks.

Computer Implementation

The computer program for the validation version of a PART algorithm using "independent multiple arcs" approximation based on the first-available-arc-with-Property 1 is in Appendix D; the computer program for the validation version of a PART algorithm using sequential approximation is in Appendix E; the computer program for the validation version of a PART algorithm for identifying the K most stochastically dominating paths for large network approximation and reduction is in Appendix F. These programs can accommodate networks with up to 100 nodes and up to 99 activities emanating from a node. (In a fully connected network with 100 nodes, there are 99 activities emanating from the source node to every other node.) Up to 100 "strongly random" test networks may be specified to be generated during each run of the programs which approximate the throughput distribution; there is no number-of-runs restriction on the program for large network approximation and reduction. The number of nodes, N, in the test networks to be generated may be specified between 2 and 100; if 0 is input, the number of nodes is randomly selected from between 2 and 100. The number of activities, A, in each of the test networks to be generated may be specified between $L_A = N - 1$, the minimum number of activities in a network with N nodes, and $U_A = N(N-1)/2$, the maximum number of activities in a fully connected network with N nodes; if 0 is input, the number of activities is drawn as a random value from the normal distribution which closely approximates the distribution of A:

$$\text{normal}\left(\frac{(L_A + U_A)}{2} - \frac{(U_A - L_A)^2}{500}, \frac{(U_A - L_A)}{2.5}\right)$$

(Section 2.7.2 [above]). The specified number of random test networks with N nodes and A activities are generated by subroutine GENRAN, following the deletion method (DM) in Figure 20 if $A \geq N(N-1)/4$ or the addition method (AM) in Figure 21 if otherwise. Each test network is then reduced by the PART algorithm of the particular validation program being run. To obtain a surrogate for each test network's solution, the network is extensively simulated; up to 10,000 simulation replications may be specified. The simulation results are then statistically compared with the algorithm output. Viewed collectively across all the test networks generated and reduced, these comparisons characterize the performance of the PART algorithm of the particular validation program being run.

3.5.2 Throughput Approximation with Simulation

In the computer programs for the validation versions of PART algorithms using an "independent multiple arcs" approximation method and the sequential approximation method, subroutine SIMULT simulates the throughput distribution during the k^{th} simulation replication using the following procedure:

1. Generate a random value, t_{ij}, from the activity duration distribution of each activity, a_{ij}, in the network.

2. Generate the critical path to each node by computing the simulated duration (time) through each node i, $TSIM(i)$.

    a. $TSIM(1) = 0$.

    At Step 2.b., the procedure is forward recursive. Assume nodes 2 through $i-1$ have been processed and that node i, $i \leq N$, is next to be processed.

    b. At node i:

    (1) Determine the predecessor nodes j to node i and the activities a_{ji} which connect nodes j to node i.

    (2) Compute the simulated time through node i as:

    $$TSIM(i) = \max[TSIM(j) + t_{ji}]$$

    (3) If $i < N$, go to Step 2.b. If $i = N$, go to Step 3.

3. Store the simulated time through the network, $TSIM(N)$, for the k^{th} simulation replication.

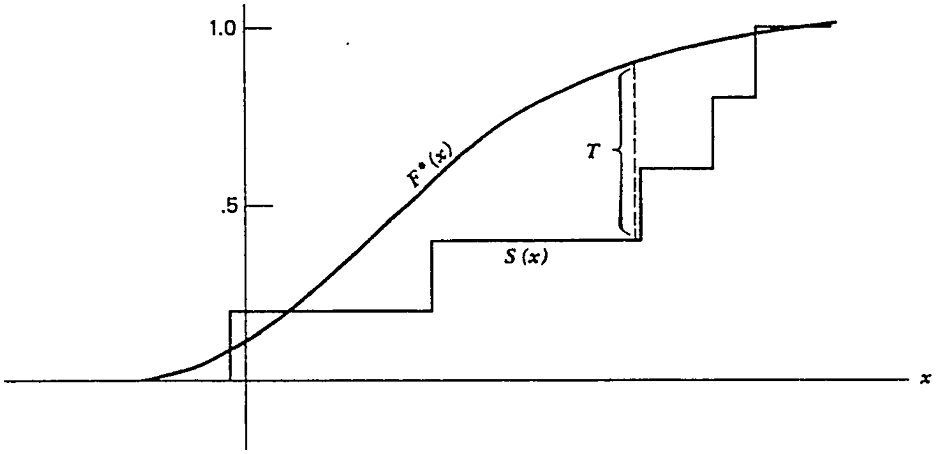


Figure 30. The Kolmogorov-Smirnov Goodness-of-Fit Test. [Adapted from Conover (1980)]

The Kolmogorov-Smirnov goodness-of-fit test is then used to compare the polygonal approximation of the throughput distribution developed by the PART algorithm of the particular validation program being run with the simulation approximation of the throughput distribution.

The Kolmogorov-Smirnov Goodness-of-Fit Test[2]

The Kolmogorov-Smirnov (K-S) one-sample goodness-of-fit test is used when examining a random sample from some unknown distribution in order to test the null hypothesis that the unknown distribution is, in fact, a known, specified distribution with cdf $F^*(x)$. The random sample is compared with $F^*(x)$ by means of the empirical distribution function (sample cdf), $S(x)$, to see if there is "good" agreement. If there is not sufficiently "good" agreement between $F^*(x)$ and $S(x)$, the null hypothesis is rejected; otherwise, it is not rejected. The K-S test statistic T is the maximum vertical distance between the graphs of $F^*(x)$ and $S(x)$, as shown in Figure 30:

[2] Discussions of the Kolmogorov-Smirnov goodness-of-fit tests can found in numerous nonparametric statistics texts. For example, Conover (1980) presents a thorough treatment.

$$T = \sup_x |F^*(x) - S(x)|$$

("sup" is the supremum, or greatest). Quantiles of the distribution of T are tabulated in the literature. They are exact whenever the hypothesized distribution is continuous (Noether, 1967).

When the number of simulation replications is large, the simulation approximation of the throughput distribution may safely be considered the "true" throughput distribution, and hence a one-sample Kolmogorov-Smirnov goodness-of-fit test may be used. To compute the value of the K-S test statistic, the domain of the polygonal approximation of the throughput distribution, obtained from the PART algorithm of the particular validation program being run, is partitioned with a set of equal size classes. Fergeson and Shortell (1978) found the best results are achieved when the number of classes in the partition is 50. The linear polynomial functional representations of the polynomial approximation of the throughput distribution are used to compute the area under the approximated pdf in each class of the partition. Then an approximated cdf of the throughput distribution, *TOTAAR*, is constructed by summing the areas under the approximated pdf through each of the 51 boundary points of the partition. A cumulative frequency histogram, *SIMTOT*, of the stored, simulated times through the network, *TSIM*(N), is then constructed over the partition by sorting the time values into the 50 classes of the partition and then summing the numbers of time values in the classes through each of the 51 boundary points of the partition. Constructing the approximated cdf of the throughput distribution, *TOTAAR*, and the cumulative frequency histogram of the simulated throughput times, *SIMTOT*, on a common partition of the domain of the approximated throughput distribution facilitates the computation of the K-S test statistic.

By virtue of its construction, the cumulative frequency histogram of the simulated throughput times, *SIMTOT*, behaves like a grouped data statistic. When viewed as such, the maximum separation between *TOTAAR* and *SIMTOT* occurs either just to the right or just to the left of one of the boundary points, so the computed value of the K-S test statisitc would be the maximum of the "just-to-the-left" and "just-to-the-right" differences. However, when the number of simulation replications is large, the simulation approximation of the throughput distribution may be viewed as a continuous function, and the values which its frequency histogram takes on at the boundary points may be taken as "exact" values of the throughput cdf (Dodin, 1985a). Then, the computed value of the K-S test statistic is just the maximum of the differences between *TOTAAR* and *SIMTOT* computed at the boundary points, b_i, of the partition:

$$T = \max_i \left| SIMTOT(b_i) - TOTAAR(b_i) \right| \qquad i = 2,\ldots,51$$

The computed value of the K-S test statistic is compared with the 1%, 2%, 5%, 10% and 20% quantiles of the distribution of T in order to bound the ***prob value:***

$$\text{prob value} = \text{P(Type I error)} = \text{P(reject } H_0 | H_0 \text{ true)}$$

in an interval (<1%, 1% - 2%, 2% - 5%, 5% - 10%, 10% - 20%, or >20%). If the prob value is >5%, the conclusion of the K-S goodness of fit test would be that there is insufficient evidence to reject the null hypothesis that the polygonal approximation of the throughput distribution, obtained from the PART algorithm of the particular validation program being run, is different from the true throughput distribution at the at-least 5% level of statistical significance.

3.5.3 Path Criticality Approximation with Simulation

In the computer program for the validation version of a PART algorithm for identifying the K most stochastically dominating paths for large network approximation and reduction, subroutine SIMULC simulates each test network, computes simulation-approximated activity criticality and node criticality indices, and identifies the K most critical paths. The activity criticality and node criticality indices are approximated using the following procedure. During the l^{th} simulation replication:

1. Generate a random value, t_{ij}, from the activity duration distribution of each activity, a_{ij}, in the network.

2. Compute the earliest completion duration (time) of each node i, $TEARLY(i)$.

   a. $TEARLY(1) = 0$.

   At Step 2.b., the procedure is forward recursive. Assume nodes 2 through $i-1$ have been processed and that node i, $i \leq N$, is next to be processed.

   b. At node i:

      (1) Determine the predecessor nodes j to node i and the activities a_{ji} which connect nodes j to node i.

      (2) Compute the earliest completion time of node i as:

      $$TEARLY(i) = \max[TEARLY(j) + t_{ji}]$$

      (3) If $i < N$, go to Step 2.b. If $i = N$, go to Step 3.

3. Compute the latest completion time of each node i, $TLATE(i)$, and the number of critical nodes succeeding node i, $NOCS(i)$.

   a. $TLATE(N) = TEARLY(N)$ and $NOCS(N) = 1$.

   At Step 3.b., the procedure is backward recursive. Assume nodes $N-1$ through $i+1$ have been processed and that node i, $i \geq 1$, is next to be processed.

b. At node i:

(1) Determine the successor nodes j to node i and the activities a_{ij} which connect node i to nodes j.

(2) Compute the float of the activities a_{ij}, $TFLOAT(a_{ij})$ as:

$$TFLOAT(a_{ij}) = [TLATE(j) - t_{ij}] - TEARLY(i)$$

If $TFLOAT(a_{ij}) = 0$, activity a_{ij} is on a critical path(s). Set:

$$NOCS(i) = NOCS(i) + NOCS(j)$$

(3) Compute the latest completion time of node i as:

$$TLATE(i) = \min[TLATE(j) - t_{ij}]$$

(4) If $i > 1$, go to Step 3.b. If $i = 1$, go to Step 4.

At Step 4., the procedure is forward recursive. Assume nodes 2 through $i-1$ have been processed and that node i, $i \leq N$, is next to be processed.

4. Determine the number of times each activity a_{ij}, the k th predecessor of node i, appears on a critical path, $NCA(i,k)$, and the number of critical nodes preceding node i, $NOCP(i)$. At node i:

a. Determine the predecessor nodes j to node i and the activities a_{ji} which connect nodes j to node i.

b. If activity a_{ji} is the k th predecessor of node i and $TFLOAT(a_{ji}) = 0$, activity a_{ij} is on a critical path(s). Set:

$$NCA(i,k) = NCA(i,k) + NOCS(i) \cdot NOCP(j)$$
$$NCCP(i) = NOCP(i) + NOCP(j)$$

c. If $i < N$, go to Step 4. If $i = N$, stop.

After *NSIM* replications of the simulation have been run, the simulation-approximated activity criticality indices *CA* and node criticality indices are computed as follows:

$$\text{sim.-approx. } CA(k\text{th predecessor activity of node } i) = \frac{NCA(i,k)}{NSIM}$$

$$\text{sim.-approx. criticality index of node } i = \sum_{j} CA(j\text{th successor activity to node } i),$$

$$i = 1,\ldots,N-1$$

$$\text{sim.-approx. criticality index of node } N = \sum_{j} CA(j\text{th predecessor activity to node } N)$$

The simulation-approximated normalized activity criticality and node criticality indices follow as:

$$\begin{array}{c}\text{sim.-approx.}\\ \text{normalized activity criticality index}\end{array} = \frac{\text{sim.-approx. activity criticality index}}{\text{sim.-approx. node criticality index of node } N}$$

$$\begin{array}{c}\text{sim.-approx.}\\ \text{normalized node criticality index}\end{array} = \frac{\text{sim.-approx. node criticality index}}{\text{sim.-approx. node criticality index of node } N}$$

To identify the *K* most critical paths in a test network requires knowledge of the values of the path criticality indices *CR* of all the paths through the network. Simulation-approximated *CR*s could be obtained by enumerating all the paths in the network before simulation, and then, after each simulation replication, calculating the *CR* of each path and updating the average value of each path's *CR* through the current simulation replication. This is a burdensome task, and for large networks is impractical, because of the computer storage required to hold the enumerated network paths during the simulation. Alternatively, the computer program for the validation version of a PART algorithm for large network approximation and reduction uses the average of the simulation-approximated *CA*s of the activities on a path as the value of the simulation-approximated *CR* of the path. If a test network has no common arcs among its paths, or if all common arcs have *CA*s equal to zero, then the average of the simulation-approximated *CA*s of the activities on a path would be the same as the simulation-approximated *CR* of that path obtained from path enumeration. If the test network has common arcs with non-zero *CA* s, then the average of the simulation-approximated *CA* s of the activities on a path with one or more of these common arcs would be greater than the simulation-approximated *CR* of that path obtained from path enumeration, because the non-zero *CA* s of the common arcs on that path reflect the occurrence of those common arcs on other critical paths as well. Consequently, the averages of the simulation-approximated *CA* s of the activities on paths which contain common arcs with non-zero *CA* s are inflated relative to the averages of the simulation-approximated *CA* s of the activities on paths which contain no common arcs or whose common arcs have *CA* s equal to zero. Moreover, this inflation is not uniform, since the more common arcs with non-zero *CA* s a path contains and the higher the *CA* s of these arcs, the higher the average of the simulation-approximated *CA* s of the activities on the path, i.e., such a path would appear more critical relative to other paths. However, especially in large networks, the relative ranking of the paths, based on the average of the simulation-approximated *CA* s of the activities on a path as the value of the simulation-approximated *CR* of the path, should not change compared to the ranking of the paths based on the simulation-approximated *CR* s obtained from path enumeration, if they could be obtained, except under unusual circumstances. Since the objective of critical path analysis is to identify critical paths and activities which appear on critical paths, especially those activities which appear on several critical paths, so that management attention can be focused on them to insure that they don't incur schedule slips, the computed values of approximated path criticality indices are not nearly as important as the identification of the relatively most critical paths and any activities which appear on several critical paths, all of which may be deserving of special management attention.

To identify the *K* most critical paths in a test network based on averaged, simulation-approximated *CR* s, all paths through the network must be enumerated and ranked.

Subroutine SIMULC employs a procedure adapted from Asano and Sato (1985). The following notation was used in their development:

> For each node v, the successor nodes of v are ordered so that the first successor of v is denoted by $fs(v)$ and the next successor of v following from v' is denoted $\text{next}(v,v')$. If v' is the last successor of v, then $\text{next}(v,v') = \emptyset$.

The path enumeration procedure is as follows:

1. Identify a first path: Starting at the source node, node 1, trace the pointer $fs(\ )$ to the first successor of each node, performing the operation $v(i) = fs(v(i-1))$ for $i = 1,2,\ldots$ until $v(i) = N$.
2. Identify a new path: Find the largest i such that $\text{next}(v(i), v(i+1)) \neq \emptyset$.
    a. Next path: Set $v(i+1) = \text{next}(v(i), v(i+1))$. Then trace the pointer $fs(\ )$ from $v(i+1)$ to the sink node, node N. Go to Step 2.
    b. If such an i does not exist, then all paths have been enumerated. Stop.

Subroutine SIMULC identifies the K most critical paths using the following procedure:

1. Enumerate K paths. Compute the averaged, simulation-approximated CRs of the paths. Rank the paths in descending order on the basis of their CRs. Store the rank-ordered set of K paths.
2. Enumerate a next path. If none, stop. Compute the averaged, simulation-approximated CR of the path.
    a. If this path's CR is > the CR of the K^{th}-ranked path in the rank-ordered set of K paths, delete the K^{th}-ranked path from the set, insert the new path into the set, rank the paths in the set in descending order on the basis of their CRs, and re-store the rank-ordered set of K paths. Go to Step 2.

b. If this path's *CR* is ≤ the *CR* of the K^{th}-ranked path in the rank-ordered set of K paths, discard the path. Go to Step 2.

The rank-ordered set of K paths contains the K most critical paths based on averaged, simulation-approximated *CR*s. This path set is compared with the K most stochastically dominating paths obtained from subroutine DOMPTH by counting the number of activities in common between each pair of k^{th}-ranked paths, $k = 1,\ldots,K$.

3.6 Summary

In this chapter, a new method for numerical approximation of continuous activity resource consumption (duration) distributions of stochastic project management networks, linear polynomial (polygonal) approximation, was developed, derived from the spline approximations used in numerical differentiation and integration and motivated by the shortcomings of the first attempt at an exact solution to stochastic network reduction by Martin (1965). Series-parallel reduction operations based on the new method were described. The method was mated with three network reduction approaches - "independent multiple arcs" (dual arcs), sequential approximation, and a heuristic for identifying the K most critical paths - to form the members of a new family of Polygonal Approximation and Reduction Techniques (PART). The PART algorithm using "independent multiple arcs" (dual arcs) represents the first successful implementation of an arc-duplication reduction method. A design-of-experiments framework employing "strongly randomized" networks was proposed for the validation of the method and its associated PART algorithms, and validation versions of the PART algorithms were constructed with capabilities to simulate "true" network solutions. The next chapter reports the results of performance tests conducted to validate the polygonal approximation method and its associated PART algorithms.

CHAPTER 4

RESULTS

In this chapter, polygonal approximation of continuous activity resource consumption distributions and series-parallel reduction operations based on polygonal approximation are validated for the distributions frequently encountered in stochastic project management networks. Then, the PART algorithms using "independent multiple arcs" approximation based on the first-available-arc-with-Property 1 method and sequential approximation to obtain network throughput distributions are validated, as is a PART algorithm for identifying the *K* most stochastically dominating paths for large network approximation and reduction, through comparisons with simulation results and the results obtained from competing approximation procedures. Finally, the run time and storage requirements of these PART algorithms are contrasted with those of competing approximation procedures.

4.1 Polygonal Approximation

The validation of polygonal approximation of continuous activity resource consumption distributions (Section 3.2.1 [above]) is illustrated with two distributions representative of the set which was tested. Figure 31 depicts a truncated normal distribution, defined on [0,10], with a mean of 5 and a standard deviation of 1.67, and its polygonal approximation constructed on 10 classes using:

$$n = 10 + \text{integer}(\text{class width} * 3) = 13$$

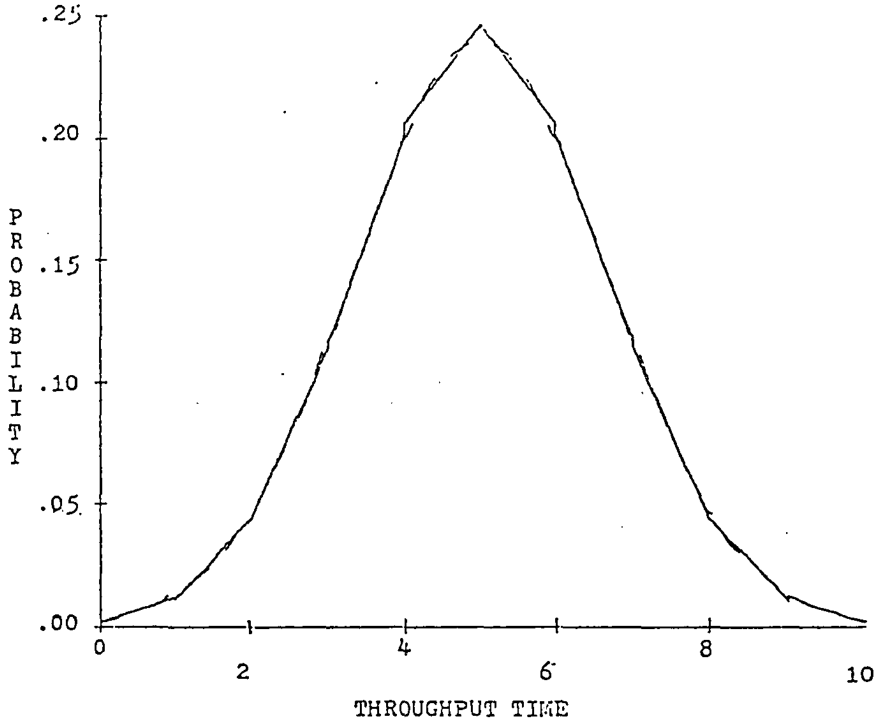
regression fitting points uniformly distributed across each class. Figure 32 depicts a truncated exponential distribution, also defined on [0,10], with a mean of 3 and its polygonal approximation, which was similarly constructed. (In these and later similar figures, the dashed line represents the pdf of the distribution, and the solid line represents its polygonal approximation. When the polygonal approximation is very close to the actual pdf, it may be difficult to distinguish between them in the illustrations.) The tabular data presented below the graphs permit direct comparisons of the values of the pdf (column labeled ACTUAL) and the polygonal approximation (column labeled APPROX) at selected points across the domains of definition. Tables 13 and 14 contain the class boundaries and the coefficients b_0 and b_1 of the approximating polynomial:

$$b_0 + b_1 t$$

in each class. The values of the polygonal approximation (APPROX column) were computed with the approximating polynomial for the class containing each of the selected T values. At a class boundary, there are two approximations of the distribution at that point, corresponding to the approximating line segments of the class which terminates and the class which begins at that class boundary. Averaging the two approximation values at a class boundary produces a smoother fit of the polygonal approximation, but otherwise adversely affects the performance of polygonal approximation and reduction techniques. Consequently, only one of those values is presented in the tabular data at a class boundary.

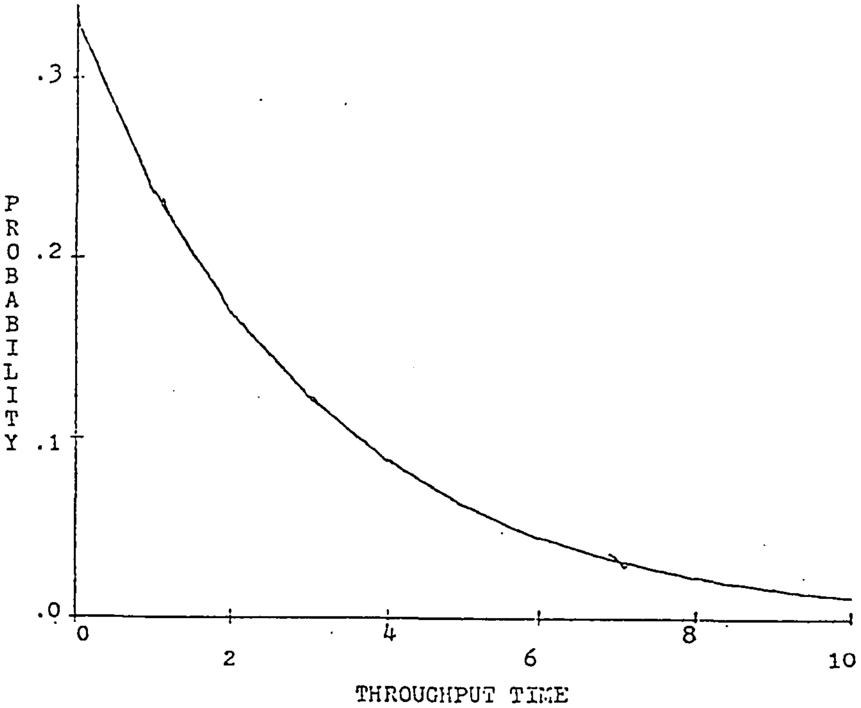
When a continuous distribution which is defined over the real numbers, such as a normal distribution, or over the positive reals, such as an exponential distribution or a gamma distribution, is used to model activity resource consumption, a truncated version defined on a finite domain must be used with polygonal approximation. Typically, truncation is effected by selecting the minimum and maximum values of the finite domain such that the area under the pdf between these values is at least 99% of the total area under the pdf. If a normal distribution is truncated such that the standard deviation of the truncation-approximation is one-sixth of the finite domain, then 99.7% of the area under the theoretical normal curve will be captured by the truncation-approximation. These truncation conventions have been employed in the results reported here.

Figures 31 and 32 illustrate that the essential character of a continuous, curvilinear distribution function is retained by polygonal approximation with SLR. For the truncated



T	ACTUAL	APPROX
0.0	0.0027	0.0016
0.4	0.0053	0.0058
0.8	0.0100	0.0100
1.0	0.0134	0.0109
1.4	0.0232	0.0244
1.8	0.0379	0.0378
2.0	0.0474	0.0445
2.4	0.0709	0.0722
2.8	0.1002	0.1000
3.0	0.1165	0.1169
3.4	0.1510	0.1507
3.8	0.1847	0.1846
4.0	0.1999	0.2056
4.4	0.2243	0.2217
4.8	0.2376	0.2378
5.0	0.2394	0.2457
5.4	0.2326	0.2297
5.8	0.2133	0.2138
6.0	0.1999	0.2014
6.4	0.1682	0.1676
6.8	0.1336	0.1338
7.0	0.1165	0.1139
7.4	0.0849	0.0861
7.8	0.0584	0.0583
8.0	0.0474	0.0446
8.4	0.0299	0.0311
8.8	0.0178	0.0176
9.0	0.0134	0.0121
9.4	0.0073	0.0079
9.8	0.0038	0.0037

Figure 31. Polygonal approximation of a normal distribution.



T	ACTUAL	APPROX
0.0	0.3333	0.3310
0.4	0.2917	0.2930
0.8	0.2553	0.2549
1.0	0.2388	0.2372
1.4	0.2090	0.2099
1.8	0.1829	0.1827
2.0	0.1711	0.1699
2.4	0.1498	0.1504
2.8	0.1311	0.1309
3.0	0.1226	0.1218
3.4	0.1073	0.1078
3.8	0.0939	0.0938
4.0	0.0879	0.0873
4.4	0.0769	0.0773
4.8	0.0673	0.0673
5.0	0.0630	0.0625
5.4	0.0551	0.0553
5.8	0.0482	0.0481
6.0	0.0451	0.0448
6.4	0.0395	0.0396
6.8	0.0346	0.0345
7.0	0.0323	0.0321
7.4	0.0283	0.0284
7.8	0.0248	0.0247
8.0	0.0232	0.0230
8.4	0.0203	0.0204
8.8	0.0177	0.0177
9.0	0.0166	0.0165
9.4	0.0145	0.0146
9.8	0.0127	0.0127

Figure 32. Polygonal approximation of an exponential distribution.

Table 13. Normal Distribution Approximation.

INTERVAL	LOWER LIMIT	UPPER LIMIT	B(O)	B(1)
1	0.0	1.0	.00160552	.01048282
2	1.0	2.0	-.02264429	.03358418
3	2.0	3.0	-.09430650	.06937860
4	3.0	4.0	-.13677374	.08456000
5	4.0	5.0	.04434678	.04031159
6	5.0	6.0	.44545077	-.03994642
7	6.0	7.0	.70844264	-.08450183
8	7.0	8.0	.60075794	-.06954927
9	8.0	9.0	.31456379	-.03374476
10	9.0	10.0	.10712047	-.01055497

Table 14. Exponential Distribution Approximation.

INTERVAL	LOWER LIMIT	UPPER LIMIT	B(0)	B(1)
1	0.0	1.0	.33101131	-.09510278
2	1.0	2.0	.30532417	-.06814418
3	2.0	3.0	.26760181	-.04882745
4	3.0	4.0	.22673172	-.03498647
5	4.0	5.0	.18752920	-.02506888
6	5.0	6.0	.15233265	-.01796254
7	6.0	7.0	.12202449	-.01287113
8	7.0	8.0	.09665428	-.00922222
9	8.0	9.0	.07586382	-.00660801
10	9.0	10.0	.05909387	-.00473487

normal distribution in Figure 31, the maximum of the absolute values of the deviations (MADV) between the pdf and the polygonal approximation is 0.0064; for the truncated exponential in Figure 32, 0.0023. In both cases, these MADVs occur at or near the peak/maximum value of the pdf, where it should be expected that the greatest difficulty would be encountered in approximating a curvilinear function with straight line segments. However, as the figures illustrate, the general shapes of both distributions are preserved under polygonal approximation, and the approximated and actual values of the pdfs are nearly identical. Similar results were achieved when polygonal approximation was applied to the uniform, triangular, gamma, and beta distributions.

4.2 Series-Parallel Reduction Operations

Validation of the application of series-parallel reduction operations to polygonal approximations of continuous activity resource consumption distributions (Sections 3.2.2 and 3.2.3 [above]) is illustrated with four representatives of the combinations tested.

4.2.1 Series Reduction

The first example is the series reduction (convolution) of two uniform distributions. Suppose:

$$f \sim \text{uniform } [4,8] \quad \text{and} \quad g \sim \text{uniform } [5,10]$$

$$f(t) = 0.25,\ 4 \le t \le 8 \quad \text{and} \quad g(t) = 0.20,\ 5 \le t \le 10$$

Then their convolution is:

$$f \oplus g = \int f(\tau) g(t-\tau)\, d\tau$$

$$= \int_{\text{lower limit}}^{\text{upper limit}} (0.25)(0.20)\, d\tau$$

$$= \left[0.05\tau\right]_{\text{lower limit}}^{\text{upper limit}}$$

The class boundary points of the theoretical solution are the $bf_i + bg_j$: 9, 13, 14, and 18; so, the convolution integral must be evaluated over the intervals: [9,13], [13,14], and [14,18]. From Equation (5), the upper and lower limits of integration are:

Interval	Lower limit	Upper limit
[9,13]	1	$t-5$
[13,14]	4	8
[14,18]	$t-10$	8

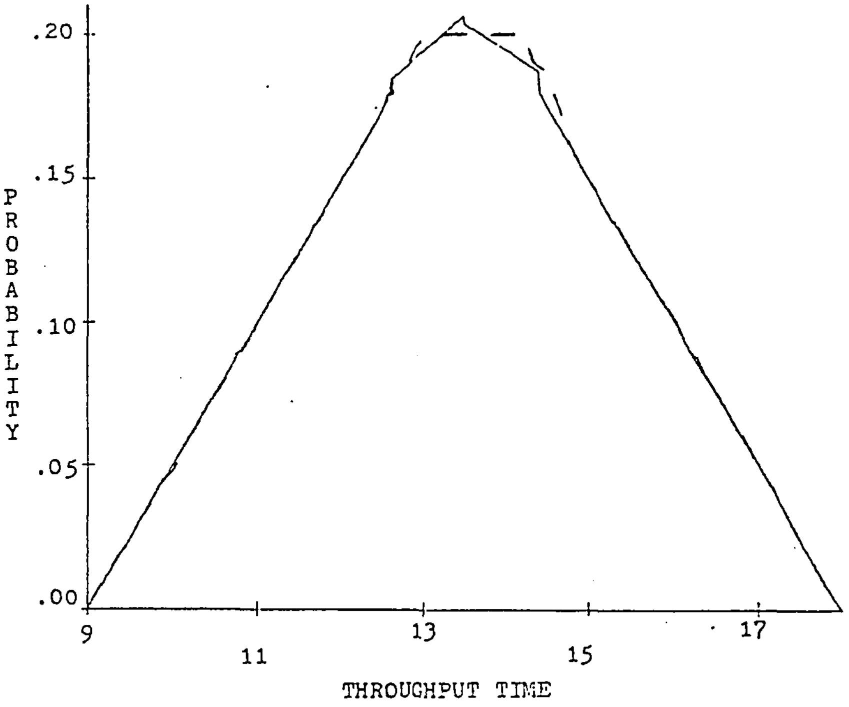
The evaluated convolution integral yields the following final form of the series reduction:

Interval	$(f \oplus g)(t)$
[9,13]	$0.05t - 0.45$
[13,14]	0.20
[14,18]	$-0.05t + 0.90$

Figure 33 depicts the pdf of the convolution and its polygonal approximation, which is defined on 10 classes over [9,18] (class width = 0.9). Table 15 presents a comparison of the first and second moments of the series reduction as calculated from the theoretical convolution, as obtained from a Monte Carlo simulation (3,000 replications), and as calculated with grouped data statistics from the polygonal approximation (labeled PART). Figure 33 shows, again, that polygonal approximation experiences its greatest error near the peak of the distribution, but that the errors are still small (the MADV is 0.0900 near the peak) and the shape of the convolution distribution is maintained.

Table 15. Moments of the Series Reduction of Two Uniform Distributions.

Source	Mean	Standard Deviation
Theoretical	13.500	1.850
Simulation	13.487	1.853
PART	13.501	1.847



T	ACTUAL	APPROX
9.0	0.0000	0.0000
9.5	0.0270	0.0270
9.9	0.0450	0.0450
10.1	0.0540	0.0540
10.4	0.0720	0.0720
10.8	0.0900	0.0900
11.0	0.0990	0.0990
11.3	0.1170	0.1170
11.7	0.1350	0.1350
11.9	0.1440	0.1440
12.2	0.1620	0.1620
12.6	0.1800	0.1800
12.8	0.1890	0.1891
13.1	0.2000	0.1977
13.5	0.2000	0.2063
13.7	0.2000	0.2005
14.0	0.2000	0.1941
14.4	0.1890	0.1877
14.6	0.1800	0.1710
14.9	0.1530	0.1530
15.3	0.1350	0.1350
15.5	0.1260	0.1260
15.8	0.1080	0.1080
16.2	0.0900	0.0899
16.4	0.0810	0.0810
16.7	0.0630	0.0630
17.1	0.0450	0.0499
17.3	0.0360	0.0360
17.6	0.0180	0.0180
18.0	0.0000	0.0000

Figure 33. Series reduction of two uniform distributions.

The second example is the series reduction (convolution) of two normal distributions. Suppose:

$$f \sim \text{normal } (7, 2), \text{ truncated on } [1,13]$$

and

$$g \sim \text{normal } (13,1), \text{ truncated on } [10,16]$$

The normal distribution is auto-reproductive under convolution (Parzen, 1960). The parameters of the convolution of two normal distributions are:

$$\mu = \mu_f + \mu_g$$
$$\sigma = \sqrt{\sigma_f^2 + \sigma_g^2}$$
$$\text{minimum} = \min f + \min g$$
$$\text{maximum} = \max f + \max g$$

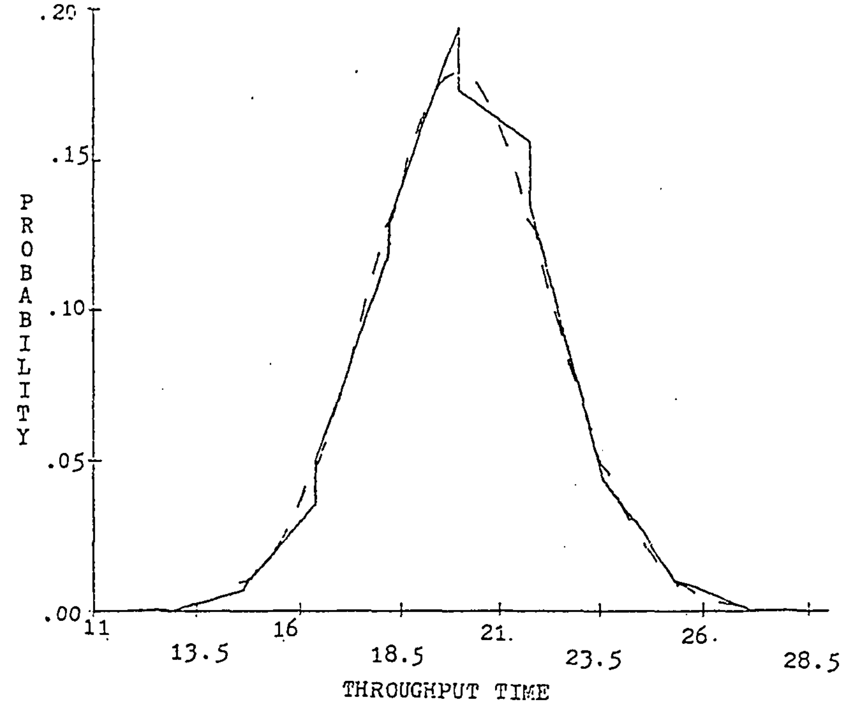
and the final form of this example convolution is:

$$f \oplus g \sim \text{Normal } (20, 2.236), \text{ truncated on } [11,29]$$

Figure 34 depicts the pdf of the convolution and its polygonal approximation, which is defined on 10 classes over [11,29] (class width = 1.8). Table 16 presents a comparison of the first and second moments of the series reduction obtained by the same methods as the data in Table 15. In Figure 34, the straight line segments appear to only roughly approximate the pdf of the convolution in the region near the peak. The polygonal approximation exhibits a spiking feature as it overshoots the peak, just before it falls back to a value just below the peak. Spiking occurs whenever a distribution has a peak, and the effect is the greatest when a class boundary point falls at or very near the peak. However, the MADV for this convolution of two normal distributions, at the peak, is only 0.0158.

Table 16. Moments of the Series Reduction of Two Normal Distributions.

Source	Mean	Standard Deviation
Theoretical	20.000	2.236
Simulation	20.020	2.222
PART	20.085	2.208



T	ACTUAL	APPROX
11.0	0.0001	0.0000
12.2	0.0004	0.0003
12.8	0.0010	0.0000
14.0	0.0049	0.0046
14.6	0.0097	0.0081
15.2	0.0178	0.0174
15.8	0.0306	0.0267
16.4	0.0488	0.0359
16.7	0.0600	0.0617
17.3	0.0861	0.0840
17.9	0.1148	0.1064
18.2	0.1290	0.1304
18.8	0.1545	0.1514
19.4	0.1721	0.1723
20.0	0.1784	0.1933
20.3	0.1768	0.1696
20.9	0.1645	0.1639
21.5	0.1425	0.1582
21.8	0.1290	0.1347
22.4	0.1003	0.1043
23.0	0.0725	0.0739
23.6	0.0488	0.0435
23.9	0.0390	0.0381
24.5	0.0235	0.0264
25.1	0.0132	0.0146
25.4	0.0096	0.0102
26.3	0.0034	0.0051
27.2	0.0011	0.0000
27.5	0.0006	0.0005
28.7	0.0001	0.0001

Figure 34. Series reduction of two normal distributions.

The third example is the series reduction (convolution) of two exponential distributions. Suppose:

$$f \sim \text{exponential } (\mu = 3), \text{ shifted onto } [1,10]$$

and

$$g \sim \text{exponential } (\mu = 25), \text{ shifted onto } [20,43]$$

Then, after shifting:

$$f(t) = \tfrac{1}{2}e^{-\frac{1}{2}(t-1)},\ 1 \le t \le 10 \quad \text{and} \quad g(t) = \tfrac{1}{5}e^{-\frac{1}{5}(t-20)},\ 20 \le t \le 43$$

and their convolution is:

$$f \oplus g = \int f(\tau)g(t-\tau)\,d\tau$$

$$= \int_{\text{lower limit}}^{\text{upper limit}} \tfrac{1}{10}e^{-0.3\tau-0.2t+4.5}\,d\tau$$

$$= \left[-\tfrac{1}{3}e^{-0.3\tau-0.2t+4.5}\right]_{\text{lower limit}}^{\text{upper limit}}$$

The class boundary points of the theoretical solution are the $bf_i + bg_j$: 21, 30, 44, and 53; so, the convolution integral must be evaluated over the intervals: [21,30], [30,44], and [44,53]. From Equation (5), the upper and lower limits of integration are:

Interval	Lower limit	Upper limit
[21,30]	1	$t-20$
[30,44]	1	10
[44,53]	$t-43$	10

The evaluated convolution integral yields the following final form of the series reduction:

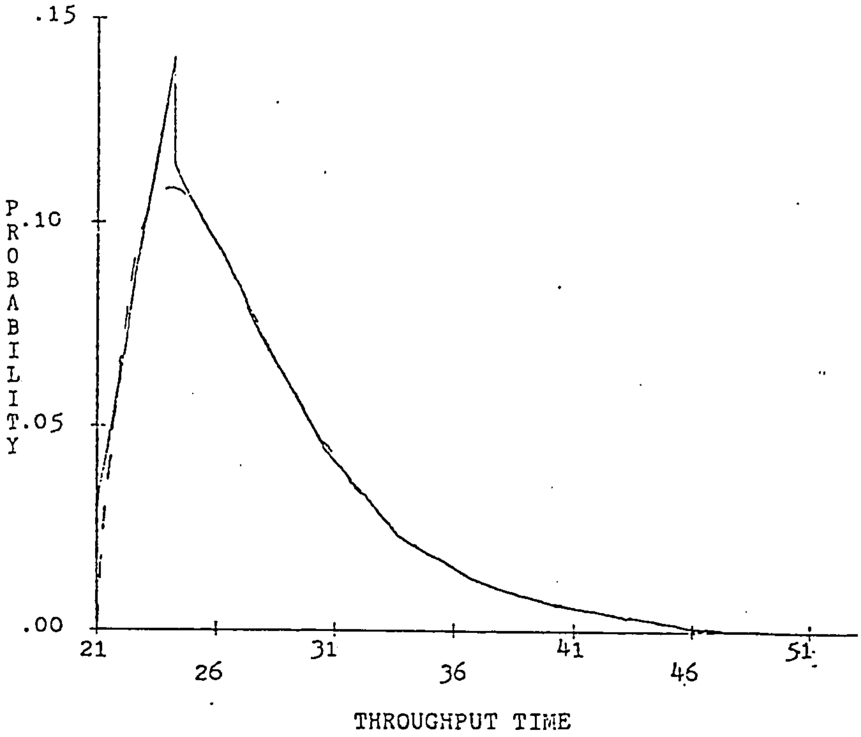
Interval	$(f \oplus g)(t)$
[21,30]	$-\frac{1}{3}e^{-0.2t+4.5}\left(e^{-0.3(t-20)} - e^{-0.3}\right)$
[30,44]	$-\frac{1}{3}e^{-0.2t+4.5}\left(e^{-3} - e^{-0.3}\right)$
[44,53]	$-\frac{1}{3}e^{-0.2t+4.5}\left(e^{-3} - e^{-0.3(t-43)}\right)$

Figure 35 depicts the pdf of the convolution and its polygonal approximation, which is defined on 10 classes over [21,53] (class width = 3.2). Table 17 presents a comparison of the first and second moments of the series reduction obtained by the same methods as the data in Table 15. In Figure 35, the straight line segments again appear to only roughly approximate the pdf of the convolution in the region near the peak; the MADV, at the class boundary 24.2, is 0.0320. The polygonal approximation again exhibits a spiking feature as it overshoots the peak, just before it falls back to a value which is, this time, just above the peak. Unfortunately, this type of "extreme" spiking is characteristic of the series reduction of any pair of exponential distributions, because of the steep slopes of the convolution pdf on either side of the peak. This is the same effect which motivated Dodin (1980,1985a) to employ the equal probabilities method only in the discretization of exponential distributions, while employing the equal intervals method for all other continuous distributions. Because of this effect, the performance of polygonal approximation and reduction techniques degrades as the percentage of exponential activity resource consumption distributions increases in project management networks. In such cases, to maintain accuracy, the number of classes in the partition of the domain of each distribution must be increased in the polygonal approximation.

When series-parallel reduction operations were applied to polygonal approximations of triangular, gamma, and beta distributions, the behavior was similar to that experienced with the normal. Although some spiking occurred near all distribution peaks, the "extreme" spiking effect was restricted to the convolution of exponentials.

Table 17. Moments of the Series Reduction of Two Exponential Distributions.

Source	Mean	Standard Deviation
Theoretical	27.672	4.796
Simulation	27.975	5.110
PART	27.491	4.754



T	ACTUAL	APPROX
21.0	0.0000	0.0310
21.8	0.0606	0.0584
22.6	0.0923	0.0857
23.4	0.1059	0.1131
24.2	0.1085	0.1404
24.6	0.1073	0.1100
25.4	0.1013	0.1017
26.2	0.0931	0.0933
27.0	0.0838	0.0850
27.4	0.0791	0.0786
29.0	0.0612	0.0614
30.6	0.0456	0.0442
31.4	0.0388	0.0389
33.0	0.0282	0.0283
34.6	0.0205	0.0204
36.2	0.0149	0.0150
37.0	0.0127	0.0123
38.6	0.0092	0.0093
40.2	0.0067	0.0063
41.0	0.0057	0.0057
42.6	0.0041	0.0041
43.4	0.0035	0.0035
45.0	0.0018	0.0020
46.6	0.0008	0.0005
47.4	0.0005	0.0005
49.0	0.0002	0.0002
49.8	0.0001	0.0001
51.4	0.0000	0.0001
53.0	0.0000	0.0000

Figure 35. Series reduction of two exponential distributions.

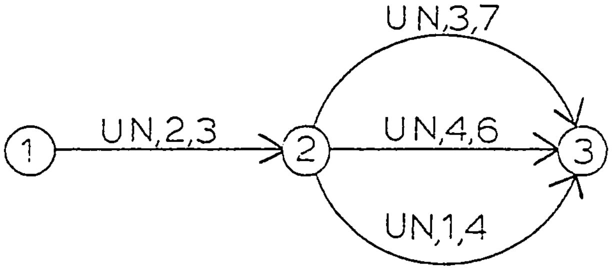


Figure 36. Network with three parallel activities.

4.2.2 Parallel Reduction

The fourth example is the reduction of the simple network in Figure 36, which involves the parallel reduction (maximum) of three uniform distributions. Suppose:

$$f_{12} \sim \text{uniform } [2,3] \quad \text{and} \quad \begin{cases} f_{23_1} = f_1 \sim \text{uniform } [3,7] \\ f_{23_2} = f_2 \sim \text{uniform } [4,6] \\ f_{23_3} = f_3 \sim \text{uniform } [1,4] \end{cases}$$

Then the cdfs of the three parallel activities are:

$$F_1(t) = \begin{cases} 0\ ;\ t < 3 \\ \dfrac{t-3}{4}\ ;\ 3 \le t \le 7 \\ 1\ ;\ t > 7 \end{cases}$$

$$F_2(t) = \begin{cases} 0\ ;\ t < 4 \\ \dfrac{t-4}{2}\ ;\ 4 \le t \le 6 \\ 1\ ;\ t > 6 \end{cases}$$

$$F_3(t) = \begin{cases} 0 \; ; t < 1 \\ \dfrac{t-1}{3} \; ; 1 \le t \le 4 \\ 1 \; ; t > 4 \end{cases}$$

and of their maximum is:

$$F_{\max(f_1, f_2, f_3)} = F_1 * F_2 * F_3$$

The lower bound of the domain of the maximum is:

$$[\max(f_1, f_2, f_3)]_{\text{lower bound}} = \max(f_1, f_2, f_3 \text{ - lower bounds}) = 4$$

and the class boundary points of the theoretical solution are the $\{bf_1\text{'s}\} \cup \{bf_2\text{'s}\} \cup \{bf_3\text{'s}\}$: 4, 6, and 7; so, the maximum must be evaluated over the intervals: [4,6] and [6,7].

$$F_{123}(t) = F_{\max(f_1, f_2, f_3)}(t) = \begin{cases} 0.125t^2 - 0.875t + 1.5 \; ; \; 4 \le t \le 6 \\ 0.25t - 0.75 \; ; \; 6 \le t \le 7 \end{cases}$$

$$f_{123}(t) = f_{\max(f_1, f_2, f_3)}(t) = \begin{cases} 0.25t - 0.875 ; \; 4 \le t \le 6 \\ 0.25 \; ; \; 6 \le t \le 7 \end{cases}$$

Figure 37 depicts the pdf of the maximum and its polygonal approximation, which is defined on 10 classes over [4,7] (class width = 0.3). In this figure, the solid line is the theoretical solution, and the histogram is the polygonal approximation. (The histogram is an output from a PART algorithm.) The polygonal approximation is identical to the theoretical solution except in a small neighborhood of the jump discontinuity at $t = 6$, which is contained in the [5.8,6.1] class of the approximation. As the histogram illustrates, the polygonal approximation is representing the jump discontinuity as a steeply negatively sloped line segment over the [5.8,6.1] class. Had the jump discontinuity fallen on a class boundary point of the polygonal approximation, there would be complete agreement between the theoretical solution and the approximation of the maximum.

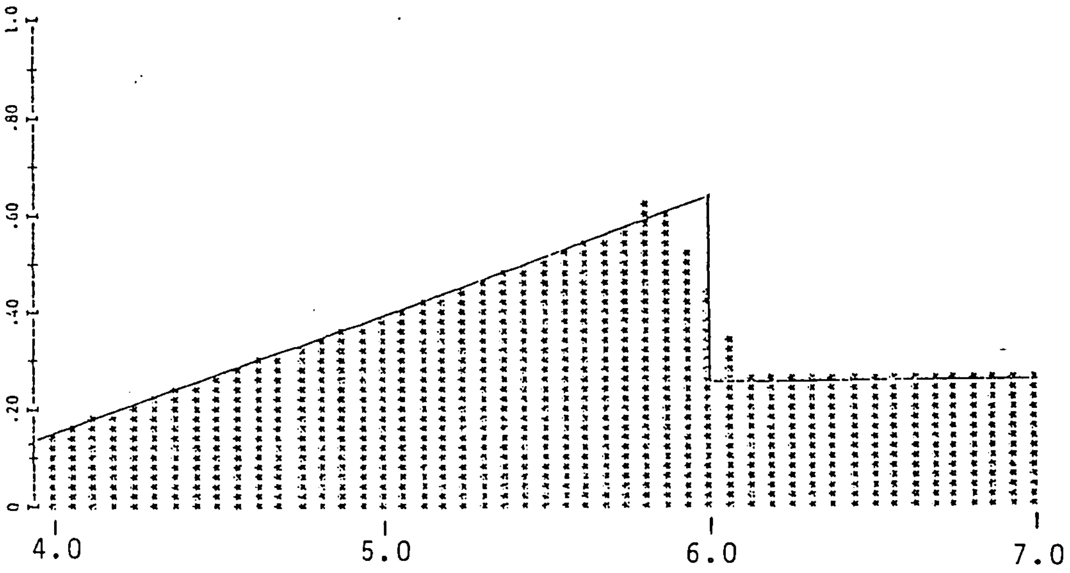
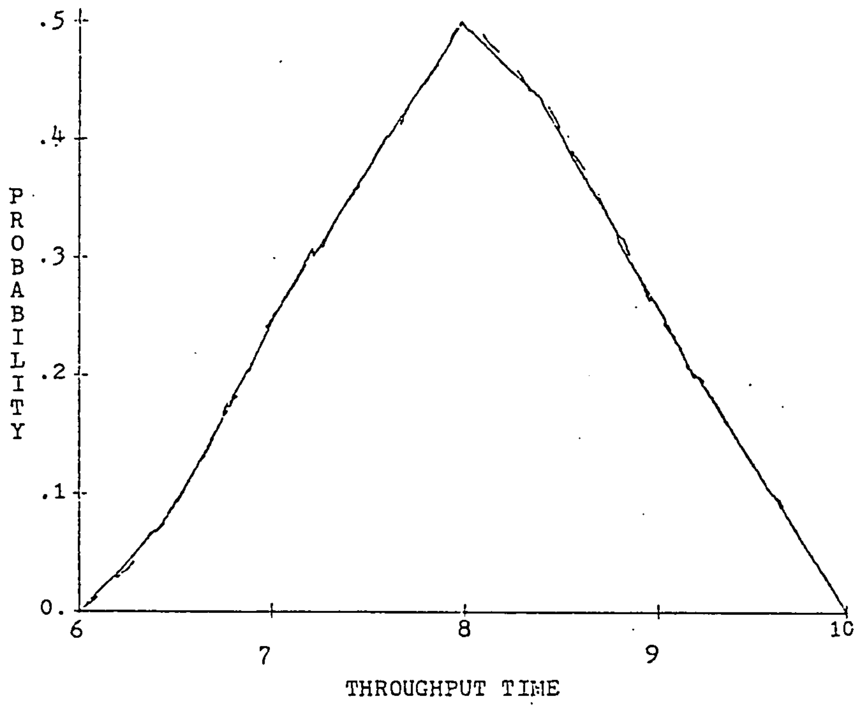


Figure 37. Parallel reduction of three uniform distributions (intermediate network solution).

The throughput distribution of the network is obtained from the series reduction (convolution) of f_{12} and f_{123}:

$$f_{\text{throughput}} = f_{12} \oplus f_{123} = \begin{cases} 0.125t^2 - 1375t + 0.375 & ; 6 \le t \le 7 \\ 0.25t - 1.5 & ; 7 \le t \le 8 \\ -0.125t^2 + 1.875t - 6.5 & ; 8 \le t \le 9 \\ -0.25t + 2.5 & ; 9 \le t \le 10 \end{cases}$$

Figure 38 depicts the pdf of the throughput distribution and its polygonal approximation, which is defined on 10 classes over [6,10] (class width = 0.4). Table 18 presents comparisons of the first and second moments of the parallel reduction (intermediate network solution) and the throughput distribution obtained by the same means as the data in Table 15. The polygonal approximation is a close fit to the theoretical pdf. The greatest disagreement is again near the peak, but even there the errors are small. The MADV is 0.0079 near the peak.



T	ACTUAL	APPROX
6.0	0.0000	0.0000
6.2	0.0300	0.0342
6.4	0.0700	0.0684
6.5	0.0938	0.0951
6.7	0.1488	0.1493
6.8	0.1800	0.1837
7.0	0.2500	0.2453
7.2	0.3000	0.3079
7.3	0.3250	0.3265
7.5	0.3750	0.3767
7.6	0.4000	0.4032
7.8	0.4500	0.4511
8.0	0.5000	0.4990
8.1	0.4863	0.4793
8.3	0.4513	0.4489
8.4	0.4300	0.4292
8.6	0.3800	0.3742
8.8	0.3200	0.3205
8.9	0.2863	0.2864
9.1	0.2250	0.2278
9.2	0.2000	0.2007
9.4	0.1500	0.1506
9.6	0.1000	0.1005
9.7	0.0750	0.0753
9.9	0.0250	0.0251

Figure 38. Reduction of a network with three parallel activities (throughput distribution).

Table 18. Moments of the Reduction of a Network with Three Parallel Activities.

Source	Mean	Standard Deviation
Intermediate network solution:		
Theoretical	5.500	0.755
Simulation	5.541	0.722
PART	5.524	0.732
Throughput distribution:		
Theoretical	8.020	0.800
Simulation	8.039	0.779
PART	8.039	0.791

Similar results were achieved when series-parallel reduction operations were applied to other combinations of polygonal approximations of continuous distributions, including the uniform, triangular, normal, exponential, gamma, and beta distributions. The polygonal approximation results were close to the theoretical solutions, where they could be obtained, and the simulation results, as measured by the MADVs between the approximations and the theoretical/simulation pdfs and the differences between their first and second moments. The MADVs occurred at or near the peaks/maximum values of the pdfs, where, again, it should be expected that the greatest difficulty would be encountered in approximating curvilinear functions with straight line segments. Spiking occurred at and in the vicinity of distribution peaks, but "extreme" spiking was limited to situations involving the combinations of exponential distributions. When graphs of theoretical solutions and histograms of polygonal approximation and simulation results were compared, the general shapes of the distributions were seen to be the same, and the approximated and actual/ simulated values of the pdfs were nearly identical almost everywhere.

The collective results of these tests validated polygonal approximation of continuous activity resource consumption distributions and series-parallel reduction operations based on polygonal approximation, when employed in the reduction of simple networks. The next phase of the validation process involved demonstrating the performance of polygonal approximation and reduction algorithms for general project management networks.

4.3 PART Algorithm Performance Based on Throughput Distributions

In this section, validation of the PART algorithms using "independent multiple arcs" approximation based on the first-available-arc-with-Property 1 method and sequential approximation to obtain throughput distributions (Section 3.3 [above]) is discussed. The validation was accomplished in three steps:

1. Performance of the algorithms against selected test networks.
2. Comparison of the performance of the algorithms against the reported performance of competing approximation procedures.
3. Performance of the algorithms against a set of "strongly randomized" test networks constructed in accordance with the experimental design for algorithm testing discussed in Section 3.5.1 [above].

For all of the test networks against which the algorithms were exercised, the results of extensive Monte Carlo simulations of these networks (Section 3.5.2 [above]) were taken as the "true" throughput distributions, since the actual solutions cannot be obtained. Example data inputs and outputs for the programs in Appendices A, B, D, and E are included here.

4.3.1 Performance Against Selected Test Networks

Although a small number of test networks (cases) are reported in the literature, they are not necessarily either realistic in terms of project management applications or representative of the challenges which a network approximation and reduction procedure may face, and consequently "strongly randomized" test networks should be employed in algorithm validation (Dodin, 1980 and 1985a; Hagstrom, 1990). (Table 6 contains references to 19 of these networks.) Nonetheless, if properly selected, previously reported test networks can illustrate important characteristics of algorithm performance. Three previously reported test networks were chosen on functional considerations for PART algorithm validation.

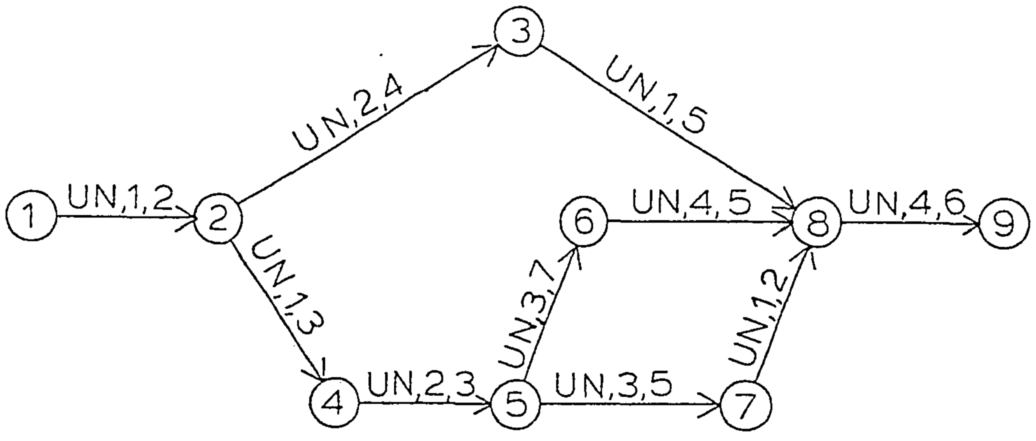
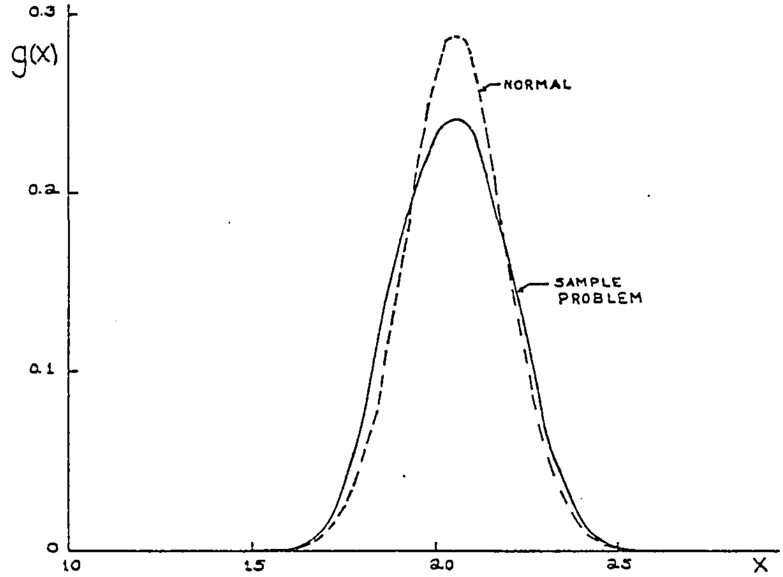


Figure 39. Martin's sample problem.
[Adapted from Martin (1965)]

Martin (1965) used the simple network in Figure 39 with nine nodes and eight arcs and all-uniformly distributed times to demonstrate the concept of a series-parallel algorithm for the exact analytic solution of acyclic, directed networks (Section 3.1 [above]). Because of the problem of "exploding coefficient storage," Martin's solution was a piecewise-defined polynomial throughput time density with terms of as-high-as the fifth degree. Martin's solution is shown in Part (a) of Figure 40; a normal distribution with the same first and second moments as Martin's solution is shown in Part (b) of Figure 40, superimposed over the graph of his solution. Figure 41 depicts the pdf of Martin's solution and its polygonal approximation obtained from the PART algorithm using "independent multiple arcs" approximation. Table 19 presents a comparison of the first and second moments of Martin's network as calculated from Martin's solution, as obtained from a Monte Carlo simulation (10,000 replications), and as calculated from grouped data statistics from the polygonal approximations of the throughput distributions developed by the PART algorithms using "independent multiple arcs" approximation (labeled PART-ind) and sequential approximation (labeled PART-seq), and the MADVs between the simulation and PART algorithm approximations of the throughput distribution.

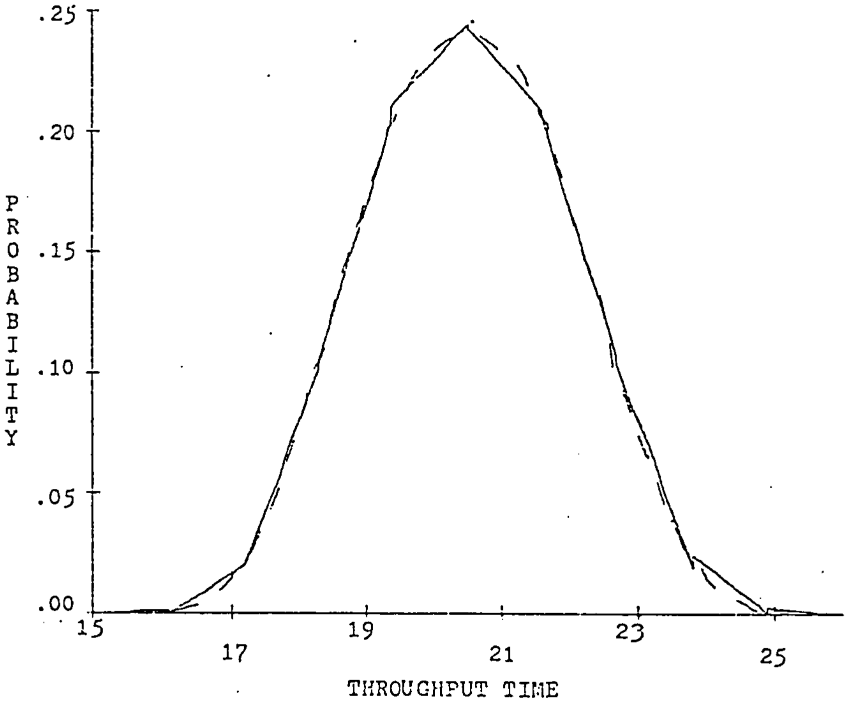
$$
\begin{aligned}
g(x) &= 0 && x \leqq 15,\\
&= -395.50789 + 131.83594x - 17.578125x^2 + 1.171875x^3\\
&\quad -0.039062499x^4 + 0.0005208334x^5 && 15 \leqq x \leqq 16,\\
&= 1242.8923 - 380.16405x + 46.421873x^2 - 2.8281249x^3\\
&\quad +0.085937497x^4 - 0.0010416667x^5 && 16 \leqq x \leqq 17,\\
&= 503.38341 - 162.66145x + 20.833333x^2 - 1.3229167x^3\\
&\quad +0.041666667x^4 - 0.0005208334x^5 && 17 \leqq x \leqq 18,\\
&= -4417.3669 + 1204.2135x - 131.04166x^2 + 7.114583x^3\\
&\quad -0.19270832x^4 + 0.002083334x^5 && 18 \leqq x \leqq 19,\\
&= 3320.4433 - 832.05207x + 83.30208x^2 - 4.1666665x^3\\
&\quad +0.10416666x^4 - 0.0010416667x^5 && 19 \leqq x \leqq 20,\\
&= -12.891052 + 1.2812805x - 0.031254768x^2 && 20 \leqq x \leqq 21,\\
&= -4267.1633 + 1014.2031x - 96.499995x^2 + 4.5937498x^3\\
&\quad -0.10937499x^4 + 0.0010416667x^5 && 21 \leqq x \leqq 22,\\
&= 11837.938 - 2646.0468x + 236.24999x^2 - 10.53125x^3\\
&\quad +0.23437499x^4 - 0.002083334x^5 && 22 \leqq x \leqq 23,\\
&= -4923.3717 + 997.7161x - 80.598956x^2 + 3.2447915x^3\\
&\quad -0.065104164x^4 + 0.0005208334x^5 && 23 \leqq x \leqq 24,\\
&= -9070.5724 + 1861.7161x - 152.59895x^2 + 6.2447913x^3\\
&\quad -0.12760416x^4 + 0.0010416667x^5 && 24 \leqq x \leqq 25,\\
&= 6188.2161 - 1190.0416x + 91.541663x^2 - 3.5208332x^3\\
&\quad +0.06770833x^4 - 0.0005208334x^5 && 25 \leqq x \leqq 26,\\
&= 0, && x \geqq 26.
\end{aligned}
$$

(a)



(b)

Figure 40. Martin's solution of Martin's sample problem. [Adapted from Martin (1965)]



T	ACTUAL	APPROX
15.0	0.0000	0.0000
15.5	0.0000	0.0006
15.9	0.0003	0.0012
16.1	0.0008	0.0000
16.6	0.0053	0.0091
17.0	0.0150	0.0164
17.2	0.0230	0.0209
17.7	0.0527	0.0575
18.1	0.0861	0.0868
18.3	0.1052	0.1052
18.8	0.1545	0.1506
19.2	0.1885	0.1869
19.4	0.2031	0.2106
19.9	0.2295	0.2258
20.3	0.2392	0.2380
20.5	0.2404	0.2424
21.0	0.2325	0.2270
21.4	0.2181	0.2148
21.6	0.2058	0.2042
22.1	0.1594	0.1606
22.5	0.1292	0.1258
22.7	0.0992	0.1027
23.2	0.0596	0.0649
23.6	0.0324	0.0346
23.8	0.0222	0.0244
24.3	0.0082	0.0133
24.7	0.0010	0.0044
24.9	0.0001	0.0022
25.4	0.0001	0.0012
25.8	0.0000	0.0004

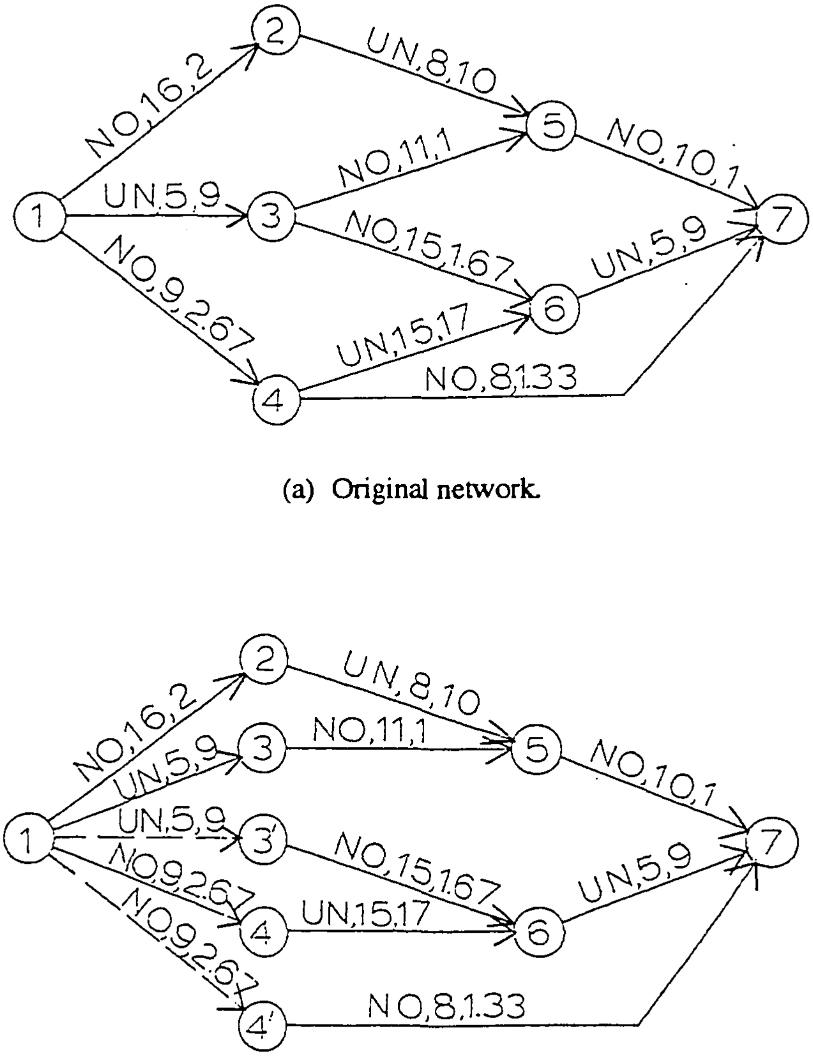
Figure 41. PART solution of Martin's sample problem.

Table 19. Comparisons of Solutions of Martin's Sample Problem.

Source	Mean	Standard Deviation	MADV
Martin's	20.524	1.380	n/a
Simulation	20.489	1.487	n/a
PART-ind	20.515	1.539	0.0167
PART-seq	20.518	1.537	0.0171

The polygonal approximations from both the PART-ind and PART-seq algorithms were very close to both Martin's solution and the simulation results. The maximum relative error in the estimation of the mean of the throughput distribution was 0.13%; in the estimation of the standard deviation, 3.50%. The MADVs, as the computed values of the K-S goodness-of-fit test statistic T, have prob values >20%.

The "conditional" network in Part (a) Figure 42 was adapted from Martin (1965) to illustrate the effect of dependencies among paths (Section 2.4.3 [above]). This network has two starting nodes for cross-connections between two paths, node 3 and node 4, and two terminal nodes for cross-connections also between two paths, nodes 5 and 6. Activities (1,3) and (1,4) are arcs with Property 1 (a activities), and activities (5,7) and (6,7) are arcs with Property 2 (b activities). Part (b) of Figure 42 depicts the completely reducible form of the original network which was created by the PART-ind algorithm using the first-available-arc-with-Property 1 method: first, node 3 was duplicated once, creating the new node $3'$, and activity (1,3) was "independently multiplied" (duplicated) once, creating activity $(1,3')$; then, node 4 was duplicated once, creating the new node $4'$, and activity (1,4) was "independently multiplied" (duplicated) once, creating activity $(1,4')$. The activity duration distributions were specified so that there were pairs of distributions with "high" variances in series, such as the equivalent activity of subnetwork $(1,3') \rightarrow (3',6)$, $(1,4) \rightarrow (4,6)$ and activity (6,7), and parallel subnetworks with near-maximally overlapping distributions, such as $(1,3') \rightarrow (3',6)$, $(1,4) \rightarrow (4,6)$ (Section 3.5.1 [above]).



(a) Original network.

(b) Completely reducible form.

Figure 42. "Conditional" network.
[Adapted from Martin (1965)]

Figure 43 shows the input data files for both the PART-ind and PART-seq programs for the "conditional" network: Part (a) shows the control data file (CONTROL.DAT), Part (b) shows the network data file (DATAN.DAT), and Part (c) shows the activity duration distributions data file (DATAH.DAT). The formats of these files are described in comment statements at the top of the code listings of the PART-ind and PART-seq programs in Appendices A and B, respectively. Figure 44 depicts the output from the PART-ind program under output option 7 (combination of output options 1 - 6), which includes: for the polygonal approximation of the network throughput distribution - its cdf, pdf, mean and standard deviation, the probabilities that the project throughput duration will fall in an interval ±1, 2, or 3 standard deviations from the mean, the number of nodes in the completely reducible network, and the number of cross-connections removed by the "independent multiple arcs" method; for the simulation approximation - its cdf, pdf, mean and standard deviation, and the probabilities that the project throughput duration will fall in an interval ±1, 2, or 3 standard deviations from the mean; and for the K-S goodness-of-fit test - the computed value of the K-S test statistic (labeled D-MAX), selected K-S test critical values, and the conclusion of the goodness-of-fit hypothesis test at the 5% level of statistical significance. Figure 45 depicts the output from the PART-seq program also under option 7, which is similar to the PART-ind output, except that there is no report of the number of nodes in the completely reducible network or the number of cross-connections removed, since sequential approximation reduces a network without the "independent multiplication" (duplication) of arcs. Additionally, the PART-seq program permits any node to be designated on an Output Critical List (OCL) for output report of the throughput distribution through that node. Node 5 was designated on the OCL for the PART-seq reduction of the "conditional" network, so there is a report for both the polygonal approximation and the simulation approximation of the throughput distribution through node 5 in the output in Figure 45.

```
007 0010 7 10000
```

(a) Control data file.

```
1 2 3 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 00 0 3
2 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 00 1 1
3 5 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 00 1 2
4 6 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 00 1 2
5 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 01 2 1
6 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 00 2 1
7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 01 3 0
```

(b) Network data file.

```
01 01 2 16.00000 0.000000 10.00000 22.00000
01 02 6 0.250000 0.000000 5.000000 9.000000
01 03 2 9.000000 0.000000 1.000000 17.00000
02 01 6 0.500000 0.000000 8.000000 10.00000
03 01 2 11.00000 0.000000 8.000000 14.00000
03 02 2 15.00000 0.000000 10.00000 20.00000
04 01 6 0.500000 0.000000 15.00000 17.00000
04 02 2 8.000000 0.000000 4.000000 12.00000
05 01 2 10.00000 0.000000 7.000000 13.00000
06 01 6 0.250000 0.000000 5.000000 9.000000
```

(c) Activity duration distributions data file.

Figure 43. PART-ind and PART-seq program input.

```
THE POLYGONAL APPROXIMATION OF THE TIME DISTRIBUTION THROUGH THE PROJECT
IS:

INTERVAL  1    LOWER LIMIT =   25.00   UPPER LIMIT =   27.00

               X = (  -.00005277) + (   .00000211) T

INTERVAL  2    LOWER LIMIT =   27.00   UPPER LIMIT =   29.00

               X = (  -.00908850) + (   .00033661) T

INTERVAL  3    LOWER LIMIT =   29.00   UPPER LIMIT =   31.00

               X = (  -.20893960) + (   .00720481) T

INTERVAL  4    LOWER LIMIT =   31.00   UPPER LIMIT =   33.00

               X = ( -1.22777647) + (   .03999845) T

INTERVAL  5    LOWER LIMIT =   33.00   UPPER LIMIT =   35.00

               X = ( -1.52304873) + (   .04921934) T

INTERVAL  6    LOWER LIMIT =   35.00   UPPER LIMIT =   37.00

               X = (  1.04511179) + (  -.02417275) T

INTERVAL  7    LOWER LIMIT =   37.00   UPPER LIMIT =   39.00

               X = (  2.10435497) + (  -.05298885) T

INTERVAL  8    LOWER LIMIT =   39.00   UPPER LIMIT =   41.00

               X = (   .82898677) + (  -.02018400) T

INTERVAL  9    LOWER LIMIT =   41.00   UPPER LIMIT =   43.00

               X = (   .09095823) + (  -.00209500) T

INTERVAL 10    LOWER LIMIT =   43.00   UPPER LIMIT =   45.00

               X = (   .00610929) + (  -.00013576) T
```

CUMULATIVE DISTRIBUTION FUNCTION

T	F(T)
25.000	.00000000
25.400	.00000017
25.800	.00000067
26.200	.00000152
26.600	.00000270
27.000	.00000380
27.400	.00003112
27.800	.00011182
28.200	.00024632
28.600	.00043462
29.000	.00060948
29.400	.00125251
29.800	.00297983
30.200	.00585872
30.600	.00988915
31.000	.01484796
31.400	.02313267
31.800	.03758719
32.200	.05843471
32.600	.08567522
33.000	.12020949
33.400	.16367516
33.800	.21590836
34.200	.27600834
34.600	.34397510
35.000	.41975245
35.400	.49741904
35.800	.57116589
36.200	.64104918
36.600	.70706891
37.000	.76853057
37.400	.82243676
37.800	.86717915
38.200	.90345229
38.600	.93125616
39.000	.95099245
39.400	.96568446
39.800	.97755212
40.200	.98619376
40.600	.99160936
41.000	.99416057
41.400	.99565460
41.800	.99717541
42.200	.99836137
42.600	.99921249
43.000	.99966867
43.400	.99982641
43.800	.99990235
44.200	.99995660
44.600	.99998915
45.000	1.00000000

```
                PROBABILITY DENSITY FUNCTION

      0        .05       .10       .15       .20       .25
      I----+----I----+----I----+----I----+----I----+----I
25.000  *
25.400  *
25.800  *
26.200  *
26.600  *
27.000  *
27.400  *
27.800  *
28.200  *
28.600  *
29.000  *
29.400  **
29.800  **
30.200  ***
30.600  ***
31.000  ****
31.400  *******
31.800  **********
32.200  *************
32.600  ****************
33.000  ********************
33.400  *************************
33.800  *****************************
34.200  *********************************
34.600  *************************************
35.000  *****************************************
35.400  ***************************************
35.800  *************************************
36.200  ***********************************
36.600  *********************************
37.000  ******************************
37.400  **************************
37.800  *********************
38.200  *****************
38.600  *************
39.000  *********
39.400  ********
39.800  ******
40.200  *****
40.600  ***
41.000  **
41.400  **
41.800  **
42.200  **
42.600  *
43.000  *
43.400  *
43.800  *
44.200  *
44.600  *
45.000  *
```

```
                 EXPECTED VALUE OF T     =  35.46154927
                 STANDARD DEVIATION OF T =   2.11260360

         THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
            33.349 TIME UNITS AND   37.574 TIME UNITS IS ABOUT 68.24 %.

         THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
            31.236 TIME UNITS AND   39.687 TIME UNITS IS ABOUT 95.44 %.

         THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
            29.124 TIME UNITS AND   41.799 TIME UNITS IS ABOUT 99.73 %.

         THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
            27.011 TIME UNITS AND   43.912 TIME UNITS IS ABOUT 99.99 %.


         THE NUMBER OF NODES IN THE FINAL NETWORK WAS   9.

         THE NUMBER OF CROSS-CONNECTIONS REDUCED WAS  2.
```

```
THE SIMULATED TIME DISTRIBUTION THROUGH THE PROJECT IS:

                CUMULATIVE DISTRIBUTION FUNCTION

                      T              F(T)
                   25.000        .00000000
                   25.400        .00000000
                   25.800        .00000000
                   26.200        .00000000
                   26.600        .00000000
                   27.000        .00000000
                   27.400        .00010000
                   27.800        .00010000
                   28.200        .00010000
                   28.600        .00020000
                   29.000        .00040000
                   29.400        .00100000
                   29.800        .00260000
                   30.200        .00450000
                   30.600        .00770000
                   31.000        .01330000
                   31.400        .02230000
                   31.800        .03670000
                   32.200        .05550000
                   32.600        .08360000
                   33.000        .11830000
                   33.400        .16190000
                   33.800        .21770000
                   34.200        .28090000
                   34.600        .35150000
                   35.000        .42580000
                   35.400        .49760000
                   35.800        .57680000
                   36.200        .64780000
                   36.600        .71560000
                   37.000        .77500000
                   37.400        .83020000
                   37.800        .86980000
                   38.200        .90640000
                   38.600        .93280000
                   39.000        .95400000
                   39.400        .97070000
                   39.800        .98050000
                   40.200        .98870000
                   40.600        .99340000
                   41.000        .99640000
                   41.400        .99840000
                   41.800        .99930000
                   42.200        .99980000
                   42.600        .99990000
                   43.000       1.00000000
                   43.400       1.00000000
                   43.800       1.00000000
                   44.200       1.00000000
                   44.600       1.00000000
                   45.000       1.00000000
```

```
                 SIMULATION FREQUENCY HISTOGRAM

          0         .02       .04       .06       .08       .10
          I----+----I----+----I----+----I----+----I----+----I
25.200    *
25.600    *
26.000    *
26.400    *
26.800    *
27.200    *
27.600    *
28.000    *
28.400    *
28.800    *
29.200    *
29.600    **
30.000    **
30.400    ***
30.800    ****
31.200    ******
31.600    ********
32.000    **********
32.400    ***************
32.800    ******************
33.200    ***********************
33.600    *****************************
34.000    *********************************
34.400    ************************************
34.800    **************************************
35.200    *************************************
35.600    *****************************************
36.000    *************************************
36.400    ***********************************
36.800    *******************************
37.200    *****************************
37.600    *********************
38.000    *******************
38.400    **************
38.800    ************
39.200    *********
39.600    ******
40.000    *****
40.400    ***
40.800    ***
41.200    **
41.600    *
42.000    *
42.400    *
42.800    *
43.200    *
43.600    *
44.000    *
44.400    *
44.800    *
```

```
            EXPECTED VALUE OF T      =  35.43308000
            STANDARD DEVIATION OF T =   2.06821897


THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   33.365 TIME UNITS AND   37.501 TIME UNITS IS ABOUT 68.24 %.


THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   31.297 TIME UNITS AND   39.570 TIME UNITS IS ABOUT 95.44 %.


THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   29.228 TIME UNITS AND   41.638 TIME UNITS IS ABOUT 99.73 %.


THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   27.160 TIME UNITS AND   43.706 TIME UNITS IS ABOUT 99.99 %.


KOLMOGOROV-SMIRNOV ONE-SAMPLE TEST COMPARISON OF POLYGONAL APPROXIMATION
OF NETWORK THROUGHPUT DISTRIBUTION AND SIMULATED NETWORK THROUGHPUT
DISTRIBUTION:

K-S TEST STATISTIC D-MAX =  .0085

K-S CRITICAL VALUES:
              20 PERCENT =  .1517
              10 PERCENT =  .1731
               5 PERCENT =  .1921
               2 PERCENT =  .2146
               1 PERCENT =  .2302

FAIL TO REJECT THE NULL HYPOTHESIS THAT THE DISTRIBUTIONS ARE THE SAME
AT THE 5% LEVEL OF STATISTICAL SIGNIFICANCE.
```

Figure 44. PART-ind program output.

```
THE POLYGONAL APPROXIMATION OF THE TIME DISTRIBUTION THROUGH NODE  5 IS:

INTERVAL  1     LOWER LIMIT =    18.00   UPPER LIMIT =   19.40

                X = (  -.05021934) + (   .00278996) T

INTERVAL  2     LOWER LIMIT =    19.40   UPPER LIMIT =   20.80

                X = (  -.26933624) + (   .01404307) T

INTERVAL  3     LOWER LIMIT =    20.80   UPPER LIMIT =   22.20

                X = (  -.73713341) + (   .03654872) T

INTERVAL  4     LOWER LIMIT =    22.20   UPPER LIMIT =   23.60

                X = ( -1.14517485) + (   .05504311) T

INTERVAL  5     LOWER LIMIT =    23.60   UPPER LIMIT =   25.00

                X = (  -.57698953) + (   .03106648) T

INTERVAL  6     LOWER LIMIT =    25.00   UPPER LIMIT =   26.40

                X = (   .87408961) + (  -.02707143) T

INTERVAL  7     LOWER LIMIT =    26.40   UPPER LIMIT =   27.80

                X = (  1.64205667) + (  -.05631819) T

INTERVAL  8     LOWER LIMIT =    27.80   UPPER LIMIT =   29.20

                X = (  1.16601964) + (  -.03921378) T

INTERVAL  9     LOWER LIMIT =    29.20   UPPER LIMIT =   30.60

                X = (   .44160704) + (  -.01434302) T

INTERVAL 10     LOWER LIMIT =    30.60   UPPER LIMIT =   32.00

                X = (   .08799734) + (  -.00273934) T
```

CUMULATIVE DISTRIBUTION FUNCTION

T	F(T)
18.000	.00000000
18.280	.00010935
18.560	.00043739
18.840	.00098413
19.120	.00174957
19.400	.00267724
19.680	.00415175
19.960	.00667058
20.240	.01029021
20.520	.01501062
20.800	.02085426
21.080	.02872561
21.360	.03948433
21.640	.05310798
21.920	.06959657
22.200	.08912744
22.480	.11260280
22.760	.14057016
23.040	.17285218
23.320	.20944884
23.600	.25052371
23.880	.29530060
24.160	.34267625
24.440	.39248709
24.720	.44473314
25.000	.49924863
25.280	.55358916
25.560	.60564189
25.840	.65557257
26.120	.70338121
26.400	.74877754
26.680	.79032501
26.960	.82716760
27.240	.85959559
27.520	.88760898
27.800	.91117036
28.080	.93091271
28.360	.94754381
28.640	.96110107
28.920	.97158448
29.200	.97912099
29.480	.98481229
29.760	.98950622
30.040	.99307585
30.320	.99552117
30.600	.99694457
30.880	.99790321
31.160	.99874950
31.440	.99938106
31.720	.99979789
32.000	1.00000000

```
                 PROBABILITY DENSITY FUNCTION

       0        .05       .10       .15       .20       .25
       I----+----I----+----I----+----I----+----I----+----I
18.000  *
18.280  *
18.560  *
18.840  *
19.120  **
19.400  **
19.680  **
19.960  ***
20.240  ****
20.520  *****
20.800  ******
21.080  ********
21.360  **********
21.640  ************
21.920  **************
22.200  ****************
22.480  *******************
22.760  ***********************
23.040  **************************
23.320  *****************************
23.600  ********************************
23.880  **********************************
24.160  ************************************
24.440  *************************************
24.720  ***************************************
25.000  *****************************************
25.280  ***************************************
25.560  *************************************
25.840  ************************************
26.120  **********************************
26.400  ********************************
26.680  *****************************
26.960  **************************
27.240  ***********************
27.520  *******************
27.800  ****************
28.080  **************
28.360  ************
28.640  **********
28.920  *******
29.200  *****
29.480  *****
29.760  ****
30.040  ***
30.320  **
30.600  **
30.880  **
31.160  **
31.440  *
31.720  *
32.000  *
```

```
            EXPECTED VALUE OF T     =  25.00218256
            STANDARD DEVIATION OF T =   2.06795371

THE PROBABILITY OF NODE   5 THROUGHPUT TIME FALLING BETWEEN
   22.934 TIME UNITS AND  27.070 TIME UNITS IS ABOUT 68.24 %.

THE PROBABILITY OF NODE   5 THROUGHPUT TIME FALLING BETWEEN
   20.866 TIME UNITS AND  29.138 TIME UNITS IS ABOUT 95.44 %.

THE PROBABILITY OF NODE   5 THROUGHPUT TIME FALLING BETWEEN
   18.798 TIME UNITS AND  31.206 TIME UNITS IS ABOUT 99.73 %.

THE PROBABILITY OF NODE   5 THROUGHPUT TIME FALLING BETWEEN
   16.730 TIME UNITS AND  33.274 TIME UNITS IS ABOUT 99.99 %.
```

```
THE POLYGONAL APPROXIMATION OF THE TIME DISTRIBUTION THROUGH THE PROJECT
IS:

INTERVAL  1    LOWER LIMIT =   25.00   UPPER LIMIT =   27.00

               X = (  -.00005277) + (   .00000211) T

INTERVAL  2    LOWER LIMIT =   27.00   UPPER LIMIT =   29.00

               X = (  -.00908850) + (   .00033661) T

INTERVAL  3    LOWER LIMIT =   29.00   UPPER LIMIT =   31.00

               X = (  -.20893960) + (   .00720481) T

INTERVAL  4    LOWER LIMIT =   31.00   UPPER LIMIT =   33.00

               X = ( -1.22777647) + (   .03999845) T

INTERVAL  5    LOWER LIMIT =   33.00   UPPER LIMIT =   35.00

               X = ( -1.52304873) + (   .04921934) T

INTERVAL  6    LOWER LIMIT =   35.00   UPPER LIMIT =   37.00

               X = (  1.04511179) + (  -.02417275) T

INTERVAL  7    LOWER LIMIT =   37.00   UPPER LIMIT =   39.00

               X = (  2.10435497) + (  -.05298885) T

INTERVAL  8    LOWER LIMIT =   39.00   UPPER LIMIT =   41.00

               X = (   .82898677) + (  -.02018400) T

INTERVAL  9    LOWER LIMIT =   41.00   UPPER LIMIT =   43.00

               X = (   .09095823) + (  -.00209500) T

INTERVAL 10    LOWER LIMIT =   43.00   UPPER LIMIT =   45.00

               X = (   .00610929) + (  -.00013576) T
```

CUMULATIVE DISTRIBUTION FUNCTION

T	F(T)
25.000	.00000000
25.400	.00000017
25.800	.00000067
26.200	.00000152
26.600	.00000270
27.000	.00000380
27.400	.00003112
27.800	.00011182
28.200	.00024632
28.600	.00043462
29.000	.00060948
29.400	.00125251
29.800	.00297983
30.200	.00585872
30.600	.00988915
31.000	.01484796
31.400	.02313267
31.800	.03758719
32.200	.05843471
32.600	.08567522
33.000	.12020949
33.400	.16367516
33.800	.21590836
34.200	.27600834
34.600	.34397510
35.000	.41975245
35.400	.49741904
35.800	.57116589
36.200	.64104918
36.600	.70706891
37.000	.76853057
37.400	.82243676
37.800	.86717915
38.200	.90345229
38.600	.93125616
39.000	.95099245
39.400	.96568446
39.800	.97755212
40.200	.98619376
40.600	.99160936
41.000	.99416057
41.400	.99565460
41.800	.99717541
42.200	.99836137
42.600	.99921249
43.000	.99966867
43.400	.99982641
43.800	.99990235
44.200	.99995660
44.600	.99998915
45.000	1.00000000

```
                 PROBABILITY DENSITY FUNCTION

          0         .05       .10       .15       .20       .25
          I----+----I----+----I----+----I----+----I----+----I
   25.000  *
   25.400  *
   25.800  *
   26.200  *
   26.600  *
   27.000  *
   27.400  *
   27.800  *
   28.200  *
   28.600  *
   29.000  *
   29.400  **
   29.800  **
   30.200  ***
   30.600  ***
   31.000  ****
   31.400  *******
   31.800  **********
   32.200  *************
   32.600  ****************
   33.000  ********************
   33.400  *************************
   33.800  *****************************
   34.200  *********************************
   34.600  *************************************
   35.000  *****************************************
   35.400  ***************************************
   35.800  *************************************
   36.200  ***********************************
   36.600  *********************************
   37.000  ******************************
   37.400  **************************
   37.800  *********************
   38.200  *****************
   38.600  *************
   39.000  *********
   39.400  ********
   39.800  ******
   40.200  *****
   40.600  ***
   41.000  **
   41.400  **
   41.800  **
   42.200  **
   42.600  *
   43.000  *
   43.400  *
   43.800  *
   44.200  *
   44.600  *
   45.000  *
```

```
                EXPECTED VALUE OF T     =  35.46154927
                STANDARD DEVIATION OF T =   2.11260360

THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   33.349 TIME UNITS AND   37.574 TIME UNITS IS ABOUT 68.24 %.

THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   31.236 TIME UNITS AND   39.687 TIME UNITS IS ABOUT 95.44 %.

THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   29.124 TIME UNITS AND   41.799 TIME UNITS IS ABOUT 99.73 %.

THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   27.011 TIME UNITS AND   43.912 TIME UNITS IS ABOUT 99.99 %.
```

```
THE SIMULATED TIME DISTRIBUTION THROUGH NODE  5 IS:

              CUMULATIVE DISTRIBUTION FUNCTION

                    T                F(T)
                 18.000          .00000000
                 18.280          .00010000
                 18.560          .00030000
                 18.840          .00040000
                 19.120          .00060000
                 19.400          .00150000
                 19.680          .00320000
                 19.960          .00600000
                 20.240          .00920000
                 20.520          .01430000
                 20.800          .02080000
                 21.080          .03020000
                 21.360          .03920000
                 21.640          .05320000
                 21.920          .07220000
                 22.200          .09140000
                 22.480          .11570000
                 22.760          .14320000
                 23.040          .17210000
                 23.320          .20910000
                 23.600          .24800000
                 23.880          .29560000
                 24.160          .34340000
                 24.440          .39390000
                 24.720          .44980000
                 25.000          .50300000
                 25.280          .55830000
                 25.560          .60850000
                 25.840          .66300000
                 26.120          .70720000
                 26.400          .74970000
                 26.680          .79310000
                 26.960          .82930000
                 27.240          .86280000
                 27.520          .88860000
                 27.800          .91220000
                 28.080          .93290000
                 28.360          .94870000
                 28.640          .96170000
                 28.920          .97200000
                 29.200          .97990000
                 29.480          .98600000
                 29.760          .99110000
                 30.040          .99470000
                 30.320          .99610000
                 30.600          .99760000
                 30.880          .99890000
                 31.160          .99950000
                 31.440          .99970000
                 31.720          .99990000
                 32.000         1.00000000
```

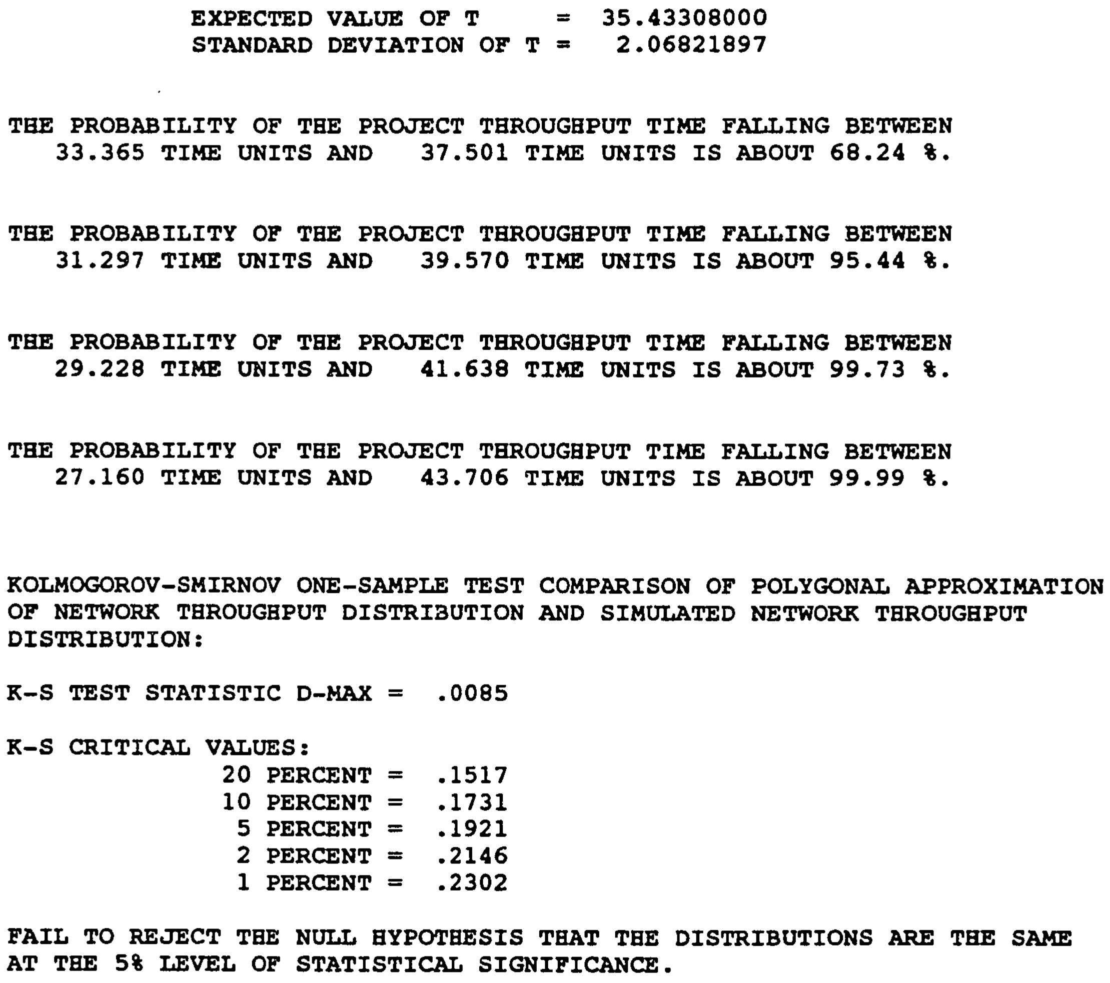
```
                    SIMULATION FREQUENCY HISTOGRAM

       0        .02       .04       .06       .08       .10
       I----+----I----+----I----+----I----+----I----+----I
18.140  *
18.420  *
18.700  *
18.980  *
19.260  *
19.540  **
19.820  **
20.100  ***
20.380  ****
20.660  ****
20.940  ******
21.220  ******
21.500  ********
21.780  ***********
22.060  ***********
22.340  *************
22.620  ***************
22.900  ***************
23.180  ********************
23.460  ********************
23.740  *************************
24.020  *************************
24.300  **************************
24.580  *****************************
24.860  ****************************
25.140  *****************************
25.420  **************************
25.700  ****************************
25.980  ***********************
26.260  **********************
26.540  ***********************
26.820  *******************
27.100  ******************
27.380  **************
27.660  *************
27.940  ***********
28.220  *********
28.500  ********
28.780  ******
29.060  *****
29.340  ****
29.620  ****
29.900  ***
30.180  **
30.460  **
30.740  **
31.020  *
31.300  *
31.580  *
31.860  *
```

```
                EXPECTED VALUE OF T     =  24.98661600
                STANDARD DEVIATION OF T =   2.05452688

THE PROBABILITY OF NODE   5 THROUGHPUT TIME FALLING BETWEEN
   22.932 TIME UNITS AND  27.041 TIME UNITS IS ABOUT 68.24 %.

THE PROBABILITY OF NODE   5 THROUGHPUT TIME FALLING BETWEEN
   20.878 TIME UNITS AND  29.096 TIME UNITS IS ABOUT 95.44 %.

THE PROBABILITY OF NODE   5 THROUGHPUT TIME FALLING BETWEEN
   18.823 TIME UNITS AND  31.150 TIME UNITS IS ABOUT 99.73 %.

THE PROBABILITY OF NODE   5 THROUGHPUT TIME FALLING BETWEEN
   16.769 TIME UNITS AND  33.205 TIME UNITS IS ABOUT 99.99 %.
```

```
THE SIMULATED TIME DISTRIBUTION THROUGH THE PROJECT IS:

              CUMULATIVE DISTRIBUTION FUNCTION

                   T              F(T)
                 25.000        .00000000
                 25.400        .00000000
                 25.800        .00000000
                 26.200        .00000000
                 26.600        .00000000
                 27.000        .00000000
                 27.400        .00010000
                 27.800        .00010000
                 28.200        .00010000
                 28.600        .00020000
                 29.000        .00040000
                 29.400        .00100000
                 29.800        .00260000
                 30.200        .00450000
                 30.600        .00770000
                 31.000        .01330000
                 31.400        .02230000
                 31.800        .03670000
                 32.200        .05550000
                 32.600        .08360000
                 33.000        .11830000
                 33.400        .16190000
                 33.800        .21770000
                 34.200        .28090000
                 34.600        .35150000
                 35.000        .42580000
                 35.400        .49760000
                 35.800        .57680000
                 36.200        .64780000
                 36.600        .71560000
                 37.000        .77500000
                 37.400        .83020000
                 37.800        .86980000
                 38.200        .90640000
                 38.600        .93280000
                 39.000        .95400000
                 39.400        .97070000
                 39.800        .98050000
                 40.200        .98870000
                 40.600        .99340000
                 41.000        .99640000
                 41.400        .99840000
                 41.800        .99930000
                 42.200        .99980000
                 42.600        .99990000
                 43.000       1.00000000
                 43.400       1.00000000
                 43.800       1.00000000
                 44.200       1.00000000
                 44.600       1.00000000
                 45.000       1.00000000
```

SIMULATION FREQUENCY HISTOGRAM

```
      0        .02       .04       .06       .08       .10
      I----+----I----+----I----+----I----+----I----+----I
25.200  *
25.600  *
26.000  *
26.400  *
26.800  *
27.200  *
27.600  *
28.000  *
28.400  *
28.800  *
29.200  *
29.600  **
30.000  **
30.400  ***
30.800  ****
31.200  ******
31.600  ********
32.000  **********
32.400  ***************
32.800  ******************
33.200  ***********************
33.600  *****************************
34.000  *********************************
34.400  ************************************
34.800  **************************************
35.200  *************************************
35.600  *****************************************
36.000  *************************************
36.400  ***********************************
36.800  *******************************
37.200  *****************************
37.600  *********************
38.000  *******************
38.400  **************
38.800  ************
39.200  *********
39.600  ******
40.000  *****
40.400  ***
40.800  ***
41.200  **
41.600  *
42.000  *
42.400  *
42.800  *
43.200  *
43.600  *
44.000  *
44.400  *
44.800  *
```



```
             EXPECTED VALUE OF T      =  35.43308000
             STANDARD DEVIATION OF T =   2.06821897


THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   33.365 TIME UNITS AND   37.501 TIME UNITS IS ABOUT 68.24 %.


THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   31.297 TIME UNITS AND   39.570 TIME UNITS IS ABOUT 95.44 %.


THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   29.228 TIME UNITS AND   41.638 TIME UNITS IS ABOUT 99.73 %.


THE PROBABILITY OF THE PROJECT THROUGHPUT TIME FALLING BETWEEN
   27.160 TIME UNITS AND   43.706 TIME UNITS IS ABOUT 99.99 %.


KOLMOGOROV-SMIRNOV ONE-SAMPLE TEST COMPARISON OF POLYGONAL APPROXIMATION
OF NETWORK THROUGHPUT DISTRIBUTION AND SIMULATED NETWORK THROUGHPUT
DISTRIBUTION:

K-S TEST STATISTIC D-MAX =  .0085

K-S CRITICAL VALUES:
             20 PERCENT =  .1517
             10 PERCENT =  .1731
              5 PERCENT =  .1921
              2 PERCENT =  .2146
              1 PERCENT =  .2302

FAIL TO REJECT THE NULL HYPOTHESIS THAT THE DISTRIBUTIONS ARE THE SAME
AT THE 5% LEVEL OF STATISTICAL SIGNIFICANCE.
```

Figure 45. PART-seq program output.

Table 20. Comparisons of Reductions of a "Conditional" Network.

Source	Mean	Standard Deviation	MADV
Simulation	35.433	2.068	n/a
PART-ind	35.462	2.113	0.0085
PART-seq	35.462	2.113	0.0085

Table 20 presents comparisons of the PART-ind and PART-seq algorithm reductions and the simulation approximation of the "conditional" network obtained by the same means as the data in Table 19. The PART-ind and PART-seq approximations were essentially identical and were very close to the simulation results. The relative error in the estimation of the mean of the throughput distribution was 0.08%; in the estimation of the standard deviation, 2.18%. The MADVs, as the computed values of the K-S goodness-of-fit test statistic T, again have prob values >20%.

The doubly tied, double Wheatstone Bridge network in Figure 46 consists of a double Wheatstone Bridge (center "square" of nodes 2, 3, 4, 5, and 6), whose front edge is tied by two activities to a source node (node 1) and whose back edge is tied by two activities to a sink node (node 7) (Whitehouse, 1973; Elmaghraby, 1977). Although the network has

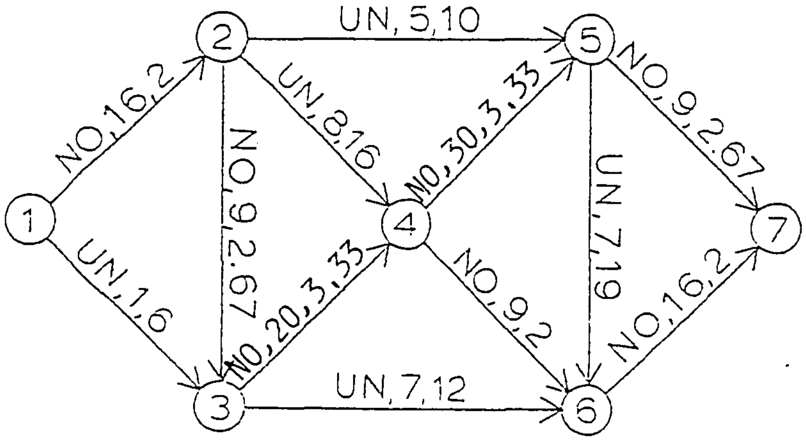


Figure 46. Doubly tied, double Wheatstone Bridge network.

only seven nodes and 12 activities, there are 13 paths from source to sink. The activity duration distributions were suggested by Solberg (1978). The five uniform distributions have "high" variances and appear in series in one pair in the original network. The domains of the uniform distributions were chosen so that paths of about the same average length have near-maximally overlapping distributions; for example, the distribution of path (1,2) → (2,5) → (5,6) → (6,7) has average length 52.5 over domain [32,73], and the distribution of path (1,2) → (2,4) → (4,6) → (6,7) has average length 53 over domain [31,75]. Because of these selections of the activity duration distributions, the polygonal approximation-based series-parallel reductions operators in the PART algorithms had greater difficulty maintaining accuracy during the reduction of this network than would have been the case if the distributions had been randomly selected (Section 3.5.1 [above]).

Table 21 presents comparisons of the PART-ind and PART-seq algorithm reductions and the simulation approximation of the doubly tied, double Wheatstone Bridge network obtained by the same means as the data in Table 19, and the CPM estimates of the first and second moments of the throughput distribution. Again, the PART-ind and PART-seq approximations were essentially identical and were very close to the simulation results. The relative error in the estimation of the mean was 0.23%; in the estimation of the standard deviation, 9.74%. The MADVs again have prob values > 20%. As expected, CPM underestimated the mean (Moder and Phillips, 1964), and the PART-ind and PART-seq algorithms overestimated it (Section 2.5.2, and Equation (9), Section 3.3.1 [above]).

Table 21. Comparisons of Reductions of a Doubly Tied, Double Wheatstone Bridge Network.

Source	Mean	Standard Deviation	MADV
CPM	104.000	7.024	n/a
Simulation	103.960	6.970	n/a
PART-ind	104.194	7.649	0.0267
PART-seq	104.194	7.649	0.0267

4.3.2 Performance Against Reported Performance of Competing Procedures

Only two researchers have reported performance results for competing approximation procedures: Dodin (1980, 1985a) for sequential approximation, and Hagstrom (1990) for ordered recursive conditioning. Hagstrom reported only the computation time (in CPU microseconds) to compute either moments or values of the cumulative throughput distributions of a set of 19 test networks with an ordered recursive conditioning algorithm based on discretization of activity duration distributions (Section 2.5.3 [above]). Since she reported no information concerning the accuracy of her algorithm, it is not possible to compare directly the performance of ordered recursive conditioning with the performance of polygonal approximation except on the basis of computation times for specific networks.

From a performance analysis of a sequential approximation algorithm, Dodin (1980, 1985a) reported results for two experimental combinations: different activity distribution functions with a fixed network, and different size networks with a fixed activity distribution function for each activity. Table 3 (in Section 2.5.2 [above]) shows how the distribution type affects the performance for a randomly generated network with 10 nodes and 15 activities. Table 4 reflects the effect of network size; a uniform distribution was applied to each of eight networks. Table 22 shows the distribution functions used by Dodin. He held that the choice of the distributions and their parameters is of no particular significance except as it reflects the general applicability of an approximating procedure.

Dodin's results (Tables 3 and 4) are reported for "a randomly generated network" tested with a particular distribution or pair of numbers of nodes and arcs. It is not clear whether the reported performance measures are for a single network randomly generated to test all the experimental cases, or whether they are the response values of the "best" network from a set of randomly generated networks each of which was tested under all the experimental cases - in either case the results are without value to the characterization of the performance of the sequential approximation algorithm - or whether they were obtained in some other way. In any event, it is not possible to compare directly the reported performance of sequential approximation with the performance of polygonal approximation.

To effect as reasonable and meaningful a comparison as possible between the reported performance of sequential approximation and the performance of polygonal approximation, 20 "strongly random" networks were generated and their throughput distributions then approximated by both the PART-ind and PART-seq algorithms under each of Dodin's experimental cases. In order to relate the performance of polygonal approximation, as reflected by the output products of the PART-ind and PART-seq algorithms, the raw data performance measures reported by Dodin were first converted to relative error performance measures. Tables 23 and 24 contain Dodin's reported results from Tables 3 and 4 expressed in relative error terms: the relative error of the average (standard deviation) is the difference between the averages (standard deviations) obtained from the sequential approximation algorithm and the simulation approximation, expressed as a percentage of the value obtained from the simulation approximation.

The test networks were generated and then reduced by the PART-ind and PART-seq validation programs [labeled PART-ind(val) and PART-seq(val), respectively]. Figure 47 shows the input data files for both the PART-ind(val) and PART-seq(val) programs for the "all" types-of-distribution case from Tables 3 and 23: Part (a) shows the control file (CONTROL.VAL) and Part (b) shows the activity duration distributions data file (DATAH.VAL). The formats of these files are described in comment statements at the top of the code listings of the PART-ind(val) and PART-seq(val) programs in Appendices D and E, respectively. Figure 48 depicts the output from the PART-ind(val) program under output option 9 (only statistical comparisons), a summary table which includes for each network generated: for the K-S goodness-of-fit test for the network throughput distribution - the computed value of the K-S test statistic (labeled D-MAX) and its associated prob value; the relative errors of the approximations of the mean and standard deviation;

Table 22. Probability Distribution Functions Used in Performance Analysis of a Sequential Approximation Algorithm. [Adapted from Dodin (1980)]

Distribution Type	Mean or First Parameter	Stn. Dev. or Second Parameter	Minimum	Maximum
Uniform	5.0	n/a	0.0	10.0
Triangular	5.0	n/a	1.0	11.0
Normal	8.0	2.0	2.0	14.0
Exponential	2.0	n/a	0.0	15.0
Gamma	3.0	1.0	0.0	10.0
Beta	3.0	2.0	1.0	11.0

Table 23. Performance Measures for Sequential Approximation for Various Distribution Functions (Recomputed). [Adapted from Dodin (1980, 1985a)]

Distribution Type	Comparison of the Approximate DF with that of MCS			
	Rel. Error of Average	Rel. Error of Standard Deviation	MADV	AADV
Uniform	0.25%	-8.43%	0.0426	0.0115
Triangular	-0.27%	-7.76%	0.0513	0.0180
Normal	0%	-2.89%	0.0585	0.0206
Exponential	0.08%	-9.95%	0.0328	0.0099
Gamma	1.44%	3.30%	0.0436	0.0122
Beta	-0.54%	-7.51%	0.0772	0.0275
Discrete	-0.05%	-1.01%	0.0083	0.0016
All	1.39%	-4.92%	0.0274	0.0070

Table 24. Effect of Network Size on the Performance Measures for Sequential Approximation (Recomputed). [Adapted from Dodin (1980, 1985a)]

No. Nodes	No. Arcs	Comparison of the Approximate DF with that of MCS			
		Rel. Error of Average	Rel. Error of Standard Deviation	MADV	AADV
10	15	0.25%	-8.43%	0.0426	0.0115
20	40	1.94%	-11.70%	0.0557	0.0162
30	50	0.79%	-12.44%	0.0525	0.0169
40	60	2.20%	-14.83%	0.0633	0.0182
40	80	1.67%	-9.17%	0.0303	0.0115
50	75	-0.06%	-6.40%	0.0651	0.0259
50	100	2.78%	-20.25%	0.0880	0.0263
60	150	3.46%	-21.15%	0.1082	0.0306

```
020 010 0015 9 10000 06 0.00
```

(a) Control data file.

```
1 5.000000 0.000000 1.000000 11.00000
2 8.000000 0.000000 2.000000 14.00000
3 2.000000 0.000000 0.000000 15.00000
4 3.000000 1.000000 0.000000 10.00000
5 3.000000 2.000000 1.000000 11.00000
6 0.100000 0.000000 0.000000 10.00000
```

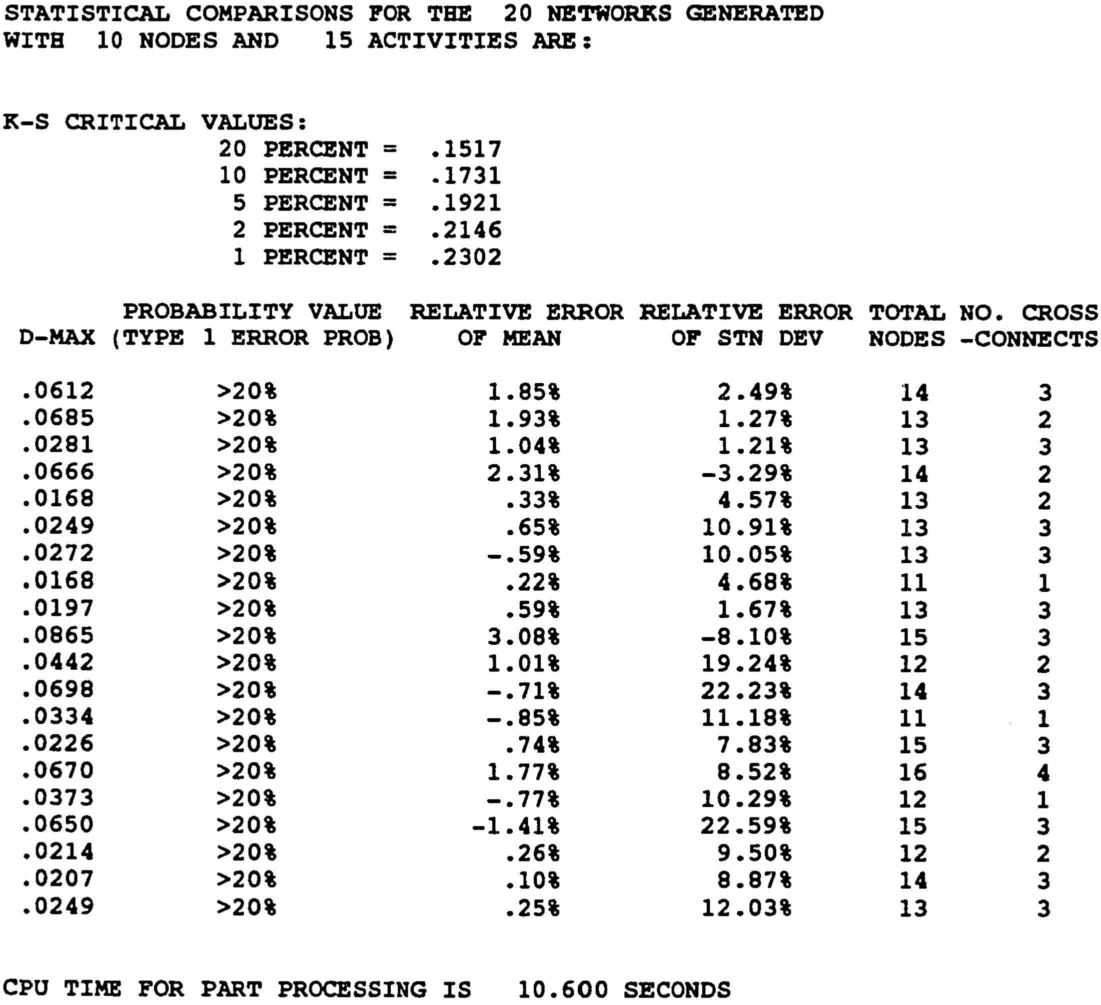
(b) Activity duration distributions data file.

Figure 47. PART-ind(val) and PART-seq(val) program input.

the number of nodes in the completely reducible network; and the number of cross-connections removed by the "independent multiple arcs" method. Figure 49 depicts the output from the PART-seq(val) program also under option 9, which is similar to the PART-ind(val) output, except that there is no report of the number of nodes in the completely reducible network or the number of cross-connections removed, since sequential approximation reduces a network without the "independent multiplication" (duplication) of arcs.

Summaries of the performance of the PART-ind and PART-seq algorithms against Dodin's experimental cases are presented in Table 25 (different activity distribution functions) and Table 26 (different size networks). The summaries contain the ranges of values experienced and the average absolute errors (of the "average" network) of the three performance measures - relative error of the average, relative error of the standard deviation, and MADV - and the values of these performance measures for the "best" network among the 20 "strongly random" networks generated for each experimental case.

Among the experimental cases involving the six different activity distribution functions with a network of fixed size, the performance measures of the "best" network reduced by



```
STATISTICAL COMPARISONS FOR THE  20 NETWORKS GENERATED
WITH  10 NODES AND   15 ACTIVITIES ARE:


K-S CRITICAL VALUES:
              20 PERCENT =  .1517
              10 PERCENT =  .1731
               5 PERCENT =  .1921
               2 PERCENT =  .2146
               1 PERCENT =  .2302

        PROBABILITY VALUE  RELATIVE ERROR RELATIVE ERROR TOTAL NO. CROSS
 D-MAX (TYPE 1 ERROR PROB)    OF MEAN        OF STN DEV   NODES -CONNECTS

 .0612         >20%            1.85%            2.49%      14        3
 .0685         >20%            1.93%            1.27%      13        2
 .0281         >20%            1.04%            1.21%      13        3
 .0666         >20%            2.31%           -3.29%      14        2
 .0168         >20%             .33%            4.57%      13        2
 .0249         >20%             .65%           10.91%      13        3
 .0272         >20%           -.59%            10.05%      13        3
 .0168         >20%             .22%            4.68%      11        1
 .0197         >20%             .59%            1.67%      13        3
 .0865         >20%            3.08%           -8.10%      15        3
 .0442         >20%            1.01%           19.24%      12        2
 .0698         >20%           -.71%            22.23%      14        3
 .0334         >20%           -.85%            11.18%      11        1
 .0226         >20%             .74%            7.83%      15        3
 .0670         >20%            1.77%            8.52%      16        4
 .0373         >20%           -.77%            10.29%      12        1
 .0650         >20%          -1.41%            22.59%      15        3
 .0214         >20%             .26%            9.50%      12        2
 .0207         >20%             .10%            8.87%      14        3
 .0249         >20%             .25%           12.03%      13        3

CPU TIME FOR PART PROCESSING IS   10.600 SECONDS
```

Figure 48. PART-ind(val) program output.

```
STATISTICAL COMPARISONS FOR THE  20 NETWORKS GENERATED
WITH  10 NODES AND   15 ACTIVITIES ARE:


K-S CRITICAL VALUES:
              20 PERCENT =  .1517
              10 PERCENT =  .1731
               5 PERCENT =  .1921
               2 PERCENT =  .2146
               1 PERCENT =  .2302

          PROBABILITY VALUE    RELATIVE ERROR  RELATIVE ERROR
 D-MAX   (TYPE 1 ERROR PROB)      OF MEAN        OF STN DEV

 .0612         >20%              1.85%            2.49%
 .0686         >20%              1.93%            1.25%
 .0281         >20%              1.04%            1.21%
 .0666         >20%              2.31%           -3.29%
 .0168         >20%               .33%            4.57%
 .0249         >20%               .65%           10.91%
 .0272         >20%              -.59%           10.05%
 .0168         >20%               .22%            4.68%
 .0197         >20%               .59%            1.67%
 .0865         >20%              3.08%           -8.10%
 .0313         >20%               .68%           13.99%
 .0698         >20%              -.71%           22.23%
 .0327         >20%              -.80%           11.58%
 .0226         >20%               .74%            7.83%
 .0670         >20%              1.77%            8.52%
 .0373         >20%              -.77%           10.29%
 .0650         >20%             -1.41%           22.59%
 .0214         >20%               .26%            9.50%
 .0200         >20%               .15%            9.21%
 .0249         >20%               .25%           12.03%

CPU TIME FOR PART PROCESSING IS     8.090 SECONDS
```

Figure 49. PART-seq(val) program output.

Table 25. Performance Measures for Polygonal Approximation of 20 Networks (with 10 Nodes and 15 Activities) with Various Distribution Functions.

Distribution Type	Performance Measure	"Independent Multiple Arcs"			Sequential Approximation		
		Relative Error of Average	Relative Error of Stn Dev	MADV	Relative Error of Average	Relative Error of Stn Dev	MADV
All Uniform	Range	-0.40% to 4.92%	-3.52% to 4.50%	0.0044 to 0.1523	-0.40% to 4.92%	-3.52% to 4.50%	0.0044 to 0.1523
	Best	0.14%	0.73%	0.0044	0.14%	0.73%	0.0044
	Avg Abs Error	2.20%	4.37%	0.0568	2.20%	4.40%	0.0569
All Triangular	Range	-0.28% to 3.84%	-7.31% to 14.28%	0.0110 to 0.1273	-0.28% to 3.84%	-7.31% to 14.28%	0.0110 to 0.1273
	Best	-0.05%	5.08%	0.0110	-0.05%	5.08%	0.0110
	Avg Abs Error	1.05%	5.49%	0.0434	1.05%	5.48%	0.0436
All Normal	Range	0.07% to 2.91%	8.03% to 20.34%	0.0296 to 0.1459	0.06% to 2.91%	7.78% to 20.34%	0.0309 to 0.1459
	Best	0.34%	8.03%	0.0296	0.36%	7.78%	0.0335
	Avg Abs Error	1.02%	14.74%	0.0626	1.02%	14.75%	0.0630
All Exponential	Range	-15.53% to 2.96%	10.03% to 51.02%	0.0597 to 0.2275	-15.53% to 1.38%	9.74% to 51.02%	0.0597 to 0.2275
	Best	0.70%	18.37%	0.0597	0.70%	18.37%	0.0597
	Avg Abs Error	7.66%	24.28%	0.1396	7.49%	24.35%	0.1338
All Gamma	Range	-6.34% to 2.68%	2.43% to 20.07%	0.0178 to 0.1226	-6.34% to 2.68%	2.43% to 20.07%	0.0178 to 0.1226
	Best	-0.23%	2.43%	0.0178	-0.23%	2.43%	0.0178
	Avg Abs Error	1.89%	9.80%	0.0444	1.90%	9.82%	0.0428
All Beta	Range	0.14% to 2.57%	-4.35% to 11.66%	0.0142 to 0.0918	0.11% to 2.57%	-4.35% to 11.66%	0.0142 to 0.0918
	Best	0.27%	5.93%	0.0142	0.27%	5.93%	0.0142
	Avg Abs Error	1.04%	6.88%	0.0422	1.04%	6.88%	0.0421
Mixed (All)	Range	-1.41% to 3.08%	-8.10% to 22.59%	0.0168 to 0.0865	-1.41% to 3.08%	-8.10% to 22.59%	0.0168 to 0.0865
	Best	0.22%	4.68%	0.0168	0.22%	4.68%	0.0168
	Avg Abs Error	1.02%	9.03%	0.0411	1.01%	8.80%	0.0404

Table 26. Effect of Network Size on the Performance Measures for Polygonal Approximation of 20 Networks (with All-Uniform Activity Distributions).

No. Nodes/ No. Arcs		Perfor-mance Measure	"Independent Multiple Arcs"			Sequential Approximation		
			Relative Error of Average	Relative Error of Stn Dev	MADV	Relative Error of Average	Relative Error of Stn Dev	MADV
10	15	Range	-0.40% to 4.92%	-13.52% to 4.50%	0.0044 to 0.1523	-0.40% to 4.92%	-13.52% to 4.50%	0.0044 to 0.1523
		Best	0.14%	0.73%	0.0044	0.14%	0.73%	0.0044
		Av Abs Er	2.20%	4.37%	0.0568	2.20%	4.40%	0.0569
20	40	Range	1.61% to 5.88%	-13.96% to 4.46%	0.0453 to 0.1716	1.61% to 5.88%	-13.96% to 4.46%	0.0469 to 0.1716
		Best	1.61%	0.90%	0.0453	1.61%	0.78%	0.0469
		Av Abs Er	3.96%	5.40%	0.1204	3.95%	5.39%	0.1203
30	50	Range	0.88% to 5.69%	-8.15% to 8.18%	0.0354 to 0.1652	0.88% to 5.69%	-8.15% to 8.18%	0.0366 to 0.1642
		Best	0.88%	8.18%	0.0356	0.88%	8.18%	0.0356
		Av Abs Er	3.14%	3.62%	0.1007	3.14%	3.65%	0.1009
40	60	Range	0.19% to 3.64%	-2.00% to 11.00%	0.0127 to 0.1174	0.12% to 3.63%	-1.83% to 11.04%	0.0130 to 0.1153
		Best	0.19%	5.85%	0.0127	0.20%	5.88%	0.0130
		Av Abs Er	1.67%	4.66%	0.0591	1.64%	4.63%	0.0586
40	80	Range	1.45% to 7.48%	-8.04% to 11.65%	0.0479 to 0.2446	1.50% to 7.48%	-7.82% to 11.41%	0.0478 to 0.2456
		Best	1.45%	3.25%	0.0479	1.50%	3.36%	0.0478
		Av Abs Er	4.31%	4.64%	0.1482	4.30%	4.55%	0.1482
50	75	Range	-1.71% to 5.64%	-5.67% to 13.71%	0.0222 to 0.2145	-1.53% to 6.03%	-5.09% to 13.90%	0.0178 to 0.2239
		Best	0.58%	4.13%	0.0222	0.48%	3.82%	0.0178
		Av Abs Er	2.14%	6.58%	0.0758	2.15%	6.61%	0.0758
50	100	Range	-0.65% to 5.40%	-1.94% to 11.52%	0.0356 to 0.1690	-0.61% to 5.45%	-1.02% to 12.08%	0.0347 to 0.1734
		Best	1.21%	4.72%	0.0358	0.51%	12.08%	0.0347
		Av Abs Er	2.13%	6.80%	0.0818	2.14%	6.80%	0.0823
60	150	Range	0.43% to 8.99%	-4.92% to 18.11%	0.0327 to 0.3026	0.35% to 9.19%	-5.77% to 18.45%	0.0313 to 0.3100
		Best	0.43%	10.25%	0.0327	0.35%	9.88%	0.0313
		Av Abs Er	4.80%	7.33%	0.1845	4.83%	7.45%	0.1858

polygonal approximation were about the same or better than the results reported by Dodin for all except the exponential distribution. To control the build-up of error from initial distribution approximation and subsequent series-reduction operations, Dodin used 50% more discretization points for each exponential distribution than for all the other distributions, and he used the equal probabilities method for discretization of each exponential distribution as opposed to the equal intervals method for the discretization of all the other distributions, because the exponential has such a high coefficient of variation (Section 2.4.4 [above]). If the number of classes for polygonal approximation is similarly increased in PART algorithms, the error build-up due to exponential distributions is likewise attenuated. Among his treatment cases, Dodin included a seventh distribution type, the discrete distribution, which he did not have to discretize at the start of a network reduction with his sequential approximation algorithm. Because there were no initial discretization errors with discrete distributions, the performance measures for the discrete type-of-distribution case were the best of all of the eight experimental cases which Dodin considered, and the performance measures for the "all" types-of-distribution case benefited from the presence of some discrete distributions which were randomly selected as activity distributions for the network reduced by his sequential approximation algorithm (Tables 3 and 23). However, the performance measures of the "best" network reduced by polygonal approximation for the "all" types-of-distribution case (labeled "Mixed (All)" in Table 25), which did not include any discrete distributions, were better than Dodin's reported results by a factor of about two, and the performance measures of the "average" network were comparable to Dodin's reported results for a single network reduced by his sequential approximation algorithm; this was Dodin's only experimental case involving a mix of activity distribution types.

Among the experimental cases involving the eight different size networks with a fixed activity distribution function (uniform) for each activity, the performance measures for the "best" network reduced by polygonal approximation were about the same or better than the results reported by Dodin. For the two largest network sizes considered (50 nodes and 100 activities, and 60 nodes and 150 activities), the performance measures for the "average" network reduced by polygonal approximation were comparable to Dodin's reported results for a single network of each size reduced by his sequential approximation algorithm. As expected, all the performance measures - ranges of values, "best," and "average" of he relative errors of the average, standard deviation, and MADV - fell off for the networks reduced by polygonal approximation as the number of activities increased for a fixed number of nodes (Table 26: compare the results for networks with 40 nodes and 60 activities, then 80 activities, and for networks with 50 nodes and 75 activities, then 100 activities), because of the increased number of initial distribution approximations and subsequent series-parallel reduction operations. Dodin reported a contrary finding for a network with 40 nodes and 60 activities, then 80 activities (Tables 4 and 24). However, since Dodin's reported results are for a single network of each size reduced by his sequential approximation algorithm, this disagreement between his reported results and the performance of polygonal approximation is not meaningful.

For both of Dodin's experimental combinations, the performance of the polygonal approximation-based PART-ind and PART-seq algorithms was virtually identical (Tables 25 and 26), and about the same or better than Dodin's reported results for his sequential approximation algorithm (Tables 3 and 23), based on comparisons of relative error-based performance measurements. Since each of Dodin's experimental combinations ignored a source of variability - different activity distribution functions with a fixed network ignored variability due to network size, and different size networks with a fixed activity distribution function for each activity ignored variability due to activity distributions - his results cannot be generalized. However, the performance of the polygonal approximation-based algorithms for his two experimental combinations failed to support two of his original findings based on his discretization-based sequential approximation algorithm (Section 2.5.2 [above]). First, algorithm-approximated standard deviations of throughput distributions were not less than simulation-approximated standard deviations, even for a majority of the time, when different activity distributions were tested with a fixed network; consequently, Dodin's finding that approximated distribution functions have less variation than distribution functions obtained by simulation was not supported. Rather, the variation of an approximated throughput distribution is a function of both the amount of variation in the activity distribution functions and the ability of an algorithm operating in a particular configuration, e.g., number of classes for a polygonal approximation-based algorithm, to capture that variation in the initial distribution approximation and subsequent series-parallel reduction operations. When activity distribution variation is low and an algorithm is effectively configured to control error build-up, algorithm-approximated standard deviations may be less than simulation-approximated values, which are functions of randomly generated deviates. However, when activity variation increases, i.e., when activity distributions have increasing coefficients of variation, unless an algorithm's configuration is adjusted to account for the increasing variation (as Dodin did for the exponential distribution-type case), the relative error of the standard deviation will increase, as shown in Table 25. Second, although MADV increased as network size increased, when different size networks were tested with a fixed activity distribution function for each activity and a constant number of simulation replications (10,000) was used, the number of simulation replications was reasonably large (10 times Dodin's number of 1,000 replications); consequently, Dodin's finding that increasing MADV with increasing network size is the consequence of fixed simulation sample sizes also was not supported. Rather, increasing MADV with increasing network size is explained by the increasing numbers of initial activity distribution approximations and subsequent series-parallel reduction operations needed to reduce the network with an approximation-based method.

4.3.3 Performance Against Test Networks Constructed within an Experimental Design

Following the experimental design for algorithm testing discussed in Section 3.5.1 [above], a series of validation experiments was conducted for the PART-ind and PART-seq algorithms based on how great a challenge a "strongly randomized" test network presented to the algorithms. As similar results were experienced for all of the experiments, only one is discussed in detail here. A three-factor ANOVA was selected. The design factors were specified as: the two PART algorithms (factor label METHOD); network size and structure, measured by the number of nodes and the number of activities in a network (factor label NETSIZE); and the challenge of the probability distribution functions of activity duration (factor label CHALLNG). The networks in Table 26 were repeated as the eight levels of the network size and structure factor. Three levels of the challenge factor were specified: low, medium, and high. A set of activity duration distributions was specified for each challenge level. Since pairs of activity duration distributions with "high" variances, i.e., whose standard deviations are high percentages of their means (high coefficients of variation), are the most difficult convolutions for a numerical approximation-based series-reduction operator, and pairs of maximally overlapping activity duration distributions are the most difficult for a numerical approximation-based parallel-reduction operator to reduce, these properties were used to specify the parameters of the distributions at each challenge factor level. Table 27 contains the coefficients of variation of the six probability distribution functions available in PART programs and shows how the coefficients can be increased by manipulation of the parameters of the distribution. The parameters and the domains of the distributions at each challenge factor level are in Table 28. As the challenge factor level rises, the coefficients of variation of the distributions and the amount of overlap of their domains both increase. The response variables were the three performance measures: MADV, the absolute relative error of the average, and the absolute error of the standard deviation.

Table 27. Coefficients of Variation of Probability Distribution Functions.

Distribution	Coefficient of Variation $\left(\frac{\sigma}{\mu}\right)$	To increase coefficient:
Uniform on $[a, b]$	$\dfrac{\left(\frac{b-a}{\sqrt{12}}\right)}{\left(\frac{a+b}{2}\right)} = \dfrac{1}{\sqrt{3}}\left[1-\dfrac{2}{\left(1+\frac{b}{a}\right)}\right]$	Increase difference between b and a.
Triangular on $[a, b]$ with mode m	$\dfrac{\left(\frac{\sqrt{a^2+m^2+b^2-m(a+b)-ab}}{3\sqrt{2}}\right)}{\left(\frac{a+m+b}{3}\right)}$ $= \dfrac{1}{\sqrt{2}}\left(\dfrac{\sqrt{a^2+m^2+b^2-m(a+b)-ab}}{a+m+b}\right)$	Increase b and m relative to a.
Normal (μ, σ)	$\dfrac{\sigma}{\mu}$	Increase σ relative to μ or decrease μ relative to σ.
Exponential (μ)	$\dfrac{\mu}{\mu} = 1$	Coefficient is constant.
Gamma (α, β) with shape parameter α and scale parameter β	$\dfrac{\sqrt{\alpha}\beta}{\alpha\beta} = \dfrac{1}{\sqrt{\alpha}}$	Decrease α.
Beta (α_1, α_2) with shape parameters α_1 and α_2.	$\dfrac{\left(\frac{\sqrt{\alpha_1\alpha_2}}{(\alpha_1+\alpha_2)\sqrt{\alpha_1+\alpha_2+1}}\right)}{\left(\frac{\alpha_1}{\alpha_1+\alpha_2}\right)}$ $= \sqrt{\dfrac{\alpha_2}{\alpha_1(\alpha_1+\alpha_2+1)}}$	Decrease α_1 relative to α_2.

Table 28. Probability Distribution Functions Used in ANOVA of Performance Measures.

(a) Low Challenge Factor Level.

Distribution Type	Mean or First Parameter	Stn. Dev. or Second Parameter	Minimum	Maximum
Uniform	42.5	n/a	40.0	45.0
Triangular	15.0	n/a	10.0	20.0
Normal	30.0	3.0	21.0	39.0
Exponential	2.0	2.0	0.0	15.0
Gamma	4.0	1.0	0.0	10.0
Beta	1.0	1.0	5.0	15.0

(b) Medium Challenge Factor Level.

Distribution Type	Mean or First Parameter	Stn. Dev. or Second Parameter	Minimum	Maximum
Uniform	5.0	n/a	0.0	10.0
Triangular	10.0	n/a	5.0	20.0
Normal	15.0	2.0	9.0	21.0
Exponential	10.0	10.0	5.0	25.0
Gamma	3.0	1.0	0.0	10.0
Beta	3.0	2.0	10.0	20.0

(c) High Challenge Factor Level.

Distribution Type	Mean or First Parameter	Stn. Dev. or Second Parameter	Minimum	Maximum
Uniform	10.0	n/a	0.0	20.0
Triangular	16.0	n/a	0.0	20.0
Normal	9.0	3.0	0.0	20.0
Exponential	5.0	5.0	0.0	20.0
Gamma	0.5	1.0	0.0	20.0
Beta	0.2	2.0	0.0	20.0

Twenty "strongly random" networks were generated by the PART-ind(val) and PART-seq(val) programs for each of the 48 experimental cases: method by network size and structure by challenge. Part (a) of Table 29 presents the ANOVA results for the performance measure MADV, Part (b) for absolute relative error of the average (AVGERR), and Part (c) for absolute relative error of the standard deviation (STDERR). As expected, the performance of the PART-ind and PART-seq algorithms was virtually identical; method was not a statistically significant factor, so it was dropped from the models. Table 30 shows the results for the two-factor ANOVAs, excluding the method factor. While challenge was the overwhelmingly dominant factor in the models for MADV and AVGERR, network size and structure and the challenge-network size and structure interaction were also statistically significant. For STDERR, the factor strengths of challenge and network size and structure were about the same, and the challenge-network size and structure interaction was also statistically significant. The first and second moments of the performance measures for each of the two significant factors, challenge and network size and structure, are in Table 31. For MADV, the moments are about the same magnitude at the low and medium challenge factor levels, then jump by a factor of three at the high level; they increase slowly, almost monotonically with the number of activities (network size and structure). For AVGERR, the moments jump by a factor of about five at each challenge factor level, and also increase slowly, almost monotonically with the number of activities. For STDERR, the moments are virtually the same for the low and medium challenge factor levels, then jump by a factor of about 1.5 at the high level; they also increase slowly, almost monotonically with the number of nodes. Table 32 contains the moments and ranges of values of the performance measures by factors; Table 33 presents these data by experimental cases.

The validation experiments were conducted with versions of the PART-ind(val) and PART-seq(val) programs which employ ten classes for polygonal approximation, without

Table 29. ANOVA of Performance Measures for Three Factors.

(a) Performance Measure: MADV.

```
Dependent Variable: MADV

Source                  DF        Sum of Squares          Mean Square      F Value        Pr > F

Model                   47            3.53615306           0.07523730        28.23        0.0001

Error                  912            2.43085693           0.00266541

Corrected Total        959            5.96700999

                  R-Square                  C.V.             Root MSE                  MADV Mean

                  0.592617              51.51404           0.05162764                 0.10022052


Source                  DF           Type I SS          Mean Square      F Value        Pr > F

METHOD                   1          0.00000938           0.00000938         0.00        0.9527
CHALLNG                  2          2.48333445           1.24166722       465.84        0.0001
METHOD*CHALLNG           2          0.00006826           0.00003413         0.01        0.9873
NETSIZE                  7          0.80119819           0.11445688        42.94        0.0001
METHOD*NETSIZE           7          0.00005804           0.00000829         0.00        1.0000
CHALLNG*NETSIZE         14          0.25135323           0.01795380         6.74        0.0001
METHOD*CHALLN*NETSIZ    14          0.00013152           0.00000939         0.00        1.0000

Source                  DF         Type III SS          Mean Square      F Value        Pr > F

METHOD                   1          0.00000938           0.00000938         0.00        0.9527
CHALLNG                  2          2.48333445           1.24166722       465.84        0.0001
METHOD*CHALLNG           2          0.00006826           0.00003413         0.01        0.9873
NETSIZE                  7          0.80119819           0.11445688        42.94        0.0001
METHOD*NETSIZE           7          0.00005804           0.00000829         0.00        1.0000
CHALLNG*NETSIZE         14          0.25135323           0.01795380         6.74        0.0001
METHOD*CHALLN*NETSIZ    14          0.00013152           0.00000939         0.00        1.0000
```

Table 29 cont.

(b) Performance Measure: Absolute Relative Error of the Average.

```
Dependent Variable: AVGERR

Source                    DF        Sum of Squares         Mean Square      F Value          Pr > F

Model                     47      6529.67047406        138.92915902         31.04          0.0001

Error                    912      4081.30867500          4.47511916                            .

Corrected Total          959     10610.97914906

                    R-Square                C.V.            Root MSE                     AVGERR Mean

                    0.615369            89.12895          2.11544774                      2.37346875


Source                    DF             Type I SS         Mean Square      F Value          Pr > F

METHOD                     1          0.02194594          0.02194594          0.00          0.9442
CHALLNG                    2       5965.36351187       2982.68175594        666.50          0.0001
METHOD*CHALLNG             2          0.07365813          0.03682906          0.01          0.9918
NETSIZE                    7        261.38451323         37.34064475          8.34          0.0001
METHOD*NETSIZE             7          0.07306823          0.01043832          0.00          1.0000
CHALLNG*NETSIZE           14        302.57522646         21.61251618          4.83          0.0001
METHOD*CHALLN*NETSIZ      14          0.17855021          0.01275359          0.00          1.0000

Source                    DF           Type III SS         Mean Square      F Value          Pr > F

METHOD                     1          0.02194594          0.02194594          0.00          0.9442
CHALLNG                    2       5965.36351187       2982.68175594        666.50          0.0001
METHOD*CHALLNG             2          0.07365813          0.03682906          0.01          0.9918
NETSIZE                    7        261.38451323         37.34064475          8.34          0.0001
METHOD*NETSIZE             7          0.07306823          0.01043832          0.00          1.0000
CHALLNG*NETSIZE           14        302.57522646         21.61251618          4.83          0.0001
METHOD*CHALLN*NETSIZ      14          0.17855021          0.01275359          0.00          1.0000
```

Table 29 cont.

(c) Performance Measure: Absolute Relative Error of the Standard Deviation.

```
Dependent Variable: STDERR

Source                    DF          Sum of Squares           Mean Square        F Value            Pr > F

Model                     47          70920.08712999         1508.93802404          14.96            0.0001

Error                    912          91970.56231002          100.84491481                                .

Corrected Total          959         162890.64944001

                    R-Square                    C.V.              Root MSE                      STDERR Mean

                    0.435385                51.15459           10.04215688                      19.63100000

Source                    DF               Type I SS           Mean Square        F Value            Pr > F

METHOD                     1              0.06733500            0.06733500           0.00            0.9794
CHALLNG                    2          17987.35609312         8993.67804656          89.18            0.0001
METHOD*CHALLNG             2              0.58448438            0.29224219           0.00            0.9971
NETSIZE                    7          48866.98891166         6980.99841595          69.23            0.0001
METHOD*NETSIZE             7              5.07702000            0.72528857           0.01            1.0000
CHALLNG*NETSIZE           14           4037.26853521          288.37632394           2.86            0.0003
METHOD*CHALLN*NETSIZ      14             22.74475063            1.62462504           0.02            1.0000

Source                    DF             Type III SS           Mean Square        F Value            Pr > F

METHOD                     1              0.06733500            0.06733500           0.00            0.9794
CHALLNG                    2          17987.35609312         8993.67804656          89.18            0.0001
METHOD*CHALLNG             2              0.58448438            0.29224219           0.00            0.9971
NETSIZE                    7          48866.98891166         6980.99841595          69.23            0.0001
METHOD*NETSIZE             7              5.07702000            0.72528857           0.01            1.0000
CHALLNG*NETSIZE           14           4037.26853521          288.37632394           2.86            0.0003
METHOD*CHALLN*NETSIZ      14             22.74475062            1.62462504           0.02            1.0000
```

Table 30. ANOVA of Performance Measures for Two Factors.

(a) Performance Measure: MADV.

```
Dependent Variable: MADV

Source                DF         Sum of Squares          Mean Square       F Value          Pr > F

Model                 23             3.53588586           0.15373417         59.19          0.0001

Error                936             2.43112412           0.00259735

Corrected Total      959             5.96700999

                R-Square                   C.V.             Root MSE                     MADV Mean

                0.592572               50.85211           0.05096425                    0.10022052


Source                DF              Type I SS          Mean Square       F Value          Pr > F

CHALLNG                2             2.48333445           1.24166722        478.05          0.0001
NETSIZE                7             0.80119819           0.11445688         44.07          0.0001
CHALLNG*NETSIZE       14             0.25135323           0.01795380          6.91          0.0001

Source                DF            Type III SS          Mean Square       F Value          Pr > F

CHALLNG                2             2.48333445           1.24166722        478.05          0.0001
NETSIZE                7             0.80119819           0.11445688         44.07          0.0001
CHALLNG*NETSIZE       14             0.25135323           0.01795380          6.91          0.0001
```

Table 30 cont.

(b) Performance Measure: Absolute Relative Error of the Average.

```
Dependent Variable: AVGERR

Source                 DF         Sum of Squares          Mean Square       F Value          Pr > F

Model                  23         6529.32325156         283.88361963         65.10          0.0001

Error                 936         4081.65589750           4.36074348

Corrected Total       959        10610.97914906

                 R-Square                  C.V.             Root MSE                    AVGERR Mean

                 0.615337              87.98259           2.08823933                     2.37346875

Source                 DF            Type I SS          Mean Square       F Value          Pr > F

CHALLNG                 2         5965.36351187        2982.68175594        683.98          0.0001
NETSIZE                 7          261.38451323          37.34064475          8.56          0.0001
CHALLNG*NETSIZE        14          302.57522646          21.61251618          4.96          0.0001

Source                 DF          Type III SS          Mean Square       F Value          Pr > F

CHALLNG                 2         5965.36351187        2982.68175594        683.98          0.0001
NETSIZE                 7          261.38451323          37.34064475          8.56          0.0001
CHALLNG*NETSIZE        14          302.57522646          21.61251618          4.96          0.0001
```

Table 30 cont.

(c) Performance Measure: Absolute Relative Error of the Standard Deviation.

```
Dependent Variable: STDERR

Source                  DF          Sum of Squares          Mean Square       F Value          Pr >·F

Model                   23          70891.61353999        3082.24406696         31.36          0.0001

Error                  936          91999.03590002          98.28956827

Corrected Total        959         162890.64944001

                  R-Square                    C.V.             Root MSE                   STDERR Mean

                  0.435210                50.50232           9.91410956                   19.63100000

Source                  DF             Type I SS          Mean Square       F Value          Pr > F

CHALLNG                  2        17987.35609312        8993.67804656         91.50          0.0001
NETSIZE                  7        48866.98891166        6980.99841595         71.02          0.0001
CHALLNG*NETSIZE         14         4037.26853521         288.37632394          2.93          0.0002

Source                  DF           Type III SS          Mean Square       F Value          Pr > F

CHALLNG                  2        17987.35609312        8993.67804656         91.50          0.0001
NETSIZE                  7        48866.98891166        6980.99841595         71.02          0.0001
CHALLNG*NETSIZE         14         4037.26853521         288.37632394          2.93          0.0002
```

Table 31. Moments of Performance Measures for Significant Factors (by Measures).

(a) Performance Measure: MADV.

Level of		MADV	
CHALLNG	N	Mean	SD
1	320	0.05291875	0.03181785
2	320	0.07694469	0.04766603
3	320	0.17079812	0.08738528

Level of		MADV	
NETSIZE	N	Mean	SD
1	120	0.04974917	0.04521256
2	120	0.08348500	0.07325959
3	120	0.08566333	0.05817537
4	120	0.09168333	0.06406115
5	120	0.11879500	0.07304904
6	120	0.09674333	0.05375412
7	120	0.12404750	0.08892510
8	120	0.15159750	0.11125419

(b) Performance Measure: Absolute Relative Error of the Average.

Level of		AVGERR	
CHALLNG	N	Mean	SD
1	320	0.23900000	0.28359360
2	320	1.01090625	0.91793560
3	320	5.87050000	3.69324076

Level of		AVGERR	
NETSIZE	N	Mean	SD
1	120	1.65216667	2.56419338
2	120	2.46466667	3.49425193
3	120	2.01066667	2.71845907
4	120	2.12725000	2.91916990
5	120	2.67633333	3.23805599
6	120	1.88900000	2.48214037
7	120	2.83716667	3.92840169
8	120	3.33050000	4.49405823

(c) Performance Measure: Absolute Relative Error of the Standard Deviation.

Level of		STDERR	
CHALLNG	N	Mean	SD
1	320	16.6824062	9.8132803
2	320	16.4593750	9.4180198
3	320	25.7512187	16.4086176

Level of		STDERR	
NETSIZE	N	Mean	SD
1	120	5.8103333	3.6089664
2	120	11.1362500	8.1841620
3	120	18.3445833	10.3571760
4	120	21.4225833	12.2668376
5	120	21.6905833	10.8426521
6	120	25.6892500	14.0754225
7	120	24.9766667	12.1602751
8	120	27.9777500	12.5250773

Table 32. Moments and Ranges of Performance Measures for Significant Factors (by Factors).

(a) Factor: Challenge of Probability Distribution Functions.

```
 CHALLNG  N Obs  Variable    N        Mean         Std Dev        Minimum        Maximum
-----------------------------------------------------------------------------------------
       1   320   MADV       320    0.0529187     0.0318178      0.0089000      0.1626000
                 AVGERR     320    0.2390000     0.2835936              0      2.2100000
                 STDERR     320   16.6824062     9.8132803      0.4000000     41.8800000

       2   320   MADV       320    0.0769447     0.0476660      0.0077000      0.2690000
                 AVGERR     320    1.0109062     0.9179356      0.0200000      4.5000000
                 STDERR     320   16.4593750     9.4180198      0.0700000     40.9800000

       3   320   MADV       320    0.1707981     0.0873853      0.0196000      0.5094000
                 AVGERR     320    5.8705000     3.6932408      0.0400000     18.4000000
                 STDERR     320   25.7512187    16.4086176      0.1600000     88.7500000
-----------------------------------------------------------------------------------------
```

Table 32 cont.

(b) Factor: Network Size.

NETSIZE	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
1	120	MADV	120	0.0497492	0.0452126	0.0077000	0.1941000
		AVGERR	120	1.6521667	2.5641934	0.0200000	10.9300000
		STDERR	120	5.8103333	3.6089664	0.1600000	16.1700000
2	120	MADV	120	0.0834850	0.0732596	0.0094000	0.3194000
		AVGERR	120	2.4646667	3.4942519	0.0300000	14.0300000
		STDERR	120	11.1362500	8.1841620	0.1200000	43.5600000
3	120	MADV	120	0.0856633	0.0581754	0.0170000	0.2558000
		AVGERR	120	2.0106667	2.7184591	0	9.9500000
		STDERR	120	18.3445833	10.3571760	0.5100000	54.5700000
4	120	MADV	120	0.0916833	0.0640612	0.0143000	0.2887000
		AVGERR	120	2.1272500	2.9191699	0	11.1600000
		STDERR	120	21.4225833	12.2668376	0.2600000	59.9400000
5	120	MADV	120	0.1187950	0.0730490	0.0214000	0.3610000
		AVGERR	120	2.6763333	3.2380560	0.0200000	12.6200000
		STDERR	120	21.6905833	10.8426521	0.0700000	50.9200000
6	120	MADV	120	0.0967433	0.0537541	0.0203000	0.2593000
		AVGERR	120	1.8890000	2.4821404	0.0200000	10.8300000
		STDERR	120	25.6892500	14.0754225	7.8500000	88.7500000
7	120	MADV	120	0.1240475	0.0889251	0.0188000	0.3853000
		AVGERR	120	2.8371667	3.9284017	0	17.4100000
		STDERR	120	24.9766667	12.1602751	9.0000000	72.1700000
8	120	MADV	120	0.1515975	0.1112542	0.0212000	0.5094000
		AVGERR	120	3.3305000	4.4940582	0.0200000	18.4000000
		STDERR	120	27.9777500	12.5250773	5.6800000	62.5600000

Table 33. Moments and Ranges of Performance Measures for Significant Factors (by Experimental Cases).

(a) Factors: Low Challenge Level by Network Size.

CHALLNG	NETSIZE	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
1	1	40	MADV	40	0.0267075	0.0184505	0.0089000	0.0740000
			AVGERR	40	0.1937500	0.1857996	0.0400000	0.6400000
			STDERR	40	5.5230000	3.0257936	0.4000000	12.4200000
	2	40	MADV	40	0.0304700	0.0208383	0.0094000	0.0985000
			AVGERR	40	0.1530000	0.1600513	0.0300000	0.6600000
			STDERR	40	8.4567500	4.1304792	1.6800000	16.9500000
	3	40	MADV	40	0.0538150	0.0323698	0.0170000	0.1486000
			AVGERR	40	0.3042500	0.4766167	0	2.2100000
			STDERR	40	17.3652500	9.9626073	2.5800000	40.6900000
	4	40	MADV	40	0.0479625	0.0277126	0.0143000	0.1200000
			AVGERR	40	0.1740000	0.2287850	0	0.8200000
			STDERR	40	16.9205000	8.9079950	3.7500000	36.5500000
	5	40	MADV	40	0.0670350	0.0278773	0.0214000	0.1231000
			AVGERR	40	0.3067500	0.2441562	0.0200000	0.9400000
			STDERR	40	20.1752500	8.0525505	8.4600000	37.9400000
	6	40	MADV	40	0.0685850	0.0305579	0.0203000	0.1626000
			AVGERR	40	0.3667500	0.3319591	0.0600000	1.5100000
			STDERR	40	19.9585000	6.5631956	8.1100000	29.6000000
	7	40	MADV	40	0.0601600	0.0314311	0.0188000	0.1515000
			AVGERR	40	0.2122500	0.2540491	0	1.0700000
			STDERR	40	21.2255000	10.0763241	9.0000000	41.8800000
	8	40	MADV	40	0.0686150	0.0313261	0.0212000	0.1302000
			AVGERR	40	0.2012500	0.2044905	0.0200000	0.7700000
			STDERR	40	23.8345000	8.8658289	9.4200000	38.1100000

Table 33 cont.

(b) Factors: Medium Challenge Level by Network Size.

CHALLNG	NETSIZE	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
2	1	40	MADV	40	0.0288125	0.0190300	0.0077000	0.0766000
			AVGERR	40	0.5122500	0.5006790	0.0200000	1.8000000
			STDERR	40	4.9185000	2.5598784	0.1600000	11.6000000
	2	40	MADV	40	0.0658950	0.0441338	0.0228000	0.1726000
			AVGERR	40	1.0375000	1.0392126	0.0600000	4.2400000
			STDERR	40	8.6050000	6.2344362	0.1200000	24.1900000
	3	40	MADV	40	0.0615300	0.0275670	0.0266000	0.1397000
			AVGERR	40	0.6565000	0.6533760	0.0200000	2.5600000
			STDERR	40	15.8530000	6.2201122	0.5100000	25.2600000
	4	40	MADV	40	0.0715275	0.0298441	0.0226000	0.1336000
			AVGERR	40	0.9652500	0.7353736	0.1400000	2.8000000
			STDERR	40	16.9037500	8.4382851	0.2600000	32.8800000
	5	40	MADV	40	0.0958775	0.0423116	0.0413000	0.1838000
			AVGERR	40	1.3832500	0.8720044	0.1500000	3.0500000
			STDERR	40	17.6270000	9.0397692	0.0700000	37.0400000
	6	40	MADV	40	0.0720075	0.0235411	0.0416000	0.1186000
			AVGERR	40	0.8392500	0.5492064	0.0200000	2.5400000
			STDERR	40	20.6875000	7.4274635	7.8500000	38.0300000
	7	40	MADV	40	0.0966350	0.0590836	0.0365000	0.2690000
			AVGERR	40	1.1122500	1.1407791	0.0200000	4.5000000
			STDERR	40	22.0332500	6.4877942	11.4500000	32.9600000
	8	40	MADV	40	0.1232725	0.0549691	0.0391000	0.2368000
			AVGERR	40	1.5810000	1.1521102	0.0300000	4.1000000
			STDERR	40	25.0470000	7.9702602	12.9800000	40.9800000

Table 33 cont.

(c) Factors: High Challenge Level by Network Size.

CHALLNG	NETSIZE	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
3	1	40	MADV	40	0.0937275	0.0505667	0.0196000	0.1941000
			AVGERR	40	4.2505000	3.0559047	0.2800000	10.9300000
			STDERR	40	6.9895000	4.6590683	0.1600000	16.1700000
	2	40	MADV	40	0.1540900	0.0753739	0.0405000	0.3194000
			AVGERR	40	6.2035000	3.7739624	0.2200000	14.0300000
			STDERR	40	16.3470000	10.3278226	0.8700000	43.5600000
	3	40	MADV	40	0.1416450	0.0605554	0.0594000	0.2558000
			AVGERR	40	5.0712500	2.7240243	0.1700000	9.9500000
			STDERR	40	21.8155000	13.0255136	3.1100000	54.5700000
	4	40	MADV	40	0.1555600	0.0655839	0.0326000	0.2887000
			AVGERR	40	5.2425000	3.1870119	0.3700000	11.1600000
			STDERR	40	30.4435000	13.5421614	9.1500000	59.9400000
	5	40	MADV	40	0.1934725	0.0686187	0.0566000	0.3610000
			AVGERR	40	6.3390000	3.1527398	0.2100000	12.6200000
			STDERR	40	27.2695000	12.6863549	3.1300000	50.9200000
	6	40	MADV	40	0.1496375	0.0549402	0.0722000	0.2593000
			AVGERR	40	4.4610000	2.8452577	0.0400000	10.8300000
			STDERR	40	36.4217500	18.1324863	12.5700000	88.7500000
	7	40	MADV	40	0.2153475	0.0783795	0.0706000	0.3853000
			AVGERR	40	7.1870000	4.0246505	0.5900000	17.4100000
			STDERR	40	31.6712500	15.4367022	9.8100000	72.1700000
	8	40	MADV	40	0.2629050	0.1147681	0.1110000	0.5094000
			AVGERR	40	8.2092500	4.7603773	1.2700000	18.4000000
			STDERR	40	35.0517500	16.0875387	5.6800000	62.5600000

special consideration for the presence of any exponential distributions. If the number of classes for polygonal approximation is increased, errors build up at a slower rate, so the performance of PART algorithms would be improved compared to the results of the validation experiments. Nonetheless, the results of the validation experiments were comparable to previously reported results (Tables 23 an 24) except at the upper end of the experimental cases: networks with 50 or more nodes and 75 or more activities, coupled with the high challenge factor level distributions; only there were some MADVs experienced which were statistically significant at the 5% level. The validation experiments demonstrated the following properties of PART algorithm performance:

(1) The performance of the PART-ind and PART-seq algorithms is virtually identical, as expected.

(2) The accuracy of PART algorithms to approximate the throughput distribution of a stochastic activity network is a function of both network size and structure and how challenging the activity duration distributions are to polygonal approximation-based series-parallel reduction operations, and there is a weak interaction between these two factors.

(3) Lack-of-fit of the approximated throughput distribution and error in the approximation of the mean increase slowly, almost monotonically with increasing network size and structure and challenge of the activity distributions.

(4) Error in approximation of the standard deviation of the throughput distribution also increases slowly, almost monotonically with increasing network size and structure, but is more sensitive to how "high" the variances of activity distributions are and how much overlap there is among their domains.

4.4 PART Algorithm Performance Based on Activity, Node, and Path Criticality

In this section, validation of a PART algorithm for identifying the K most stochastically dominating paths for large network approximation and reduction is discussed. Since the topic of identifying the K most critical paths in a stochastic activity network has received little attention in the literature, there is only limited reported experience in testing large-network algorithms upon which to draw. Dodin (1984) proposed an approach: approximating the K most critical paths with the K most stochastically dominating paths, obtained from a heuristic based on the sequential approximation method, with which he reported preliminary computational experience (Section 2.8.2 [above]). Validation of the PART algorithm was patterned after Dodin's testing of the heuristic, extended to include the broad elements of the experimental design for algorithm testing (Section 3.5.1 [above]). The PART algorithm approximates activity criticality, normalized activity criticality, node criticality, and normalized node criticality indices, in addition to identifying the K most stochastically dominating paths (Section 3.4.2 [above]). For all the test networks against which the algorithm was exercised, the simulation-approximated criticality indices from extensive Monte Carlo simulations of these networks (Section 3.5.3 [above]) were taken as the "true" values of these indices, since the actual indices cannot be obtained. From these simulations, the K most critical paths were obtained from the rankings of the averaged simulation-approximated activity criticality indices CA of the activities on each path, as the value of the simulation-approximated path criticality index CR of the path. While these results were taken as the "true" K most critical paths, it must be recognized that they may not be exact, since the averages of simulation-approximated CAs of activities on paths which contain common arcs are non-uniformly inflated relative to the averages of simulation-approximated CAs of activities on paths which contain no common arcs or whose common arcs have CAs equal to zero, and this inflation can alter the ranking of the paths. However, in large networks the relative

```
007 0010 5
```

Figure 50. PART-paths program input (control data file).

ranking of paths based on averaged simulation-approximated *CA* s of activities on paths should not change compared to the ranking which could be obtained from path enumeration, except under unusual circumstances (Section 3.5.3 [above]). Example data inputs and outputs for the programs included in Appendices C and F are included here.

4.4.1 Performance Against Selected Test Networks

Since Dodin (1984) is the only researcher who has reported test results for *K*-most-critical-paths approximation, the only test networks discussed in the literature are those which were used in his tests. These networks, which were randomly generated, are discussed in the next section. Consequently, to verify the PART algorithm for identifying the *K* most stochastically dominating paths (labeled PART-paths), a small set of small-sized networks which could be manually reduced was tested by comparing the *K* most critical paths obtained by manual reduction with the *K* most stochastically dominating paths obtained from the PART-paths algorithm. As an example, one of these networks, the "conditional" network in Figure 42 (Section 4.3 [above]), is discussed here. This network has seven nodes and ten activities, and there are five paths through the network from source to sink. CPM analysis of the network ranks the five paths as follows:

Rank 1: 1→2→5→7
Rank 2: 1→4→6→7
Rank 3: 1→3→6→7
Rank 4: 1→3→5→7
Rank 5: 1→4→7

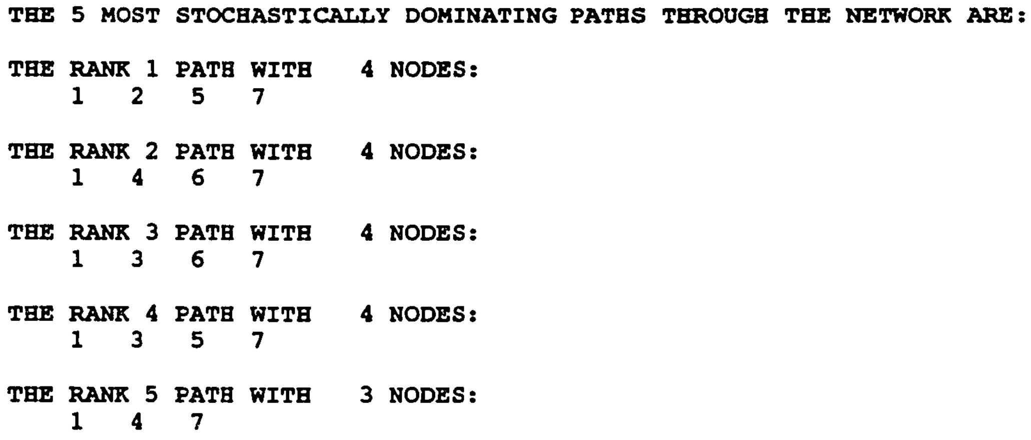
and it is easily verified that this is the "true" ranking of the critical paths.

Figure 50 shows the control data input file (CONTROL.PATHS) for the PART-paths program; the network data file (DATAN.PATHS) is the same as the DATAN.DAT file in

Part (b) of Figure 43, and the activity duration distributions data file (DATAH.PATHS) is the same as the DATAH.DAT file in Part (c) of Figure 43. The formats of these files are described in comment statements at the top of the code listing in Appendix C. Figure 51 depicts the output from the PART-paths program, which includes the approximated activity criticality, normalized activity criticality, node criticality, and normalized node criticality indices, and the K most stochastically dominating paths through the network. The activity criticality and normalized activity criticality indices are presented backward through the network from the sink node, as predecessor activities of their respective end nodes, in the order in which they are computed (Section 3.4.2 [above]). The node criticality and normalized criticality indices are presented below the activity indices, since their computation depends on the values of the activity indices. The node criticality and normalized node criticality indices of the source node and the sink node are always equal to one, since all critical paths depart from the source node and terminate at the sink node. In the PART-paths program, the node criticality indices are normalized by the scale factor necessary to make the normalized node criticality index of the sink node equal to exactly one. If the normalized node criticality index of the source node is not exactly equal to one, the difference reflects built-up error from polygonal approximation.

Up to the top five most stochastically dominating paths may be requested from the PART-paths program, since the heuristic can not only be used to identify the most critical path, but in networks with low densities, the second, third,..., and K^{th} critical paths. In dense networks it can only be used to approximate the set of the K most critical paths without consideration of the path's rank in the set (Section 2.8.2 [above]). For the "conditional" network, the five most stochastically dominating paths obtained from the heuristic implemented with polygonal approximation were identical to the five critical paths obtained manually. Although the K paths identified by the heuristic are, in most cases, the K most critical paths, there are counterinstances where this does not hold (Dodin, 1984).

```
FROM THE POLYGONAL APPROXIMATION AND REDUCTION TECHNIQUE:

THE ACTIVITIES ENDING AT NODE   7 AND THEIR CRITICALITY INDICES ARE:
                                                    NORMALIZED
STARTING   CRITICALITY   CRITICALITY
   NODE        INDEX         INDEX
    4         .00000        .00000
    5         .77696        .77706
    6         .22291        .22294

THE ACTIVITIES ENDING AT NODE   6 AND THEIR CRITICALITY INDICES ARE:
                                                    NORMALIZED
STARTING   CRITICALITY   CRITICALITY
   NODE        INDEX         INDEX
    3         .02026        .02027
    4         .20777        .20779

THE ACTIVITIES ENDING AT NODE   5 AND THEIR CRITICALITY INDICES ARE:
                                                    NORMALIZED
STARTING   CRITICALITY   CRITICALITY
   NODE        INDEX         INDEX
    2         .77230        .77240
    3         .00331        .00331

THE ACTIVITIES ENDING AT NODE   4 AND THEIR CRITICALITY INDICES ARE:
                                                    NORMALIZED
STARTING   CRITICALITY   CRITICALITY
    NODE        INDEX         INDEX
     1         .20777        .20779

THE ACTIVITIES ENDING AT NODE   3 AND THEIR CRITICALITY INDICES ARE:
                                                    NORMALIZED
STARTING   CRITICALITY   CRITICALITY
   NODE        INDEX         INDEX
    1         .02358        .02358

THE ACTIVITIES ENDING AT NODE   2 AND THEIR CRITICALITY INDICES ARE:
                                                    NORMALIZED
STARTING   CRITICALITY   CRITICALITY
   NODE        INDEX         INDEX
    1         .77230        .77240

THE CRITICALITY INDICES OF THE NODES ARE:
                                            NORMALIZED
                  CRITICALITY   CRITICALITY
   NODE        INDEX         INDEX
    1        1.00365       1.00378
    2         .77230        .77240
    3         .02358        .02358
    4         .20777        .20779
    5         .77696        .77706
    6         .22291        .22294
    7         .99987       1.00000
```



```
THE 5 MOST STOCHASTICALLY DOMINATING PATHS THROUGH THE NETWORK ARE:

THE RANK 1 PATH WITH   4 NODES:
    1   2   5   7

THE RANK 2 PATH WITH   4 NODES:
    1   4   6   7

THE RANK 3 PATH WITH   4 NODES:
    1   3   6   7

THE RANK 4 PATH WITH   4 NODES:
    1   3   5   7

THE RANK 5 PATH WITH   3 NODES:
    1   4   7
```

Figure 51. PART-paths program output.

4.4.2 Performance Against Reported Performance of Competing Procedures

Dodin (1984) is the only researcher who has reported performance results for a competing approximation procedure implementing the heuristic: discretization approximation. In a performance analysis conducted when he originally proposed the heuristic, he applied it to 14 "strongly random" networks to identify the three most stochastically dominating paths, and he simulated the networks to identify the three most critical paths. In each network, the pdf's in Table 10 (in Section 2.8.2 [above]) were used equally. The accuracy of the heuristic was measured by the closeness between the set of stochastically dominating paths, identified by the procedure implemented with discretization approximation, and the set of corresponding critical paths, identified by the simulation. Table 11 shows the sizes of the test networks and the closeness between the set of stochastically dominating paths and the set of critical paths. A performance measure entry of "1" indicates that the K^{th} stochastically dominating path was identical to the corresponding critical path, whereas an entry of "0" indicates that the two paths were not the same, but differed by at least one arc. Dodin observed that most of the nonmatching paths occurred in networks with arc to node densities higher than two, which he attributed to the large number of paths in dense networks with similar or close durations. For the 14 networks tested, he reported that the three most critical (or stochastically dominating) paths had many arcs in common, i.e., many of the arcs of the most critical (stochastically dominating) paths were constituents of the other critical (stochastically dominating) paths.

As is the case for his reported performance analysis of a sequential approximation algorithm for approximating network throughput distribution, Dodin's results of the performance of the heuristic implemented with discretization approximation were reported for a single "randomly generated network" tested with a particular network size and assignment of activity distributions. In each case, only one network was tested because the network's critical paths had to be obtained through exhaustive enumeration by simulation, which required extensive computational resources. However, since only one network was tested for each case considered, the results are without value to the characterization of the performance of the approximation algorithm implementing the heuristic. Again, it is not possible to compare directly the reported performance of the heuristic implemented with discretization approximation with the performance of polygonal approximation.

4.4.3 Performance Against Test Networks Constructed within an Experimental Design

Again following the experimental design for algorithm testing (Section 3.5.1 [above]), a series of validation experiments was conducted for the PART-paths algorithm based on how great a challenge a randomly constructed test network presented to the algorithm. Since the performance of the heuristic implemented with polygonal approximation is reflected in the closeness between the sets of the K most stochastically dominating paths and the K most critical paths - here obtained from the rankings of the averaged simulation-approximated activity criticality indices - in order to effectively describe the performance in quantitative terms, a better performance measure was needed than Dodin's "1"s and "0"s. Based on the degree of matching between the first most stochastically dominant path and the first most critical path, the seconds, etc., as reported by Dodin, a "matching category" performance measure was constructed, with three categories, based on the nodes of a path:

1. "All": all the nodes in the K^{th} most stochastically dominant path match the nodes in the K^{th} most critical path.
2. "1-4": the number of mismatches between the nodes in the K^{th} most stochastically dominant path and the nodes in the K^{th} most critical path is between one and four.
3. ">4": the number of mismatches between the nodes in the K^{th} most stochastically dominant path and the nodes in the K^{th} most critical path is greater than four.

Since this "matching category" performance measure is a categorical variable, not a numerically valued variable, a traditional ANOVA experimental design could not be

```
020 020 0060 3 10000 06
```

Figure 52. PART-paths(val) program input (control data file).

employed. As in the validation of the PART-ind and PART-seq algorithms, two design factors were of interest: network size and structure, measured by the number of nodes and number of activities in a network; and challenge of the probability distribution functions of activity duration. To effect as reasonable and meaningful comparison as possible with Dodin's reported results, the 14 network sizes in Table 11 were used, although both the number of nodes and the number of activities could have been randomly generated (Section 3.5.1 [above]). The three sets of activity duration distributions with the high, medium, and low challenge-factor levels were again used, along with the distributions in Table 10. The three most stochastically dominant paths were identified and compared with the three most critical paths obtained from the rankings of the averaged simulation-approximated activity criticality indices. The "matching category" variable value was developed for each path for each network tested, so that the performance of the heuristic implemented by polygonal approximation could be described by its matching characteristics based on the path number. Additionally, the combined set of nodes in the three most stochastically dominant paths and the combined set of nodes in the three most critical paths were developed and compared. As similar results were experienced for all the experiments, only one is discussed in detail here - the experiment based on Dodin's reported results.

The test networks were generated and then the heuristic applied to them by the PART-paths validation program [labeled PART-paths(val)]. Figure 52 shows the control data input file (CONTROL.PATHS-RNETGEN) for the PART-paths(val) program; the activity duration distributions data file (DATAH.PATHS-RNETGEN) is the same as the DATAH.DAT file in Part (c) of Figure 43. The formats for these files are described in comment statements at the top of the code listing in Appendix F. Figure 53 depicts the output from the PART-paths(val) program for the first network from the set of 20 "strongly random" networks generated and tested with 20 nodes and 60 activities. The output includes the approximated activity criticality, normalized activity criticality, node criticality, and normalized criticality indices obtained from the heuristic implemented with polygonal approximation and from simulation (Section 3.5.3 [above]), a comparison table of the activity criticality indices, the *K* most stochastically dominating paths obtained from the heuristic implemented with polygonal approximation, the *K* most critical paths obtained from the rankings of the averaged simulation-approximated activity criticality indices, and a comparison table of the two path sets which shows the number of nodes in common in each path pair. Table 34 presents the values of the "matching category" performance measure for the pairwise comparisons between the sets of nodes of the first, second, and third most stochastically dominant paths and the sets of nodes of the first, second, and third most critical paths for the first network from the set of 20 networks generated and tested for each of the 14 different network sizes.

Taken collectively across the 20 networks generated and tested, the results of the "matching category" performance measure were as follows. For the first path, "all" nodes matched between the set of stochastically dominant paths and the set of critical paths for 86% of the comparisons, there were "1-4" node mismatches for 12% of the comparisons, and there were ">4" node mismatches for 2% of the comparisons. For the second path, "all" was 26%, and "1-4" was 74%. For the third path, "all" was 19%, "1-4" was 79%, and ">4" was 2%. The "matching category" values of "1-4" and ">4", however, constitute a "Heisenberg uncertainty"-like area: because the *K* most critical paths obtained from the rankings of the averaged simulation-approximated activity criticality indices may not be exact, it is not possible to ascribe the cause of any node mismatch necessarily to either error from polygonal approximation or error from critical path determination. To isolate error from only polygonal approximation, critical paths must be obtained through exhaustive

```
THE RESULTS FOR NETWORK NUMBER   1 OF   20 NETWORKS GENERATED ARE:


FROM THE POLYGONAL APPROXIMATION AND REDUCTION TECHNIQUE:

THE ACTIVITIES ENDING AT NODE  20 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING   CRITICALITY  CRITICALITY
  NODE        INDEX        INDEX
    2         .00000       .00000
   10         .00091       .00090
   11         .00025       .00025
   18         .39895       .39668
   19         .60561       .60217

THE ACTIVITIES ENDING AT NODE  19 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING   CRITICALITY  CRITICALITY
  NODE        INDEX        INDEX
    1         .00000       .00000
    7         .00026       .00026
    9         .00000       .00000
   12         .03000       .02983
   13         .02440       .02426
   14         .22520       .22392
   15         .31462       .31283
   16         .15806       .15716
   17         .04454       .04429

THE ACTIVITIES ENDING AT NODE  18 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING   CRITICALITY  CRITICALITY
  NODE        INDEX        INDEX
    2         .00000       .00000
    5         .00001       .00001
    6         .00000       .00000
    8         .00205       .00204
   10         .00517       .00514
   15         .30067       .29895
   16         .14400       .14318
   17         .04617       .04591

THE ACTIVITIES ENDING AT NODE  17 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING   CRITICALITY  CRITICALITY
  NODE        INDEX        INDEX
    2         .00001       .00001
    3         .00001       .00001
   11         .00655       .00651
   13         .05975       .05941

THE ACTIVITIES ENDING AT NODE  16 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING   CRITICALITY  CRITICALITY
  NODE        INDEX        INDEX
    2         .00014       .00014
   13         .30654       .30480
```

```
THE ACTIVITIES ENDING AT NODE  15 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    1        .00019       .00019
   14        .69499       .69104

THE ACTIVITIES ENDING AT NODE  14 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
   13        .92019       .91495

THE ACTIVITIES ENDING AT NODE  13 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    1        .00291       .00289
    2        .01405       .01397
    4        .02696       .02680
   11        .39036       .38814
   12        .60716       .60371

THE ACTIVITIES ENDING AT NODE  12 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    1        .01174       .01167
    3        .01687       .01677
    5        .08005       .07960
   10        .60135       .59793

THE ACTIVITIES ENDING AT NODE  11 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    2        .06223       .06187
    3        .05151       .05122
    4        .17731       .17631
    5        .19198       .19089
    6        .09573       .09519
    8        .32711       .32525

THE ACTIVITIES ENDING AT NODE  10 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    3        .04102       .04079
    7        .35263       .35062
    8        .41343       .41108

THE ACTIVITIES ENDING AT NODE   9 AND THEIR CRITICALITY INDICES ARE:
                                NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    3        .00000       .00000
```

```
THE ACTIVITIES ENDING AT NODE   8 AND THEIR CRITICALITY INDICES ARE:
                                 NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    2       .06519       .06482
    5       .16952       .16856
    7       .22144       .22018

THE ACTIVITIES ENDING AT NODE   7 AND THEIR CRITICALITY INDICES ARE:
                                 NORMALIZED
STARTING  CRITICALITY  CRITICALITY '
  NODE       INDEX        INDEX
    2       .15224       .15137
    6       .22779       .22649

THE ACTIVITIES ENDING AT NODE   6 AND THEIR CRITICALITY INDICES ARE:
                                 NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    3       .32352       .32168

THE ACTIVITIES ENDING AT NODE   5 AND THEIR CRITICALITY INDICES ARE:
                                 NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    2       .44158       .43906

THE ACTIVITIES ENDING AT NODE   4 AND THEIR CRITICALITY INDICES ARE:
                                 NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    3       .20427       .20311

THE ACTIVITIES ENDING AT NODE   3 AND THEIR CRITICALITY INDICES ARE:
                                 NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    1       .63720       .63358

THE ACTIVITIES ENDING AT NODE   2 AND THEIR CRITICALITY INDICES ARE:
                                 NORMALIZED
STARTING  CRITICALITY  CRITICALITY
  NODE       INDEX        INDEX
    1       .73543       .73124
```

```
THE CRITICALITY INDICES OF THE NODES ARE:
                                NORMALIZED
                CRITICALITY   CRITICALITY
 NODE              INDEX         INDEX
   1             1.38747       1.37958
   2              .73543        .73124
   3              .63720        .63358
   4              .20427        .20311
   5              .44158        .43906
   6              .32352        .32168
   7              .57433        .57106
   8              .74260        .73838
   9              .00000        .00000
  10              .60742        .60397
  11              .39717        .39491
  12              .63716        .63354
  13             1.31088       1.30342
  14              .92019        .91495
  15              .61529        .61179
  16              .30206        .30035
  17              .09071        .09019
  18              .39895        .39668
  19              .60561        .60217
  20             1.00572       1.00000
```

```
FROM MONTE CARLO SIMULATION:

THE ACTIVITIES ENDING AT NODE  20 AND THEIR CRITICALITY INDICES ARE:
          NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE        PATH          INDEX         INDEX
    2            0         .00000        .00000
   10            0         .00000        .00000
   11            0         .00000        .00000
   18         4708         .47080        .46734
   19         5366         .53660        .53266

THE ACTIVITIES ENDING AT NODE  19 AND THEIR CRITICALITY INDICES ARE:
          NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE        PATH          INDEX         INDEX
    1            0         .00000        .00000
    7            0         .00000        .00000
    9            0         .00000        .00000
   12            1         .00010        .00010
   13            0         .00000        .00000
   14         1159         .11590        .11505
   15         3552         .35520        .35259
   16          643         .06430        .06383
   17           11         .00110        .00109

THE ACTIVITIES ENDING AT NODE  18 AND THEIR CRITICALITY INDICES ARE:
          NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE        PATH          INDEX         INDEX
    2            0         .00000        .00000
    5            0         .00000        .00000
    6            0         .00000        .00000
    8            0         .00000        .00000
   10            0         .00000        .00000
   15         3962         .39620        .39329
   16          735         .07350        .07296
   17           11         .00110        .00109

THE ACTIVITIES ENDING AT NODE  17 AND THEIR CRITICALITY INDICES ARE:
          NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE        PATH          INDEX         INDEX
    2            0         .00000        .00000
    3            0         .00000        .00000
   11            2         .00020        .00020
   13           20         .00200        .00199

THE ACTIVITIES ENDING AT NODE  16 AND THEIR CRITICALITY INDICES ARE:
          NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE        PATH          INDEX         INDEX
    2            0         .00000        .00000
   13         1378         .13780        .13679

THE ACTIVITIES ENDING AT NODE  15 AND THEIR CRITICALITY INDICES ARE:
          NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE        PATH          INDEX         INDEX
    1            0         .00000        .00000
   14         7514         .75140        .74588
```

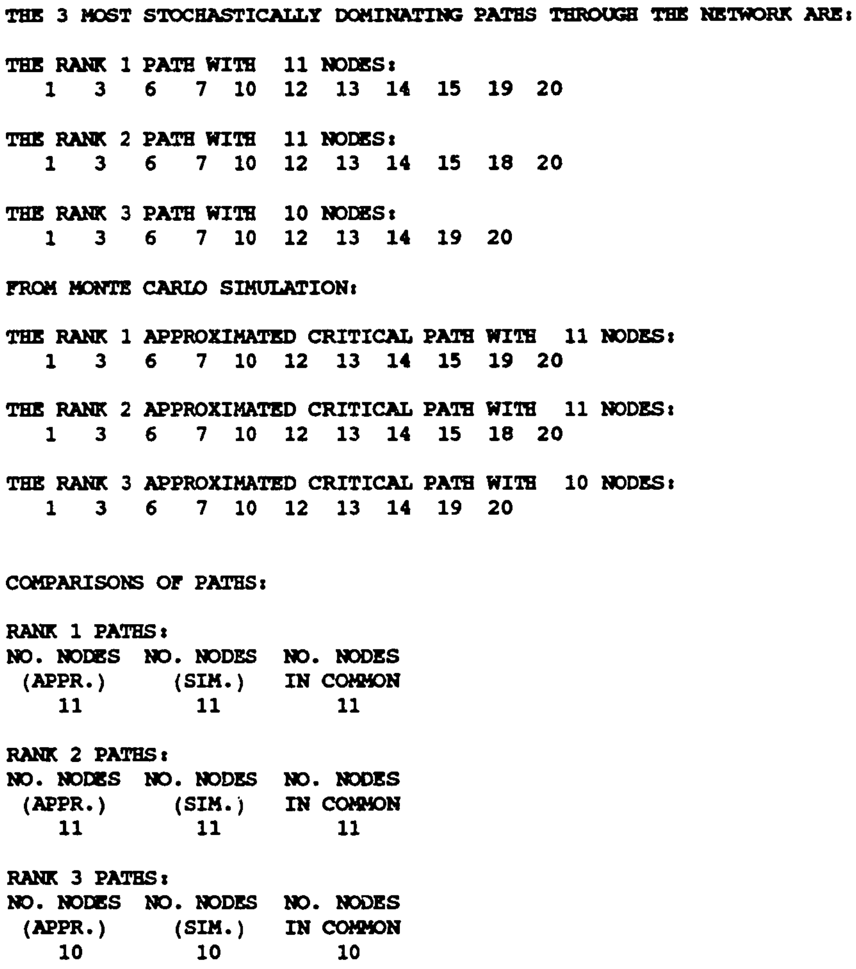
```
THE ACTIVITIES ENDING AT NODE  14 AND THEIR CRITICALITY INDICES ARE:
           NO. TIMES ON               NORMALIZED
STARTING    A CRITICAL   CRITICALITY  CRITICALITY
  NODE         PATH         INDEX        INDEX
   13          8673        .86730       .86093

THE ACTIVITIES ENDING AT NODE  13 AND THEIR CRITICALITY INDICES ARE:
           NO. TIMES ON               NORMALIZED
STARTING    A CRITICAL   CRITICALITY  CRITICALITY
  NODE         PATH         INDEX        INDEX
    1             0        .00000       .00000
    2             0        .00000       .00000
    4             4        .00040       .00040
   11          3157        .31570       .31338
   12          6910        .69100       .68592

THE ACTIVITIES ENDING AT NODE  12 AND THEIR CRITICALITY INDICES ARE:
           NO. TIMES ON               NORMALIZED
STARTING    A CRITICAL   CRITICALITY  CRITICALITY
  NODE         PATH         INDEX        INDEX
    1             0        .00000       .00000
    3             0        .00000       .00000
    5            59        .00590       .00586
   10          6852        .68520       .68017

THE ACTIVITIES ENDING AT NODE  11 AND THEIR CRITICALITY INDICES ARE:
           NO. TIMES ON               NORMALIZED
STARTING    A CRITICAL   CRITICALITY  CRITICALITY
  NODE         PATH         INDEX        INDEX
    2             0        .00000       .00000
    3             0        .00000       .00000
    4           602        .06020       .05976
    5           542        .05420       .05380
    6            10        .00100       .00099
    8          2005        .20050       .19903

THE ACTIVITIES ENDING AT NODE  10 AND THEIR CRITICALITY INDICES ARE:
           NO. TIMES ON               NORMALIZED
STARTING    A CRITICAL   CRITICALITY  CRITICALITY
  NODE         PATH         INDEX        INDEX
    3             0        .00000       .00000
    7          4366        .43660       .43339
    8          2486        .24860       .24677

THE ACTIVITIES ENDING AT NODE   9 AND THEIR CRITICALITY INDICES ARE:
           NO. TIMES ON               NORMALIZED
STARTING    A CRITICAL   CRITICALITY  CRITICALITY
  NODE         PATH         INDEX        INDEX
    3             0        .00000       .00000

THE ACTIVITIES ENDING AT NODE   8 AND THEIR CRITICALITY INDICES ARE:
           NO. TIMES ON               NORMALIZED
STARTING    A CRITICAL   CRITICALITY  CRITICALITY
  NODE         PATH         INDEX        INDEX
    2            46        .00460       .00457
    5          1792        .17920       .17788
    7          2653        .26530       .26335
```

```
THE ACTIVITIES ENDING AT NODE   7 AND THEIR CRITICALITY INDICES ARE:
               NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE         PATH          INDEX          INDEX
    2          1197         .11970         .11882
    6          5822         .58220         .57792

THE ACTIVITIES ENDING AT NODE   6 AND THEIR CRITICALITY INDICES ARE:
               NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE         PATH          INDEX          INDEX
    3          5832         .58320         .57892

THE ACTIVITIES ENDING AT NODE   5 AND THEIR CRITICALITY INDICES ARE:
               NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE         PATH          INDEX          INDEX
    2          2393         .23930         .23754

THE ACTIVITIES ENDING AT NODE   4 AND THEIR CRITICALITY INDICES ARE:
               NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE         PATH          INDEX          INDEX
    3           606         .06060         .06015

THE ACTIVITIES ENDING AT NODE   3 AND THEIR CRITICALITY INDICES ARE:
               NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE         PATH          INDEX          INDEX
    1          6438         .64380         .63907

THE ACTIVITIES ENDING AT NODE   2 AND THEIR CRITICALITY INDICES ARE:
               NO. TIMES ON                  NORMALIZED
STARTING   A CRITICAL    CRITICALITY   CRITICALITY
  NODE         PATH          INDEX          INDEX
    1          3636         .36360         .36093

THE CRITICALITY INDICES OF THE NODES ARE:
                            NORMALIZED
               CRITICALITY  CRITICALITY
  NODE            INDEX         INDEX
    1          1.00740       1.00000
    2           .36360        .36093
    3           .64380        .63907
    4           .06060        .06015
    5           .23930        .23754
    6           .58320        .57892
    7           .70190        .69674
    8           .44910        .44580
    9           .00000        .00000
   10           .68520        .68017
   11           .31590        .31358
   12           .69110        .68602
   13          1.00710        .99970
   14           .86730        .86093
   15           .75140        .74588
   16           .13780        .13679
   17           .00220        .00218
   18           .47080        .46734
   19           .53660        .53266
   20          1.00740       1.00000
```

COMPARISONS OF ACTIVITY CRITICALITY INDICES:

				NORM.	NORM.	NO. ACTS.	NO. ACTS.	RELATIVE
		CRIT.	CRIT.	CRIT.	CRIT.	GREATER	GREATER	ERROR
START	END	INDEX	INDEX	INDEX	INDEX	CRIT. IN.	CRIT. IN.	OF NORM.
NODE	NODE	(APPR.)	(SIM.)	(APPR.)	(SIM.)	(APPR.)	(SIM.)	CRIT. INDEX
2	20	.00000	.00000	.00000	.00000	57	36	UNDEFINED
10	20	.00091	.00000	.00090	.00000	46	36	UNDEFINED
11	20	.00025	.00000	.00025	.00000	48	36	UNDEFINED
18	20	.39895	.47080	.39668	.46734	9	8	-15.12%
19	20	.60561	.53660	.60217	.53266	5	7	13.05%
1	19	.00000	.00000	.00000	.00000	57	36	UNDEFINED
7	19	.00026	.00000	.00026	.00000	47	36	UNDEFINED
9	19	.00000	.00000	.00000	.00000	54	36	UNDEFINED
12	19	.03000	.00010	.02983	.00010	36	35	********%
13	19	.02440	.00000	.02426	.00000	38	36	UNDEFINED
14	19	.22520	.11590	.22392	.11505	18	21	94.63%
15	19	.31462	.35520	.31283	.35259	14	12	-11.28%
16	19	.15806	.06430	.15716	.06383	24	23	146.23%
17	19	.04454	.00110	.04429	.00109	34	30	3955.96%
2	18	.00000	.00000	.00000	.00000	57	36	UNDEFINED
5	18	.00001	.00000	.00001	.00000	51	36	UNDEFINED
6	18	.00000	.00000	.00000	.00000	54	36	UNDEFINED
8	18	.00205	.00000	.00204	.00000	45	36	UNDEFINED
10	18	.00517	.00000	.00514	.00000	43	36	UNDEFINED
15	18	.30067	.39620	.29895	.39329	16	10	-23.99%
16	18	.14400	.07350	.14318	.07296	26	22	96.25%
17	18	.04617	.00110	.04591	.00109	33	30	4104.25%
2	17	.00001	.00000	.00001	.00000	53	36	UNDEFINED
3	17	.00001	.00000	.00001	.00000	52	36	UNDEFINED
11	17	.00655	.00020	.00651	.00020	42	34	3180.58%
13	17	.05975	.00200	.05941	.00199	31	29	2892.57%
2	16	.00014	.00000	.00014	.00000	50	36	UNDEFINED
13	16	.30654	.13780	.30480	.13679	15	19	122.83%
1	15	.00019	.00000	.00019	.00000	49	36	UNDEFINED
14	15	.69499	.75140	.69104	.74588	2	1	-7.35%
13	14	.92019	.86730	.91495	.86093	0	0	6.28%
1	13	.00291	.00000	.00289	.00000	44	36	UNDEFINED
2	13	.01405	.00000	.01397	.00000	40	36	UNDEFINED
4	13	.02696	.00040	.02680	.00040	37	33	6650.31%
11	13	.39036	.31570	.38814	.31338	10	13	23.86%
12	13	.60716	.69100	.60371	.68592	4	2	-11.99%
1	12	.01174	.00000	.01167	.00000	41	36	UNDEFINED
3	12	.01687	.00000	.01677	.00000	39	36	UNDEFINED
5	12	.08005	.00590	.07960	.00586	28	27	1259.11%
10	12	.60135	.68520	.59793	.68017	6	3	-12.09%
2	11	.06223	.00000	.06187	.00000	30	36	UNDEFINED
3	11	.05151	.00000	.05122	.00000	32	36	UNDEFINED
4	11	.17731	.06020	.17631	.05976	22	25	195.03%
5	11	.19198	.05420	.19089	.05380	21	26	254.80%
6	11	.09573	.00100	.09519	.00099	27	32	9489.10%
8	11	.32711	.20050	.32525	.19903	12	17	63.42%
3	10	.04102	.00000	.04079	.00000	35	36	UNDEFINED
7	10	.35263	.43660	.35062	.43339	11	9	-19.10%
8	10	.41343	.24860	.41108	.24677	8	15	66.58%
3	9	.00000	.00000	.00000	.00000	54	36	UNDEFINED
2	8	.06519	.00460	.06482	.00457	29	28	1319.51%
5	8	.16952	.17920	.16856	.17788	23	18	-5.24%
7	8	.22144	.26530	.22018	.26335	19	14	-16.39%
2	7	.15224	.11970	.15137	.11882	25	20	27.40%
6	7	.22779	.58220	.22649	.57792	17	6	-60.81%

```
   3     6  .32352  .58320  .32168  .57892      13          5       -44.43%
   2     5  .44158  .23930  .43906  .23754       7         16        84.84%
   3     4  .20427  .06060  .20311  .06015      20         24       237.64%
   1     3  .63720  .64380  .63358  .63907       3          4         -.86%
   1     2  .73543  .36360  .73124  .36093       1         11       102.60%

THE MAXIMUM (ABSOLUTE) RELATIVE ERROR OF
NORMALIZED ACTIVITY CRITICALITY INDEX IS: ********%
```



```
THE 3 MOST STOCHASTICALLY DOMINATING PATHS THROUGH THE NETWORK ARE:

THE RANK 1 PATH WITH  11 NODES:
   1   3   6   7  10  12  13  14  15  19  20

THE RANK 2 PATH WITH  11 NODES:
   1   3   6   7  10  12  13  14  15  18  20

THE RANK 3 PATH WITH  10 NODES:
   1   3   6   7  10  12  13  14  19  20

FROM MONTE CARLO SIMULATION:

THE RANK 1 APPROXIMATED CRITICAL PATH WITH  11 NODES:
   1   3   6   7  10  12  13  14  15  19  20

THE RANK 2 APPROXIMATED CRITICAL PATH WITH  11 NODES:
   1   3   6   7  10  12  13  14  15  18  20

THE RANK 3 APPROXIMATED CRITICAL PATH WITH  10 NODES:
   1   3   6   7  10  12  13  14  19  20


COMPARISONS OF PATHS:

RANK 1 PATHS:
NO. NODES  NO. NODES  NO. NODES
 (APPR.)     (SIM.)   IN COMMON
    11         11         11

RANK 2 PATHS:
NO. NODES  NO. NODES  NO. NODES
 (APPR.)     (SIM.)   IN COMMON
    11         11         11

RANK 3 PATHS:
NO. NODES  NO. NODES  NO. NODES
 (APPR.)     (SIM.)   IN COMMON
    10         10         10
```

Figure 53. PART-paths(val) program output.

Table 34. "Matching Category" Performance Measure for Comparisons of Paths.

No. of Nodes	No. of Arcs	Order of Paths		
		Path 1	Path 2	Path 3
10	15	All	1-4	All
10	20	All	All	1-4
10	30	All	1-4	1-4
10	40	All	1-4	All
20	30	All	1-4	1-4
20	40	All	All	1-4
20	50	All	1-4	1-4
20	60	All	All	All
20	70	All	1-4	1-4
20	80	All	All	1-4
30	45	All	1-4	1-4
30	60	All	1-4	All
30	75	All	1-4	1-4
30	90	All	1-4	1-4

enumeration by simulation, which requires extensive computational resources. Also viewed collectively across the 20 networks generated and tested, the combined set of nodes in the three most stochastically dominant paths and the combined set of nodes in the three most critical path were the same in virtually every comparison. Only very infrequently were there node mismatches between the two sets of nodes; when a mismatch occurred, it involved one or a small number of nodes whose connecting activities had very close criticality indices which were lower than the indices of all the activities connecting the other nodes in the combined set.

Again, the validation experiments were conducted with a version of the PART-paths(val) program which employs ten classes for polygonal approximation, without special consideration for the presence of any exponential distributions. If the number of classes for polygonal approximation is increased, errors build up at a slower rate, so the performance of this PART algorithm would be improved relative to the results of the validation experiments. Nonetheless, the results of the validation experiments appeared comparable to the previously reported results (Table 11), to the extent that information could be distilled from Dodin's "1"s and "0"s. Dodin reported that for all the paths that did not match, the stochastically dominating path differed from the corresponding critical path in no more than four arcs, which means, depending on how it is interpreted, that the paths differed by no more than two or no more than four nodes. The PART-paths(val) algorithm experienced mismatches involving more than four nodes in less than 2% of the pairwise comparisons in the validation experiments. A great majority of the path mismatches occurred in networks with arc to node densities higher than two, which confirmed one of Dodin's observations. Often the three most stochastically dominating paths and the three most critical paths had several to many arcs in common, as illustrated by the results in Figure 51, which confirmed another of Dodin's observations.

The validation experiments demonstrated that the implementation with polygonal approximation of the heuristic to determine the *K* most stochastically dominant paths has the following properties:

(1) The most critical path is correctly determined by the most stochastically dominating path approximately 85% of the time, and differs from it by no more than four nodes the rest of the time. The second most critical path is similarly, correctly determined approximately 25% of the time, and the third most critical path is similarly, correctly determined approximately 20% of the time.

(2) The closeness between a critical path and its corresponding stochastically dominating path is not affected by the challenge of the activity distributions, although accuracy in approximation of activity and node criticality indices is. While the accuracy of approximating path criticality degrades with increasing challenge, the relative criticality of paths is maintained.

Table 35. CPU Time Comparison of Sequential Approximation and PART Algorithms (Throughput Distributions).

Distribution Type	Network size		CPU time in seconds		
	No. Nodes	No. Arcs	Sequential Approx.	PART-ind	PART-seq
Uniform	10	15	1.665	0.433	0.320
Triangular	10	15	1.673	0.447	0.324
Normal	10	15	1.680	0.452	0.343
Exponential	10	15	1.678	0.452	0.340
Gamma	10	15	1.695	0.497	0.375
Beta	10	15	1.766	0.561	0.452
Discrete	10	15	0.994	n/a	n/a
All	10	15	1.623	0.530	0.405
Uniform	20	40	4.448	0.877	0.496
Uniform	30	50	5.335	1.242	0.610
Uniform	40	60	5.974	1.573	0.657
Uniform	40	80	7.919	2.261	0.873
Uniform	50	75	7.499	2.231	0.772
Uniform	50	100	10.867	3.244	1.002
Uniform	60	150	15.822	5.984	1.806

4.5 Run Time and Storage Requirements

In this section, the computer run time and array storage requirements of the PART programs, which were executed interactively on an IBM/RS/6000/580 computer, are discussed and compared with the requirements of competing programs, where known.

4.5.1 Run Time Requirements

Dodin (1980 and 1985a, 1984) reported computational experience with his computer implementations of sequential approximation and a heuristic for identifying the *K* most stochastically dominating paths, both based on discretization. Table 35 gives the run times in CPU seconds on a UNIVAC 1100/80 for 15 networks which Dodin reported for his sequential approximation algorithm; each network had all activity duration distributions of a single type. Twenty "strongly random" replications of these networks were reduced with the PART-ind and PART-seq algorithms; the average run times are also shown in Table 35.

Table 36. CPU Time Comparison of Sequential Approximation and PART Algorithms (Heuristic for the *K* Most Stochastically Dominating Paths).

Problem No.	Network size		CPU time in seconds	
	No. Nodes	No. Arcs	Discretization	PART-path
1	10	15	0.92	0.703
2	10	20	1.94	0.963
3	10	30	7.80	2.040
4	10	40	11.60	2.943
5	20	30	9.47	2.117
6	20	40	9.92	3.270
7	20	50	15.61	4.453
8	20	60	26.81	5.473
9	20	70	31.55	5.487
10	20	80	39.84	8.123
11	30	45	16.70	3.207
12	30	60	29.26	6.377
13	30	75	33.96	7.337
14	30	90	43.65	9.103

For the smallest networks compared, PART-ind was almost four times as fast as sequential approximation based on discretization, and PART-seq was over five times as fast; for the largest networks, PART-ind was over 2.5 times as fast, and PART-seq was almost nine times as fast. Table 36 gives the run times in CPU seconds also for a UNIVAC 1100/80 for 14 networks which Dodin reported for his algorithm which implemented a heuristic for identifying the *K* most stochastically dominating paths. The activity duration distributions were assigned in blocks of ten from the distributions in Table 10. Twenty "strongly random" replications of these networks were analyzed with the PART-path algorithm; to achieve greater randomness than the block assignments, the activity duration distributions were individually randomly selected. The average run times are also shown in Table 36. For the smallest networks compared, PART-path was two to three times faster than the discretization implementation of the heuristic; for the largest networks, PART-path was almost five times as fast. While some of the differences in these run times may be

Table 37. CPU Time Comparison of Ordered Recursive Conditioning and PART Algorithms (Throughput Distributions).

	Network size		CPU time in seconds		
Case	No. Nodes	No. Arcs	Ordered Rec. Conditioning	PART-ind	PART-seq
TEST5A	5	Random	0.013	0.388	0.312
TEST6A	6	Random	0.035	0.449	0.370
TEST7A	7	Random	0.082	0.513	0.428
TEST8A	8	Random	0.741	0.567	0.462
TEST9A	9	Random	2.284	0.719	0.473
TEST10A	10	Random	5.215	0.735	0.609
TEST11A	11	Random	17.880	0.813	0.658
TEST12A	12	Random	75.730	1.128	0.949

accounted for by the fact that the IBM/RS/6000/580 is a faster computer than the UNIVAC 1100/80 which Dodin used in the early 1980s, the polygonal approximation-based algorithms appear to be faster than the discretization-based algorithms, conservatively by a factor of two or three, possibly higher.

Hagstrom (1990) reported computational experience with her implementation, also based on discretization, of ordered recursive conditioning. Table 37 gives the run times (from Table 6, in CPU microseconds) on an IBM3081 for 19 test networks, including eight (TEST5A - TEST12A) which were generated with a random number of activities for a fixed number of nodes. Twenty "strongly random" replications of these networks were reduced with the PART-ind and PART-seq algorithms; the average run times are also shown in Table 37. Only for the very small networks with five to seven nodes were the run times of ordered recursive conditioning better than the run times of the polygonal-approximation based algorithms, which otherwise decisively outperformed the competition. Even for small networks with only 12 nodes, PART-ind and PART-seq were about 75 times faster. Even with a generous allowance for differences in speed between the two computer systems, ordered recursive conditioning is not as efficient for stochastic network reduction as polygonal approximation and reduction.

Run time prediction models were constructed and tested for the PART-ind, PART-seq, and PART-path algorithms. Dodin (1980 and 1985a, 1984, 1985b) established that the complexity of "independent multiple arcs" (dual arcs) is $O(CA)$ where C is the complexity of series-parallel reduction operations, the complexity of sequential approximation is also $O(CA)$, and the complexity of the heuristic for identifying the K most stochastically dominating paths is $O(CKN^2)$. For polygonal approximation, C is $O(cn_r)$, where c is the number of classes and n_r is the number of regression fitting points. Consequently, the number of arithmetic computations performed by both the PART-ind and PART-seq algorithms is $O(cn_rA)$, and run time models are of the form:

$$CPU\ Time \leq (\text{constant})cn_rA$$

When equality is assumed, the models can be log-linearized as:

$$\log(CPU\ Time) = \log(\text{constant}) + \log(c) + \log(n_r) + \log(A)$$

For our implementations of polygonal approximation, $c = 10$ and $n_r = 50$ (Section 3.2.1 [above]), so the models reduce to:

$$\log(CPU\ Time) = \text{constant} + \log(A)$$

SLRs of run time data from the validation ANOVA experiments and the run time comparisons with previously reported results were performed on the SAS System, and the following models were obtained:

$$\log(CPU\ Time_{ind}) = -3.742 + 1.089\log(A)$$

$$\log(CPU\ Time_{seq}) = -3.139 + 0.776\log(A)$$

The PART-ind run time model had a prob value for statistical significance of regression of 0.0001 and an adjusted R^2 of 0.968; the PART-seq model had a prob value of 0.0001 and an adjusted R^2 of 0.874. According to the theory, the PART-path algorithm should have a crude run time model, ignoring lower order terms, of the form:

$$CPU\ Time_{paths} \leq (\text{constant})cn_rKN^2$$

A SLR of run time data from the run time comparisons with previously reported results was performed on the log-linearized model (equality assumed) on the SAS System, but the adjusted R^2 was only 0.56. The data were then fit to an $O(cn_rKA)$ model of the form:

$$CPU\ Time_{paths} \leq (\text{constant})cn_rKA$$

and the resulting model for $K = 3$ was:

$$\log(CPU\ Time_{paths}) = -4.257 + 1.449\log(A)$$

with a prob value of 0.0001 and an adjusted R^2 of 0.983. All three run time models fit well. To test the predictive power of the models, 20 "strongly randomized" replications of a small number of networks with sizes different from the model development data were reduced, and the averages of the actual CPU times were compared with the 95% confidence intervals on the mean response from the models. Tables 38 and 39 summarize the results of the predictive tests. The models failed to predict well the average CPU time for the networks tested, indicating there is more to be understood about the modeling of run times of polygonal approximation-based network reduction algorithms.

Table 38. Predictive Performance of CPU Run Time Models (PART-ind and PART-seq).

Network size		CPU time in seconds			
No.	No.	PART-ind		PART-seq	
Nodes	Arcs	Average	Confidence Int.	Average	Confidence Int.
15	30	0.925	0.907 - 1.023	0.690	0.555 - 0.664
25	45	1.473	1.422 - 1.577	0.937	0.770 - 0.898
35	55	1.854	1.769 - 1.963	1.094	0.899 - 1.050
45	70	2.716	2.289 - 2.565	1.375	1.077 - 1.277
55	125	5.553	4.200 - 4.943	2.537	1.629 - 2.075

Table 39. Predictive Performance of CPU Run Time Models (PART-path).

Network size		CPU time in seconds	
No.	No.	PART-path	
Nodes	Arcs	Average	Confidence Int.
15	25	1.471	1.375 - 1.642
25	55	4.425	4.422 - 5.017
30	70	5.976	6.181 - 7.221

4.5.2 Computer Storage Requirements

The PART programs in the appendices were constructed as proof-of-concept demonstrators and, as such, are limited in the sizes of network they can analyze. Each program can accommodate networks with up to and including 100 nodes, N, and the maximum number of activities that may exist in a network with N nodes, $N(N-1)/2$. A maximum of 100 "strongly random" networks can be generated upon request, no more than five most stochastically dominating paths can be identified, and the number of simulation replications is capped at 10,000. To reduce error build up, the programs use double precision arithmetic. This requires approximately 3.728 megabytes of computer storage for arrays for PART-ind, 2.883 megabytes for PART-seq, and 4.583 megabytes for PART-paths. Unfortunately, no other researcher has reported storage requirements for a competing program. Hagstrom (1990) stated that no analysis of computer storage requirements was performed for her implementation of ordered recursive conditioning, and Dodin (1980 and 1985a, 1984) did not address the question of computer storage requirements for his programs for sequential approximation and the heuristic for identifying the K most stochastically dominating paths. Analysis of unpublished computer code for Dodin's programs suggested that his sequential approximation program requires about 0.994 megabytes computer storage and is limited to 50 nodes and 100 activities (although he has reported performance results for larger sized networks) and 1,000 simulation replications, using single precision arithmetic. Similarly, his program for identifying the K most stochastically dominating paths appears to require about 0.493 megabytes of memory for array storage exclusive of network simulation to determine the actual critical paths. Crude projections of equivalency of Dodin's programs to the PART programs are that his sequential approximation program would require 3 to 4 megabytes of memory to match the PART-ind and PART-seq capabilities and that his critical path algorithm would require about 2 megabytes plus a critical path simulator to match the PART-path capability.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Summary

A method for the linear polynomial (polygonal) approximation of continuous activity resource consumption (duration) distributions of stochastic project management networks was developed in this research. The method was derived from the spline approximations used in numerical differentiation and integration and motivated by the shortcomings of the first attempt at an exact solution to stochastic network reduction by Martin (1965). It is the first new method for network approximation and reduction to be advanced since discretization, the basis for all previously developed algorithms. The method was successfully mated with three network reduction approaches - arc duplication, sequential approximation, and a heuristic for identifying the *K* most critical paths - to form the members of a new family of Polygonal Approximation and Reduction Techniques (PART), and proof-of-concept computer programs were written for these three PART algorithms. The PART algorithm using "independent multiple arcs" (dual arcs) represents the first successful implementation of an arc-duplication reduction method. PART algorithms approximate the throughput distribution for small-to-moderate size networks, and the *K* most critical paths for large networks. Collectively, PART algorithms constitute an analytic reduction capability operative across the entire range of project management networks.

To validate the performance of polygonal approximation-based algorithms, an experimental framework for algorithm testing was developed following the principles of design of experiments; no such framework had previously been employed in the testing of network reduction algorithms. Validation testing confirmed that the accuracy of polygonal approximation is a function of network size, as driven by the number of activities much more strongly than the number of nodes, how great a challenge the activity distribution functions present to series-parallel reduction operations based on polygonal approximation, and the number of classes in the partitions of the domains of continuous activity distributions over which linear polynomials are piecewise-defined to approximate activity probability density functions, just as accuracy of discretization approximation is a function of how densely discretization points are packed over the domains. Compared to other discretization-based methods - sequential approximation and ordered recursive conditioning - PART algorithms were demonstrated to be as accurate or more accurate in the characterization of the throughput distribution function. Important findings were the observations that the PART algorithm for identifying the K most stochastically dominating paths performs differently with respect to its ability to match the first most critical path, the second, the third, etc., and that the algorithm is insensitive to challenging distributions in its ability to discern relative criticality among network paths. PART algorithms, by virtue of their design, execute faster than their discretization competitors; speeds as many as nine times as fast as their competitors were experienced during performance testing, while across the board it is conservative to say that PART algorithms are two to three times as fast as all other network reduction methods. As analytic techniques, PART algorithms are orders of magnitude faster than simulation-approximations without significant losses in accuracy when simulation results are taken as "true."

In conclusion, polygonal approximation and associated PART algorithms represent a new and innovative concept in the analytic arsenal aimed at stochastic project management networks. In validation testing, conducted on network reduction algorithms for the first time in accordance with a design of experiments framework, they exhibited performance at worst comparable, but usually superior, to their competitors in terms of accuracy and speed. They have the potential to put the power of network management into the hands of anyone in possession of a desktop computing capability. In this research, they have demonstrated their worthiness for continued development.

5.2 Recommendations for Further Research

The following topics are recommended for further research:

1. Investigate the sensitivity of the performance of PART-ind and PART-seq algorithms to network size and structure and the challenge of the activity distribution functions to polygonal approximation-based series-reduction operations.
2. Investigate the sensitivity of the performance of PART algorithms to the number of classes in the partitions of domains of activity distributions and the number of regression fitting points.
3. Develop more accurate predictive models for run times of PART algorithms.
4. Refine the "matching category" performance measure for the PART-paths algorithm and characterize the ability of the algorithm to identify both specific critical paths and membership sets of highly critical activities which should receive management focus.
5. Investigate the incorporation of probabilistic branching into PART algorithms to widen their scope of application outside of the class of acyclic, directed networks.
6. Investigate expansion of the set of distributions generally used to model activity resource consumption (duration) and the possible potential to enhance PART algorithm performance. In particular, consider Dodin's suggestions concerning the extreme value distribution and the engineered distributions developed by Golenko-Ginzburg which are auto-reproductive under series-parallel reduction operations.
7. Investigate possible applications of polygonal approximation to other discretization-based engineering algorithms, particularly in scheduling.

REFERENCES

ADLAKHA, V. G., AND V. G. KULKARNI. 1989. Stochastic PERT Networks: Review and Classified Bibliography (1966-1987). *INFOR* **27**, 272-296.

ALVAREZ-VALDES, R. 1988. Computational Comparison of Classical and New Heuristic Algorithms for Resource-Constrained Project Scheduling. Paper presented at the First International Workshop on Project Management and Scheduling, Lisbon, July 11-13.

AMERICAN SOCIETY OF TOOL AND MANUFACTURING ENGINEERS. 1967. *Effective Project Management, Basic Network Time Techniques.* Argyle, New York.

ASANO, T., AND S. SATO. 1985. Long Path Enumeration Algorithms for Timing Verification on Large Digital Systems. In Y. Alavi, G. Chartrand, L. Lesniak, D. R. Lick, and C. E. Wall (Eds.), *Graph Theory with Applications to Algorithms and Computer Science.* John Wiley, New York.

AVRAMIDIS, A. N., K. W. BAUER, AND J. R. WILSON. 1991. Simulation of Stochastic Activity Networks Using Path Control Variates. *Naval Research Logistics* **38**, 183-201.

BADIRU, A. B. 1988. *Project Management in Manufacturing and High Technology Operations.* John Wiley, New York.

BADIRU, A. B. 1991a. A Simulation Approach to PERT Network Analysis. *Simulation* **57**, 245-255.

BADIRU, A. B. 1991b. STARC 2.0: An Improved PERT Network Simulation Tool. *Computers & Industrial Engineering* **20**, 389-400

BADIRU, A. B., AND G. E. WHITEHOUSE. 1989. *Computer Tools, Models and Techniques for Project Management.* Tab Books, Blue Ridge Summit, PA.

BAILEY, M. P. 1992. Measuring Performance of Integrated Air Defense Networks Using Stochastic Networks. *Operations Research* **40**, 647-659.

BARD, J. F., AND J. E. BENNETT. 1991. Arc Reduction and Path Preference in Stochastic Acyclic Networks. *Management Science* **37**, 198-215.

BATTERSBY, A. 1970. *Network Analysis for Planning and Scheduling.* John Wiley, New York.

BENGTSON, N. M. 1988. Using Operational Analysis in Simulation: A Queueing Network Example. *Journal of the Operational Research Society* **39**, 1125-1136.

BIRGE, J. R., AND J. K. HO. 1993. Optimal Flows in Stochastic Dynamic Networks with Congestion. *Operations Research* **41**, 203-216.

BURT, J. M. 1969. *Stochastic PERT Networks.* Ph.D. Dissertation. College of Business Administration, Carnegie-Mellon University, Pittsburgh, PA.

BURT, J. M., AND M. B. GARMAN. 1971. Conditional Monte Carlo: A Simulation Technique for Stochastic Network Analysis. *Management Science* **18**, 207-217.

BURT, J. M., D. P. GAVER, AND M. PERLAS. 1970. Simple Stochastic Networks: Some Problems and Procedures. *Naval Research Quarterly* **17**, 439-458.

BUSACKER, R. G., AND T. L. SAATY. 1965. *Finite Graphs and Networks: An Introduction with Applications.* McGraw-Hill, New York.

CARRUTHERS, J. A., AND A. BATTERSBY. 1966. Advances in Critical Path Methods. *Operational Research Quarterly* **17**, 359-380.

CHAE, H. C. 1990. A Geometric Interpretation of the PERT Assumption on the Activity Time. *International Journal of Mathematical Education* **21**, 283-288.

CHAE, K. C., AND S. KIM. 1990. Estimating the Mean and Variance of PERT Activity Time Using Likelihood-Ratio of the Mode and Midpoint. *IEE Transactions* **22**, 198-203.

CHAPMAN, C. B., AND J. DEL HOYO. 1972. Progressive Basic Decision CPM. *Operational Research Quarterly* **23**, 345-359.

CHARNES, A., W. W. COOPER, AND G. L. THOMPSON. 1964. Critical Path Analyses Via Chance Constrained and Stochastic Programming. *Operations Research* **12**, 460-470.

CHARNES A., M. KIRBY, AND W. RAIKE. 1966. Chance-Constrained Generalized Networks. *Operations Research* **14**, 1113.

CHRISTOFIDES, N., R. ALVAREZ-VALDES, AND J. M. TAMARIT. 1987. Project Scheduling With Resource Constraints: A Branch and Bound Approach. *European Journal of Operational Research* **29**, 262-273.

CLARK, C. E. 1961. The Greatest of a Finite Set of Random Variables. *Operations Research* **9**, 145-162.

CLEROUX, R., AND D. J. MCCONALOGUE. 1976. A Numerical Convolution Algorithm for Recursively-Defined Convolution Integrals Involving Distribution Functions. *Management Science* **22**, 1138-1146.

CLINGEN, C. T. 1964. A Modification of Fulkerson's Algorithm for Expected Duration of a PERT Project When Activities Have Continuous Density Functions. *Operations Research* **4**, 629-632.

CONNERS, M. M., AND W. I. ZANGWILL. 1971. Cost Minimization in Networks with Stochastic Discrete Requirements. *Operations Research* **19**, 794-821.

CONOVER, W. J. 1980. *Practical Nonparametric Statistics,* 2nd Ed. John Wiley, New York.

CONTE, S. D., AND C. DEBOOR. 1972. *Elementary Numerical Analysis: An Algorithmic Approach,* 2nd Ed. McGraw-Hill, New York.

COOK, T. M. AND R. H. JENNINGS. 1979. Estimating a Project's Completion Time Using Intelligent Simulation Methods. *Journal of the Operational Research Society* **30**, 1103-1108.

COREA, G. A., AND V. G. KULKARNI. 1990. Minimum Cost Routing on Stochastic Networks. *Operations Research* **38**, 527-536.

COREA, G. A., AND V. G. KULKARNI. 1993. Shortest Paths in Stochastic Networks with Arc Lengths Having Discrete Distributions. *Networks* **23**, 175-183.

CRAMER, H. 1946. *Mathematical Methods of Statistics*. Princeton University Press, Princeton, NJ.

DANESE, A. E. 1965. *Advanced Calculus: An Introduction to Applied Mathematics.*, Vol. I. Allyn and Bacon, Boston, MA.

DEMEULEMEESTER, E., B. M. DODIN, AND W. S. HERROELEN. 1993. A Random Activity Network Generator. *Operations Research* **41**, 972-980.

DEO, N. 1974. *Graph Theory with Applications to Engineering and Computer Sciences.* Prentice-Hall, Englewood Cliffs, NJ.

DEVROYE, L. P. 1980. Inequalities for the Completion Times of Stochastic PERT Networks. *Math. Ops. Res.* **4**, 441-447.

DIAL, R., F. GLOVER, AND D. KLINGMAN. 1979. A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees. *Networks* **9**, 215-248.

DODIN, B. M. 1980. *On Estimating the Probability Distribution Functions in PERT-Type Networks.* OR Report No. 153 (Revised). North Carolina State University, Raleigh, NC.

DODIN, B. M. 1984. Determining the K Most Critical Paths in PERT Networks. *Operations Research* **32**, 859-877.

DODIN, B. M. 1985a. Approximating the Distribution Functions in Stochastic Networks. *Computers and Operations Research* **12**, 251-264.

DODIN, B. M. 1985b. Reducibility of Stochastic Networks. *OMEGA* **13**, 223-232.

DODIN, B. M. 1985c. Bounding the Project Completion Time Distribution in PERT Networks. *Operations Research* **33**, 862-881.

DODIN, B. M., AND M. SIRVANCI. 1990. Stochastic Networks and the Extreme Value Distribution. *Computers & Operations Research* **17**, 397-409.

DONALDSON, W. A. 1965. The Estimation of the Mean and Variance of a 'PERT' Activity Time. *Operations Research* **13**, 382-387.

DUFFIN, R. J. 1965. Topology of Series-Parallel Networks. *J. Math. Anal. Appl.* **10**, 303-318.

DUFFY, C. A. 1993. Buyers Guide: Project Managers Seek to Transcend Niche. *PC Week* **10** **(32)**, 91-94.

DUNNE, E. J. 1971. *Network Theory and Project Resource Management.* Ph.D. Dissertation. Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, IL.

DUNNE, E. J., AND M. I. DESSOUKY. 1971. *Structural Properties of Networks.* Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, IL.

EISNER, H. 1962. A Generalized Network Approach to the Planning and Scheduling of a Research Project *Operations Research* **10**, 115-122.

ELMAGHRABY, S. E. 1964. An Algebra for the Analysis of Generalized Activity Networks. *Management Science* **10**, 494-514.

ELMAGHRABY, S. E. 1966. On Generalized Activity Networks. *Journal of Industrial Engineering* **18**, 621-631.

ELMAGHRABY, S. E. 1967. On the Expected Duration of PERT Type Networks. *Management Science* **13**, 299-306.

ELMAGHRABY, S. E. 1968. The Determination of Optimal Activity Duration in Project Scheduling. *Journal of Industrial Engineering* **19**, 48-51.

ELMAGHRABY, S. E. 1970a. Theory of Networks and Management Science: Parts I and II. *Management Science* **17**, 1-34 & B-54 - B-71.

ELMAGHRABY, S. E. 1970b. Some Network Models in Management Science. *Lecture Notes in Operations Research and Mathematical Systems,* Vol. 29. Springer-Verlag, New York.

ELMAGHRABY, S. E. 1977 *Activity Networks: Project Planning and Control by Network Models.* Wiley Interscience, New York.

ELMAGHRABY, S. E. AND W. S. HERROELEN. 1980. On the Measurement of Complexity in Activity Networks. *European Journal of Operational Research* **5**, 223-234.

ESARY, J. D., F. PROSCHAN AND D. W. WALKUP. 1967. Association of Random Variables, with Applications. *Annals of Math. Stat.* **38**, 1466-1474.

EVANS, J. R. 1972. *On Determining the Distribution of Maximum Flows in Networks with Discrete Probabilistic Branch Capacities.* Master's Thesis. School of Industrial Engineering, Purdue University, West Lafayette, IN.

EVANS, J. R. 1976. Maximum Flows in Probabilistic Graphs - The Discrete Case. *Networks* **6**, 161-183.

FELLER, W. 1971. *An Introduction to Probability Theory and Its Applications*, 2nd Ed. John Wiley, New York.

FERGESON, J. A., AND P. H. SHORTELL. 1978. *Polygonal Approximation and Reduction of Activity Resource Consumption Distributions in Program Management Networks.* Master's Thesis. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB, OH.

FILLEY, R. D. 1986. 1986 Project Management Software Buyer's Guide. *Industrial Engineering* **18 (1)**, 51-63.

FIX, W., K. NEUMANN, AND U. STEINHARDT. 1975. *Introduction to Time and Cost Scheduling of Decision Activity Networks.* Discussion Paper. Institut Für Operations Research, Unversität Karlsruhe, Germany.

FOLDES, S., AND F. SOUMIS. 1993. PERT and Crashing Revisited: Mathematical Generalizations. *European Journal of Operational Research* **64**, 286-294.

FORD, L. R., AND D. R. FULKERSON. 1956. Maximal Flow Through a Network. *Canadian Journal of Mathematics* **8**, 399-404.

FORD, L. R., AND D. R. FULKERSON. 1962. *Flows in Networks.* Princeton University Press, Princeton, NJ.

FRANCIS, V. E. 1985. Aggregation of Network Flow Problems. Ph.D. Dissertation. University of California, Los Angeles, CA.

FRANCIS, V. E. 1986. Aggregation in PERT/CPM. Presented at the TIMS/ORSA Joint National Meeting, Miami, FL.

FRANK, H. 1969. Shortest Paths in Probabilistic Graphs. *Operations Research* **17**, 583-599.

FRANK, H., AND S. L. HAKIMI. 1965. Probabilistic Flows Through a Communication Network. *IEEE Transactions on Circuit Theory* **CT-12**, 413-414.

FRANK, H., AND I. T. FRISCH. 1971. *Communication, Transmission and Transportation Networks.* Addison-Wesley, Reading, MA.

FULKERSON, D. R. 1962. Expected Critical Path Lengths in PERT Networks. *Operations Research* **10**, 808-817.

GARMAN, M. B. 1972. More on Conditioned Sampling in the Simulation of Stochastic Networks. *Management Science* **19**, 90-95.

GAREY, M. R., AND D. S. JOHNSON. 1979. *Computers and Intractibility A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co. San Francisco, CA.

GIDO, J. 1985. *Project Management Software Directory.* Industrial Press, New York.

GOLENKO-GINZBURG, D. 1988. On the Distribution of Activity Time in PERT. *Journal of the Operational Research Society* **39**, 767-771.

GOLENKO-GINZBURG, D. 1989. A New Approach to the Activity-Time Distribution on PERT. *Journal of the Operational Research Society* **40**, 389-393.

HABER, S. 1970. Numerical Evaluation of Multiple Integrals. *SIAM Review* **12**, 481-526.

HACKMAN, S. T. AND R. C. LEACHMAN. 1989 A General Framework for Modeling Production. *Management Science* **35**, 478-495.

HAGSTROM, J. 1988. Computational Complexity of PERT Problems. *Networks* **18**, 139-147.

HAGSTROM, J. 1990. Computing the Probability Distribution of Project Duration in a PERT Network. *Networks* **20**, 231-244.

HAMMING, R. W. 1962. *Numerical Methods for Scientists and Engineers*. McGraw-Hill, New York.

HANNAH, H. D. 1978. Personal Interviews. Engineering Project Planning, Hobart Corporation, Troy, OH.

HARTLEY, H. O., AND A. W. WORTHAM. 1966. A Statistical Theory for PERT Critical Path Analysis. *Management Science* **12**, B-469 - B-482.

HERROELEN, W. AND G. CAESTECKER. 1979. *The Generation of Random Activity Networks*. Report No. 7906, Department of Applied Economic-Sciences, Catholic University of Leuven, Belgium (March).

HILL, T. W., JR. 1969. *On Determining a Distribution Function Known Only by Its Moments and/or Moment Generating Function*. Ph.D. Dissertation. College of Engineering Sciences, Arizona State University, Tempe.

HILLIER, F., AND G. J. LIEBERMAN. 1967. *Introduction to Operations Research*. Holden-Day, San Francisco.

HOPFINGER, E., AND U. STEINHARDT. 1975. *Two New Procedures for the Evaluation of Finite Acyclic Activity Networks with Stochastic Durations of Activities*. Discussion Paper. Institut Für Operations Research, Universität Karlsruhe, Germany.

HOPFINGER, E., AND U. STEINHARDT. 1976. On the Exact Evaluation of Finite Activity Networks with Stochastic Durations of Activities. *Lecture Notes in Economics and Mathematical Systems*, Vol. 117, *Optimization and Operations Research*. Springer-Verlag, New York.

HOROWITZ, E., AND S. SAHNI. 1976. *Fundamentals of Data Structures*. Computer Science Press, Rockville, MD.

HOROWITZ, E., AND S. SAHNI. 1978. *Fundamentals of Computer Algorithms*. Computer Science Press, Rockville, MD.

HOWARD. R. A. 1960. *Dynamic Programming and Markov Processes.* Technology Press, Cambridge, Mass. and John Wiley, London.

HOWARD. R. A. 1964. Systems Analysis of Semi-Markov Processes. *IEEE Transactions on Military Electronics* April, 114-124.

HUGGINS, W. H. 1957. Signal Flow Graphs. *Proceedings of the IRE* **9**, 74-86

HUNG, M. S., AND DIVOKY, J. J. 1988. A Computational Study of Efficient Shortest Path Algorithms. *Computers and Operations Research* **15**, 567-576.

International Mathematical and Statistical Library (IMSL), 1991, Version 2.0, IMSL Inc., Houston, TX.

ISAAC, S., AND W. B. MICHAEL. 1981. *Handbook in Research and Evaluation.* EdITS, San Diego, CA.

IZUCHUKWU, J. I. 1990. Project Management: Shortening the Critical Path. *Mechanical Engineering* **112**, 59-60.

KAMBUROWSKI, J. 1985. Bounds in Temporal Analysis of Stochastic Networks. *Foundations of Control Engineering* **10**, 177-189.

KAMBUROWSKI, J. 1992. Bounding the Distribution of Project Duration in PERT Networks. *Operations Research Letters* **12**, 17-22.

KAPLAN, W. 1962. *Operational Methods for Linear Systems.* Addison-Wesley, Reading, MA.

KARNS, L. A., AND L. A. SWANSON. 1973. The Effects of Activity Time Variance on Critical Path Planning. *Project Management Quarterly* **4**, 17-25.

KENDALL, M. G., AND A. STUART. 1969. *The Advanced Theory of Statistics,* Vol. 1, 3rd ed. Hafner, New York.

KENNEDY, K. 1991. Using Simulation for Expanding the PERT/CPM Model in a Quantitative Analysis Class. *Collegiate Microcomputer* **9**, 244-252.

KING, W. R., AND P. A. LUCAS. 1973. An Experimental Analysis of Network Planning. *Management Science* **19**, 1423-1432.

KLEIJNEN, J. P. C. 1974. *Statistical Techniques in Simulation,* Vol. I. Marcel Dekker, New York.

KLEIN, C. M. 1993. Modeling Uncertainty in Networks. *Mathematical and Computer Modelling* **17**, 31-40.

KLEINDORFER, G. B. 1971. Bounding Distributions for a Stochastic Acyclic Network. *Operations Research* **19**, 1586-1601.

KLEINDORFER, G. B., AND P. R. KLEINDORFER. 1974. Bounding Distributions for Stochastic Logic Networks. *Operations Research Quarterly* **25**, 465-479.

KLINGEL, A. R., JR. 1966. Bias in PERT Project Completion Time Calculations for a Real Network. *Management Science* **13**, B-194 - B-201.

KLINGMAN, D. A. NAPIER AND J. STUTZE. 1974. NETGEN: A Program for Generating Large Scale Caoacitated Assignment, Transportation, and Minimum Cost Flow Network Programs. *Management Science* **20**, 814-821.

KUKLAN, H. 1993. Effective Project Management: An Expanded Network Approach. *Journal of Systems Management* **44**, 12-16.

KURTULUS, I. AND E. W. DAVIS. 1982. Multiproject Scheduling: Categorization of Heuristic Rules Performance. *Management Science* **28**, 161-172.

LAW, A. M., AND W. D. KELTON. 1982. *Simulation Modeling and Analysis*. McGraw-Hill, New York.

LEE, S. M., G. L. MOELLER, AND L. A. DIGMAN. 1982. *Network Analysis for Management Decisions A Stochastic Approach*. Kluwer Nijhoff, Boston, MA.

LINDSEY, J. H., JR. 1972. An Estimate of Expected Critical Path Length in PERT Networks. *Operations Research* **20**, 800-812.

LITTLEFIELD, T. K., AND P. H. RANDOLPH. 1991. PERT Duration Times: Mathematics or MBO. *Interfaces* **21**, 92-95.

LUKASZEWICZ, J. 1965. On the Estimation of Errors Introduced by Standard Assumptions Concerning the Distribution of Activity Duration in PERT Calculations. *Operations Research* **13**, 326-327.

MACCRIMMON, K. R., AND C. A. RYAVEC. 1964. An Analytical Study of the PERT Assumptions. *Operations Research* **12**, 16-37.

MARKS, N. B. 1991. Introduction of Criticality Indices into PERT Instruction. *International Journal of Mathematical Education in Science and Technology* **22**, 223-228.

MARTIN, J. J. 1965. Distribution of the Time Through a Directed Acyclic Network. *Operations Research* **13**, 46-66.

MIRCHANDANI, P. B. 1976. Shortest Distance and Reliability of Probabilistic Networks. *Computers and Operations Research* **3**, 347-355.

MIRHRAM, G. A. 1972. *Simulation: Statistical Foundations and Methodology*. Academic Press, New York.

MODER, J., AND C. PHILLIPS. 1964. *Project Management with CPM and PERT*. Reinhold, New York.

MOORE, L. J., AND E. R. CLAYTON. 1976. *GERT Modeling and Simulation: Fundamentals and Applications*. Petrocelli/Charter, New York.

MORET, B. M. E., AND H. D. SHAPIRO. 1991a. *Algorithms from P to NP Volume 1 Design and Efficiency*. Benjamin/Cummings, Redwood City, CA.

MORET, B. M. E., AND H. D. SHAPIRO. 1991b. *Algorithms from P to NP Volume 2 Heuristics and Complexity*. Benjamin/Cummings, Redwood City, CA.

MUKHOPADHYAY, K., AND B. P. SINHA. 1992. Reliability Analysis of Networks Using Stochastic Models. *Information Sciences* **65**, 225-237.

NETER, J., AND W. WASSERMAN. 1974. *Applied Linear Statistical Models*. Richard D. Irwin, Homewood, IL.

NOETHER, G. E. 1967. *Elements of Nonparametric Statistics*. John Wiley, New York.

O'CONNER, D. R. 1981. *Exact Distributions of Stochastic PERT Networks*. University College Dublin, Ireland.

O'NEAL, K. R. 1987. Project Management Microcomputer Software Buyer's Guide. *Industrial Engineering* **19** (1), 53-63.

PAPOULIS, A. 1965. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York.

PARZEN, E. 1960. *Modern Probability Theory and Its Applications*. John Wiley, New York.

PATTERSON, J. 1984. A Comparison of Exact Approaches for Solving the Multiple Constrained Resource Project Scheduling Problem. *Management Science* **30**, 845-867.

PHILLIPS, D. T., A. RAVINDRAN, AND J. J. SOLBERG. 1976. *Operations Research: Principles and Practice*. John Wiley, New York.

PRITSKER, A. A. B. 1966. *GERT: Graphical Evaluation and Review Technique*. Memorandum RM-4973. The Rand Corporation, Santa Monica, CA.

PRITSKER, A. A. B. 1978. *Modeling and Analysis Using Q-GERT Networks*. Halsted Press, New York.

PRITSKER, A. A. B. 1986. *Introduction to Simulation and SLAM II*. Halsted Press, New York.

PRITSKER, A. A. B., AND W. W. HAPP. 1966. GERT: Graphical Evaluation and Review Technique, Part I, Fundamentals. *Journal of Industrial Engineering* **17**, 267-274.

PRITSKER, A. A. B., AND P. J. KIVIAT. 1969. *Simulation with GASP II*. Prentice-Hall, Englewood Cliffs, NJ.

PRITSKER, A. A. B., AND G. E. WHITEHOUSE. 1966. GERT: Graphical Evaluation and Review Technique, Part II, Probabilistic and Industrial Engineering Applications. *Journal of Industrial Engineering* **17**, 293-301.

RAGSDALE, C. 1989. The Current State of Network Simulation in Project Management Theory and Practice. *Omega* **17**, 21-25.

RINGER, L. 1966. *Analytic and Monte Carlo Distribution Theory for PERT.* Ph.D. Dissertation. Department of Statistics, Texas A&M University, College Station, TX.

RINGER, L. 1969. Numerical Operators for Statistical PERT Critical Path Analysis. *Management Science* **16**, B-136 - B-143.

ROBILLARD; P., AND M. TRAHAN. 1976. Expected Completion Time in PERT Networks. *Operations Research* **24**, 177-182.

ROBILLARD, P., AND M. TRAHAN. 1977. Completion Time of PERT Networks. *Operations Research* **25**, 15-29.

ROGERS, D. F., R. D. PLANTE, R. T. WONG, AND J. R. EVANS. 1991. Aggregation and Disaggregation Techniques and Methodology in Optimization. *Operations Research* **39**, 553-582.

ROGERS, T. 1993. Project Management: Emerging as a Requisite for Success. *Industrial Engineering* **25** **(6)**, 42-43.

SAS INSTITUTE INC. 1989. *SAS Language and Procedures. Usage.* Version 6, 1st Ed. Cary, NC.

SAS INSTITUTE INC. 1990. *SAS /STAT User's Guide.* Vol.2. *GLM-VARCOMP.* Version 6, 4th Ed., Cary, NC.

SCHONBERGER, R. J. 1981. Why Projects Are "Always" Late: A Rationale Based on Manual Simulation of a PERT/CPM Network. *Interfaces* **11**, 66-70.

SCULLI, D. 1989. A Historical Note on PERT Times. *Omega* **17**, 195-196.

SCULLI, D., AND Y. W. SHUM 1991. An Approximate Solution to the PERT Problem. *Computers & Mathematics with Applications* **21**, 1-7.

SCULLI, D., AND K. L. WONG. 1985. The Maximum and Sum of Two Beta Variables and the Analysis of PERT Networks. *OMEGA* **13**, 233-240.

SEDGEWICK, R. 1983. *Algorithms.* Addison-Wesley, Reading, MA.

SHOGAN, A. W. 1977. Bounding Distributions for a Stochastic PERT Network. *Networks* **7**, 359-381.

SIGAL, C. E. 1977. *The Stochastic Shortest Route Problem.* Ph.D. Dissertation. School of Industrial Engineering, Purdue University, West Lafayette, IN.

SIGAL, C. E., A. A. B. PRITSKER, AND J. J. SOLBERG. 1979. The Use of Cutset in Monte Carlo Analysis of Stochastic Networks. *Math. Comp. Simulation* **21**,379-384.

SIGAL, C. E., A. A. B. PRITSKER, AND J. J. SOLBERG. 1980. The Stochastic Shortest Route Problem. *Operations Research* **28**, 1122-1129.

SIMAO, H. P., AND W. B. POWELL. 1992a. Numerical Methods for Simulating Transient, Stochastic Queueing Networks - I: Methodology. *Transportation Science* **26**, 296-311.

SIMAO, H. P., AND W. B. POWELL. 1992b. Numerical Methods for Simulating Transient, Stochastic Queueing Networks - II: Experimental Design. *Transportation Science* **26**, 312-329.

SMITH, D. J. 1992. Modelling Small Project Networks with a Microcomputer. *Management Services* **36**, 26-31.

SOLBERG, J. J. 1978 and 1979. Personal Interviews. School of Industrial Engineering, Purdue University, West Lafayette, IN.

STROUD, A. H. 1971. *Approximate Calculation of Multiple Integrals.* Prentice-Hall, Englewood Cliffs, NJ.

TALBOT, F. 1982. Resource-Constrained Project Scheduling With Time-Resource Tradeoffs: The Non-preemptive Case. *Management Science* **28**, 1197-1210.

THOMAS, W. 1969. Four Float Measures for Critical Path Scheduling. *Industrial Engineering* **1**, 19-23.

TVERSKY, 1969. Intransitivity of Reference. *Psychology Review* **76**,31-48.

VALDES,J., R. E. TARJAN, AND E. L. LAWLER. 1979. The Recognition of Series-Parallel Digraphs. *Proceedings of 11th Ann. AMC Symp. on Theory of Computing*, Atlanta, GA.

VANSLYKE, R. M. 1963. Monte Carlo Methods and the PERT Problem. *Operations Research* **11**, 839-860.

WAGNER, H. 1969. *Principles of Operations Research with Applications to Managerial Decisions.* Prentice Hall, Englewood Cliffs, NJ.

WAGNER, H. 1975. *Principles of Operations Research,* 2nd ed. Prentice-Hall, Englewood Cliffs, NJ.

WALLACE, S. W. 1989. Bounding the Expected Time-Cost Curve for a Stochastic PERT Network from Below. *Operations Research Letters* **8**, 89-94.

WANG, L., AND W. E. WILHELM. 1993. A PERT-Based Paradigm for Modeling Assembly Operations. *IEE Transactions* **25**, 88-103.

WELSH, D. J. A. 1965. Errors Introduced by a PERT Assumption. *Operations Research* **13**, 141-143.

WHEELWRIGHT, J. C. 1986. How To Choose The Project Management Microcomputer Software That's Right For You. *Industrial Engineering* **18 (1)**, 46-50.

WHITEHOUSE, G. E. 1973. *Systems Analysis and Design Using Network Techniques.* Prentice-Hall, Englewood Cliffs, NJ.

WHITEHOUSE, G. E. AND A. A. B. PRITSKER. 1969. GERT: Part III - Further Statistical Results; Counters; Renewal Times and Correlations. *IE Transactions* **1**, 45-50.

WIEST, J. D., AND F. K. LEVY. 1977. *A Management Guide to PERT/CPM with GERT/PDM/DCPM and Other Networks.* Prentice-Hall, Englewood Cliffs, NJ.

WILLIAMS, T. M. 1992. Criticality in Stochastic Networks. *Journal of the Operational Research Society* **43**, 353-357.

WOLLMER, R. D. 1968. Stochastic Sensitivity Analysis of Maximum Flow and Shortest Route Networks. *Management Science* **14**, 551-564.

WOLLMER, R. D. 1985. Critical Path Planning Under Uncertainty. *Mathematical Programming Study* **25**, 164-171.

YAKOWITZ, S. 1977. *Computational Probability and Simulation.* Addison-Wesley, Reading, MA.

APPENDIX A

PART PROGRAM WITH "INDEPENDENT MULTIPLE ARCS" APPROXIMATION

```
C
C
C             POLYGONAL APPROXIMATION AND REDUCTION TECHNIQUE
C                                 (PART)
C                               ALGORITHM
C                                  FOR
C                      ACYCLIC, DIRECTED NETWORKS
C                                 USING
C                 "INDEPENDENT MULTIPLE ARCS" NETWORKS
C                            TO APPROXIMATE
C                         NONSEPARABLE NETWORKS
C
C
C     THIS PROGRAM IS WRITTEN IN FORTRAN 77 AND IS PRESENTLY DESIGNED
C     TO BE OPERATED IN A TIME SHARING MODE WITH ALL DATA INPUT FROM
C     THREE (3) DATA FILES.  THE PROGRAM DIRECTS OUTPUT IN EIGHT (8)
C     OPTIONAL FORMATS TO A TIME SHARING TERMINAL.  IF DESIRED, THE
C     READ STATEMENTS AT THE BEGINNING OF THE MAIN PROGRAM CAN BE
C     MODIFIED TO ALLOW DATA INPUT DIRECTLY FROM THE TIME SHARING
C     TERMINAL.
C
C     THE CURRENT DIMENSIONS OF THE PROGRAM ALLOW A NETWORK WITH A
C     MAXIMUM OF 100 NODES AND A MAXIMUM OF 99 ACTIVITIES BEGINNING
C     AT EACH NODE.  THESE LIMITS CAN BE EXPANDED BY CHANGING THE
C     DIMENSIONS OF THE XINT AND VALUE ARRAYS.
C
C
C     ***********************************************************
C
C                 O P E R A T I N G   I N S T R U C T I O N S
C
C     ***********************************************************
C
C     INSTRUCTIONS FOR BUILDING DATA FILES
C     ------------------------------------
C
C           DATA FILE DATAN.DAT
C
C     THIS DATA FILE CONTAINS A DESCRIPTION OF THE NETWORK STRUCTURE.
C     EACH NODE REQUIRES 4 RECORDS WITH A TOTAL OF 103 FIELDS:
C        FIELD 1 IS THE BEGINNING NODE.
C        FIELDS 2 THRU 100 ARE THE NUMBERS OF THE NODES AT WHICH THE
C            ACTIVITIES WHICH BEGIN AT THE NODE IN FIELD 1 TERMINATE.
C        FIELD 101 IS A DUMMY FIELD AND SHOULD BE SET EQUAL TO 0 OR 1.
C        FIELD 102 INDICATES HOW MANY ACTIVITIES TERMINATE AT THE NODE
C            INDICATED IN FIELD NUMBER 1.
C        FIELD 103 INDICATES HOW MANY ACTIVITIES BEGIN AT THE NODE INDI-
C            CATED IN FIELD NUMBER 1.
C     RECORD 1 CONTAINS FIELDS 1-26; RECORD 2 CONTAINS FIELDS 27-52;
C     RECORD 3 CONTAINS FIELDS 53-78; RECORD 4 CONTAINS FILEDS 79-103.
C
C
C           DATA FILE DATAH.DAT
C
C     THIS DATA FILE CONTAINS A DESCRIPTION OF THE DISTRIBUTIONS OF
C     EACH OF THE ACTIVITIES IN THE NETWORK.
```

```
C
C     THERE ARE 7 FIELDS OF DATA.
C     FIELD 1 IS THE NODE NUMBER.
C     FIELD 2 IS THE NUMBER OF THE ACTIVITY COMING FROM THE NODE.
C     FIELD 3 IS THE CODE FOR THE TYPE OF DISTRIBUTION.
C         1 = TRIANGULAR DISTRIBUTION
C         2 = NORMAL DISTRIBUTION
C         3 = EXPONENTIAL DISTRIBUTION
C         4 = GAMMA DISTRIBUTION
C         5 = BETA DISTRIBUTION
C         6 = UNIFORM DISTRIBUTION
C     FIELD 4 IS
C         MODE FOR A TRIANGULAR DISTRIBUTION.
C         MEAN FOR A NORMAL DISTRIBUTION.
C         MEAN FOR AN EXPONENTIAL DISTRIBUTION.
C         ALPHA FOR A GAMMA OR A BETA DISTRIBUTION.
C         1/(B-A) FOR A UNIFORM DISTRIBUTION.
C     FIELD 5 IS BETA FOR A GAMMA OR A BETA DISTRIBUTION.
C     FIELD 6 IS THE MINIMUM VALUE OF THE DISTRIBUTION.
C     FIELD 7 IS THE MAXIMUM VALUE OF THE DISTRIBUTION.
C
C
C           DATA FILE CONTROL.DAT
C
C     THIS IS A SINGLE LINE DATA FILE WHICH CONTAINS CONTROL
C     PARAMETERS FOR INPUT, OUTPUT, AND MONTE CARLO SIMULATION.
C
C     THERE ARE 4 FIELDS OF DATA.
C     FIELD 1 IS THE NUMBER OF NODES IN THE NETWORK.
C     FIELD 2 IS THE NUMBER OF ACTIVITIES IN THE NETWORK.
C     FIELD 3 IS THE OUTPUT OPTION DESIRED FOR THE PART RESULTS.
C         1 = A DESCRIPTION OF EACH OF THE 10 CLASSES OF THE
C             FINAL DISTRIBUTION IN THE FORM OF Y = B(O) + B(1) X.
C         2 = A CUMULATIVE DISTRIBUTION FUNCTION OF THE FINAL
C             DISTRIBUTION.
C         3 = A DISCRETE PROBABILITY DENSITY FUNCTION AND A
C             SIMULATION FREQUENCY HISTOGRAM IN GRAPHICAL FORMAT.
C         4 = A COMBINATION OF 1 AND 2 ABOVE.
C         5 = A COMBINATION OF 1 AND 3 ABOVE.
C         6 = A COMBINATION OF 2 AND 3 ABOVE.
C         7 = A COMBINATION OF 1, 2, AND 3 ABOVE.
C         8 = ONLY THE EXPECTED VALUE AND STANDARD DEVIATION.
C     FIELD 4 IS THE NUMBER OF ITERATIONS OF THE MONTE CARLO
C     SIMULATION REQUESTED (MAXIMUM = 10,000).
C         0 = NO MONTE CARLO SIMULATION IS REQUESTED.
C
C
C     NOTE
C
C     ALL UNUSED FIELDS MUST BE ZEROED OUT.
C
C
C
C     ***********************************************************
C
C                   M A I N     P R O G R A M
```

```
C
C     ***********************************************************
C
      REAL*8 XINT(200,99,12),VALUE(200,99,10,3),A(130)
      REAL*8 ZVAL(130,5),XX(100,2),TOTAAR(51)
      REAL*8 SIMT(100,10000),SIMTOT(51)
      REAL*8 AREA,AVG,COUNT,SIG,SIZE
     *                ,XD1,XD2,XD3,XD4,X,XSIZE
     *                ,DIFF
      REAL*4 HIGH,HLOW,PERCNT,KSCR20,KSCR10,KSCR05,KSCR02,KSCR01,DMAX
      INTEGER I,IDUAL,IEDN,IFLAG,IPRINT,ISTN,ISTNP
     *       ,MM
     *       ,N,NACTS,NAN,NCL,NCROSS,NET,NETT,NSIM,NSTART
     *       ,J,J1,J2,J3
     *       ,K,KK
     *       ,L,L1,L2,L3,L4,LASTK,LASTMM,L3COUNT
      DIMENSION NET(200,103),NETT(103)
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      COMMON/PARA4/XX
      COMMON/PARA5/NET
      COMMON/PARA6/SIMT
      CHARACTER*1 KBL,KBM
      DATA KBL/' '/,KBM/'*'/
      DATA NCL/0/,NCROSS/0/
C
C     OPEN INPUT AND OUTPUT FILES
C
      OPEN (UNIT = 11, FILE = 'datan.dat')
      OPEN (UNIT = 12, FILE = 'datah.dat')
      OPEN (UNIT = 13, FILE = 'control.dat')
C
C     READ INFORMATION INTO DATA MATRICES.
C
      READ (13,1900) N,NACTS,NAN,NSIM
      DO 0910 I = 1,N
      READ (11,1901) (NETT(J), J = 1,103)
      L1 = NETT(1)
      DO 0900 K = 1,103
      NET(L1,K) = NETT(K)
 0900 CONTINUE
 0910 CONTINUE
C
C     DO 1010 READS DATA FROM DATAH AND LOADS THIS DATA INTO
C     THE VALUE AND XINT ARRAYS.  THIS DO ALSO DETERMINES IF
C     THE ACTIVITY DISTRIBUTION IS OTHER THAN UNIFORM, AND,
C     AND, IF SO, CALLS LINEAR TO APPROXIMATE IT WITH A
C     PIECEWISE POLYGONAL FUNCTION.
C
      DO 1010 I = 1,NACTS
      READ (12,1902) L1,L2,XD1,XD2,XD3,XD4,XD5
      VALUE(L1,L2,1,3) = XD1
      VALUE(L1,L2,2,3) = XD2
      VALUE(L1,L2,3,3) = XD3
      XINT(L1,L2,1) = XD4
      XINT(L1,L2,2) = XD5
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 6) THEN
      CALL LINEAR (L1,L2,NCL)
      GO TO 1010
C
C     DO 1000 CONVERTS DATA FOR UNIFORM DISTRIBUTIONS INTO A USABLE
C     FORM FOR SUBROUTINES SERIES AND PARA.
C
      ELSE
      XINT(L1,L2,11) = XINT(L1,L2,2)
      X = XINT(L1,L2,1)
      XSIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      DO 1000 J = 1,10
      VALUE(L1,L2,J,1) = VALUE(L1,L2,2,3)
      XINT(L1,L2,J) = X
      X = X+XSIZE
 1000 CONTINUE
      END IF
 1010 CONTINUE
C
C     MONTE CARLO SIMULATION OF THE NETWORK.
C
      IF (NSIM .EQ. 0) GO TO 1030
      NSTART = N
      CALL SIMULT (N,NSIM)
C
C     REDUCTION OF THE NETWORK BEGINS.
C
C     DO 1040 CHECKS IF A CONVOLUTION (SERIES-REDUCTION) OPERATION
C     IS POSSIBLE, i.e., IF THERE EXISTS A NODE I NOT ON THE OUTPUT
C     CRITICAL LIST SUCH THAT
C         IN-DEGREE NODE I = OUT-DEGREE NODE I = 1.
C
 1030 L3COUNT = 2
 1035 DO 1040 I=L3COUNT,N-1
      L3 = I
      IF ((NET(I,102)+NET(I,103)) .EQ. 2) GO TO 1050
 1040 CONTINUE
C
C     IF (IN-DEGREE NODE I + OUT-DEGREE NODE I) > 2 FOR ALL I NOT = 1
C     OR N, NETWORK IS NONSEPARABLE, SO PROCEED TO "INDEPENDENT
C     MULTIPLE ARCS" APPROXIMATION.
C
      IF (L3COUNT .EQ. 2) GO TO 1145
      GO TO 1080
C
C     A CONVOLUTION IS POSSIBLE WITH THE TWO ACTIVITIES, ONE OF WHICH
C     TERMINATES AT NODE L3 AND THE OTHER OF WHICH STARTS AT NODE L3.
C     DO 1060 IDENTIFIES THE STARTING NODE NUMBER AND THE ACTIVITY
C     NUMBER OF THE ACTIVITY TERMINATING AT NODE L3.  THEN THE SERIES
C     SUBNETWORK CONSISTING OF THESE TWO ACTIVITIES IS CONVOLUTED INTO
C     AN EQUIVALENT ACTIVITY.
C
 1050 DO 1060 I=1,L3-1
      DO 1060 J=2,NET(I,103)+1
      L1 = I
      L2 = J-1
      IF (NET(I,J) .EQ. L3) GO TO 1070
 1060 CONTINUE
 1070 CALL SERIES(L1,L2,L3,1)
      NET(L1,L2+1) = NET(L3,2)
      NET(L3,2) = 0
      NET(L3,101) = 0
      NET(L3,102) = 0
      NET(L3,103) = 0
      L3COUNT = L3+1
      IF (L3COUNT .EQ. N) GO TO 1080
      GO TO 1035
C
C     DO 1140 CHECKS IF A MAXIMUM (PARALLEL-REDUCTION) OPERATION IS
C     POSSIBLE, i.e., IF THERE EXIST TWO DIFFERENT ACTIVITIES, A1 AND
C     A2, SUCH THAT
C         STARTING NODE (A1) = STARTING NODE (A2), AND
C           ENDING NODE (A1) = ENDING NODE (A2).
C     THEN THE PARALLEL SUBNETWORK CONSISTING OF THESE TWO ACTIVITIES
C     IS PARALLEL-REDUCED WITH A MAXIMUM OPERATION INTO AN EQUIVALENT
C     ACTIVITY.
C
 1080 DO 1140 I=1,N-1
      L1 = I
 1085 DO 1090 J=2,NET(L1,103)
      L2 = J-1
      IF (NET(L1,J) .EQ. 0) GO TO 1140
      DO 1090 K=J+1,NET(L1,103)+1
      L3 = K-1
      IF (NET(L1,J) .EQ. NET(L1,K)) THEN
      IEDN = NET(L1,J)
      GO TO 1110
      ELSE
      GO TO 1090
      END IF
 1090 CONTINUE
      GO TO 1140
 1110 CALL PARA(L1,L2,L3)
      NET(L1,103) = NET(L1,103)-1
      NET(IEDN,102) = NET(IEDN,102)-1
      DO 1120 K=L3,NET(L1,103)
      NET(L1,K+1) = NET(L1,K+2)
      DO 1115 L=1,10
      XINT(L1,K,L)= XINT(L1,K+1,L)
      VALUE(L1,K,L,1) = VALUE(L1,K+1,L,1)
      VALUE(L1,K,L,2) = VALUE(L1,K+1,L,2)
 1115 CONTINUE
      XINT(L1,K,11) = XINT(L1,K+1,11)
 1120 CONTINUE
      K = NET(L1,103)+1
      NET(L1,K+1) = 0
      DO 1130 L=1,10
      XINT(L1,K,L) = 0.
      VALUE(L1,K,L,1) = 0.
      VALUE(L1,K,L,2) = 0.
 1130 CONTINUE
```

```
      XINT(L1,K,11) = 0.
      GO TO 1085
 1140 CONTINUE
      GO TO 1030
C
C     THROUGH 1146 CHECKS IF THE NETWORK HAS BEEN SERIES-PARALLEL
C     REDUCED TO A SINGLE EQUIVALENT ACTIVITY.
C
 1145 IF (NET(1,103) .NE. 1) GO TO 1147
      IF (NET(N,102) .NE. 1) GO TO 1147
      DO 1146 I=2,N-1
      IF (NET(I,102) + NET(I,103)) 1540,1146,1147
 1146 CONTINUE
C
C     THE NETWORK HAS BEEN SERIES-PARALLEL REDUCED TO A SINGLE EQUIVA-
C     LENT ACTIVITY.
C
      GO TO 1240
C
C     TO 1240 REDUCES THE NONSEPARABLE NETWORK USING "INDEPENDENT MULTI-
C     PLE ARCS" APPROXIMATION WITH THE "FIRST AVAILABLE ARC WITH
PROPERTY
C     1" METHOD.
C
 1147 NCROSS = NCROSS+1
C
C     DO 1148 IDENTIFIES THE START NODE ISTN OF THE FIRST AVAILABLE
C     CROSS-CONNECTION IN THE NONSEPARABLE NETWORK.
C
      DO 1148 I=2,N-2
      IF ((NET(I,102) .EQ. 1) .AND. (NET(I,103) .GT. 1)) THEN
      ISTN = I
      GO TO 1149
      ELSE
      GO TO 1148
      END IF
 1148 CONTINUE
C
C     THROUGH 1151 IDENTIFIES THE START NODE ISTNP AND THE ACTIVITY
C     NUMBER IDUAL OF THE SINGLE ACTIVITY TERMINATING AT NODE ISTN.
C     THIS ACTIVITY IS THE "A" ACTIVITY CONNECTING NODE ISTNP AND NODE
C     ISTN WHICH MUST BE "INDEPENDENTLY MULTIPLIED."
C
 1149 DO 1151 I=1,N-3
      DO 1150 J=2,NET(I,103)+1
      IF (NET(I,J) .EQ. ISTN) THEN
      ISTNP = I
      IDUAL = J-1
      GO TO 1152
      ELSE
      GO TO 1150
      END IF
 1150 CONTINUE
 1151 CONTINUE
C
C     NDUAL IS THE NUMBER OF "INDEPENDENT MULTIPLES" OF ACTIVITY IDUAL
```

```
C     WHICH MUST BE INSERTED INTO THE NETWORK.
C
 1152 NDUAL = NET(ISTN,103)-1
      NNEW = N+NDUAL
C
C     DO 1153 INCREASES NODE NUMBERS ABOVE ISTN BY NDUAL.
C
      DO 1153 J=1,N
      DO 1153 J1=1,NET(J,103)+1
      IF (NET(J,J1) .GT. ISTN) NET(J,J1) = NET(J,J1)+NDUAL
 1153 CONTINUE
C
C     DO 1154 SHIFTS ROWS OF NET ARRAY, FROM N DOWN TO ISTN+1, AHEAD
C     NDUAL ROWS.
C
      DO 1154 J=NNEW,ISTN+NDUAL+1,-1
      DO 1154 J1=1,103
      NET(J,J1) = NET(J-NDUAL,J1)
      NET(J-NDUAL,J1)=0
 1154 CONTINUE
C
C     DO 1155 INSERTS NDUAL NODES AFTER NODE ISTN FOR NDUAL "INDEPENDENT
C     MULTIPLES" OF THE "PROPERTY 1" ACTIVITY IN THE NET ARRAY.
C
      DO 1155 J=ISTN+1,ISTN+NDUAL
      NET(J,1) = J
      NET(J,2) = NET(ISTN,J-ISTN+2)
      NET(ISTN,J-ISTN+2) = 0
      NET(J,101) = 1
      NET(J,102) = 1
      NET(J,103) = 1
 1155 CONTINUE
      NET(ISTN,103) = 1
C
C     DO 1156 SHIFTS THE TERMINATING NODE NUMBERS OF ACTIVITIES STARTING
C     AT NODE ISTNP AFTER ISTN AHEAD NDUAL POSITIONS IN THE NET ARRAY.
C
      DO 1156 J=NET(ISTNP,103)+NDUAL+1,IDUAL+NDUAL+2,-1
      NET(ISTNP,J) = NET(ISTNP,J-NDUAL)
      NET(ISTNP,J-NDUAL) = 0
 1156 CONTINUE
C
C     DO 1157 INSERTS NDUAL NODES - ISTN+1,...,ISTN+NDUAL - AS THE
C     TERMINATING NODES OF THE "INDEPENDENTLY MULTIPLIED" "A" ACTIVITY
C     FROM NODE ISTNP IN THE NET ARRAY.
C
      DO 1157 J=1,NDUAL
      NET(ISTNP,IDUAL+J+1) = ISTN+J
 1157 CONTINUE
      NET(ISTNP,103) = NET(ISTNP,103)+NDUAL
C
C     DO 1158 SHIFTS ROWS OF XINT ARRAY, FROM N DOWN TO ISTN+1, AHEAD
C     NDUAL ROWS.
C
      DO 1158 J=NNEW,ISTN+NDUAL+1,-1
      DO 1158 J1=1,99
      DO 1158 J2=1,12
      XINT(J,J1,J2) = XINT(J-NDUAL,J1,J2)
      XINT(J-NDUAL,J1,J2) = 0.
 1158 CONTINUE
C
C     DO 1159 SHIFTS ACTIVITY LINEARIZATION POINTS FOR ACTIVITIES
C     2,...,NDUAL+1 FROM NODE ISTN TO THE FIRST AND ONLY ACTIVITIES
C     FROM THE NEW NODES - ISTN+1,...,ISTN+NDUAL - IN THE XINT ARRAY.
C
      DO 1159 J1=1,NDUAL
      DO 1159 J2=1,12
      XINT(ISTN+J1,1,J2) = XINT(ISTN,J1+1,J2)
      XINT(ISTN,J1+1,J2) = 0.
 1159 CONTINUE
C
C     DO 1160 SHIFTS ACTIVITY LINEARIZATION POINTS FOR ACTIVITIES
C     AFTER IDUAL FROM NODE ISTN AHEAD NDUAL POSITIONS IN THE XINT
ARRAY.
C
      DO 1160 J1=NET(ISTNP,103),IDUAL+NDUAL+1,-1
      DO 1160 J2=1,12
      XINT(ISTNP,J1,J2) = XINT(ISTNP,J1-NDUAL,J2)
      XINT(ISTNP,J1-NDUAL,J2) = 0.
 1160 CONTINUE
C
C     DO 1161 INSERTS NDUAL COPIES OF THE ACTIVITY LINEARIZATION POINTS
C     OF ACTIVITY IDUAL FROM NODE ISTNP TO THE NEW NODES - ISTN+1,...,
C     ISTN+NDUAL - IN THE XINT ARRAY.
C
      DO 1161 J1=1,NDUAL
      DO 1161 J2=1,12
      XINT(ISTNP,IDUAL+J1,J2) = XINT(ISTNP,IDUAL,J2)
 1161 CONTINUE
C
C     DO 1162 SHIFTS ROWS OF VALUE ARRAY, FROM N DOWN TO ISTN+1, AHEAD
C     NDUAL ROWS.
C
      DO 1162 J=NNEW,ISTN+NDUAL+1,-1
      DO 1162 J1=1,99
      DO 1162 J2=1,10
      DO 1162 J3=1,3
      VALUE(J,J1,J2,J3) = VALUE(J-NDUAL,J1,J2,J3)
      VALUE(J-NDUAL,J1,J2,J3) = 0.
 1162 CONTINUE
C
C     DO 1163 SHIFTS ACTIVITY POLYGONAL APPROXIMATION COEFFICIENTS FOR
C     ACTIVITIES 2,...NDUAL+1 FROM NODE ISTN TO THE FIRST AND ONLY
C     ACTIVITIES FROM THE NEW NODES - ISTN+1,...,ISTN+NDUAL - IN THE
C     VALUE ARRAY.
C
      DO 1163 J1=1,NDUAL
      DO 1163 J2=1,10
      DO 1163 J3=1,3
      VALUE(ISTN+J1,1,J2,J3) = VALUE(ISTN,J1+1,J2,J3)
      VALUE(ISTN,J1+1,J2,J3) = 0.
 1163 CONTINUE
```

```
C
C     DO 1164 SHIFTS ACTIVITY POLYGONAL APPROXIMATION COEFFICIENTS FOR
C     ACTIVITIES AFTER IDUAL FROM NODE ISTNP AHEAD NDUAL POSITIONS IN
THE
C     VALUE ARRAY.
C
      DO 1164 J1=NET(ISTNP,103),IDUAL+NDUAL+1,-1
      DO 1164 J2=1,10
      DO 1164 J3=1,3
      VALUE(ISTNP,J1,J2,J3) = VALUE(ISTNP,J1-NDUAL,J2,J3)
      VALUE(ISTNP,J1-NDUAL,J2,J3) = 0.
 1164 CONTINUE
C
C     DO 1165 INSERTS NDUAL COPIES OF THE ACTIVITY POLYGONAL APPROXIMA-
C     TION COEFFICIENTS OF ACTIVITY IDUAL FROM NODE ISTNP TO THE NEW
C     NODES - ISTN+1,...,ISTN+NDUAL - IN THE VALUE ARRAY.
C
      DO 1165 J1=1,NDUAL
      DO 1165 J2=1,10
      DO 1165 J3=1,3
      VALUE(ISTNP,IDUAL+J1,J2,J3) = VALUE(ISTNP,IDUAL,J2,J3)
 1165 CONTINUE
      N = NNEW
      GO TO 1030
C
C     AT THIS POINT IN THE MAIN PROGRAM ALL CALCULATIONS HAVE
C     BEEN COMPLETED AND DATA IS PREPARED FOR FINAL OUTPUT.
C
 1240 PRINT 1910
      L = 1
      KK = 0
      DO 1270 I = 1,10
C
C     THE XX ARRAY IS USED FOR HISTOGRAM AND CDF CALCULATIONS.
C
      XX(1,1) = XINT(1,1,I)
      SIZE = (XINT(1,1,2)-XINT(1,1,1))/5.
      LASTK = L+4
      DO 1250 K = L,LASTK
      KK = KK+1
      XX(K,2) = VALUE(1,1,I,1)+(VALUE(1,1,I,2)*XX(K,1))
      IF ((KK .LE. 1).AND.(L .GT. 4)) XX(K,2) = (((VALUE(1,1,I,2)
     &*XX(K,1))+VALUE(1,1,I,1))+(VALUE(1,1,I-1,2)*XX(K,1))+
     &VALUE(1,1,I-1,1))/2.
      XX(K+1,1) = XX(K,1)+SIZE
1250  CONTINUE
      KK = 0
      L = L+5
      IF ((NAN .EQ. 1).OR.(NAN .EQ. 4).OR.(NAN .EQ. 5).OR.
     &(NAN .EQ. 7)) GO TO 1260
      GO TO 1270
 1260 IF (I .EQ. 1) PRINT 1920
      PRINT 1930,I,XINT(1,1,I),XINT(1,1,I+1)
      PRINT 1940,VALUE(1,1,I,1),VALUE(1,1,I,2)
 1270 CONTINUE
      XX(51,2)=VALUE(1,1,10,1)+VALUE(1,1,10,2)*XX(51,1)
```

```
C
C     TOTAAR IS USED FOR CDF CALCULATIONS.
C
      AREA = 0.0
      DO 1280 I = 1,50
      AREA = AREA+((XX(I,2)+XX(I+1,2))*SIZE*.5)
      TOTAAR(I) = AREA
 1280 CONTINUE
      AREA = 1.0/AREA
      DO 1290 I = 1,50
      TOTAAR(I) = TOTAAR(I)*AREA
 1290 CONTINUE
      DO 1295 I = 51,2,-1
      TOTAAR(I) = TOTAAR(I-1)
 1295 CONTINUE
      TOTAAR(1) = 0.0
      XX(1,1) = XINT(1,1,1)
      IF ((NAN .EQ. 2).OR.(NAN .EQ. 4).OR.(NAN .EQ. 6).OR.
     &(NAN .EQ. 7)) GO TO 1300
      GO TO 1320
 1300 PRINT 1910
      PRINT 1950
      DO 1310 I = 1,51
      PRINT 1960,XX(I,1),TOTAAR(I)
 1310 CONTINUE
 1320 IF ((NAN .EQ. 3).OR.(NAN .EQ. 5).OR.(NAN .EQ. 6).OR.
     &(NAN .EQ. 7)) GO TO 1330
      GO TO 1340
 1330 IPRINT = 51
      IFLAG = 0
      CALL PLOT (IPRINT,KBL,KBM,IFLAG)
 1340 CONTINUE
C
C     DO 1360 COMPUTES AN APPROXIMATED EXPECTED VALUE USING
C     GROUPED DATA.  DO 1370 COMPUTES AN APPROXIMATED STANDARD
C     DEVIATION.
C
      AVG = 0.0
      SIG = 0.0
      LASTMM = IPRINT-1
      DO 1360 MM = 1,LASTMM
      AVG = AVG+((XX(MM,1)+(SIZE/2.))*(TOTAAR(MM+1)-TOTAAR(MM)))
 1360 CONTINUE
      DO 1370 MM = 1,LASTMM
      SIG = SIG+(((XX(MM,1)+(SIZE/2.)-AVG)**2)*(TOTAAR(MM+1)-
     &TOTAAR(MM)))
 1370 CONTINUE
      SIG = DSQRT(SIG)
      PRINT 1910
      PRINT 1970,AVG,SIG
      DO 1380 MM = 1,4
      HLOW = AVG-(FLOAT(MM)*SIG)
      HIGH = AVG+(FLOAT(MM)*SIG)
C
C     THESE ARE FIXED PERCENTAGES.  THE ASSUMPTION IS MADE THAT
C     THE FINAL PRODUCT WILL RESEMBLE A NORMAL DISTRIBUTION.  THEY
```

```
C     CORRESPOND TO 1, 2, 3, AND 4 STANDARD DEVIATIONS RESPECTIVELY.
C
      IF (MM .EQ. 1) PERCNT = 68.24
      IF (MM .EQ. 2) PERCNT = 95.44
      IF (MM .EQ. 3) PERCNT = 99.73
      IF (MM .EQ. 4) PERCNT = 99.99
      PRINT 1980,HLOW,HIGH,PERCNT
 1380 CONTINUE
      PRINT 1985,N,NCROSS
      IF (NSIM .EQ. 0) GO TO 1530
C
C     THROUGH 1500 COMPILES OUTPUT FROM THE MONTE CARLO SIMULATION.
C
C     DO 1410 COMPILES THE CUMULATIVE DISTRIBUTION FUNCTION.
C
      COUNT = 0.0
      SIMTOT(1) = 0.0
      DO 1410 J = 1,50
      DO 1400 K = 1,NSIM
      IF((XX(J,1) .LE. SIMT(NSTART,K)) .AND. (SIMT(NSTART,K) .LT.
     &XX(J+1,1))) COUNT = COUNT+1.
 1400 CONTINUE
      XX(J+1,2) = COUNT/DFLOAT(NSIM)
      SIMTOT(J+1) = SIMTOT(J)+XX(J+1,2)
      COUNT = 0.0
 1410 CONTINUE
      PRINT 1910
      PRINT 1925
      IF ((NAN .EQ. 2) .OR. (NAN .EQ. 4) .OR. (NAN .EQ. 6) .OR.
     &(NAN .EQ. 7)) GO TO 1420
      GO TO 1440
 1420 PRINT 1950
      DO 1430 J = 1,51
      PRINT 1960,XX(J,1),SIMTOT(J)
 1430 CONTINUE
 1440 IF ((NAN .EQ. 3) .OR. (NAN .EQ. 5) .OR. (NAN .EQ. 6) .OR.
     &(NAN .EQ. 7)) GO TO 1450
      GO TO 1470
 1450 DO 1460 J = 1,50
      XX(J,1) = XX(J,1)+(SIZE/2.)
      XX(J,2) = XX(J+1,2)
 1460 CONTINUE
      IPRINT = 50
      IFLAG = 1
      CALL PLOT (IPRINT,KBL,KBM,IFLAG)
 1470 CONTINUE
C
C     DO 1480 COMPUTES AN APPROXIMATED EXPECTED VALUE AND
C     DO 1490 COMPUTES AN APPROXIMATED STANDARD DEVIATION
C     USING GROUPED DATA.
C
      AVG = 0.0
      SIG = 0.0
      DO 1480 J = 1,50
      AVG = AVG+(XX(J,1)*(SIMTOT(J+1)-SIMTOT(J)))
 1480 CONTINUE
```

```
      DO 1490 J = 1,50
      SIG = SIG+(((XX(J,1)-AVG)**2)*(SIMTOT(J+1)-SIMTOT(J)))
 1490 CONTINUE
      SIG = DSQRT(SIG)
      PRINT 1910
      PRINT 1970,AVG,SIG
      DO 1500 MM = 1,4
      HLOW = AVG-(FLOAT(MM)*SIG)
      HIGH = AVG+(FLOAT(MM)*SIG)
C
C     IT IS ASSUMED THAT THE DISTRIBUTION THROUGH NODE I RESEMBLES
C     A NORMAL DISTRIBUTION.  THE FIXED PERCENTAGES CORRESPOND TO
C     1, 2, 3, AND 4 STANDARD DEVIATIONS, RESPECTIVELY, FROM THE
C     EXPECTED VALUE.
C
      IF (MM .EQ. 1) PERCNT = 68.24
      IF (MM .EQ. 2) PERCNT = 95.44
      IF (MM .EQ. 3) PERCNT = 99.73
      IF (MM .EQ. 4) PERCNT = 99.99
      PRINT 1980,HLOW,HIGH,PERCNT
 1500 CONTINUE
C
C     COMPARE POLYGONAL APPROXIMATION OF THROUGHPUT DISTRIBUTION
C     WITH SIMULATED THROUGHPUT DISTRIBUTION USING THE KOLMOGOROV-
C     SMIRNOV ONE-SAMPLE TEST.
C
      KSCR20 = 1.0730/SQRT(50.)
      KSCR10 = 1.2239/SQRT(50.)
      KSCR05 = 1.3581/SQRT(50.)
      KSCR02 = 1.5174/SQRT(50.)
      KSCR01 = 1.6276/SQRT(50.)
C
C     COMPUTE THE K-S TEST STATISTIC D-MAX.
C
      DMAX = 0.0
      DO 1520 I = 2,51
      DIFF = DABS(SIMTOT(I)-TOTAAR(I))
      IF (DIFF .GT. DMAX) DMAX = DIFF
 1520 CONTINUE
      PRINT 1910
      PRINT 1991,DMAX
      PRINT 1992,KSCR20,KSCR10,KSCR05,KSCR02,KSCR01
      IF (DMAX .LE. KSCR05) PRINT 1993
 1530 STOP
 1540 PRINT 1990
      STOP
C
C     FORMAT STATEMENTS
C
 1900 FORMAT (I3,1X,I4,1X,I1,1X,I5)
 1901 FORMAT (3(I2,25(1X,I2)/),I2,21(1X,I2),1X,I1,2(1X,I2))
 1902 FORMAT (2(I2,1X),F1.0,4(1X,F8.2))
 1910 FORMAT (1H1)
 1920 FORMAT (1X,'THE POLYGONAL APPROXIMATION OF THE TIME DISTRIBUTION',
     &' THROUGH THE PROJECT IS:' ///)
 1925 FORMAT (1X,'THE SIMULATED TIME DISTRIBUTION',
     &' THROUGH THE PROJECT IS:' ///)
 1930 FORMAT (1X,'INTERVAL',I3,4X,'LOWER LIMIT =',F8.2,3X,
     &'UPPER LIMIT =',F8.2 //)
 1940 FORMAT (15X,'X = (',F12.8,') + (',F12.8,') T' ///)
 1950 FORMAT (14X,'CUMULATIVE DISTRIBUTION FUNCTION' //
     &21X,'T',13X,'F(T)')
 1960 FORMAT (16X,F9.3,F17.8)
 1970 FORMAT (12X,'EXPECTED VALUE OF T    =',F13.8 /
     &12X,'STANDARD DEVIATION OF T =',F13.8 //)
 1980 FORMAT (1X,'THE PROBABILITY OF THE PROJECT THROUGHPUT TIME',
     &' FALLING BETWEEN' / 1X,F8.3,' TIME UNITS AND ',F8.3,
     &' TIME UNITS IS ABOUT ',F5.2,' %.'//)
 1985 FORMAT (/ 1X,'THE NUMBER OF NODES IN THE FINAL NETWORK WAS ',I3,
     &'.' // 1X,'THE NUMBER OF CROSS-CONNECTIONS REDUCED WAS ',I2,'.')
 1990 FORMAT (1X,'PROGRAM STOPPED' / 1X,'IMPROPER NODE NUMBER(S) '
     &,'ENCOUNTERED')
 1991 FORMAT (1X,'KOLMOGOROV-SMIRNOV ONE-SAMPLE TEST COMPARISON OF ',
     &'POLYGONAL APPROXIMATION' / 1X,'OF NETWORK THROUGHPUT DISTRIBUTION
     & AND SIMULATED NETWORK THROUGHPUT' / 1X,'DISTRIBUTION:' //
     &1X,'K-S TEST STATISTIC D-MAX = ', F6.4 /)
 1992 FORMAT (1X,'K-S CRITICAL VALUES:' / 15X,'20 PERCENT = ',F6.4 /
     &15X,'10 PERCENT = ',F6.4 / 16X,'5 PERCENT = ',F6.4 /
     &16X,'2 PERCENT = ',F6.4 / 16X,'1 PERCENT = ',F6.4 /)
 1993 FORMAT (1X,'FAIL TO REJECT THE NULL HYPOTHESIS THAT THE ',
     &'DISTRIBUTIONS ARE THE SAME' / 1X,'AT THE 5% LEVEL OF ',
     &'STATISTICAL SIGNIFICANCE.')
      END
C     END MAIN PROGRAM
C
C     *************************************************************
C
C        S U B R O U T I N E           P A R A
C
C     *************************************************************
C
      SUBROUTINE PARA (L1,L2,L3)
      REAL*8 VALUE(200,99,10,3),XINT(200,99,12)
      REAL*8 XVAL,ZVAL(130,5),PAR(2,15,6),FACT,B(130)
      REAL*4 Z
      INTEGER L1,L2,L3,NV1,NV2
      INTEGER K4(2,30)
      INTEGER I,IINT,N,NCL,J,K,K3,L6,LASTJ,LASTK
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/B
C
C     SUBROUTINE PARA IS USED TO REDUCE PARALLEL ARCS INTO A SINGLE
C     EQUIVALENT ARC.  IT FINDS THE MAX OPERATOR BY MULTIPLYING CAP
C     F(X) AGAINST CAP G(X) OVER THE INTERVALS OF VALIDITY.
C
      NV1 = 10
      NV2 = 10
      DO 2020 N = 1,2
      L6 = L2
      IF (N .EQ. 2) L6 = L3
      FACT = 0
      DO 2010 J = 1,10
      B(1) = XINT(L1,L6,J)
C
C     DO 2000 CONVERTS EACH LINEAR POLYNOMIAL PIECE OF LITTLE F(X)
C     INTO THE CORRESPONDING QUADRATIC POLYNOMIAL PIECE OF ITS
C     CUMULATIVE DISTRIBUTION CAP F(X).
C
      DO 2000 I = 1,2
      XVAL = VALUE(L1,L6,J,I)
      Z = FLOAT(I)
      PAR(N,J,I+1) = XVAL/Z
      PAR(N,J,1) = PAR(N,J,1)+((-1.0)*(XVAL/Z)*(B(1)**I))
      K4(N,J) = I+1
 2000 CONTINUE
      IF (J .GT. 1) PAR(N,J,1) = PAR(N,J,1)+FACT
      FACT = PAR(N,J,1)+(PAR(N,J,2)*XINT(L1,L6,J+1))+(PAR(N,J,3)
     &*(XINT(L1,L6,J+1)**2))
 2010 CONTINUE
 2020 CONTINUE
C
C     DO 2040 ASSIGNS INTERVAL BOUNDARY VALUES TO THE B ARRAY.
C
      DO 2040 I = 1,22
      IF (I .GT. 11) GO TO 2030
      B(I) = XINT(L1,L2,I)
      GO TO 2040
 2030 B(I) = XINT(L1,L3,I-11)
 2040 CONTINUE
      NCL = 21
      CALL SORT(NCL)
C
C     DO 2080 DETERMINES THE POINT AT WHICH THE DISTRIBUTION DOMAINS
C     OF THE TWO ARCS BEING COMBINED OVERLAP.  ONCE THIS POINT IS
C     DETERMINED, THE B ARRAY IS ADJUSTED TO REFLECT THE OVERLAP
C     (ALL VALUES LESS THAN THIS POINT OF FIRST OVERLAP NEED NOT BE
C     CONSIDERED, BECAUSE ONE OF THE DISTRIBUTIONS EQUALS ZERO AT
C     THESE VALUES). IF THE DOMAINS ARE DISJOINT OR OVERLAP AT ONLY.
C     ONE BOUNDARY POINT, THE RESULT OF THE APPLICATION OF THE
C     MAXIMUM OPERATOR IS JUST THE UNCHANGED APPROXIMATED PROBABILITY
C     DENSITY FUNCTION OF THE DISTRIBUTION DEFINED ON THE HIGHER-
C     VALUED DOMAIN.  GO TO 2180 OR GO TO 2160 RETURNS THIS FUNCTION
C     DIRECTLY WITHOUT FURTHER PROCESSING.
C
      IINT = 0
      LASTJ = NCL+1
      DO 2080 J = 1,LASTJ
      IF ((XINT(L1,L2,1) .GE. XINT(L1,L3,1)-0.001) .AND.
     &(XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001)) GO TO 2080
      IF (IINT .GE. 1) GO TO 2060
      IF (XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001) GO TO 2050
      IF (XINT(L1,L3,J+1) .GE. XINT(L1,L2,1)-0.001) IINT = J
      IF ((XINT(L1,L3,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L2,1) .GE. XINT(L1,L3,J+1)-0.001)
     &.AND. (XINT(L1,L2,1) .LE. XINT(L1,L3,J+1)+0.001)
     &.AND. (XINT(L1,L3,J+2) .LE. 0.001))) GO TO 2180
      GO TO 2080
 2050 IF (XINT(L1,L2,J+1) .GE. XINT(L1,L3,1)-0.001) IINT = J
      IF ((XINT(L1,L2,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L3,1) .GE. XINT(L1,L2,J+1)-0.001)
     &.AND. (XINT(L1,L3,1) .LE. XINT(L1,L2,J+1)+0.001)
     &.AND. (XINT(L1,L2,J+2) .LE. 0.001))) GO TO 2160
      GO TO 2080
 2060 LASTK = NCL-(IINT-1)
      DO 2070 K = 1,LASTK
      B(K) = B(K+IINT)
      B(K+IINT) = 0
 2070 CONTINUE
      GO TO 2090
 2080 CONTINUE
 2090 NCL = NCL-IINT
C
C     DO 2150 IS THE OUTER LOOP FOR THE PROCESS OF CREATING THE
C     EQUIVALENT ARC.  NCL IS THE NUMBER OF CLASSES INVOLVED
C     BETWEEN THE TWO ARCS.
C
      N1 = 0
      N2 = 0
      DO 2150 I = 1,NCL
      DO 2110 J = 1,11
C
C     DO 2110 DETERMINES THE APPROPRIATE INTERVALS OF EACH DISTRIBUTION
C     THAT ARE VALID FOR THE B(I) VALUE BEING CONSIDERED.  N1 AND
C     N2 ARE THE CONTROLS FOR UPPER AND LOWER ARCS RESPECTIVELY.
C
      IF (N1 .GE. 1) GO TO 2100
      IF (((B(I) .GE. XINT(L1,L2,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L2,J+1)+0.001)) .OR. (XINT(L1,L2,J+1) .LE. 0.001))
     &N1 = J
 2100 CONTINUE
      IF (N2 .GE. 1) GO TO 2110
      IF (((B(I) .GE. XINT(L1,L3,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L3,J+1)+0.001)) .OR. (XINT(L1,L3,J+1) .LE. 0.001))
     &N2 = J
 2110 CONTINUE
      IF (N2 .GT. NV2) K4(2,N2) = 1
      IF (N1 .GT. NV1) K4(1,N1) = 1
C
C     DO 2130 AND DO 2120 PERFORM THE POLYGONAL MULTIPLICATION FOR
C     CAP F(X) AND CAP G(X).
C
      LASTJ = K4(2,N2)
      LASTK = K4(1,N1)
      DO 2130 J = 1,LASTJ
      DO 2120 K = 1,LASTK
      IF (N2 .GT. NV2) PAR(2,N2,J) =  1
      IF (N1 .GT. NV1) PAR(1,N1,K) =  1
      K3 = J+K-1
      ZVAL(I,K3) = ZVAL(I,K3)+(PAR(1,N1,K)*PAR(2,N2,J))
 2120 CONTINUE
 2130 CONTINUE
C
C     DO 2140 OBTAINS THE FIRST DERIVATIVE OF THE RESULT OF THE
```

```
C     MULTIPLICATION OF CAP F(X) AND CAP G(X) IN THE FORM OF A
C     LITTLE H(X) FOR THAT PRODUCT.
C
      DO 2140 J = 1,4
      ZVAL(I,J) = ZVAL(I,J+1)*FLOAT(J)
      ZVAL(I,J+1) = 0
 2140 CONTINUE
      N1 = 0
      N2 = 0
 2150 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE RESULTS OF THE
C     PARALLEL REDUCTION WITH THE B(O) AND B(1) FORM IN EACH OF 10
C     CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR(L1,L2,NCL)
      GO TO 2180
 2160 DO 2170 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L3,I,1)
      VALUE(L1,L2,I,2) = VALUE(L1,L3,I,2)
      XINT(L1,L2,I) = XINT(L1,L3,I)
 2170 CONTINUE
      XINT(L1,L2,11) = XINT(L1,L3,11)
 2180 VALUE(L1,L2,1,3) = 0
      DO 2210 I = 1,2
      DO 2200 J = 1,10
      DO 2190 K = 1,3
      PAR(I,J,K) = 0
 2190 CONTINUE
 2200 CONTINUE
 2210 CONTINUE
      RETURN
      END
C     END SUBROUTINE PARA
C
C     **********************************************************
C
C                 S U B R O U T I N E       S E R I E S
C
C     ***********************************************************
C
      SUBROUTINE SERIES (L1,L2,L3,L4)
      REAL*8 VALUE(200,99,10,3),XINT(200,99,12)
      REAL*8 ZVAL(130,5),XLIM(2),A(130)
      REAL*8 F0,F1,G0,G1,XL
      INTEGER L1,L2,L3,L4
      INTEGER ISEL(2)
      INTEGER I,IK,J,K,NCL,NCL1,NE
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
C
C     SUBROUTINE SERIES PERFORMS THE CONVOLUTION OF TWO PROBABILITY
C     DISTRIBUTIONS BY INTEGRATING THE PRODUCT OF THEIR PIECEWISE
C     POLYGONAL APPROXIMATIONS IN THE FORMS OF F(X) AND G(T-X).
```

```
C
C     THIS SECTION DETERMINES THE INTERVALS OF VALIDITY FOR THE
C     CONVOLUTION.
C
C     THE A ARRAY IS USED FOR THE SAME PURPOSE AS THE B ARRAY IN PARA.
C
      K = 0
C
C     DO 3010 CREATES ALL POSSIBLE INTERVALS OF THE NEW DISTRIBUTION
C     BY ADDING THE INTERVALS OF THE TWO DISTRIBUTIONS BEING WORKED.
C
      DO 3010 I = 1,12
      IF ((XINT(L3,L4,I) .LE. 0).AND.(I .GT. 1)) GO TO 3020
      DO 3000 J = 1,12
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) NCL1 = J-2
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) GO TO 3010
      K = K+1
      A(K) = XINT(L1,L2,J)+XINT(L3,L4,I)
 3000 CONTINUE
 3010 CONTINUE
 3020 NINT = I-2
      NCL = K-1
C
C     DO 3120 IS CONTROLLED BY THE NUMBER OF CLASSES IN THE F(X)
C     DISTRIBUTION.  DO 3110 IS CONTROLLED BY THE NUMBER OF CLASSES
C     CREATED BY COMBINING F(X) AND G(T-X).  DO 3100 IS CONTROLLED
C     BY THE NUMBER OF CLASSES IN THE G(T-X) DISTRIBUTION.  THIS
C     ALLOWS THE EVALUATION OF ALL OF THE CREATED CLASSES FOR EVERY
C     CLASS IN BOTH DISTRIBUTIONS.
C
      CALL SORT (NCL)
      DO 3120 K = 1,NCL1
      DO 3110 I = 1,NCL
      DO 3100 J = 1,NINT
      IK = 0
C
C     THIS IF STATEMENT DETERMINES WHICH INTERVALS ARE VALID FOR THE
C     INTERVAL END POINT A(I) BEING EVALUATED AND FOR THE VALUE OF K
C     BEING CONTROLLED BY DO 3120.
C
      IF ((A(I) .GE. XINT(L1,L2,K)+XINT(L3,L4,J)-0.001) .AND. (A(I+1)
     &.LE. XINT(L1,L2,K+1)+XINT(L3,L4,J+1)+0.001)) IK = J
      IF (IK .GE. 1) GO TO 3030
      GO TO 3100
 3030 ISEL(1) = 0
      ISEL(2) = 0
C
C     THE IF STATEMENTS INVOLVING XLIM ARE USED TO DETERMINE THE
C     UPPER AND LOWER LIMITS OF INTEGRATION.  IT IS DETERMINED WHETHER
C     THE LIMIT COMES FROM THE F(X) OR THE G(T-X) DISTRIBUTION.  ISEL
C     IS USED TO DESIGNATE VALUES FROM THE G(T-X) DISTRIBUTION.
C
      IF (XINT(L1,L2,K) .GE. (A(I+1)-XINT(L3,L4,J+1)-0.001)) GO TO 3040
      XLIM(1) = XINT(L3,L4,J+1)
      ISEL(1) = 999
      GO TO 3050
 3040 XLIM(1) = XINT(L1,L2,K)
 3050 IF (XINT(L1,L2,K+1) .LE. (A(I)-XINT(L3,L4,J)+0.001)) GO TO 3060
      XLIM(2) = XINT(L3,L4,J)
      ISEL(2) = 999
      GO TO 3070
 3060 XLIM(2) = XINT(L1,L2,K+1)
 3070 CONTINUE
      DO 3090 NE = 1,2
      F0 = VALUE(L1,L2,K,1)
      F1 = VALUE(L1,L2,K,2)
      G0 = VALUE(L3,L4,IK,1)
      G1 = VALUE(L3,L4,IK,2)
      XL = XLIM(NE)
      Z = 1.0
      IF (NE .EQ. 1) Z = -1.0
      IF (ISEL(NE) .EQ. 999) GO TO 3080
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL AT A FINITE
C     LIMIT.  THE INTEGRATION IS BROKEN DOWN INTO ITS COMPONENT PARTS
C     BY THE POWER OF THE COEFFICIENT THAT RESULTS.  Z CONTROLS THE
C     SIGN OF THE INTEGRAL BASED ON WHETHER THE LOWER OR UPPER LIMIT
C     IS BEING EVALUATED.
C
      ZVAL(I,1) = ZVAL(I,1)+((F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((-1.0*F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+(((F1*G1*XL**2)/2.)+(F0*G1*XL))*Z
      ZVAL(I,3) = ZVAL(I,3)+((-1.0*F0*G1)/2.)*Z
      GO TO 3090
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL FOR LIMITS.
C     IN THE FORM OF (T-X).  THE FORMULAS ARE DIFFERENT BECAUSE
C     OF THE DIFFERENT POLYNOMIAL CREATED WHEN THE INTEGRATION
C     INVOLVES LIMITS IN THE FORM OF (T-X).
C
 3080 ZVAL(I,1) = ZVAL(I,1)+((-1.0*F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+((-1.0*F1*G0*XL)+(F0*G0)-
     &((F1*G1*XL**2)/2.))*Z
      ZVAL(I,3) = ZVAL(I,3)+((F1*G0)/2.)*Z
      ZVAL(I,4) = ZVAL(I,4)+((F1*G1)/6.)*Z
 3090 CONTINUE
 3100 CONTINUE
 3110 CONTINUE
 3120 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE CONVOLUTION
C     RESULTS WITH THE B(0) AND B(1) FORM IN EACH OF 10 CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR (L1,L2,NCL)
      RETURN
      END
C     END SUBROUTINE SERIES
C
C     ***************************************************************
C
```

```
C                 S U B R O U T I N E     P L O T
C
C     ************************************************************
C
      SUBROUTINE PLOT (IPRINT,KBL,KBM,IFLAG)
      REAL*8 XX(100,2),SORT
      CHARACTER*1 KBL,KBM,LINE(101)
      INTEGER I,IFLAG,IPRINT,J,JPLOT,K,NN
      COMMON/PARA4/XX
C
C     PLOT IS USED TO CREATE THE HISTOGRAM FOR FINAL OUTPUT.
C     THE VARIABLE SORT IN THIS SUBROUTINE IS NOT RELATED TO
C     THE SUBROUTINE SORT.
C
      SORT = XX(1,2)
      DO 4000 I = 2,IPRINT
      IF (SORT .LE. XX(I,2)) SORT = XX(I,2)
 4000 CONTINUE
      PRINT 4900
      IF (IFLAG .EQ. 0) THEN
      PRINT 4910
      ELSE
      PRINT 4915
      END IF
      IF (SORT .GT. 0.5) PRINT 4920
      IF ((SORT .GT. 0.25).AND.(SORT .LE. 0.50)) PRINT 4930
      IF ((SORT .GT. 0.10).AND.(SORT .LE. 0.25)) PRINT 4940
      IF ((SORT .GT. 0.05).AND.(SORT .LE. 0.10)) PRINT 4950
      IF (SORT .LE. 0.05) PRINT 4960
      PRINT 4970
      DO 4030 I = 1,IPRINT
      DO 4010 J = 1,51
      LINE(J) = KBL
 4010 CONTINUE
      IF (SORT .GT. 0.5) JPLOT = (INT((XX(I,2)*50.)+0.5))+1
      IF ((SORT .GT. 0.25).AND.(SORT .LE. 0.50))
     &JPLOT = (INT((XX(I,2)*100.)+0.5))+1
      IF ((SORT .GT. 0.10).AND.(SORT .LE. 0.25))
     &JPLOT = (INT((XX(I,2)*200.)+0.5))+1
      IF ((SORT .GT. 0.05).AND.(SORT .LE. 0.10))
     &JPLOT = (INT((XX(I,2)*500.)+0.5))+1
      IF (SORT .LE. 0.05) JPLOT = (INT((XX(I,2)*1000.0)+0.5))+1
      IF (JPLOT .LE. 0) JPLOT = 1
      IF (JPLOT .GT. 51) JPLOT = 51
      DO 4020 NN = 1,JPLOT
      LINE(NN) = KBM
 4020 CONTINUE
      PRINT 4980,XX(I,1),(LINE(K), K = 1,JPLOT)
 4030 CONTINUE
C
C     FORMAT STATEMENTS
C
 4900 FORMAT (1H1)
 4910 FORMAT (15X,'PROBABILITY DENSITY FUNCTION' //)
 4915 FORMAT (15X,'SIMULATION FREQUENCY HISTOGRAM' //)
 4920 FORMAT (9X,'0        .20       .40       .60       .80       1.0')
```

```
4930 FORMAT (9X,'0        .10       .20       .30       .40       .50')
4940 FORMAT (9X,'0        .05       .10       .15       .20       .25')
4950 FORMAT (9X,'0        .02       .04       .06       .08       .10')
4960 FORMAT (9X,'0        .01       .02       .03       .04       .05')
4970 FORMAT (9X,'I----+----I----+----I----+----I----+----I----+----I')
4980 FORMAT (1X,F8.3,2X,51A1)
      RETURN
      END
C     END SUBROUTINE PLOT
C
C     ************************************************************
C
C                  S U B R O U T I N E         L I N E A R
C
C     ************************************************************
C
      SUBROUTINE LINEAR (L1,L2,NCL)
      REAL*8 VALUE(200,99,10,3),XINT(200,99,12),ZVAL(130,5),A(130)
      REAL*8 Q,Q1,Q2,STD,SUMX,SUMY,SUMXY,SUMSQ
      REAL*8 ALPHA,AREA,BETA,FACT,SIZE,W,X,XLMBDA,XMEAN
      REAL*8 XMODE,XSIZE,Y
      INTEGER L1,L2
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      EXTERNAL DGAMMA
C
C     SUBROUTINE LINEAR PIECEWISE POLYGONALIZES DISTRIBUTION DATA
C     FROM THE MAIN PROGRAM AND SUBROUTINES PARA AND SERIES WITH
C     THE B(O) AND B(1) FORM IN EACH OF 10 CLASSES THROUGH THE USE
C     OF SIMPLE LINEAR REGRESSION.
C
      XMODE = VALUE(L1,L2,2,3)
      XMEAN = VALUE(L1,L2,2,3)
      STD = ((VALUE(L1,L2,2,3)-XINT(L1,L2,1))/3.)
      XLMBDA = VALUE(L1,L2,2,3)-XINT(L1,L2,1)
      ALPHA = VALUE(L1,L2,2,3)
      BETA = VALUE(L1,L2,3,3)
      SIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) SIZE = (A(NCL+1)-A(1))/10.
      XINT(L1,L2,11) = XINT(L1,L2,2)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) XINT(L1,L2,11) = A(NCL+1)
      X = XINT(L1,L2,1)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) X = A(1)
      DO 5000 I = 1,10
      XINT(L1,L2,I) = X
      X = X+SIZE
 5000 CONTINUE
      DO 5050 I = 1,10
      X = XINT(L1,L2,I)
      SUMY = 0.
      SUMX = 0.
      SUMXY = 0.
      SUMSQ = 0.
C
C     W CONTROLS THE NUMBER OF DATA POINTS USED IN THE REGRESSION
```

```
C     COMPUTATIONS.
C
      W = 10.+IDINT(SIZE*3.)
      XSIZE = SIZE/W
      LASTJ = IDINT(W)
      DO 5040 J = 1,LASTJ
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 99)  GO TO 5030
      DO 5010 K3 = 1,NCL
      K  = 0
      IF ((X .GE. A(K3)).AND.(X .LE. A(K3+1))) K = K3
      IF (K .GE. 1) GO TO 5020
 5010 CONTINUE
C
C     SERIES OR PARA GENERATED DISTRIBUTIONS.
C
 5020 Y = ZVAL(K,1)+(ZVAL(K,2)*X)+(ZVAL(K,3)*(X**2))
     &+(ZVAL(K,4)*(X**3))
 5030 CONTINUE
C
C     TRIANGULAR DISTRIBUTION.
C
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 1) THEN
         IF (XINT(L1,L2,1) .LE. X .AND. X .LE. XMODE) THEN
         Y = (2.*(X-XINT(L1,L2,1)))/((XMODE-XINT(L1,L2,1))*10.*SIZE)
         ELSE
         Y = (2.*(XINT(L1,L2,11)-X))/((XINT(L1,L2,11)-XMODE)*10.*SIZE)
         END IF
C
C     NORMAL  DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 2) THEN
      Y = (1./(STD*2.506628275))*(DEXP((-1.0)*(((X-XMEAN)/STD)**2)/2.))
C
C     EXPONENTIAL DISTRIBUTION  (SHIFTED).
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 3) THEN
      Y = (1./XLMBDA)*(DEXP((-1.0)*((X-XINT(L1,L2,1))/XLMBDA)))
C
C     GAMMA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 4) THEN
      Y = (1./(DGAMMA(ALPHA)*(BETA**ALPHA)))*DEXP(-X/BETA)*(X**(ALPHA-1.
     &))
C
C     BETA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 5) THEN
      Y = (DGAMMA(ALPHA+BETA)/(DGAMMA(ALPHA)*DGAMMA(BETA)))*
     &(1./(10.*SIZE)**(ALPHA+BETA-2.))*
     &((X-XINT(L1,L2,1))**(ALPHA-1.))*
     &((XINT(L1,L2,11)-X)**(BETA-1.))
      END IF
      IF (Y .LT. 0) Y = 0
      SUMX = SUMX+X
      SUMY = SUMY+Y
      SUMXY = SUMXY+(X*Y)
      SUMSQ = SUMSQ+(X**2)
      X = X+XSIZE
 5040 CONTINUE
      VALUE(L1,L2,I,2) = (SUMXY-((SUMX*SUMY)/W))/(SUMSQ-((SUMX**2)/W))
      VALUE(L1,L2,I,1) = (SUMY/W)-(VALUE(L1,L2,I,2)*(SUMX/W))
 5050 CONTINUE
C
C     DO 5060 CALCULATES THE AREA UNDER THE APPROXIMATED DISTRIBUTION.
C     AN ADJUSTMENT FACTOR FOR THE AMOUNT THAT THIS AREA HAS BEEN
C     UNDERESTIMATED OR OVERESTIMATED IS THEREBY DETERMINED.
C
      DO 5060 I = 1,10
      Q  = XINT(L1,L2,I+1)-XINT(L1,L2,I)
      Q1 = (XINT(L1,L2,I)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      Q2 = (XINT(L1,L2,I+1)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      IF (Q1 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q1*(-1.0))
      IF (Q2 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q2*(-1.0))
      IF (Q1 .LT. 0.) Q1 = 0.
      IF (Q2 .LT. 0.) Q2 = 0.
      AREA = AREA+((Q1+Q2)*Q*0.5)
 5060 CONTINUE
      FACT = 1.0/AREA
C
C     DO 5070 ADJUSTS THE COEFFICIENTS OF ALL THE LINEAR POLYNOMIAL
C     PIECES BY THE FACTOR COMPUTED IN DO 5060 IN ORDER TO NORMALIZE
C     THE AREA BACK TO ONE.  THIS ACTS TO REDUCE ACCUMULATING ERRORS
C     DURING PROGRAM COMPUTATIONS.
C
      DO 5070 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)*FACT
      VALUE(L1,L2,I,2) = VALUE(L1,L2,I,2)*FACT
 5070 CONTINUE
      AREA = 0
      DO 5080  I = 1,130
      A(I)  = 0
      ZVAL(I,1) = 0
      ZVAL(I,2) = 0
      ZVAL(I,3) = 0
      ZVAL(I,4) = 0
 5080 CONTINUE
      RETURN
      END
C     END SUBROUTINE LINEAR
C
C     ************************************************************
C
C                  S U B R O U T I N E      S O R T
C
C     ************************************************************
C
      SUBROUTINE SORT (NCL)
      REAL*8 A(130),B
      INTEGER NCL
      INTEGER I,K1
      COMMON/PARA3/A
C
```

```
C     SUBROUTINE SORT IS USED TO CONDUCT AN ALGEBRAIC SORT OF DATA
C     CREATED IN THE SERIES AND PARA SUBROUTINES.
C
 6000 K1  = 0
      DO 6010 I = 1,NCL
      IF ((A(I) .LT. (A(I+1)+.01)).AND.(A(I) .GT. (A(I+1)-.01)))
     &GO TO 6020
      IF (A(I) .LT. A(I+1)) GO TO 6010
      IF (A(I) .GT. A(I+1)) B = A(I)
      A(I) = A(I+1)
      A(I+1) = B
      K1=K1+1
 6010 CONTINUE
      IF (K1 .GE. 1) GO TO 6000
      GO TO 6040
 6020 NCL = NCL-1
      LASTJ = NCL+1
      DO 6030 J = I,LASTJ
      A(J) = A(J+1)
      A(J+1) = 0
 6030 CONTINUE
      GO TO 6000
 6040 RETURN
      END
C     END SUBROUTINE SORT
C
C     ***********************************************************
C
C               S U B R O U T I N E       S I M U L T
C
C     ***********************************************************
C
      SUBROUTINE SIMULT (N,NSIM)
      REAL*8 XINT(200,99,12),VALUE(200,99,10,3)
      REAL*8 T(100,99),SIMT(100,10000)
      REAL*8 ALPHA,BETA,X,XLNGTH,XLMBDA,XMAX,XMEAN,XMIN,XMODE
      REAL*8 RN,STD,TTEMP,TMAX
      DIMENSION NET(200,103)
      INTEGER ISEED,ISIM,N,NDIST,NSIM
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA5/NET
      COMMON/PARA6/SIMT
      EXTERNAL DRNUN,DRNNOR,DRNEXP,DRNGAM,DRNBET,RNSET
C
C     DO 7130 GENERATES A SIMULATED NETWORK THROUGHPUT FOR EACH OF
C     NSIM ITERATIONS OF THE MONTE CARLO SIMULATION OF THE NETWORK.
C
      ISEED = 123456789
      CALL RNSET(ISEED)
      DO 7130 ISIM=1,NSIM
C
C     DO 7080 GENERATES A RANDOM VALUE FROM THE ACTIVITY RESOURCE
C     CONSUMPTION (ACTIVITY TIME) DISTRIBUTION OF EACH ACTIVITY.
C
      DO 7080 I=1,N-1
      DO 7070 J=1,NET(I,103)
      NDIST = IDINT(VALUE(I,J,1,3))
      XMIN = XINT(I,J,1)
      XMAX = XINT(I,J,11)
      XLNGTH = XMAX-XMIN
      GO TO (7010,7020,7030,7040,7050,7060) NDIST
C
C     TRIANGULAR DISTRIBUTION.
C
 7010 CALL DRNUN(1,RN)
      XMODE = VALUE(I,J,2,3)
      X = (XMODE-XMIN)/XLNGTH
      IF (RN .GT. X) GO TO 7015
      T(I,J) = XMIN+DSQRT(RN*XLNGTH*(XMODE-XMIN))
      GO TO 7070
 7015 T(I,J) = XMAX-DSQRT(XLNGTH*(XMAX-XMODE)*(1.-RN))
      GO TO 7070
C
C     NORMAL DISTRIBUTION.
C
 7020 CALL DRNNOR(1,RN)
      XMEAN = VALUE(I,J,2,3)
      STD = (XMEAN-XMIN)/3.
      T(I,J) = (RN*STD)+XMEAN
      IF ((T(I,J) .LT. XMIN) .OR. (T(I,J) .GT. XMAX)) GO TO 7020
      GO TO 7070
C
C     EXPONENTIAL DISTRIBUTION.
C
 7030 CALL DRNEXP(1,RN)
      XLMBDA = VALUE(I,J,2,3)-XMIN
      T(I,J) = (XLMBDA*RN)+XMIN
      IF (T(I,J) .GT. XMAX) GO TO 7030
      GO TO 7070
C
C     GAMMA DISTRIBUTION.
C
 7040 ALPHA = VALUE(I,J,2,3)
      BETA = VALUE(I,J,3,3)
      CALL DRNGAM(1,ALPHA,RN)
      T(I,J) = BETA*RN
      IF (T(I,J) .GT. XMAX) GO TO 7040
      GO TO 7070
C
C     BETA DISTRIBUTION.
C
 7050 ALPHA = VALUE(I,J,2,3)
      BETA = VALUE(I,J,3,3)
      CALL DRNBET(1,ALPHA,BETA,RN)
      T(I,J) = XMIN+(XLNGTH*RN)
      GO TO 7070
C
C     UNIFORM DISTRIBUTION.
C
 7060 CALL DRNUN(1,RN)
      T(I,J) = XMIN+(XLNGTH*RN)
 7070 CONTINUE
 7080 CONTINUE
C
C     DO 7120 GENERATES THE CRITICAL PATH TO EACH NODE.  THE
C     SIMULATED TIME THROUGH NODE I FROM SIMULATION ITERATION L
C     IS STORED IN SIMT(I,L).
C
      SIMT(1,ISIM) = 0.0
      DO 7120 I=2,N
      TMAX = 0.0
C
C     DO 7110 DETERMINES THE STARTING NODES AND THE ACTIVITIES WHICH
C     TERMINATE AT NODE I > STARTING NODES, AND COMPUTES THE SIMULATED
C     THROUGHPUT VALUE THROUGH NODE I AS THE MAXIMUM OF THE
C         [(THROUGHPUT VALUE THROUGH STARTING NODE) +
C          (ACTIVITY VALUE FROM STARTING NODE TO NODE I)].
C
      DO 7110 J=1,I-1
      DO 7100 J1=2,NET(J,103)+1
      IF (NET(J,J1)-I) 7100,7090,7100
 7090 TTEMP = SIMT(J,ISIM)+T(J,J1-1)
      IF (TTEMP .LT. TMAX) GO TO 7100
      TMAX = TTEMP
 7100 CONTINUE
 7110 CONTINUE
      SIMT(I,ISIM) = TMAX
 7120 CONTINUE
 7130 CONTINUE
      RETURN
      END
C     END SUBROUTINE SIMULT
```

APPENDIX B

PART PROGRAM WITH SEQUENTIAL APPROXIMATION

```
C
C
C             POLYGONAL APPROXIMATION AND REDUCTION TECHNIQUE
C                                (PART)
C                              ALGORITHM
C                                 FOR
C                     ACYCLIC, DIRECTED NETWORKS
C                                USING
C                  "SEQUENTIAL APPROXIMATION" METHOD
C
C
C     THIS PROGRAM IS WRITTEN IN FORTRAN 77 AND IS PRESENTLY DESIGNED
C     TO BE OPERATED IN A TIME SHARING MODE WITH ALL DATA INPUT FROM
C     THREE (3) DATA FILES.  THE PROGRAM DIRECTS OUTPUT IN EIGHT (8)
C     OPTIONAL FORMATS TO A TIME SHARING TERMINAL.  IF DESIRED, THE
C     READ STATEMENTS AT THE BEGINNING OF THE MAIN PROGRAM CAN BE
C     MODIFIED TO ALLOW DATA INPUT DIRECTLY FROM THE TIME SHARING
C     TERMINAL.
C
C     THE CURRENT DIMENSIONS OF THE PROGRAM ALLOW A NETWORK WITH A
C     MAXIMUM OF 100 NODES AND A MAXIMUM OF 99 ACTIVITIES BEGINNING
C     AT EACH NODE.  THESE LIMITS CAN BE EXPANDED BY CHANGING THE
C     DIMENSIONS OF THE XINT AND VALUE ARRAYS.
C
C
C     ************************************************************
C
C                O P E R A T I N G   I N S T R U C T I O N S
C
C     ************************************************************
C
C     INSTRUCTIONS FOR BUILDING DATA FILES
C     ----------------------------------
C
C           DATA FILE DATAN.DAT
C
C     THIS DATA FILE CONTAINS A DESCRIPTION OF THE NETWORK STRUCTURE.
C     EACH NODE REQUIRES 4 RECORDS WITH A TOTAL OF 103 FIELDS:
C         FIELD 1 IS THE BEGINNING NODE.
C         FIELDS 2 THRU 100 ARE THE NUMBERS OF THE NODES AT WHICH THE
C            ACTIVITIES WHICH BEGIN AT THE NODE IN FIELD 1 TERMINATE.
C         FIELD 101 INDICATES WHETHER OR NOT THE NODE IS ON THE OUTPUT
C            CRITICAL LIST.
C                1 = NODE IS ON THE OUTPUT CRITICAL LIST
C                0 = NODE IS NOT ON THE OUTPUT CRITICAL LIST
C         FIELD 102 INDICATES HOW MANY ACTIVITIES TERMINATE AT THE NODE
C            INDICATED IN FIELD NUMBER 1.
C         FIELD 103 INDICATES HOW MANY ACTIVITIES BEGIN AT THE NODE
INDI-
C            CATED IN FIELD NUMBER 1.
C
C
C           DATA FILE DATAH.DAT
C
C     THIS DATA FILE CONTAINS A DESCRIPTION OF THE DISTRIBUTIONS OF
C     EACH OF THE ACTIVITIES IN THE NETWORK.
```

```
C
C     THERE ARE 7 FIELDS OF DATA.
C     FIELD 1 IS THE NODE NUMBER.
C     FIELD 2 IS THE NUMBER OF THE ACTIVITY COMING FROM THE NODE.
C     FIELD 3 IS THE CODE FOR THE TYPE OF DISTRIBUTION.
C         1 = TRIANGULAR DISTRIBUTION
C         2 = NORMAL DISTRIBUTION
C         3 = EXPONENTIAL DISTRIBUTION
C         4 = GAMMA DISTRIBUTION
C         5 = BETA DISTRIBUTION
C         6 = UNIFORM DISTRIBUTION
C     FIELD 4 IS
C         MODE FOR A TRIANGULAR DISTRIBUTION.
C         MEAN FOR A NORMAL DISTRIBUTION.
C         MEAN FOR AN EXPONENTIAL DISTRIBUTION.
C         ALPHA FOR A GAMMA OR A BETA DISTRIBUTION.
C         1/(B-A) FOR A UNIFORM DISTRIBUTION.
C     FIELD 5 IS BETA FOR A GAMMA OR A BETA DISTRIBUTION.
C     FIELD 6 IS THE MINIMUM VALUE OF THE DISTRIBUTION.
C     FIELD 7 IS THE MAXIMUM VALUE OF THE DISTRIBUTION.
C
C
C           DATA FILE CONTROL.DAT
C
C     THIS IS A SINGLE LINE DATA FILE WHICH CONTAINS CONTROL
C     PARAMETERS FOR INPUT, OUTPUT, AND MONTE CARLO SIMULATION.
C
C     THERE ARE 4 FIELDS OF DATA.
C     FIELD 1 IS THE NUMBER OF NODES IN THE NETWORK.
C     FIELD 2 IS THE NUMBER OF ACTIVITIES IN THE NETWORK.
C     FIELD 3 IS THE OUTPUT OPTION DESIRED FOR THE PART RESULTS.
C         1 = A DESCRIPTION OF EACH OF THE 10 CLASSES OF THE
C             FINAL DISTRIBUTION IN THE FORM OF Y = B(0) + B(1) X.
C         2 = A CUMULATIVE DISTRIBUTION FUNCTION OF THE FINAL
C             DISTRIBUTION.
C         3 = A DISCRETE PROBABILITY DENSITY FUNCTION AND A
C             SIMULATION FREQUENCY HISTOGRAM IN GRAPHICAL FORMAT.
C         4 = A COMBINATION OF 1 AND 2 ABOVE.
C         5 = A COMBINATION OF 1 AND 3 ABOVE.
C         6 = A COMBINATION OF 2 AND 3 ABOVE.
C         7 = A COMBINATION OF 1, 2, AND 3 ABOVE.
C         8 = ONLY THE EXPECTED VALUE AND STANDARD DEVIATION.
C     FIELD 4 IS THE NUMBER OF ITERATIONS OF THE MONTE CARLO
C     SIMULATION REQUESTED (MAXIMUM = 10,000).
C         0 = NO MONTE CARLO SIMULATION IS REQUESTED.
C
C
C     NOTE
C
C     ALL UNUSED FIELDS MUST BE ZEROED OUT.
C
C
C
C     **********************************************************
C
C                   M A I N      P R O G R A M
```

```
C
C     ************************************************************
C
      REAL*8 XINT(101,100,12),VALUE(101,100,10,3),A(130)
      REAL*8 ZVAL(130,5),XX(100,2),TOTAAR(51)
      REAL*8 SIMT(100,10000),SIMTOT(51)
      REAL*8 AREA,AVG,COUNT,SIG,SIZE
     *                ,XD1,XD2,XD3,XD4,X,XSIZE
     *                ,DIFF
      REAL*4 HIGH,HLOW,PERCNT,KSCR20,KSCR10,KSCR05,KSCR02,KSCR01,DMAX
      INTEGER I,IEDN,ICOUNT,IACT,IFLAG,IOCL,IPRE,IPRINT,ISNODE
     *       ,MM
     *       ,N,NACTS,NAN,NCL,NET,NETT,NSIM
     *       ,J,J1
     *       ,K,KK
     *       ,L,L1,L2,L3,LASTK,LASTMM,L3COUNT
      DIMENSION NET(100,103),NETT(103),IOCL(100),IPRE(99,2)
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      COMMON/PARA4/XX
      COMMON/PARA5/NET
      COMMON/PARA6/SIMT
      CHARACTER*1 KBL,KBM
      DATA KBL/' '/,KBM/'*'/
      DATA NCL/0/
C
C     OPEN INPUT AND OUTPUT FILES
C
      OPEN (UNIT = 11, FILE = 'datan.dat')
      OPEN (UNIT = 12, FILE = 'datah.dat')
      OPEN (UNIT = 13, FILE = 'control.dat')
C
C     READ INFORMATION INTO DATA MATRICES.
C
      READ (13,1900) N,NACTS,NAN,NSIM
      DO 0910 I = 1,N
      READ (11,1901) (NETT(J), J = 1,103)
      L1 = NETT(1)
      DO 0900 K = 1,103
      NET(L1,K) = NETT(K)
 0900 CONTINUE
 0910 CONTINUE
C
C     DO 0920 LOADS THE OUTPUT CRITICAL LIST ARRAY.
C
      DO 0920 I =1,N
      IOCL(I) = NET(I,101)
      NET(I,101) = 1
 0920 CONTINUE
C
C     DO 1010 READS DATA FROM DATAH AND LOADS THIS DATA INTO
C     THE VALUE AND XINT ARRAYS.  THIS DO ALSO DETERMINES IF
C     THE ACTIVITY DISTRIBUTION IS OTHER THAN UNIFORM, AND,
C     IF SO, CALLS LINEAR TO APPROXIMATE IT WITH A
C     PIECEWISE POLYGONAL FUNCTION.
```

```
C
      DO 1010 I = 1,NACTS
      READ (12,1902) L1,L2,XD1,XD2,XD3,XD4,XD5
      VALUE(L1,L2,1,3) = XD1
      VALUE(L1,L2,2,3) = XD2
      VALUE(L1,L2,3,3) = XD3
      XINT(L1,L2,1) = XD4
      XINT(L1,L2,2) = XD5
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 6) THEN
      CALL LINEAR (L1,L2,NCL)
      GO TO 1010
C
C     DO 1000 CONVERTS DATA FOR UNIFORM DISTRIBUTIONS INTO A USABLE
C     FORM FOR SUBROUTINES SERIES AND PARA.
C
      ELSE
      XINT(L1,L2,11) = XINT(L1,L2,2)
      X = XINT(L1,L2,1)
      XSIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      DO 1000 J = 1,10
      VALUE(L1,L2,J,1) = VALUE(L1,L2,2,3)
      XINT(L1,L2,J) = X
      X = X+XSIZE
 1000 CONTINUE
      END IF
 1010 CONTINUE
C
C     MONTE CARLO SIMULATION OF THE NETWORK.
C
      IF (NSIM .EQ. 0) GO TO 1030
      CALL SIMULT (N,NSIM)
C
C     REDUCTION OF THE NETWORK BEGINS.
C
C     DO 1040 CHECKS IF A CONVOLUTION (SERIES-REDUCTION) OPERATION
C     IS POSSIBLE, i.e., IF THERE EXISTS A NODE I NOT ON THE OUTPUT
C     CRITICAL LIST SUCH THAT
C         IN-DEGREE NODE I = OUT-DEGREE NODE I = 1.
C
 1030 L3COUNT = 2
 1035 DO 1040 I=L3COUNT,N-1
      L3 = I
      IF ((NET(I,102)+NET(I,103)) .EQ. 2 .AND. IOCL(I) .EQ. 0)
     &GO TO 1050
 1040 CONTINUE
C
C     IF (IN-DEGREE NODE I + OUT-DEGREE NODE I) > 2 FOR ALL I NOT = 1
C     OR N, NETWORK IS NONSEPARABLE, SO PROCEED TO "SEQUENTIAL
C     APPROXIMATION"
C
      IF (L3COUNT .EQ. 2) GO TO 1145
      GO TO 1080
C
C     A CONVOLUTION IS POSSIBLE WITH THE TWO ACTIVITIES, ONE OF WHICH
C     TERMINATES AT NODE L3 AND THE OTHER OF WHICH STARTS AT NODE L3.
C     DO 1060 IDENTIFIES THE STARTING NODE NUMBER AND THE ACTIVITY
```

```
C     NUMBER OF THE ACTIVITY TERMINATING AT NODE L3.  THEN THE SERIES
C     SUBNETWORK CONSISTING OF THESE TWO ACTIVITIES IS CONVOLUTED INTO
C     AN EQUIVALENT ACTIVITY.
C
 1050 DO 1060 I=1,L3-1
      DO 1060 J=2,NET(I,103)+1
      L1 = I
      L2 = J-1
      IF (NET(I,J) .EQ. L3) GO TO 1070
 1060 CONTINUE
 1070 CALL SERIES(L1,L2,L3,1)
      NET(L1,L2+1) = NET(L3,2)
      NET(L3,2) = 0
      NET(L3,101) = 0
      NET(L3,102) = 0
      NET(L3,103) = 0
      L3COUNT = L3+1
      IF (L3COUNT .EQ. N) GO TO 1080
      GO TO 1035
C
C     DO 1140 CHECKS IF A MAXIMUM (PARALLEL-REDUCTION) OPERATION IS
C     POSSIBLE, i.e., IF THERE EXIST TWO DIFFERENT ACTIVITIES, A1 AND
C     A2, SUCH THAT
C         STARTING NODE (A1) = STARTING NODE (A2), AND
C           ENDING NODE (A1) = ENDING NODE (A2).
C     THEN THE PARALLEL SUBNETWORK CONSISTING OF THESE TWO ACTIVITIES
C     IS PARALLEL-REDUCED WITH A MAXIMUM OPERATION INTO AN EQUIVALENT
C     ACTIVITY.
C
 1080 DO 1140 I=1,N-1
      L1 = I
 1085 DO 1090 J=2,NET(L1,103)
      L2 = J-1
      IF (NET(L1,J) .EQ. 0) GO TO 1140
      DO 1090 K=J+1,NET(L1,103)+1
      L3 = K-1
      IF (NET(L1,J) .EQ. NET(L1,K)) THEN
      IEDN = NET(L1,J)
      GO TO 1110
      ELSE
      GO TO 1090
      END IF
 1090 CONTINUE
      GO TO 1140
 1110 CALL PARA(L1,L2,L3)
      NET(L1,103) = NET(L1,103)-1
      NET(IEDN,102) = NET(IEDN,102)-1
      DO 1120 K=L3,NET(L1,103)
      NET(L1,K+1) = NET(L1,K+2)
      DO 1115 L=1,10
      XINT(L1,K,L)= XINT(L1,K+1,L)
      VALUE(L1,K,L,1) = VALUE(L1,K+1,L,1)
      VALUE(L1,K,L,2) = VALUE(L1,K+1,L,2)
 1115 CONTINUE
      XINT(L1,K,11) = XINT(L1,K+1,11)
 1120 CONTINUE
```

```
      K = NET(L1,103)+1
      NET(L1,K+1) = 0
      DO 1130 L=1,10
      XINT(L1,K,L) = 0.
      VALUE(L1,K,L,1) = 0.
      VALUE(L1,K,L,2) = 0.
 1130 CONTINUE
      XINT(L1,K,11) = 0.
      GO TO 1085
 1140 CONTINUE
      GO TO 1030
C
C     THROUGH 1146 CHECKS IF THE NETWORK HAS BEEN SERIES-PARALLEL
C     REDUCED TO A SINGLE EQUIVALENT ACTIVITY.
C
 1145 IF (NET(1,103) .NE. 1) GO TO 1150
      IF (NET(N,102) .NE. 1) GO TO 1150
      DO 1146 I=2,N-1
      IF (NET(I,102) + NET(I,103)) 1550,1146,1150
 1146 CONTINUE
C
C     THE NETWORK HAS BEEN SERIES-PARALLEL REDUCED TO A SINGLE EQUIVA-
C     LENT PATH.  DO 1147 LOADS THE DISTRIBUTION OF THIS PATH INTO THE
C     100TH ACTIVITY POSITION OF NODE N.
C
      DO 1147 J=1,10
      XINT(N,100,J) = XINT(1,1,J)
      VALUE(N,100,J,1) = VALUE(1,1,J,1)
      VALUE(N,100,J,2) = VALUE(1,1,J,2)
 1147 CONTINUE
      XINT(N,100,11) = XINT(1,1,11)
      GO TO 1240
C
C     DO 1220 REDUCES THE NONSEPARABLE NETWORK USING THE "SEQUENTIAL
C     APPROXIMATION" METHOD.
C
 1150 DO 1220 I=2,N
      ICOUNT = 0
      IF (NET(I,101)) 1550,1220,1155
C
C     DO 1170 DETERMINES THE STARTING NODE NUMBER AND THE ACTIVITY
C     NUMBER OF ALL THE ACTIVITIES WHICH TERMINATE AT NODE I > STARTING
C     NODE.
C
 1155 DO 1170 J=1,I-1
      IF (NET(J,101)) 1550,1170,1160
 1160 DO 1169 J1=2,NET(J,103)+1
      IF (NET(J,J1) - I) 1169,1165,1169
 1165 ICOUNT = ICOUNT+1
      IPRE(ICOUNT,1) = J
      IPRE(ICOUNT,2) = J1-1
 1169 CONTINUE
 1170 CONTINUE
C
C     THROUGH 1220 CONVOLVES THE RESOURCE CONSUMPTION DISTRIBUTION
C     THROUGH THE STARTING NODE OF THE ACTIVITY AND THE RESOURCE
```

```
C     CONSUMPTION DISTRIBUTION OF EACH ACTIVITY WHICH TERMINATES
C     AT NODE I AND THEN FINDS THE MAXIMUM OF THESE CONVOLUTIONS.
C
C     IF THE FIRST STARTING NODE = NODE 1, THE CONVOLUTION IS EQUAL TO
C     THE DISTRIBUTION OF THE ACTIVITY WHICH TERMINATES AT NODE I.
C
      IF (IPRE(1,1) .EQ. 1) THEN
      IACT = IPRE(1,2)
      DO 1175 J=1,10
      XINT(101,1,J) = XINT(1,IACT,J)
      VALUE(101,1,J,1) = VALUE(1,IACT,J,1)
      VALUE(101,1,J,2) = VALUE(1,IACT,J,2)
 1175 CONTINUE
      XINT(101,1,11) = XINT(1,IACT,11)
C
C     OTHERWISE, LOAD THE DISTRIBUTION THROUGH THE FIRST STARTING NODE
C     INTO TEMPORARY LOCATION 1.
C
      ELSE
      ISNODE = IPRE(1,1)
      IACT = IPRE(1,2)
      DO 1180 J=1,10
      XINT(101,1,J) = XINT(ISNODE,100,J)
      VALUE(101,1,J,1) = VALUE(ISNODE,100,J,1)
      VALUE(101,1,J,2) = VALUE(ISNODE,100,J,2)
 1180 CONTINUE
      XINT(101,1,11) = XINT(ISNODE,100,11)
C
C     LOAD THE DISTRIBUTION OF THE FIRST ACTIVITY TERMINATING AT NODE I
C     IN TEMPORARY LOCATION 2.
C
      DO 1185 J=1,10
      XINT(101,2,J) = XINT(ISNODE,IACT,J)
      VALUE(101,2,J,1) = VALUE(ISNODE,IACT,J,1)
      VALUE(101,2,J,2) = VALUE(ISNODE,IACT,J,2)
 1185 CONTINUE
      XINT(101,2,11) = XINT(ISNODE,IACT,11)
C
C     CONVOLVE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2 AND
C     PLACE THE CONVOLUTION IN TEMPORARY LOCATION 1.
C
      CALL SERIES(101,1,101,2)
      END IF
C
C     IF THERE IS ONLY ONE ACTIVITY TERMINATING AT NODE I, THE
DISTRIBUTION
C     THROUGH NODE I IS THE CONVOLUTION IN TEMPORARY LOCATION 1.  LOAD
THIS
C     INTO THE 100TH ACTIVITY POSITION OF NODE I.
C
      IF (ICOUNT .EQ. 1) THEN
      DO 1190 J=1,10
      XINT(I,100,J) = XINT(101,1,J)
      VALUE(I,100,J,1) = VALUE(101,1,J,1)
      VALUE(I,100,J,2) = VALUE(101,1,J,2)
 1190 CONTINUE
```

```
      XINT(I,100,11) = XINT(101,1,11)
C
C     IF THERE ARE TWO OR MORE ACTIVITIES TERMINATING AT NODE I, LOAD
THE
C     DISTRIBUTION THROUGH THE STARTING NODE OF THE NEXT ACTIVITY INTO
C     TEMPORARY LOCATION 3.
C
      ELSE
      DO 1205 K=2,ICOUNT
      ISNODE = IPRE(K,1)
      IACT = IPRE(K,2)
      DO 1195 J=1,10
      XINT(101,3,J) = XINT(ISNODE,100,J)
      VALUE(101,3,J,1) = VALUE(ISNODE,100,J,1)
      VALUE(101,3,J,2) = VALUE(ISNODE,100,J,2)
 1195 CONTINUE
      XINT(101,3,11) = XINT(ISNODE,100,11)
C
C     THEN LOAD THE DISTRIBUTION OF THE NEXT ACTIVITY INTO TEMPORARY
C     LOCATION 4.
C
      DO 1200 J=1,10
      XINT(101,4,J) = XINT(ISNODE,IACT,J)
      VALUE(101,4,J,1) = VALUE(ISNODE,IACT,J,1)
      VALUE(101,4,J,2) = VALUE(ISNODE,IACT,J,2)
 1200 CONTINUE
      XINT(101,4,11) = XINT(ISNODE,IACT,11)
C
C     CONVOLUTE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 3 AND 4 AND
C     LOAD THE CONVOLUTION INTO TEMPORARY LOCATION 3.
C
      CALL SERIES(101,3,101,4)
C
C     PARALLEL-REDUCE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 3
AND
C     LOAD THE MAXIMUM INTO TEMPORARY LOCATION 1.
C
      CALL PARA(101,1,3)
 1205 CONTINUE
C
C     THE DISTRIBUTION THROUGH NODE I IS THE MAXIMUM IN TEMPORARY
LOCATION 1.
C     LOAD THIS INTO THE 100TH ACTIVITY POSITION OF NODE I.
C
      DO 1210 J=1,10
      XINT(I,100,J) = XINT(101,1,J)
      VALUE(I,100,J,1) = VALUE(101,1,J,1)
      VALUE(I,100,J,2) = VALUE(101,1,J,2)
 1210 CONTINUE
      XINT(I,100,11) = XINT(101,1,11)
      END IF
 1220 CONTINUE
C
C     THE DISTRIBUTION THROUGH NODE N IS THE FINAL EQUIVALENT ACTIVITY
OF THE
C     NETWORK.
```

```
C
 1240 CONTINUE
C
C     IF NODE I IS ON THE OUTPUT CRITICAL LIST, DO 1385 PRESENTS THE
C     DISTRIBUTION THROUGH NODE I IN THE OUTPUT.
C
      DO 1385 I = 2,N
      IF (IOCL(I) .EQ. 0) GO TO 1385
      PRINT 1910
      L = 1
      KK = 0
      DO 1270 J = 1,10
C
C     THE XX ARRAY IS USED FOR HISTOGRAM AND CDF CALCULATIONS.
C
      XX(1,1) = XINT(I,100,J)
      SIZE = (XINT(I,100,2)-XINT(I,100,1))/5.
      LASTK = L+4
      DO 1250 K = L,LASTK
      KK = KK+1
      XX(K,2) = VALUE(I,100,J,1)+(VALUE(I,100,J,2)*XX(K,1))
      IF ((KK .LE. 1).AND.(L .GT. 4)) XX(K,2) = (((VALUE(I,100,J,2)
     &*XX(K,1))+VALUE(I,100,J,1))+(VALUE(I,100,J-1,2)*XX(K,1))+
     &VALUE(I,100,J-1,1))/2.
      XX(K+1,1) = XX(K,1)+SIZE
1250  CONTINUE
      KK = 0
      L = L+5
      IF ((NAN .EQ. 1).OR.(NAN .EQ. 4).OR.(NAN .EQ. 5).OR.
     &(NAN .EQ. 7)) GO TO 1260
      GO TO 1270
 1260 IF (I .NE. N .AND. J .EQ. 1) THEN
      PRINT 1920,I
      ELSE IF (I .EQ. N .AND. J .EQ. 1) THEN
      PRINT 1925
      END IF
      PRINT 1930,J,XINT(I,100,J),XINT(I,100,J+1)
      PRINT 1940,VALUE(I,100,J,1),VALUE(I,100,J,2)
 1270 CONTINUE
      XX(51,2)=VALUE(I,100,10,1)+VALUE(I,100,10,2)*XX(51,1)
C
C     TOTAAR IS USED FOR CDF CALCULATIONS.
C
      AREA = 0.0
      DO 1280 J = 1,50
      AREA = AREA+((XX(J,2)+XX(J+1,2))*SIZE*.5)
      TOTAAR(J) = AREA
 1280 CONTINUE
      AREA = 1.0/AREA
      DO 1290 J = 1,50
      TOTAAR(J) = TOTAAR(J)*AREA
 1290 CONTINUE
      DO 1295 J = 51,2,-1
      TOTAAR(J) = TOTAAR(J-1)
 1295 CONTINUE
      TOTAAR(1) = 0.0
      XX(1,1) = XINT(I,100,1)
      IF ((NAN .EQ. 2).OR.(NAN .EQ. 4).OR.(NAN .EQ. 6).OR.
     &(NAN .EQ. 7)) GO TO 1300
      GO TO 1320
 1300 PRINT 1910
      PRINT 1950
      DO 1310 J = 1,51
      PRINT 1960,XX(J,1),TOTAAR(J)
 1310 CONTINUE
 1320 IF ((NAN .EQ. 3).OR.(NAN .EQ. 5).OR.(NAN .EQ. 6).OR.
     &(NAN .EQ. 7)) GO TO 1330
      GO TO 1340
 1330 IPRINT = 51
      IFLAG = 0
      CALL PLOT (IPRINT,KBL,KBM,IFLAG)
 1340 CONTINUE
C
C     DO 1360 COMPUTES AN APPROXIMATED EXPECTED VALUE AND
C     DO 1370 COMPUTES AN APPROXIMATED STANDARD DEVIATION
C     USING GROUPED DATA.
C
      AVG = 0.0
      SIG = 0.0
      LASTMM = IPRINT-1
      DO 1360 MM = 1,LASTMM
      AVG = AVG+((XX(MM,1)+(SIZE/2.))*(TOTAAR(MM+1)-TOTAAR(MM)))
 1360 CONTINUE
      DO 1370 MM = 1,LASTMM
      SIG = SIG+(((XX(MM,1)+(SIZE/2.)-AVG)**2)*(TOTAAR(MM+1)-
     &TOTAAR(MM)))
 1370 CONTINUE
      SIG = DSQRT(SIG)
      PRINT 1910
      PRINT 1970,AVG,SIG
      DO 1380 MM = 1,4
      HLOW = AVG-(FLOAT(MM)*SIG)
      HIGH = AVG+(FLOAT(MM)*SIG)
C
C     IT IS ASSUMED THAT THE DISTRIBUTION THROUGH NODE I RESEMBLES
C     A NORMAL DISTRIBUTION.  THE FIXED PERCENTAGES CORRESPOND TO
C     1, 2, 3, AND 4 STANDARD DEVIATIONS, RESPECTIVELY, FROM THE
C     EXPECTED VALUE.
C
      IF (MM .EQ. 1) PERCNT = 68.24
      IF (MM .EQ. 2) PERCNT = 95.44
      IF (MM .EQ. 3) PERCNT = 99.73
      IF (MM .EQ. 4) PERCNT = 99.99
      IF (I .NE. N) THEN
      PRINT 1980,I,HLOW,HIGH,PERCNT
      ELSE
      PRINT 1985,HLOW,HIGH,PERCNT
      END IF
 1380 CONTINUE
 1385 CONTINUE
      IF (NSIM .EQ. 0) GO TO 1540
C
```

```
C     DO 1510 COMPILES OUTPUT FROM THE MONTE CARLO SIMULATION FOR EACH
C     NODE ON THE OUTPUT CRITICAL LIST.
C
      DO 1510 I=2,N
      IF (IOCL(I) .EQ. 0) GO TO 1510
C
C     DO 1390 COMPUTES THE PARTITION OF THE INTERVAL OVER WHICH THE
C     THROUGHPUT DISTRIBUTION THROUGH NODE I IS DEFINED.
C
      XX(1,1) = XINT(I,100,1)
      SIZE = (XINT(I,100,2)-XINT(I,100,1))/5.
      DO 1390 J=2,51
      XX(J,1) = XX(J-1,1)+SIZE
 1390 CONTINUE
C
C     DO 1410 COMPILES THE CUMULATIVE DISTRIBUTION FUNCTION.
C
      COUNT = 0.0
      SIMTOT(1) = 0.0
      DO 1410 J=1,50
      DO 1400 K=1,NSIM
      IF ((XX(J,1) .LE. SIMT(I,K)) .AND. (SIMT(I,K) .LT. XX(J+1,1)))
     &COUNT = COUNT+1.
 1400 CONTINUE
      XX(J+1,2) = COUNT/DFLOAT(NSIM)
      SIMTOT(J+1) = SIMTOT(J)+XX(J+1,2)
      COUNT = 0.0
 1410 CONTINUE
      PRINT 1910
      IF (I .NE. N) THEN
      PRINT 1921,I
      ELSE
      PRINT 1926
      END IF
      IF ((NAN .EQ. 2) .OR. (NAN .EQ. 4) .OR. (NAN .EQ. 6) .OR.
     &(NAN .EQ. 7)) GO TO 1420
      GO TO 1440
 1420 PRINT 1950
      DO 1430 J=1,51
      PRINT 1960,XX(J,1),SIMTOT(J)
 1430 CONTINUE
 1440 IF ((NAN .EQ. 3) .OR. (NAN .EQ. 5) .OR. (NAN .EQ. 6) .OR.
     &(NAN .EQ. 7)) GO TO 1450
      GO TO 1470
 1450 DO 1460 J=1,50
      XX(J,1) = XX(J,1)+(SIZE/2.)
      XX(J,2) = XX(J+1,2)
 1460 CONTINUE
      IPRINT = 50
      IFLAG = 1
      CALL PLOT (IPRINT,KBL,KBM,IFLAG)
 1470 CONTINUE
C
C     DO 1480 COMPUTES AN APPROXIMATED EXPECTED VALUE AND
C     DO 1490 COMPUTES AN APPROXIMATED STANDARD DEVIATION
C     USING GROUPED DATA.
```

```
C
      AVG = 0.0
      SIG = 0.0
      DO 1480 J=1,50
      AVG = AVG+(XX(J,1)*(SIMTOT(J+1)-SIMTOT(J)))
 1480 CONTINUE
      DO 1490 J=1,50
      SIG = SIG+(((XX(J,1)-AVG)**2)*(SIMTOT(J+1)-SIMTOT(J)))
 1490 CONTINUE
      SIG = DSQRT(SIG)
      PRINT 1910
      PRINT 1970,AVG,SIG
      DO 1500 MM=1,4
      HLOW = AVG-(FLOAT(MM)*SIG)
      HIGH = AVG+(FLOAT(MM)*SIG)
C
C     IT IS ASSUMED THAT THE DISTRIBUTION THROUGH NODE I RESEMBLES
C     A NORMAL DISTRIBUTION.  THE FIXED PERCENTAGES CORRESPOND TO
C     1, 2, 3, AND 4 STANDARD DEVIATIONS, RESPECTIVELY, FROM THE
C     EXPECTED VALUE.
C
      IF (MM .EQ. 1) PERCNT = 68.24
      IF (MM .EQ. 2) PERCNT = 95.44
      IF (MM .EQ. 3) PERCNT = 99.73
      IF (MM .EQ. 4) PERCNT = 99.99
      IF (I .NE. N) THEN
      PRINT 1980,I,HLOW,HIGH,PERCNT
      ELSE
      PRINT 1985,HLOW,HIGH,PERCNT
      END IF
 1500 CONTINUE
 1510 CONTINUE
C
C     COMPARE POLYGONAL APPROXIMATION OF THROUGHPUT DISTRIBUTION
C     WITH SIMULATED THROUGHPUT DISTRIBUTION USING THE KOLMOGOROV-
C     SMIRNOV ONE-SAMPLE TEST.
C
      KSCR20 = 1.0730/SQRT(50.)
      KSCR10 = 1.2239/SQRT(50.)
      KSCR05 = 1.3581/SQRT(50.)
      KSCR02 = 1.5174/SQRT(50.)
      KSCR01 = 1.6276/SQRT(50.)
C
C     COMPUTE THE K-S TEST STATISTIC D-MAX.
C
      DMAX = 0.0
      DO 1530 I = 2,51
      DIFF = DABS(SIMTOT(I)-TOTAAR(I))
      IF (DIFF .GT. DMAX) DMAX = DIFF
 1530 CONTINUE
      PRINT 1910
      PRINT 1991,DMAX
      PRINT 1992,KSCR20,KSCR10,KSCR05,KSCR02,KSCR01
      IF (DMAX .LE. KSCR05) PRINT 1993
 1540 STOP
 1550 PRINT 1990
```

```
      STOP
C
C     FORMAT STATEMENTS
C
 1900 FORMAT (I3,1X,I4,1X,I1,1X,I5)
 1901 FORMAT (3(I2,25(1X,I2)/),I2,21(1X,I2),1X,I1,2(1X,I2))
 1902 FORMAT (2(I2,1X),F1.0,4(1X,F8.2))
 1910 FORMAT (1H1)
 1920 FORMAT (1X,'THE POLYGONAL APPROXIMATION OF THE TIME DISTRIBUTION',
     &' THROUGH NODE',1X,I2,1X,'IS:' ///)
 1921 FORMAT (1X,'THE SIMULATED TIME DISTRIBUTION',
     &' THROUGH NODE',1X,I2,1X,'IS:' ///)
 1925 FORMAT (1X,'THE POLYGONAL APPROXIMATION OF THE TIME DISTRIBUTION',
     &' THROUGH THE PROJECT IS:' ///)
 1926 FORMAT (1X,'THE SIMULATED TIME DISTRIBUTION',
     &' THROUGH THE PROJECT IS:' ///)
 1930 FORMAT (1X,'INTERVAL',I3,4X,'LOWER LIMIT =',F8.2,3X,
     &'UPPER LIMIT =',F8.2 //)
 1940 FORMAT (15X,'X = (',F12.8,') + (',F12.8,') T' ///)
 1950 FORMAT (14X,'CUMULATIVE DISTRIBUTION FUNCTION' //
     &21X,'T',14X,'F(T)')
 1960 FORMAT (16X,F9.3,F17.8)
 1970 FORMAT (12X,'EXPECTED VALUE OF T     =',F13.8 /
     &12X,'STANDARD DEVIATION OF T =',F13.8 //)
 1980 FORMAT (1X,'THE PROBABILITY OF NODE ',I3,' THROUGHPUT TIME',
     &' FALLING BETWEEN' / 1X,F8.3,' TIME UNITS AND',F8.3,
     &' TIME UNITS IS ABOUT ',F5.2,' %.'//)
 1985 FORMAT (1X,'THE PROBABILITY OF THE PROJECT THROUGHPUT TIME',
     &' FALLING BETWEEN' / 1X,F8.3,' TIME UNITS AND ',F8.3,
     &' TIME UNITS IS ABOUT ',F5.2,' %.'//)
 1990 FORMAT (1X,'PROGRAM STOPPED' / 1X,'IMPROPER NODE NUMBER(S) '
     &,'ENCOUNTERED')
 1991 FORMAT (1X,'KOLMOGOROV-SMIRNOV ONE-SAMPLE TEST COMPARISON OF ',
     &'POLYGONAL APPROXIMATION' / 1X,'OF NETWORK THROUGHPUT DISTRIBUTION
     & AND SIMULATED NETWORK THROUGHPUT' / 1X,'DISTRIBUTION:' //
     &1X,'K-S TEST STATISTIC D-MAX = ', F6.4 /)
 1992 FORMAT (1X,'K-S CRITICAL VALUES:' / 15X,'20 PERCENT = ',F6.4 /
     &15X,'10 PERCENT = ',F6.4 / 16X,'5 PERCENT = ',F6.4 /
     &16X,'2 PERCENT = ',F6.4 / 16X,'1 PERCENT = ',F6.4 /)
 1993 FORMAT (1X,'FAIL TO REJECT THE NULL HYPOTHESIS THAT THE ',
     &'DISTRIBUTIONS ARE THE SAME' / 1X,'AT THE 5% LEVEL OF ',
     &'STATISTICAL SIGNIFICANCE.')
      END
C     END MAIN PROGRAM
C
C     ***************************************************************
C
C        S U B R O U T I N E          P A R A
C
C     ***************************************************************
C
      SUBROUTINE PARA (L1,L2,L3)
      REAL*8 VALUE(101,100,10,3),XINT(101,100,12)
      REAL*8 XVAL,ZVAL(130,5),PAR(2,15,6),FACT,B(130)
      REAL*4 Z
      INTEGER L1,L2,L3,NV1,NV2
      INTEGER K4(2,30)
      INTEGER I,IINT,N,NCL,J,K,K3,L6,LASTJ,LASTK
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/B
C
C     SUBROUTINE PARA IS USED TO REDUCE PARALLEL ARCS INTO A SINGLE
C     EQUIVALENT ARC.  IT FINDS THE MAX OPERATOR BY MULTIPLYING CAP
C     F(X) AGAINST CAP G(X) OVER THE INTERVALS OF VALIDITY.
C
      NV1 = 10
      NV2 = 10
      DO 2020 N = 1,2
      L6 = L2
      IF (N .EQ. 2) L6 = L3
      FACT = 0
      DO 2010 J = 1,10
      B(1) = XINT(L1,L6,J)
C
C     DO 2000 CONVERTS EACH LINEAR POLYNOMIAL PIECE OF LITTLE F(X)
C     INTO THE CORRESPONDING QUADRATIC POLYNOMIAL PIECE OF ITS
C     CUMULATIVE DISTRIBUTION CAP F(X).
C
      DO 2000 I = 1,2
      XVAL = VALUE(L1,L6,J,I)
      Z = FLOAT(I)
      PAR(N,J,I+1) = XVAL/Z
      PAR(N,J,1) = PAR(N,J,1)+((-1.0)*(XVAL/Z)*(B(1)**I))
      K4(N,J) = I+1
 2000 CONTINUE
      IF (J .GT. 1) PAR(N,J,1) = PAR(N,J,1)+FACT
      FACT = PAR(N,J,1)+(PAR(N,J,2)*XINT(L1,L6,J+1))+(PAR(N,J,3)
     &*(XINT(L1,L6,J+1)**2))
 2010 CONTINUE
 2020 CONTINUE
C
C     DO 2040 ASSIGNS INTERVAL BOUNDARY VALUES TO THE B ARRAY.
C
      DO 2040 I = 1,22
      IF (I .GT. 11) GO TO 2030
      B(I) = XINT(L1,L2,I)
      GO TO 2040
 2030 B(I) = XINT(L1,L3,I-11)
 2040 CONTINUE
      NCL = 21
      CALL SORT(NCL)
C
C     DO 2080 DETERMINES THE POINT AT WHICH THE DISTRIBUTION DOMAINS
C     OF THE TWO ARCS BEING COMBINED OVERLAP.  ONCE THIS POINT IS
C     DETERMINED, THE B ARRAY IS ADJUSTED TO REFLECT THE OVERLAP
C     (ALL VALUES LESS THAN THIS POINT OF FIRST OVERLAP NEED NOT BE
C     CONSIDERED, BECAUSE ONE OF THE DISTRIBUTIONS EQUALS ZERO AT
C     THESE VALUES). IF THE DOMAINS ARE DISJOINT OR OVERLAP AT ONLY.
C     ONE BOUNDARY POINT, THE RESULT OF THE APPLICATION OF THE
C     MAXIMUM OPERATOR IS JUST THE UNCHANGED APPROXIMATED PROBABILITY
C     DENSITY FUNCTION OF THE DISTRIBUTION DEFINED ON THE HIGHER-
```

```
C     VALUED DOMAIN.  GO TO 2180 OR GO TO 2160 RETURNS THIS FUNCTION
C     DIRECTLY WITHOUT FURTHER PROCESSING.
C
      IINT = 0
      LASTJ = NCL+1
      DO 2080 J = 1,LASTJ
      IF ((XINT(L1,L2,1) .GE. XINT(L1,L3,1)-0.001) .AND.
     &(XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001)) GO TO 2080
      IF (IINT .GE. 1) GO TO 2060
      IF (XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001) GO TO 2050
      IF (XINT(L1,L3,J+1) .GE. XINT(L1,L2,1)-0.001) IINT = J
      IF ((XINT(L1,L3,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L2,1) .GE. XINT(L1,L3,J+1)-0.001)
     &.AND. (XINT(L1,L2,1) .LE. XINT(L1,L3,J+1)+0.001)
     &.AND. (XINT(L1,L3,J+2) .LE. 0.001))) GO TO 2180
      GO TO 2080
 2050 IF (XINT(L1,L2,J+1) .GE. XINT(L1,L3,1)-0.001) IINT = J
      IF ((XINT(L1,L2,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L3,1) .GE. XINT(L1,L2,J+1)-0.001)
     &.AND. (XINT(L1,L3,1) .LE. XINT(L1,L2,J+1)+0.001)
     &.AND. (XINT(L1,L2,J+2) .LE. 0.001))) GO TO 2160
      GO TO 2080
 2060 LASTK = NCL-(IINT-1)
      DO 2070 K = 1,LASTK
      B(K) = B(K+IINT)
      B(K+IINT) = 0
 2070 CONTINUE
      GO TO 2090
 2080 CONTINUE
 2090 NCL = NCL-IINT
C
C     DO 2150 IS THE OUTER LOOP FOR THE PROCESS OF CREATING THE
C     EQUIVALENT ARC.  NCL IS THE NUMBER OF CLASSES INVOLVED
C     BETWEEN THE TWO ARCS.
C
      N1 = 0
      N2 = 0
      DO 2150 I = 1,NCL
      DO 2110 J = 1,11
C
C     DO 2110 DETERMINES THE APPROPRIATE INTERVALS OF EACH DISTRIBUTION
C     THAT ARE VALID FOR THE B(I) VALUE BEING CONSIDERED.  N1 AND
C     N2 ARE THE CONTROLS FOR UPPER AND LOWER ARCS RESPECTIVELY.
C
      IF (N1 .GE. 1) GO TO 2100
      IF (((B(I) .GE. XINT(L1,L2,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L2,J+1)+0.001)) .OR. (XINT(L1,L2,J+1) .LE. 0.001))
     &N1 = J
 2100 CONTINUE
      IF (N2 .GE. 1) GO TO 2110
      IF (((B(I) .GE. XINT(L1,L3,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L3,J+1)+0.001)) .OR. (XINT(L1,L3,J+1) .LE. 0.001))
     &N2 = J
 2110 CONTINUE
      IF (N2 .GT. NV2) K4(2,N2) = 1
      IF (N1 .GT. NV1) K4(1,N1) = 1
```

```
C
C     DO 2130 AND DO 2120 PERFORM THE POLYGONAL MULTIPLICATION FOR
C     CAP F(X) AND CAP G(X).
C
      LASTJ = K4(2,N2)
      LASTK = K4(1,N1)
      DO 2130 J = 1,LASTJ
      DO 2120 K = 1,LASTK
      IF (N2 .GT. NV2) PAR(2,N2,J) =  1
      IF (N1 .GT. NV1) PAR(1,N1,K) =  1
      K3 = J+K-1
      ZVAL(I,K3) = ZVAL(I,K3)+(PAR(1,N1,K)*PAR(2,N2,J))
 2120 CONTINUE
 2130 CONTINUE
C
C     DO 2140 OBTAINS THE FIRST DERIVATIVE OF THE RESULT OF THE
C     MULTIPLICATION OF CAP F(X) AND CAP G(X) IN THE FORM OF A
C     LITTLE H(X) FOR THAT PRODUCT.
C
      DO 2140 J = 1,4
      ZVAL(I,J) = ZVAL(I,J+1)*FLOAT(J)
      ZVAL(I,J+1) = 0
 2140 CONTINUE
      N1 = 0
      N2 = 0
 2150 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE RESULTS OF THE
C     PARALLEL REDUCTION WITH THE B(O) AND B(1) FORM IN EACH OF 10
C     CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR(L1,L2,NCL)
      GO TO 2180
 2160 DO 2170 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L3,I,1)
      VALUE(L1,L2,I,2) = VALUE(L1,L3,I,2)
      XINT(L1,L2,I) = XINT(L1,L3,I)
 2170 CONTINUE
      XINT(L1,L2,11) = XINT(L1,L3,11)
 2180 VALUE(L1,L2,1,3) = 0
      DO 2210 I = 1,2
      DO 2200 J = 1,10
      DO 2190 K = 1,3
      PAR(I,J,K) = 0
 2190 CONTINUE
 2200 CONTINUE
 2210 CONTINUE
      RETURN
      END
C     END SUBROUTINE PARA
C
C     ***************************************************************
C
C                   S U B R O U T I N E      S E R I E S
C
```

```
C     *************************************************************
C
      SUBROUTINE SERIES (L1,L2,L3,L4)
      REAL*8 VALUE(101,100,10,3),XINT(101,100,12)
      REAL*8 ZVAL(130,5),XLIM(2),A(130)
      REAL*8 F0,F1,G0,G1,XL
      INTEGER L1,L2,L3,L4
      INTEGER ISEL(2)
      INTEGER I,IK,J,K,NCL,NCL1,NE
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
C
C     SUBROUTINE SERIES PERFORMS THE CONVOLUTION OF TWO PROBABILITY
C     DISTRIBUTIONS BY INTEGRATING THE PRODUCT OF THEIR PIECEWISE
C     POLYGONAL APPROXIMATIONS IN THE FORMS OF F(X) AND G(T-X).
C
C     THIS SECTION DETERMINES THE INTERVALS OF VALIDITY FOR THE
C     CONVOLUTION.
C
C     THE A ARRAY IS USED FOR THE SAME PURPOSE AS THE B ARRAY IN PARA.
C
      K = 0
C
C     DO 3010 CREATES ALL POSSIBLE INTERVALS OF THE NEW DISTRIBUTION
C     BY ADDING THE INTERVALS OF THE TWO DISTRIBUTIONS BEING WORKED.
C
      DO 3010 I = 1,12
      IF ((XINT(L3,L4,I) .LE. 0).AND.(I .GT. 1)) GO TO 3020
      DO 3000 J = 1,12
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) NCL1 = J-2
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) GO TO 3010
      K = K+1
      A(K) = XINT(L1,L2,J)+XINT(L3,L4,I)
 3000 CONTINUE
 3010 CONTINUE
 3020 NINT = I-2
      NCL = K-1
C
C     DO 3120 IS CONTROLLED BY THE NUMBER OF CLASSES IN THE F(X)
C     DISTRIBUTION.  DO 3110 IS CONTROLLED BY THE NUMBER OF CLASSES
C     CREATED BY COMBINING F(X) AND G(T-X).  DO 3100 IS CONTROLLED
C     BY THE NUMBER OF CLASSES IN THE G(T-X) DISTRIBUTION.  THIS
C     ALLOWS THE EVALUATION OF ALL OF THE CREATED CLASSES FOR EVERY
C     CLASS IN BOTH DISTRIBUTIONS.
C
      CALL SORT (NCL)
      DO 3120 K = 1,NCL1
      DO 3110 I = 1,NCL
      DO 3100 J = 1,NINT
      IK = 0
C
C     THIS IF STATEMENT DETERMINES WHICH INTERVALS ARE VALID FOR THE
C     INTERVAL END POINT A(I) BEING EVALUATED AND FOR THE VALUE OF K
C     BEING CONTROLLED BY DO 3120.
C
```

```
      IF ((A(I) .GE. XINT(L1,L2,K)+XINT(L3,L4,J)-0.001) .AND. (A(I+1)
     &.LE. XINT(L1,L2,K+1)+XINT(L3,L4,J+1)+0.001)) IK = J
      IF (IK .GE. 1) GO TO 3030
      GO TO 3100
 3030 ISEL(1) = 0
      ISEL(2) = 0
C
C     THE IF STATEMENTS INVOLVING XLIM ARE USED TO DETERMINE THE
C     UPPER AND LOWER LIMITS OF INTEGRATION.  IT IS DETERMINED WHETHER
C     THE LIMIT COMES FROM THE F(X) OR THE G(T-X) DISTRIBUTION.  ISEL
C     IS USED TO DESIGNATE VALUES FROM THE G(T-X) DISTRIBUTION.
C
      IF (XINT(L1,L2,K) .GE. (A(I+1)-XINT(L3,L4,J+1)-0.001)) GO TO 3040
      XLIM(1) = XINT(L3,L4,J+1)
      ISEL(1) = 999
      GO TO 3050
 3040 XLIM(1) = XINT(L1,L2,K)
 3050 IF (XINT(L1,L2,K+1) .LE. (A(I)-XINT(L3,L4,J)+0.001)) GO TO 3060
      XLIM(2) = XINT(L3,L4,J)
      ISEL(2) = 999
      GO TO 3070
 3060 XLIM(2) = XINT(L1,L2,K+1)
 3070 CONTINUE
      DO 3090 NE = 1,2
      F0 = VALUE(L1,L2,K,1)
      F1 = VALUE(L1,L2,K,2)
      G0 = VALUE(L3,L4,IK,1)
      G1 = VALUE(L3,L4,IK,2)
      XL = XLIM(NE)
      Z = 1.0
      IF (NE .EQ. 1) Z = -1.0
      IF (ISEL(NE) .EQ. 999) GO TO 3080
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL AT A FINITE
C     LIMIT.  THE INTEGRATION IS BROKEN DOWN INTO ITS COMPONENT PARTS
C     BY THE POWER OF THE COEFFICIENT THAT RESULTS.  Z CONTROLS THE
C     SIGN OF THE INTEGRAL BASED ON WHETHER THE LOWER OR UPPER LIMIT
C     IS BEING EVALUATED.
C
      ZVAL(I,1) = ZVAL(I,1)+((F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((-1.0*F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+(((F1*G1*XL**2)/2.)+(F0*G1*XL))*Z
      ZVAL(I,3) = ZVAL(I,3)+((-1.0*F0*G1)/2.)*Z
      GO TO 3090
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL FOR LIMITS.
C     IN THE FORM OF (T-X).  THE FORMULAS ARE DIFFERENT BECAUSE
C     OF THE DIFFERENT POLYNOMIAL CREATED WHEN THE INTEGRATION
C     INVOLVES LIMITS IN THE FORM OF (T-X).
C
 3080 ZVAL(I,1) = ZVAL(I,1)+((-1.0*F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+((-1.0*F1*G0*XL)+(F0*G0)-
     &((F1*G1*XL**2)/2.))*Z
      ZVAL(I,3) = ZVAL(I,3)+((F1*G0)/2.)*Z
      ZVAL(I,4) = ZVAL(I,4)+((F1*G1)/6.)*Z
 3090 CONTINUE
 3100 CONTINUE
 3110 CONTINUE
 3120 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE CONVOLUTION
C     RESULTS WITH THE B(0) AND B(1) FORM IN EACH OF 10 CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR (L1,L2,NCL)
      RETURN
      END
C     END SUBROUTINE SERIES
C
C     ************************************************************
C
C                S U B R O U T I N E       P L O T
C
C     ************************************************************
C
      SUBROUTINE PLOT (IPRINT,KBL,KBM,IFLAG)
      REAL*8 XX(100,2),SORT
      CHARACTER*1 KBL,KBM,LINE(101)
      INTEGER I,IFLAG,IPRINT,J,JPLOT,K,NN
      COMMON/PARA4/XX
C
C     PLOT IS USED TO CREATE THE HISTOGRAM FOR FINAL OUTPUT.
C     THE VARIABLE SORT IN THIS SUBROUTINE IS NOT RELATED TO
C     THE SUBROUTINE SORT.
C
      SORT = XX(1,2)
      DO 4000 I = 2,IPRINT
      IF (SORT .LE. XX(I,2)) SORT = XX(I,2)
 4000 CONTINUE
      PRINT 4900
      IF (IFLAG .EQ. 0) THEN
      PRINT 4910
      ELSE
      PRINT 4915
      END IF
      IF (SORT .GT. 0.5) PRINT 4920
      IF ((SORT .GT. 0.25).AND.(SORT .LE. 0.50)) PRINT 4930
      IF ((SORT .GT. 0.10).AND.(SORT .LE. 0.25)) PRINT 4940
      IF ((SORT .GT. 0.05).AND.(SORT .LE. 0.10)) PRINT 4950
      IF (SORT .LE. 0.05) PRINT 4960
      PRINT 4970
      DO 4030 I = 1,IPRINT
      DO 4010 J = 1,51
      LINE(J) = KBL
 4010 CONTINUE
      IF (SORT .GT. 0.5) JPLOT = (INT((XX(I,2)*50.)+0.5))+1
      IF ((SORT .GT. 0.25).AND.(SORT .LE. 0.50))
     &JPLOT = (INT((XX(I,2)*100.)+0.5))+1
      IF ((SORT .GT. 0.10).AND.(SORT .LE. 0.25))
     &JPLOT = (INT((XX(I,2)*200.)+0.5))+1
      IF ((SORT .GT. 0.05).AND.(SORT .LE. 0.10))
```

```
    &JPLOT = (INT((XX(I,2)*500.)+0.5))+1
      IF (SORT .LE. 0.05) JPLOT = (INT((XX(I,2)*1000.0)+0.5))+1
      IF (JPLOT .LE. 0) JPLOT = 1
      IF (JPLOT .GT. 51) JPLOT = 51
      DO 4020 NN = 1,JPLOT
      LINE(NN) = KBM
 4020 CONTINUE
      PRINT 4980,XX(I,1),(LINE(K), K = 1,JPLOT)
 4030 CONTINUE
C
C     FORMAT STATEMENTS
C
 4900 FORMAT (1H1)
 4910 FORMAT (15X,'PROBABILITY DENSITY FUNCTION' //)
 4915 FORMAT (15X,'SIMULATION FREQUENCY HISTOGRAM' //)
 4920 FORMAT (9X,'0         .20       .40       .60       .80       1.0')
 4930 FORMAT (9X,'0         .10       .20       .30       .40       .50')
 4940 FORMAT (9X,'0         .05       .10       .15       .20       .25')
 4950 FORMAT (9X,'0         .02       .04       .06       .08       .10')
 4960 FORMAT (9X,'0         .01       .02       .03       .04       .05')
 4970 FORMAT (9X,'I----+----I----+----I----+----I----+----I----+----I')
 4980 FORMAT (1X,F8.3,2X,51A1)
      RETURN
      END
C     END SUBROUTINE PLOT
C
C     **************************************************************
C
C                   S U B R O U T I N E       L I N E A R
C
C     **************************************************************
C
      SUBROUTINE LINEAR (L1,L2,NCL)
      REAL*8 VALUE(101,100,10,3),XINT(101,100,12),ZVAL(130,5),A(130)
      REAL*8 Q,Q1,Q2,STD,SUMX,SUMY,SUMXY,SUMSQ
      REAL*8 ALPHA,AREA,BETA,FACT,SIZE,W,X,XLMBDA,XMEAN
      REAL*8 XMODE,XSIZE,Y
      INTEGER L1,L2
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      EXTERNAL DGAMMA
C
C     SUBROUTINE LINEAR PIECEWISE POLYGONALIZES DISTRIBUTION DATA
C     FROM THE MAIN PROGRAM AND SUBROUTINES PARA AND SERIES WITH
C     THE B(O) AND B(1) FORM IN EACH OF 10 CLASSES THROUGH THE USE
C     OF SIMPLE LINEAR REGRESSION.
C
      XMODE = VALUE(L1,L2,2,3)
      XMEAN = VALUE(L1,L2,2,3)
      STD = ((VALUE(L1,L2,2,3)-XINT(L1,L2,1))/3.)
      XLMBDA = VALUE(L1,L2,2,3)-XINT(L1,L2,1)
      ALPHA = VALUE(L1,L2,2,3)
      BETA = VALUE(L1,L2,3,3)
      SIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) SIZE = (A(NCL+1)-A(1))/10.
      XINT(L1,L2,11) = XINT(L1,L2,2)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) XINT(L1,L2,11) = A(NCL+1)
      X = XINT(L1,L2,1)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) X = A(1)
      DO 5000 I = 1,10
      XINT(L1,L2,I) = X
      X = X+SIZE
 5000 CONTINUE
      DO 5050 I = 1,10
      X = XINT(L1,L2,I)
      SUMY = 0.
      SUMX = 0.
      SUMXY = 0.
      SUMSQ = 0.
C
C     W CONTROLS THE NUMBER OF DATA POINTS USED IN THE REGRESSION
C     COMPUTATIONS.
C
      W = 10.+IDINT(SIZE*3.)
      XSIZE = SIZE/W
      LASTJ = IDINT(W)
      DO 5040 J = 1,LASTJ
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 99)  GO TO 5030
      DO 5010 K3 = 1,NCL
      K  = 0
      IF ((X .GE. A(K3)).AND.(X .LE. A(K3+1))) K = K3
      IF (K .GE. 1) GO TO 5020
 5010 CONTINUE
C
C     SERIES OR PARA GENERATED DISTRIBUTIONS.
C
 5020 Y = ZVAL(K,1)+(ZVAL(K,2)*X)+(ZVAL(K,3)*(X**2))
     &+(ZVAL(K,4)*(X**3))
 5030 CONTINUE
C
C     TRIANGULAR DISTRIBUTION.
C
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 1) THEN
         IF (XINT(L1,L2,1) .LE. X .AND. X .LE. XMODE) THEN
         Y = (2.*(X-XINT(L1,L2,1)))/((XMODE-XINT(L1,L2,1))*10.*SIZE)
         ELSE
         Y = (2.*(XINT(L1,L2,11)-X))/((XINT(L1,L2,11)-XMODE)*10.*SIZE)
         END IF
C
C     NORMAL  DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 2) THEN
      Y = (1./(STD*2.506628275))*(DEXP((-1.0)*(((X-XMEAN)/STD)**2)/2.))
C
C     EXPONENTIAL DISTRIBUTION  (SHIFTED).
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 3) THEN
      Y = (1./XLMBDA)*(DEXP((-1.0)*((X-XINT(L1,L2,1))/XLMBDA)))
C
C     GAMMA DISTRIBUTION.
C
```

```
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 4) THEN
      Y = (1./(DGAMMA(ALPHA)*(BETA**ALPHA)))*DEXP(-X/BETA)*(X**(ALPHA-1.
     &))
C
C     BETA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 5) THEN
      Y = (DGAMMA(ALPHA+BETA)/(DGAMMA(ALPHA)*DGAMMA(BETA)))*
     &(1./(10.*SIZE)**(ALPHA+BETA-2.))*
     &((X-XINT(L1,L2,1))**(ALPHA-1.))*
     &((XINT(L1,L2,11)-X)**(BETA-1.))
      END IF
      IF (Y .LT. 0) Y = 0
      SUMX = SUMX+X
      SUMY = SUMY+Y
      SUMXY = SUMXY+(X*Y)
      SUMSQ = SUMSQ+(X**2)
      X = X+XSIZE
 5040 CONTINUE
      VALUE(L1,L2,I,2) = (SUMXY-((SUMX*SUMY)/W))/(SUMSQ-((SUMX**2)/W))
      VALUE(L1,L2,I,1) = (SUMY/W)-(VALUE(L1,L2,I,2)*(SUMX/W))
 5050 CONTINUE
C
C     DO 5060 CALCULATES THE AREA UNDER THE APPROXIMATED DISTRIBUTION.
C     AN ADJUSTMENT FACTOR FOR THE AMOUNT THAT THIS AREA HAS BEEN
C     UNDERESTIMATED OR OVERESTIMATED IS THEREBY DETERMINED.
C
      DO 5060 I = 1,10
      Q  = XINT(L1,L2,I+1)-XINT(L1,L2,I)
      Q1 = (XINT(L1,L2,I)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      Q2 = (XINT(L1,L2,I+1)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      IF (Q1 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q1*(-1.0))
      IF (Q2 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q2*(-1.0))
      IF (Q1 .LT. 0.) Q1 = 0.
      IF (Q2 .LT. 0.) Q2 = 0.
      AREA = AREA+((Q1+Q2)*Q*0.5)
 5060 CONTINUE
      FACT = 1.0/AREA
C
C     DO 5070 ADJUSTS THE COEFFICIENTS OF ALL THE LINEAR POLYNOMIAL
C     PIECES BY THE FACTOR COMPUTED IN DO 5060 IN ORDER TO NORMALIZE
C     THE AREA BACK TO ONE.  THIS ACTS TO REDUCE ACCUMULATING ERRORS
C     DURING PROGRAM COMPUTATIONS.
C
      DO 5070 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)*FACT
      VALUE(L1,L2,I,2) = VALUE(L1,L2,I,2)*FACT
 5070 CONTINUE
      AREA = 0
      DO 5080  I = 1,130
      A(I)  = 0
      ZVAL(I,1) = 0
      ZVAL(I,2) = 0
      ZVAL(I,3) = 0
      ZVAL(I,4) = 0
 5080 CONTINUE
```

```
      RETURN
      END
C     END SUBROUTINE LINEAR
C
C     ************************************************************
C
C                 S U B R O U T I N E         S O R T
C
C     ************************************************************
C
      SUBROUTINE SORT (NCL)
      REAL*8 A(130),B
      INTEGER NCL
      INTEGER I,K1
      COMMON/PARA3/A
C
C     SUBROUTINE SORT IS USED TO CONDUCT AN ALGEBRAIC SORT OF DATA
C     CREATED IN THE SERIES AND PARA SUBROUTINES.
C
 6000 K1  = 0
      DO 6010 I = 1,NCL
      IF ((A(I) .LT. (A(I+1)+.01)).AND.(A(I) .GT. (A(I+1)-.01)))
     &GO TO 6020
      IF (A(I) .LT. A(I+1)) GO TO 6010
      IF (A(I) .GT. A(I+1)) B = A(I)
      A(I) = A(I+1)
      A(I+1) = B
      K1=K1+1
 6010 CONTINUE
      IF (K1 .GE. 1) GO TO 6000
      GO TO 6040
 6020 NCL = NCL-1
      LASTJ = NCL+1
      DO 6030 J = I,LASTJ
      A(J) = A(J+1)
      A(J+1) = 0
 6030 CONTINUE
      GO TO 6000
 6040 RETURN
      END
C     END SUBROUTINE SORT
C
C     ************************************************************
C
C                S U B R O U T I N E         S I M U L T
C
C     ************************************************************
C
      SUBROUTINE SIMULT (N,NSIM)
      REAL*8 XINT(101,100,12),VALUE(101,100,10,3)
      REAL*8 T(100,99),SIMT(100,10000)
      REAL*8 ALPHA,BETA,X,XLNGTH,XLMBDA,XMAX,XMEAN,XMIN,XMODE
      REAL*8 RN,STD,TTEMP,TMAX
      DIMENSION NET(100,103)
      INTEGER ISEED,ISIM,N,NDIST,NSIM
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA5/NET
      COMMON/PARA6/SIMT
      EXTERNAL DRNUN,DRNNOR,DRNEXP,DRNGAM,DRNBET,RNSET
C
C     DO 7130 GENERATES A SIMULATED NETWORK THROUGHPUT FOR EACH OF
C     NSIM ITERATIONS OF THE MONTE CARLO SIMULATION OF THE NETWORK.
C
      ISEED = 123456789
      CALL RNSET(ISEED)
      DO 7130 ISIM=1,NSIM
C
C     DO 7080 GENERATES A RANDOM VALUE FROM THE ACTIVITY RESOURCE
C     CONSUMPTION (ACTIVITY TIME) DISTRIBUTION OF EACH ACTIVITY.
C
      DO 7080 I=1,N-1
      DO 7070 J=1,NET(I,103)
      NDIST = IDINT(VALUE(I,J,1,3))
      XMIN = XINT(I,J,1)
      XMAX = XINT(I,J,11)
      XLNGTH = XMAX-XMIN
      GO TO (7010,7020,7030,7040,7050,7060) NDIST
C
C     TRIANGULAR DISTRIBUTION.
C
 7010 CALL DRNUN(1,RN)
      XMODE = VALUE(I,J,2,3)
      X = (XMODE-XMIN)/XLNGTH
      IF (RN .GT. X) GO TO 7015
      T(I,J) = XMIN+DSQRT(RN*XLNGTH*(XMODE-XMIN))
      GO TO 7070
 7015 T(I,J) = XMAX-DSQRT(XLNGTH*(XMAX-XMODE)*(1.-RN))
      GO TO 7070
C
C     NORMAL DISTRIBUTION.
C
 7020 CALL DRNNOR(1,RN)
      XMEAN = VALUE(I,J,2,3)
      STD = (XMEAN-XMIN)/3.
      T(I,J) = (RN*STD)+XMEAN
      IF ((T(I,J) .LT. XMIN) .OR. (T(I,J) .GT. XMAX)) GO TO 7020
      GO TO 7070
C
C     EXPONENTIAL DISTRIBUTION.
C
 7030 CALL DRNEXP(1,RN)
      XLMBDA = VALUE(I,J,2,3)-XMIN
      T(I,J) = (XLMBDA*RN)+XMIN
      IF (T(I,J) .GT. XMAX) GO TO 7030
      GO TO 7070
C
C     GAMMA DISTRIBUTION.
C
 7040 ALPHA = VALUE(I,J,2,3)
      BETA = VALUE(I,J,3,3)
      CALL DRNGAM(1,ALPHA,RN)
      T(I,J) = BETA*RN
      IF (T(I,J) .GT. XMAX) GO TO 7040
      GO TO 7070
C
C     BETA DISTRIBUTION.
C
 7050 ALPHA = VALUE(I,J,2,3)
      BETA = VALUE(I,J,3,3)
      CALL DRNBET(1,ALPHA,BETA,RN)
      T(I,J) = XMIN+(XLNGTH*RN)
      GO TO 7070
C
C     UNIFORM DISTRIBUTION.
C
 7060 CALL DRNUN(1,RN)
      T(I,J) = XMIN+(XLNGTH*RN)
 7070 CONTINUE
 7080 CONTINUE
C
C     DO 7120 GENERATES THE CRITICAL PATH TO EACH NODE.  THE
C     SIMULATED TIME THROUGH NODE I FROM SIMULATION ITERATION L
C     IS STORED IN SIMT(I,L).
C
      SIMT(1,ISIM) = 0.0
      DO 7120 I=2,N
      TMAX = 0.0
C
C     DO 7110 DETERMINES THE STARTING NODES AND THE ACTIVITIES WHICH
C     TERMINATE AT NODE I > STARTING NODES, AND COMPUTES THE SIMULATED
C     THROUGHPUT VALUE THROUGH NODE I AS THE MAXIMUM OF THE
C         [(THROUGHPUT VALUE THROUGH STARTING NODE) +
C          (ACTIVITY VALUE FROM STARTING NODE TO NODE I)].
C
      DO 7110 J=1,I-1
      DO 7100 J1=2,NET(J,103)+1
      IF (NET(J,J1)-I) 7100,7090,7100
 7090 TTEMP = SIMT(J,ISIM)+T(J,J1-1)
      IF (TTEMP .LT. TMAX) GO TO 7100
      TMAX = TTEMP
 7100 CONTINUE
 7110 CONTINUE
      SIMT(I,ISIM) = TMAX
 7120 CONTINUE
 7130 CONTINUE
      RETURN
      END
C     END SUBROUTINE SIMULT
```

APPENDIX C

PART PROGRAM FOR LARGE NETWORKS

```
C
C
C             POLYGONAL APPROXIMATION AND REDUCTION TECHNIQUE
C                                 (PART)
C                                ALGORITHM
C                   TO APPROXIMATE CRITICALITY INDICES
C                                    OF
C                           ACTIVITIES AND NODES
C                                   AND
C                          TO IDENTIFY THE K MOST
C                     STOCHASTICALLY DOMINATING PATHS
C                                    OF
C                       ACYCLIC, DIRECTED NETWORKS
C                                 USING
C                   "SEQUENTIAL APPROXIMATION" METHOD
C
C
C     THIS PROGRAM APPROXIMATES THE CRITICALITY INDICES OF THE
ACTIVITIES
C     AND NODES OF AN ACYCLIC, DIRECTED NETWORK AND IDENTIFIES THE NET-
C     WORK'S K MOST STOCHASTICALLY DOMINATING PATHS WITH THE PART
ALGORITHM
C     USING "SEQUENTIAL APPROXIMATION."  THE PROGRAM IS WRITTEN IN
FORTRAN
C     77 AND IS PRESENTLY DESIGNED TO BE OPERATED IN A TIME SHARING MODE
C     WITH ALL DATA INPUT FROM THREE (3) DATA FILES.  THE PROGRAM
DIRECTS
C     OUTPUT TO A TIME SHARING TERMINAL.  IF DESIRED, THE READ
STATEMENTS
C     AT THE BEGINNING OF THE MAIN PROGRAM CAN BE MODIFIED TO ALLOW DATA
C     INPUT DIRECTLY FROM THE TIME SHARING TERMINAL.
C
C     THE CURRENT DIMENSIONS OF THE PROGRAM ALLOW A NETWORK WITH A
C     MAXIMUM OF 100 NODES AND A MAXIMUM OF 99 ACTIVITIES BEGINNING
C     AT EACH NODE.  THESE LIMITS CAN BE EXPANDED BY CHANGING THE
C     DIMENSIONS OF THE XINT AND VALUE ARRAYS.
C
C
C     ************************************************************
C
C                  O P E R A T I N G   I N S T R U C T I O N S
C
C     ************************************************************
C
C     INSTRUCTIONS FOR BUILDING DATA FILES
C     ------------------------------------
C
C           DATA FILE DATAN.PATHS
C
C     THIS DATA FILE CONTAINS A DESCRIPTION OF THE NETWORK STRUCTURE.
C     EACH NODE REQUIRES 4 RECORDS WITH A TOTAL OF 103 FIELDS:
C         FIELD 1 IS THE BEGINNING NODE.
C         FIELDS 2 THRU 100 ARE THE NUMBERS OF THE NODES AT WHICH THE
C            ACTIVITIES WHICH BEGIN AT THE NODE IN FIELD 1 TERMINATE.
C         FIELD 101 IS A DUMMY FIELD AND SHOULD BE SET EQUAL TO 0 OR 1.
C         FIELD 102 INDICATES HOW MANY ACTIVITIES TERMINATE AT THE NODE
```

```
C          INDICATED IN FIELD NUMBER 1.
C       FIELD 103 INDICATES HOW MANY ACTIVITIES BEGIN AT THE NODE
INDI-
C          CATED IN FIELD NUMBER 1.
C     RECORD 1 CONTAINS FIELDS 1-26; RECORD 2 CONTAINS FIELDS 27-52;
C     RECORD 3 CONTAINS FIELDS 53-78; RECORD 4 CONTAINS FIELDS 79-103.
C
C
C           DATA FILE DATAH.PATHS
C
C     THIS DATA FILE CONTAINS DESCRIPTIONS OF THE PRECODED DISTRIBUTIONS
C     OF ACTIVITY DURATION.
C
C     THERE ARE 5 FIELDS OF DATA.
C     FIELD 1 IS THE CODE FOR THE TYPE OF DISTRIBUTION.
C         1 = TRIANGULAR DISTRIBUTION
C         2 = NORMAL DISTRIBUTION
C         3 = EXPONENTIAL DISTRIBUTION
C         4 = GAMMA DISTRIBUTION
C         5 = BETA DISTRIBUTION
C         6 = UNIFORM DISTRIBUTION
C     FIELD 2 IS
C         MODE FOR A TRIANGULAR DISTRIBUTION.
C         MEAN FOR A NORMAL DISTRIBUTION.
C         MEAN FOR AN EXPONENTIAL DISTRIBUTION.
C         ALPHA FOR A GAMMA OR A BETA DISTRIBUTION.
C         1/(B-A) FOR A UNIFORM DISTRIBUTION.
C     FIELD 3 IS BETA FOR A GAMMA OR A BETA DISTRIBUTION.
C     FIELD 4 IS THE MINIMUM VALUE OF THE DISTRIBUTION.
C     FIELD 5 IS THE MAXIMUM VALUE OF THE DISTRIBUTION.
C
C
C           DATA FILE CONTROL.PATHS
C
C     THIS IS A SINGLE LINE DATA FILE WHICH CONTAINS CONTROL
C     PARAMETERS FOR INPUT, OUTPUT, AND MONTE CARLO SIMULATION.
C
C     THERE ARE 3 FIELDS OF DATA.
C     FIELD 1 IS THE NUMBER OF NODES IN THE NETWORK.
C     FIELD 2 IS THE NUMBER OF ACTIVITIES IN THE NETWORK.
C     FIELD 3 IS THE DESIRED NUMBER OF PATHS IN THE SET OF K-MOST
C        STOCHASTICALLY DOMINATING PATHS (MAXIMUM = 5).
C
C
C     NOTE
C
C     ALL UNUSED FIELDS MUST BE ZEROED OUT.
C
C
C
C     **********************************************************
C
C                   M A I N    P R O G R A M
C
C     **********************************************************
C
```

```
      REAL*8 XINT(104,500,12),VALUE(104,500,10,3),A(130),
     *       ZVAL(130,5),DIST(20,5),
     *       CRTA(100,99),CRTNA(100,99),CRTN(100),
     *       X,XSIZE
      REAL*8 CONST,CRTNN,CUMCRT,PR1GE2,
     *       XD1,XD2,XD3,XD4,XD5
      INTEGER I,II,IACT,ICOUNT,IENODE,IFLAG,IPATH,IPRE,ISNODE,
     *        J,J1,J2,JJ,
     *        K,KK,
     *        L,L1,L2,L3,LASTK,
     *        MM,
     *        N,NACTS,NCL,NET,NETT,
     *        NPATH,NPATHS,NSIM,NSS
      DIMENSION NET(100,103),IPRE(100,99,2),IPATH(100,99,2),NPATH(100),
     *          NETT(103)
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      COMMON/PARA4/NET,IPRE
      DATA NCL/0/
C
C     OPEN INPUT AND OUTPUT FILES
C
      OPEN (UNIT = 11, FILE = 'datan.paths')
      OPEN (UNIT = 12, FILE = 'datah.paths')
      OPEN (UNIT = 13, FILE = 'control.paths')
C
C     READ CONTROL INFORMATION.
C
      READ (13,1900) N,NACTS,NPATHS
C
      DO 0910 I = 1,N
      READ (11,1901) (NETT(J), J = 1,103)
      L1 = NETT(1)
      DO 0900 K = 1,103
      NET(L1,K) = NETT(K)
 0900 CONTINUE
 0910 CONTINUE
C
C     DO 1010 READS DATA FROM DATAH AND LOADS THIS DATA INTO
C     THE VALUE AND XINT ARRAYS.  THIS DO ALSO DETERMINES IF
C     THE ACTIVITY DISTRIBUTION IS OTHER THAN UNIFORM, AND,
C     IF SO, CALLS LINEAR TO APPROXIMATE IT WITH A
C     PIECEWISE POLYGONAL FUNCTION.
C
      DO 1010 I = 1,NACTS
      READ (12,1902) L1,L2,XD1,XD2,XD3,XD4,XD5
      VALUE(L1,L2,1,3) = XD1
      VALUE(L1,L2,2,3) = XD2
      VALUE(L1,L2,3,3) = XD3
      XINT(L1,L2,1) = XD4
      XINT(L1,L2,2) = XD5
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 6) THEN
      CALL LINEAR(L1,L2,NCL)
      GO TO 1010
C
```

```
C     DO 1000 CONVERTS DATA FOR UNIFORM DISTRIBUTIONS INTO A USABLE
C     FORM FOR SUBROUTINES SERIES AND PARA.
C
      ELSE
      XINT(L1,L2,11) = XINT(L1,L2,2)
      X = XINT(L1,L2,1)
      XSIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      DO 1000 J = 1,10
      VALUE(L1,L2,J,1) = VALUE(L1,L2,2,3)
      XINT(L1,L2,J) = X
      X = X+XSIZE
 1000 CONTINUE
      END IF
 1010 CONTINUE
      PRINT 1910
      PRINT 1916
C
C     ANALYSIS OF THE NETWORK BEGINS.
C
C     DO 1070 DETERMINES THE SET OF PREDECESSOR ACTIVITIES, I.E. THE
C     STARTING NODE NUMBER AND THE ACTIVITY NUMBER OF EACH ACTIVITY
WHICH
C     TERMINATES AT NODE I.
C
      DO 1070 I = 2,N
      ICOUNT = 0
      DO 1060 J = 1,I-1
      IF (NET(J,101)) 1550,1030,1030
 1030 DO 1050 J1 = 2,NET(J,103)+1
      IF (NET(J,J1) - I) 1050,1040,1050
 1040 ICOUNT = ICOUNT+1
      IPRE(I,ICOUNT,1) = J
      IPRE(I,ICOUNT,2) = J1-1
 1050 CONTINUE
 1060 CONTINUE
 1070 CONTINUE
C
C     DO 1220 DETERMINES THE DISTRIBUTION THROUGH EACH NODE IN THE
C     FORWARD DIRECTION IN THE NETWORK.
C
      DO 1220 I = 2,N
C
C     THROUGH 1220 CONVOLVES THE RESOURCE CONSUMPTION DISTRIBUTION
C     THROUGH THE STARTING NODE OF THE ACTIVITY AND THE RESOURCE
C     CONSUMPTION DISTRIBUTION OF EACH ACTIVITY WHICH TERMINATES
C     AT NODE I AND THEN FINDS THE MAXIMUM OF THESE CONVOLUTIONS.
C
C     IF THE FIRST STARTING NODE = NODE 1, THE CONVOLUTION IS EQUAL TO
C     THE DISTRIBUTION OF THE ACTIVITY WHICH TERMINATES AT NODE I.
C
      IF (IPRE(I,1,1) .EQ. 1) THEN
      IACT = IPRE(I,1,2)
      DO 1175 J = 1,10
      XINT(101,1,J) = XINT(1,IACT,J)
      VALUE(101,1,J,1) = VALUE(1,IACT,J,1)
      VALUE(101,1,J,2) = VALUE(1,IACT,J,2)
 1175 CONTINUE
      XINT(101,1,11) = XINT(1,IACT,11)
C
C     OTHERWISE, LOAD THE DISTRIBUTION THROUGH THE FIRST STARTING NODE
C     INTO TEMPORARY LOCATION 1.
C
      ELSE
      ISNODE = IPRE(I,1,1)
      IACT = IPRE(I,1,2)
      DO 1180 J = 1,10
      XINT(101,1,J) = XINT(ISNODE,100,J)
      VALUE(101,1,J,1) = VALUE(ISNODE,100,J,1)
      VALUE(101,1,J,2) = VALUE(ISNODE,100,J,2)
 1180 CONTINUE
      XINT(101,1,11) = XINT(ISNODE,100,11)
C
C     LOAD THE DISTRIBUTION OF THE FIRST ACTIVITY TERMINATING AT NODE I
C     IN TEMPORARY LOCATION 2.
C
      DO 1185 J = 1,10
      XINT(101,2,J) = XINT(ISNODE,IACT,J)
      VALUE(101,2,J,1) = VALUE(ISNODE,IACT,J,1)
      VALUE(101,2,J,2) = VALUE(ISNODE,IACT,J,2)
 1185 CONTINUE
      XINT(101,2,11) = XINT(ISNODE,IACT,11)
C
C     CONVOLVE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2 AND
C     PLACE THE CONVOLUTION IN TEMPORARY LOCATION 1.
C
      CALL SERIES(101,1,101,2)
      END IF
      CONTINUE
C
C     IF THERE IS ONLY ONE ACTIVITY TERMINATING AT NODE I, THE
DISTRIBUTION
C     THROUGH NODE I IS THE CONVOLUTION IN TEMPORARY LOCATION 1.  LOAD
THIS
C     INTO THE 100TH ACTIVITY POSITION OF NODE I.
C
      IF (NET(I,102) .EQ. 1) THEN
      DO 1190 J = 1,10
      XINT(I,100,J) = XINT(101,1,J)
      VALUE(I,100,J,1) = VALUE(101,1,J,1)
      VALUE(I,100,J,2) = VALUE(101,1,J,2)
 1190 CONTINUE
      XINT(I,100,11) = XINT(101,1,11)
C
C     IF THERE ARE TWO OR MORE ACTIVITIES TERMINATING AT NODE I, LOAD
THE
C     DISTRIBUTION THROUGH THE STARTING NODE OF THE NEXT ACTIVITY INTO
C     TEMPORARY LOCATION 3.
C
      ELSE
      DO 1205 K = 2,NET(I,102)
      ISNODE = IPRE(I,K,1)
      IACT = IPRE(I,K,2)
      DO 1195 J = 1,10
      XINT(101,3,J) = XINT(ISNODE,100,J)
      VALUE(101,3,J,1) = VALUE(ISNODE,100,J,1)
      VALUE(101,3,J,2) = VALUE(ISNODE,100,J,2)
 1195 CONTINUE
      XINT(101,3,11) = XINT(ISNODE,100,11)
C
C     THEN LOAD THE DISTRIBUTION OF THE NEXT ACTIVITY INTO TEMPORARY
C     LOCATION 4.
C
      DO 1200 J = 1,10
      XINT(101,4,J) = XINT(ISNODE,IACT,J)
      VALUE(101,4,J,1) = VALUE(ISNODE,IACT,J,1)
      VALUE(101,4,J,2) = VALUE(ISNODE,IACT,J,2)
 1200 CONTINUE
      XINT(101,4,11) = XINT(ISNODE,IACT,11)
C
C     CONVOLUTE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 3 AND 4 AND
C     LOAD THE CONVOLUTION INTO TEMPORARY LOCATION 3.
C
      CALL SERIES(101,3,101,4)
C
C     PARALLEL-REDUCE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 3
AND
C     LOAD THE MAXIMUM INTO TEMPORARY LOCATION 1.
C
      CALL PARA(101,1,3)
 1205 CONTINUE
C
C     THE FORWARD DISTRIBUTION THROUGH NODE I IS THE MAXIMUM IN
TEMPORARY
C     LOCATION 1.  LOAD THIS INTO THE 100TH ACTIVITY POSITION OF NODE I.
C
      DO 1210 J = 1,10
      XINT(I,100,J) = XINT(101,1,J)
      VALUE(I,100,J,1) = VALUE(101,1,J,1)
      VALUE(I,100,J,2) = VALUE(101,1,J,2)
 1210 CONTINUE
      XINT(I,100,11) = XINT(101,1,11)
      END IF
 1220 CONTINUE
C
C     DO 1270 DETERMINES THE DISTRIBUTION THROUGH EACH NODE IN THE
C     BACKWARD DIRECTION IN THE NETWORK.
C
      DO 1270 I = N-1,1,-1
C
C     THROUGH 1270 CONVOLVES THE RESOURCE CONSUMPTION DISTRIBUTION
C     THROUGH THE ENDING NODE OF THE ACTIVITY AND THE RESOURCE
C     CONSUMPTION DISTRIBUTION OF EACH ACTIVITY WHICH STARTS
C     AT NODE I AND THEN FINDS THE MAXIMUM OF THESE CONVOLUTIONS.
C
C     IF THE LAST ENDING NODE = NODE N, THE CONVOLUTION IS EQUAL TO
C     THE DISTRIBUTION OF THE ACTIVITY WHICH TERMINATES AT NODE N.
C
      IACT = NET(I,103)
      IENODE = NET(I,IACT+1)
      IF (IENODE .EQ. N) THEN
      DO 1225 J = 1,10
      XINT(101,1,J) = XINT(I,IACT,J)
      VALUE(101,1,J,1) = VALUE(I,IACT,J,1)
      VALUE(101,1,J,2) = VALUE(I,IACT,J,2)
 1225 CONTINUE
      XINT(101,1,11) = XINT(I,IACT,11)
C
C     OTHERWISE, LOAD THE DISTRIBUTION THROUGH THE LAST ENDING NODE
C     INTO TEMPORARY LOCATION 1.
C
      ELSE
      DO 1230 J = 1,10
      XINT(101,1,J) = XINT(IENODE,101,J)
      VALUE(101,1,J,1) = VALUE(IENODE,101,J,1)
      VALUE(101,1,J,2) = VALUE(IENODE,101,J,2)
 1230 CONTINUE
      XINT(101,1,11) = XINT(IENODE,101,11)
C
C     LOAD THE DISTRIBUTION OF THE LAST ACTIVITY STARTING AT NODE I
C     IN TEMPORARY LOCATION 2.
C
      DO 1235 J = 1,10
      XINT(101,2,J) = XINT(I,IACT,J)
      VALUE(101,2,J,1) = VALUE(I,IACT,J,1)
      VALUE(101,2,J,2) = VALUE(I,IACT,J,2)
 1235 CONTINUE
      XINT(101,2,11) = XINT(I,IACT,11)
C
C     CONVOLVE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2 AND
C     PLACE THE CONVOLUTION IN TEMPORARY LOCATION 1.
C
      CALL SERIES(101,1,101,2)
      END IF
      CONTINUE
C
C     IF THERE IS ONLY ONE ACTIVITY STARTING AT NODE I, THE DISTRIBUTION
C     THROUGH NODE I IS THE CONVOLUTION IN TEMPORARY LOCATION 1.  LOAD
THIS
C     INTO THE 101ST ACTIVITY POSITION OF NODE I.
C
      IF (NET(I,103) .EQ. 1) THEN
      DO 1240 J = 1,10
      XINT(I,101,J) = XINT(101,1,J)
      VALUE(I,101,J,1) = VALUE(101,1,J,1)
      VALUE(I,101,J,2) = VALUE(101,1,J,2)
 1240 CONTINUE
      XINT(I,101,11) = XINT(101,1,11)
C
C     IF THERE ARE TWO OR MORE ACTIVITIES STARTING AT NODE I, LOAD THE
C     DISTRIBUTION THROUGH THE ENDING NODE OF THE NEXT ACTIVITY INTO
C     TEMPORARY LOCATION 3.
C
      ELSE
      DO 1255 K = NET(I,103)-1,1,-1
      IACT = K
      IENODE = NET(I,IACT)+1
      DO 1245 J = 1,10
      XINT(101,3,J) = XINT(IENODE,101,J)
      VALUE(101,3,J,1) = VALUE(IENODE,101,J,1)
      VALUE(101,3,J,2) = VALUE(IENODE,101,J,2)
 1245 CONTINUE
      XINT(101,3,11) = XINT(IENODE,101,11)
C
C     THEN LOAD THE DISTRIBUTION OF THE NEXT ACTIVITY INTO TEMPORARY
C     LOCATION 4.
C
      DO 1250 J = 1,10
      XINT(101,4,J) = XINT(I,IACT,J)
      VALUE(101,4,J,1) = VALUE(I,IACT,J,1)
      VALUE(101,4,J,2) = VALUE(I,IACT,J,2)
 1250 CONTINUE
      XINT(101,4,11) = XINT(I,IACT,11)
C
C     CONVOLUTE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 3 AND 4 AND
C     LOAD THE CONVOLUTION INTO TEMPORARY LOCATION 3.
C
      CALL SERIES(101,3,101,4)
C
C     PARALLEL-REDUCE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 3
AND
C     LOAD THE MAXIMUM INTO TEMPORARY LOCATION 1.
C
      CALL PARA(101,1,3)
 1255 CONTINUE
C
C     THE BACKWARD DISTRIBUTION THROUGH NODE I IS THE MAXIMUM IN
TEMPORARY
C     LOCATION 1.  LOAD THIS INTO THE 101ST ACTIVITY POSITION OF NODE I.
C
      DO 1260 J = 1,10
      XINT(I,101,J) = XINT(101,1,J)
      VALUE(I,101,J,1) = VALUE(101,1,J,1)
      VALUE(I,101,J,2) = VALUE(101,1,J,2)
 1260 CONTINUE
      XINT(I,101,11) = XINT(101,1,11)
      END IF
 1270 CONTINUE
C
C     DO 1275 INITIALIZES THE CRITICALITY INDICES OF THE NODES (CRTN).
C
      DO 1275 I = 1,N
      CRTN(I) = 0.0
 1275 CONTINUE
C
C     THROUGH 1415 CALCULATES THE CRITICALITY INDICES OF ALL THE
C     ACTIVITIES AND NODES IN THE NETWORK.
C
      DO 1415 I = N,2,-1
      IF (I .NE. N) GO TO 1280
      NPATH(I) = NET(I,102)
      KK = 1
      NSS = NET(I,102)
      GO TO 1295
C
C     AT NODE I, THE NUMBER OF PATHS TO BE CONSIDERED IS:
C         (NUMBER OF PATHS CONSIDERED AT NODE (I+1))
C       + (IN-DEGREE OF NODE I) - (OUT-DEGREE OF NODE I).
C
 1280 NPATH(I) = NPATH(I+1)+NET(I,102)-NET(I,103)
      NSS = NPATH(I+1)
      K = 0
C
C     DO 1290 SHIFTS TO NODE I ALL THE PATHS CONSIDERED AT NODE (I+1)
C     EXCEPT THOSE PATHS WHOSE PREDECESSOR ACTIVITIES START AT NODE I.
C
      DO 1290 JJ = 1,NSS
      ISNODE = IPATH(I+1,JJ,1)
      IACT = IPATH(I+1,JJ,2)
      IF (ISNODE .EQ. I) GO TO 1290
      K = K+1
      IPATH(I,K,1) = ISNODE
      IPATH(I,K,2) = IACT
      DO 1285 J = 1,10
      XINT(102,K,J) = XINT(102,JJ,J)
      VALUE(102,K,J,1) = VALUE(102,JJ,J,1)
      VALUE(102,K,J,2) = VALUE(102,JJ,J,2)
 1285 CONTINUE
      XINT(102,K,11) = XINT(102,JJ,11)
 1290 CONTINUE
      KK = K+1
      NSS = NPATH(I)
 1295 CONTINUE
C
C     DO 1320 DETERMINES THE DISTRIBUTIONS OF ALL PATHS WHICH INCLUDE
THE
C     (IN-DEGREE - OF - NODE I) PREDECESSOR ACTIVITIES OF NODE I.
C
      DO 1320 J = KK,NSS
      JJ = J-KK+1
      IPATH(I,J,1) = IPRE(I,JJ,1)
      IPATH(I,J,2) = IPRE(I,JJ,2)
C
C     LOAD THE FORWARD DISTRIBUTION THROUGH THE STARTING NODE OF THE
JJth
C     PREDECESSOR ACTIVITY OF NODE I INTO TEMPORARY LOCATION 1.
C
      ISNODE = IPATH(I,J,1)
      IACT = IPATH(I,J,2)
      DO 1300 K = 1,10
      XINT(101,1,K) = XINT(ISNODE,100,K)
      VALUE(101,1,K,1) = VALUE(ISNODE,100,K,1)
      VALUE(101,1,K,2) = VALUE(ISNODE,100,K,2)
 1300 CONTINUE
      XINT(101,1,11) = XINT(ISNODE,100,11)
C
C     LOAD THE DISTRIBUTION OF THE JJth PREDECESSOR ACTIVITY OF NODE I
```

```
C     INTO TEMPORARY LOCATION 2.
C
      DO 1305 K = 1,10
      XINT(101,2,K) = XINT(ISNODE,IACT,K)
      VALUE(101,2,K,1) = VALUE(ISNODE,IACT,K,1)
      VALUE(101,2,K,2) = VALUE(ISNODE,IACT,K,2)
 1305 CONTINUE
      XINT(101,2,11) = XINT(ISNODE,IACT,11)
C
C     CONVOLVE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2 AND
PLACE
C     THE CONVOLUTION IN TEMPORARY LOCATION 1.
C
C     IF THE STARTING NODE OF THE JJth PREDECESSOR ACTIVITY OF NODE I IS
C     THE STARTING NODE OF THE NETWORK, NODE 1, THE CONVOLUTION IS THE
C     DISTRIBUTION OF THE JJth PREDECESSOR ACTIVITY.
C
      IF (ISNODE .EQ.1) THEN
      DO 1306 K = 1,10
      XINT(101,1,K) = XINT(101,2,K)
      VALUE(101,1,K,1) = VALUE(101,2,K,1)
      VALUE(101,1,K,2) = VALUE(101,2,K,2)
 1306 CONTINUE
      XINT(101,1,11) = XINT(101,2,11)
      ELSE
      CALL SERIES(101,1,101,2)
      END IF
      CONTINUE
      IF (I .EQ. N) GO TO 1311
C
C     LOAD THE BACKWARD DISTRIBUTION THROUGH NODE I, THE ENDING NODE OF
C     THE JJth PREDECESSOR ACTIVITY TO NODE I, INTO TEMPORARY LOCATION
2.
C
      DO 1310 K = 1,10
      XINT(101,2,K) = XINT(I,101,K)
      VALUE(101,2,K,1) = VALUE(I,101,K,1)
      VALUE(101,2,K,2) = VALUE(I,101,K,2)
 1310 CONTINUE
      XINT(101,2,11) = XINT(I,101,11)
C
C     CONVOLVE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2 AND
PLACE
C     THE CONVOLUTION IN TEMPORARY LOCATION 1.
C
      CALL SERIES(101,1,101,2)
C
C     THE CONVOLUTION IS THE DISTRIBUTION OF ALL PATHS WHICH INCLUDE THE
C     JJth PREDECESSOR ACTIVITY OF NODE I.  LOAD THIS DISTRIBUTION INTO
THE
C     Jth ACTIVITY POSITION OF NODE 102.
C
 1311 DO 1315 K = 1,10
      XINT(102,J,K) = XINT(101,1,K)
      VALUE(102,J,K,1) = VALUE(101,1,K,1)
      VALUE(102,J,K,2) = VALUE(101,1,K,2)
 1315 CONTINUE
      XINT(102,J,11) = XINT(101,1,11)
 1320 CONTINUE
      IF (NET(I,102) .EQ. 1) GO TO 1390
C
C     TO 1360 DETERMINES THE MAXIMUM OF THE DISTRIBUTIONS OF ALL THE
C     PATHS CONSIDERED AT NODE (I+1) EXCEPT THOSE PATHS WHOSE
PREDECESSOR
C     ACTIVITIES START AT NODE I.  DO 1290 SHIFTED THESE PATHS TO NODE
I.
C
      IFLAG = 0
      IF (NPATH(I)-NET(I,102)-1) 1325,1335,1345
C
C     IF
C         NUMBER OF PATHS CONSIDERED AT NODE (I+1)
C       = NUMBER OF PREDECESSOR ACTIVITIES OF NODE I,
C     THE MAXIMUM IS THE 0 DISTRIBUTION.  SET A FLAG (IFLAG = 1).
C
 1325 IFLAG = 1
      GO TO 1360
C
C     IF
C         NUMBER OF PATHS CONSIDERED AT NODE (I+1)
C       = (NUMBER OF PREDECESSOR ACTIVITIES OF NODE I) + 1,
C     THE MAXIMUM IS THE DISTRIBUTION OF THE ONE PATH WHOSE PREDECESSOR
C     ACTIVITY DOES NOT START AT NODE I.  LOAD THIS DISTRIBUTION INTO
C     TEMPORARY LOCATION 3.
C
 1335 DO 1340 K = 1,10
      XINT(101,3,K) = XINT(102,1,K)
      VALUE(101,3,K,1) = VALUE(102,1,K,1)
      VALUE(101,3,K,2) = VALUE(102,1,K,2)
 1340 CONTINUE
      XINT(101,3,11) = XINT(102,1,11)
      GO TO 1360
C
C     IF
C         NUMBER OF PATHS CONSIDERED AT NODE (I+1)
C       > (NUMBER OF PREDECESSOR ACTIVITIES OF NODE I) + 1,
C     THERE ARE TWO OR MORE PATHS WHOSE PREDECESSOR ACTIVITIES DO NOT
START
C     AT NODE I.  DO 1350 DETERMINES THE MAXIMUM OF THE DISTRIBUTIONS OF
C     THESE PATHS.
C
C     DO 1347 LOADS THE 1st PATH WHOSE PREDECESSOR ACTIVITIES DO NOT
START
C     AT NODE I INTO TEMPORARY LOCATION 4.
C
 1345 DO 1347 K = 1,10
      XINT(101,4,K) = XINT(102,1,K)
      VALUE(101,4,K,1) = VALUE(102,1,K,1)
      VALUE(101,4,K,2) = VALUE(102,1,K,2)
 1347 CONTINUE
      XINT(101,4,11) = XINT(102,1,11)
      DO 1350 K = 2,NPATH(I)-NET(I,102)
      CALL PARA(102,1,K)
 1350 CONTINUE
C
C     LOAD THIS MAXIMUM DISTRIBUTION INTO TEMPORARY LOCATION 3.
C
      DO 1355 K = 1,10
      XINT(101,3,K) = XINT(102,1,K)
      VALUE(101,3,K,1) = VALUE(102,1,K,1)
      VALUE(101,3,K,2) = VALUE(102,1,K,2)
 1355 CONTINUE
      XINT(101,3,11) = XINT(102,1,11)
C
C     RELOAD THE 1st PATH WHOSE PREDECESSOR ACTIVITIES DO NOT START AT
C     NODE I BACK INTO THE 1st ACTIVITY POSITION OF NODE 102.
C
      DO 1357 K = 1,10
      XINT(102,1,K) = XINT(101,4,K)
      VALUE(102,1,K,1) = VALUE(101,4,K,1)
      VALUE(102,1,K,2) = VALUE(101,4,K,2)
 1357 CONTINUE
      XINT(102,1,11) = XINT(101,4,11)
C
C     DO 1380 DETERMINES
C         P(ALL PATHS WHICH INCLUDE THE Jth PREDECESSOR ACTIVITY >
C           ALL OTHER PATHS)
C     FOR EACH PREDECESSOR ACTIVITY (IACT) OF NODE I.  THIS PROBABILITY
C     IS THE CRITICALITY INDEX OF THE ACTIVITY (CRTA(I,IACT)).
C
 1360 DO 1380 J = NPATH(I)-NET(I,102)+1,NPATH(I)
C
C     LOAD THE MAXIMUM DISTRIBUTION OF ALL PATHS WHOSE PREDECESSOR
C     ACTIVITIES DO NOT START AT NODE I INTO TEMPORARY LOCATION 1.
C
      DO 1365 K = 1,10
      XINT(101,1,K) = XINT(101,3,K)
      VALUE(101,1,K,1) = VALUE(101,3,K,1)
      VALUE(101,1,K,2) = VALUE(101,3,K,2)
 1365 CONTINUE
      XINT(101,1,11) = XINT(101,3,11)
      DO 1375 K = NPATH(I)-NET(I,102)+1,NPATH(I)
      IF (K .EQ. J) GO TO 1375
C
C     LOAD THE DISTRIBUTIONS OF ALL PATHS WHICH INCLUDE THE Kth
PREDECES-
C     SOR ACTIVITY OF NODE I INTO TEMPORARY LOCATION 2, WHERE K IS DIF-
C     FERENT FROM J.
C
      DO 1370 KK = 1,10
      XINT(101,2,KK) = XINT(102,K,KK)
      VALUE(101,2,KK,1) = VALUE(102,K,KK,1)
      VALUE(101,2,KK,2) = VALUE(102,K,KK,2)
 1370 CONTINUE
      XINT(101,2,11) = XINT(102,K,11)
C
C     PARALLEL-REDUCE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2
AND
```

```
C     LOAD THE MAXIMUM INTO TEMPORARY LOCATION 1.
C
C     IF THE MAXIMUM DISTRIBUTION OF ALL PATHS WHOSE PREDECESSOR ACTIVI-
C     TIES DO NOT START AT NODE I IS THE 0 DISTRIBUTION, THE PARALLEL-
C     REDUCTION WITH THE DISTRIBUTION OF ALL PATHS WHICH INCLUDE THE 1st
C     PREDECESSOR ACTIVITY OF NODE I IS THE DISTRIBUTION OF THE LATTER.
C
      IF ((K .EQ. NPATH(I)-NET(I,102)+1) .AND. (IFLAG .EQ. 1)) THEN
      DO 1372 KK = 1,10
      XINT(101,1,KK) = XINT(101,2,KK)
      VALUE(101,1,KK,1) = VALUE(101,2,KK,1)
      VALUE(101,1,KK,2) = VALUE(101,2,KK,2)
 1372 CONTINUE
      XINT(101,1,11) = XINT(101,2,11)
      ELSE
      CALL PARA(101,1,2)
      END IF
 1375 CONTINUE
      CALL COMPAR(102,J,101,1,PR1GE2)
      IACT = J-(NPATH(I)-NET(I,102))
      CRTA(I,IACT) = PR1GE2
 1380 CONTINUE
C
C     THROUGH 1410 COMPUTES NORMALIZED CRITICALITY INDICES OF THE Jth
C     PREDECESSOR ACTIVITIES OF NODE I (CRTNA(I,J)) AND THE CRITICALITY
C     INDEX OF NODE I (CRTN(I)).
C
      IF (I .NE. N) GO TO 1400
      CUMCRT = 0.0
      DO 1385 J = 1,NET(N,102)
      CUMCRT = CUMCRT + CRTA(I,J)
 1385 CONTINUE
      CRTN(N) = CUMCRT
      CONST = CUMCRT
      IF (CONST .EQ. 0.0) CONST = 1.0
      GO TO 1400
 1390 IF (I .EQ. N) GO TO 1395
      CRTA(I,1) = CRTN(I)
      CRTNA(I,1) = CRTA(I,1)/CONST
      GO TO 1400
 1395 CRTN(I) = 1.0
      CRTA(I,1) = CRTN(I)
      CRTNA(I,1) = CRTA(I,1)
      CONST = 1.0
 1400 PRINT 1920,I
      PRINT 1925
      DO 1405 J = 1,NET(I,102)
      CRTNA(I,J) = CRTA(I,J)/CONST
      PRINT 1930,IPRE(I,J,1),CRTA(I,J),CRTNA(I,J)
 1405 CONTINUE
      DO 1410 J = 1,NET(I,102)
      ISNODE = IPRE(I,J,1)
      CRTN(ISNODE) = CRTN(ISNODE)+CRTA(I,J)
 1410 CONTINUE
 1415 CONTINUE
C
```

```
C     DO 1420 COMPUTES NORMALIZED CRITICALITY INDICES OF THE NODES
C     (CRTNN).
C
      PRINT 1935
      DO 1420 I = 1,N
      CRTNN = CRTN(I)/CONST
      PRINT 1930,I,CRTN(I),CRTNN
 1420 CONTINUE
C
C     DETERMINE THE NPATHS MOST STOCHASTICALLY DOMINATING PATHS THROUGH
THE
C     NETWORK.
C
      CALL DOMPTH(N,NPATHS)
      STOP
 1550 PRINT 1995
      STOP
C
C     FORMAT STATEMENTS
C
 1900 FORMAT (I3,1X,I4,1X,I1)
 1901 FORMAT (3(I2,25(1X,I2)/),I2,21(1X,I2),1X,I1,2(1X,I2))
 1902 FORMAT (2(I2,1X),F1.0,4(1X,F8.2))
 1910 FORMAT (1H1)
 1916 FORMAT (1X,'FROM THE POLYGONAL APPROXIMATION AND REDUCTION ',
     &'TECHNIQUE:')
 1920 FORMAT (/ 1X,'THE ACTIVITIES ENDING AT NODE ',I3,' AND THEIR ',
     &'CRITICALITY INDICES ARE:')
 1925 FORMAT (1X,24X,'NORMALIZED'/
     &         1X,'STARTING',2X,'CRITICALITY',2X,'CRITICALITY'/
     &         1X,2X,'NODE',7X,'INDEX',8X,'INDEX')
 1930 FORMAT (1X,2X,I3,7X,F7.5,6X,F7.5)
 1935 FORMAT (/ 1X,'THE CRITICALITY INDICES OF THE NODES ARE:'/
     &           1X,23X,'NORMALIZED'/
     &           1X,10X,'CRITICALITY',2X,'CRITICALITY'/
     &           1X,2X,'NODE',7X,'INDEX',8X,'INDEX')
 1995 FORMAT (1X,'PROGRAM STOPPED' / 1X,'IMPROPER NODE NUMBER(S) '
     &,'ENCOUNTERED')
 1996 FORMAT (// 1X,'CPU TIME FOR PART PROCESSING IS ',F7.2,' SECONDS'
     &//)
      END
C     END MAIN PROGRAM
C
C     ***********************************************************
C
C                S U B R O U T I N E          P A R A
C
C     ***********************************************************
C
      SUBROUTINE PARA (L1,L2,L3)
      REAL*8 VALUE(104,500,10,3),XINT(104,500,12)
      REAL*8 XVAL,ZVAL(130,5),PAR(2,15,6),FACT,B(130)
      REAL*4 Z
      INTEGER L1,L2,L3,NV1,NV2
      INTEGER K4(2,30)
      INTEGER I,IINT,N,NCL,J,K,K3,L6,LASTJ,LASTK
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/B
C
C     SUBROUTINE PARA IS USED TO REDUCE PARALLEL ARCS INTO A SINGLE
C     EQUIVALENT ARC.  IT FINDS THE MAX OPERATOR BY MULTIPLYING CAP
C     F(X) AGAINST CAP G(X) OVER THE INTERVALS OF VALIDITY.
C
      NV1 = 10
      NV2 = 10
      DO 2020 N = 1,2
      L6 = L2
      IF (N .EQ. 2) L6 = L3
      FACT = 0
      DO 2010 J = 1,10
      B(1) = XINT(L1,L6,J)
C
C     DO 2000 CONVERTS EACH LINEAR POLYNOMIAL PIECE OF LITTLE F(X)
C     INTO THE CORRESPONDING QUADRATIC POLYNOMIAL PIECE OF ITS
C     CUMULATIVE DISTRIBUTION CAP F(X).
C
      DO 2000 I = 1,2
      XVAL = VALUE(L1,L6,J,I)
      Z = FLOAT(I)
      PAR(N,J,I+1) = XVAL/Z
      PAR(N,J,1) = PAR(N,J,1)+((-1.0)*(XVAL/Z)*(B(1)**I))
      K4(N,J) = I+1
 2000 CONTINUE
      IF (J .GT. 1) PAR(N,J,1) = PAR(N,J,1)+FACT
      FACT = PAR(N,J,1)+(PAR(N,J,2)*XINT(L1,L6,J+1))+(PAR(N,J,3)
     &*(XINT(L1,L6,J+1)**2))
 2010 CONTINUE
 2020 CONTINUE
C
C     DO 2040 ASSIGNS INTERVAL BOUNDARY VALUES TO THE B ARRAY.
C
      DO 2040 I = 1,22
      IF (I .GT. 11) GO TO 2030
      B(I) = XINT(L1,L2,I)
      GO TO 2040
 2030 B(I) = XINT(L1,L3,I-11)
 2040 CONTINUE
      NCL = 21
      CALL SORT(NCL)
C
C     DO 2080 DETERMINES THE POINT AT WHICH THE DISTRIBUTION DOMAINS
C     OF THE TWO ARCS BEING COMBINED OVERLAP.  ONCE THIS POINT IS
C     DETERMINED, THE B ARRAY IS ADJUSTED TO REFLECT THE OVERLAP
C     (ALL VALUES LESS THAN THIS POINT OF FIRST OVERLAP NEED NOT BE
C     CONSIDERED, BECAUSE ONE OF THE DISTRIBUTIONS EQUALS ZERO AT
C     THESE VALUES). IF THE DOMAINS ARE DISJOINT OR OVERLAP AT ONLY.
C     ONE BOUNDARY POINT, THE RESULT OF THE APPLICATION OF THE
C     MAXIMUM OPERATOR IS JUST THE UNCHANGED APPROXIMATED PROBABILITY
C     DENSITY FUNCTION OF THE DISTRIBUTION DEFINED ON THE HIGHER-
C     VALUED DOMAIN.  GO TO 2180 OR GO TO 2160 RETURNS THIS FUNCTION
C     DIRECTLY WITHOUT FURTHER PROCESSING.
```

```
C
      IINT = 0
      LASTJ = NCL+1
      DO 2080 J = 1,LASTJ
      IF ((XINT(L1,L2,1) .GE. XINT(L1,L3,1)-0.001) .AND.
     &(XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001)) GO TO 2080
      IF (IINT .GE. 1) GO TO 2060
      IF (XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001) GO TO 2050
      IF (XINT(L1,L3,J+1) .GE. XINT(L1,L2,1)-0.001) IINT = J
      IF ((XINT(L1,L3,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L2,1) .GE. XINT(L1,L3,J+1)-0.001)
     &.AND. (XINT(L1,L2,1) .LE. XINT(L1,L3,J+1)+0.001)
     &.AND. (XINT(L1,L3,J+2) .LE. 0.001))) GO TO 2180
      GO TO 2080
 2050 IF (XINT(L1,L2,J+1) .GE. XINT(L1,L3,1)-0.001) IINT = J
      IF ((XINT(L1,L2,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L3,1) .GE. XINT(L1,L2,J+1)-0.001)
     &.AND. (XINT(L1,L3,1) .LE. XINT(L1,L2,J+1)+0.001)
     &.AND. (XINT(L1,L2,J+2) .LE. 0.001))) GO TO 2160
      GO TO 2080
 2060 LASTK = NCL-(IINT-1)
      DO 2070 K = 1,LASTK
      B(K) = B(K+IINT)
      B(K+IINT) = 0
 2070 CONTINUE
      GO TO 2090
 2080 CONTINUE
 2090 NCL = NCL-IINT
C
C     DO 2150 IS THE OUTER LOOP FOR THE PROCESS OF CREATING THE
C     EQUIVALENT ARC.  NCL IS THE NUMBER OF CLASSES INVOLVED
C     BETWEEN THE TWO ARCS.
C
      N1 = 0
      N2 = 0
      DO 2150 I = 1,NCL
      DO 2110 J = 1,11
C
C     DO 2110 DETERMINES THE APPROPRIATE INTERVALS OF EACH DISTRIBUTION
C     THAT ARE VALID FOR THE B(I) VALUE BEING CONSIDERED.  N1 AND
C     N2 ARE THE CONTROLS FOR UPPER AND LOWER ARCS RESPECTIVELY.
C
      IF (N1 .GE. 1) GO TO 2100
      IF (((B(I) .GE. XINT(L1,L2,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L2,J+1)+0.001)) .OR. (XINT(L1,L2,J+1) .LE. 0.001))
     &N1 = J
 2100 CONTINUE
      IF (N2 .GE. 1) GO TO 2110
      IF (((B(I) .GE. XINT(L1,L3,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L3,J+1)+0.001)) .OR. (XINT(L1,L3,J+1) .LE. 0.001))
     &N2 = J
 2110 CONTINUE
      IF (N2 .GT. NV2) K4(2,N2) = 1
      IF (N1 .GT. NV1) K4(1,N1) = 1
C
C     DO 2130 AND DO 2120 PERFORM THE POLYGONAL MULTIPLICATION FOR
```

```
C     CAP F(X) AND CAP G(X).
C
      LASTJ = K4(2,N2)
      LASTK = K4(1,N1)
      DO 2130 J = 1,LASTJ
      DO 2120 K = 1,LASTK
      IF (N2 .GT. NV2) PAR(2,N2,J) =  1
      IF (N1 .GT. NV1) PAR(1,N1,K) =  1
      K3 = J+K-1
      ZVAL(I,K3) = ZVAL(I,K3)+(PAR(1,N1,K)*PAR(2,N2,J))
 2120 CONTINUE
 2130 CONTINUE
C
C     DO 2140 OBTAINS THE FIRST DERIVATIVE OF THE RESULT OF THE
C     MULTIPLICATION OF CAP F(X) AND CAP G(X) IN THE FORM OF A
C     LITTLE H(X) FOR THAT PRODUCT.
C
      DO 2140 J = 1,4
      ZVAL(I,J) = ZVAL(I,J+1)*FLOAT(J)
      ZVAL(I,J+1) = 0
 2140 CONTINUE
      N1 = 0
      N2 = 0
 2150 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE RESULTS OF THE
C     PARALLEL REDUCTION WITH THE B(O) AND B(1) FORM IN EACH OF 10
C     CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR(L1,L2,NCL)
      GO TO 2180
 2160 DO 2170 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L3,I,1)
      VALUE(L1,L2,I,2) = VALUE(L1,L3,I,2)
      XINT(L1,L2,I) = XINT(L1,L3,I)
 2170 CONTINUE
      XINT(L1,L2,11) = XINT(L1,L3,11)
 2180 VALUE(L1,L2,1,3) = 0
      DO 2210 I = 1,2
      DO 2200 J = 1,10
      DO 2190 K = 1,3
      PAR(I,J,K) = 0
 2190 CONTINUE
 2200 CONTINUE
 2210 CONTINUE
      RETURN
      END
C     END SUBROUTINE PARA
C
C     ***********************************************************
C
C                S U B R O U T I N E      S E R I E S
C
C     ***********************************************************
C
```

```
      SUBROUTINE SERIES (L1,L2,L3,L4)
      REAL*8 VALUE(104,500,10,3),XINT(104,500,12)
      REAL*8 ZVAL(130,5),XLIM(2),A(130)
      REAL*8 F0,F1,G0,G1,XL
      INTEGER L1,L2,L3,L4
      INTEGER ISEL(2)
      INTEGER I,IK,J,K,NCL,NCL1,NE
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
C
C     SUBROUTINE SERIES PERFORMS THE CONVOLUTION OF TWO PROBABILITY
C     DISTRIBUTIONS BY INTEGRATING THE PRODUCT OF THEIR PIECEWISE
C     POLYGONAL APPROXIMATIONS IN THE FORMS OF F(X) AND G(T-X).
C
C     THIS SECTION DETERMINES THE INTERVALS OF VALIDITY FOR THE
C     CONVOLUTION.
C
C     THE A ARRAY IS USED FOR THE SAME PURPOSE AS THE B ARRAY IN PARA.
C
      K = 0
C
C     DO 3010 CREATES ALL POSSIBLE INTERVALS OF THE NEW DISTRIBUTION
C     BY ADDING THE INTERVALS OF THE TWO DISTRIBUTIONS BEING WORKED.
C
      DO 3010 I = 1,12
      IF ((XINT(L3,L4,I) .LE. 0).AND.(I .GT. 1)) GO TO 3020
      DO 3000 J = 1,12
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) NCL1 = J-2
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) GO TO 3010
      K = K+1
      A(K) = XINT(L1,L2,J)+XINT(L3,L4,I)
 3000 CONTINUE
 3010 CONTINUE
 3020 NINT = I-2
      NCL = K-1
C
C     DO 3120 IS CONTROLLED BY THE NUMBER OF CLASSES IN THE F(X)
C     DISTRIBUTION.  DO 3110 IS CONTROLLED BY THE NUMBER OF CLASSES
C     CREATED BY COMBINING F(X) AND G(T-X).  DO 3100 IS CONTROLLED
C     BY THE NUMBER OF CLASSES IN THE G(T-X) DISTRIBUTION.  THIS
C     ALLOWS THE EVALUATION OF ALL OF THE CREATED CLASSES FOR EVERY
C     CLASS IN BOTH DISTRIBUTIONS.
C
      CALL SORT(NCL)
      DO 3120 K = 1,NCL1
      DO 3110 I = 1,NCL
      DO 3100 J = 1,NINT
      IK = 0
C
C     THIS IF STATEMENT DETERMINES WHICH INTERVALS ARE VALID FOR THE
C     INTERVAL END POINT A(I) BEING EVALUATED AND FOR THE VALUE OF K
C     BEING CONTROLLED BY DO 3120.
C
      IF ((A(I) .GE. XINT(L1,L2,K)+XINT(L3,L4,J)-0.001) .AND. (A(I+1)
     &.LE. XINT(L1,L2,K+1)+XINT(L3,L4,J+1)+0.001)) IK = J
```

```
      IF (IK .GE. 1) GO TO 3030
      GO TO 3100
 3030 ISEL(1) = 0
      ISEL(2) = 0
C
C     THE IF STATEMENTS INVOLVING XLIM ARE USED TO DETERMINE THE
C     UPPER AND LOWER LIMITS OF INTEGRATION.  IT IS DETERMINED WHETHER
C     THE LIMIT COMES FROM THE F(X) OR THE G(T-X) DISTRIBUTION.  ISEL
C     IS USED TO DESIGNATE VALUES FROM THE G(T-X) DISTRIBUTION.
C
      IF (XINT(L1,L2,K) .GE. (A(I+1)-XINT(L3,L4,J+1)-0.001)) GO TO 3040
      XLIM(1) = XINT(L3,L4,J+1)
      ISEL(1) = 999
      GO TO 3050
 3040 XLIM(1) = XINT(L1,L2,K)
 3050 IF (XINT(L1,L2,K+1) .LE. (A(I)-XINT(L3,L4,J)+0.001)) GO TO 3060
      XLIM(2) = XINT(L3,L4,J)
      ISEL(2) = 999
      GO TO 3070
 3060 XLIM(2) = XINT(L1,L2,K+1)
 3070 CONTINUE
      DO 3090 NE = 1,2
      F0 = VALUE(L1,L2,K,1)
      F1 = VALUE(L1,L2,K,2)
      G0 = VALUE(L3,L4,IK,1)
      G1 = VALUE(L3,L4,IK,2)
      XL = XLIM(NE)
      Z = 1.0
      IF (NE .EQ. 1) Z = -1.0
      IF (ISEL(NE) .EQ. 999) GO TO 3080
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL AT A FINITE
C     LIMIT.  THE INTEGRATION IS BROKEN DOWN INTO ITS COMPONENT PARTS
C     BY THE POWER OF THE COEFFICIENT THAT RESULTS.  Z CONTROLS THE
C     SIGN OF THE INTEGRAL BASED ON WHETHER THE LOWER OR UPPER LIMIT
C     IS BEING EVALUATED.
C
      ZVAL(I,1) = ZVAL(I,1)+((F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((-1.0*F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+(((F1*G1*XL**2)/2.)+(F0*G1*XL))*Z
      ZVAL(I,3) = ZVAL(I,3)+((-1.0*F0*G1)/2.)*Z
      GO TO 3090
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL FOR LIMITS.
C     IN THE FORM OF (T-X).  THE FORMULAS ARE DIFFERENT BECAUSE
C     OF THE DIFFERENT POLYNOMIAL CREATED WHEN THE INTEGRATION
C     INVOLVES LIMITS IN THE FORM OF (T-X).
C
 3080 ZVAL(I,1) = ZVAL(I,1)+((-1.0*F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+((-1.0*F1*G0*XL)+(F0*G0)-
     &((F1*G1*XL**2)/2.))*Z
      ZVAL(I,3) = ZVAL(I,3)+((F1*G0)/2.)*Z
      ZVAL(I,4) = ZVAL(I,4)+((F1*G1)/6.)*Z
 3090 CONTINUE
 3100 CONTINUE
 3110 CONTINUE
 3120 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE CONVOLUTION
C     RESULTS WITH THE B(0) AND B(1) FORM IN EACH OF 10 CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR(L1,L2,NCL)
      RETURN
      END
C     END SUBROUTINE SERIES
C
C     ***********************************************************
C
C                S U B R O U T I N E       L I N E A R
C
C     ***********************************************************
C
      SUBROUTINE LINEAR (L1,L2,NCL)
      REAL*8 VALUE(104,500,10,3),XINT(104,500,12),ZVAL(130,5),A(130)
      REAL*8 Q,Q1,Q2,STD,SUMX,SUMY,SUMXY,SUMSQ
      REAL*8 ALPHA,AREA,BETA,FACT,SIZE,W,X,XLMBDA,XMEAN
      REAL*8 XMODE,XSIZE,Y
      INTEGER L1,L2
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      EXTERNAL DGAMMA
C
C     SUBROUTINE LINEAR PIECEWISE POLYGONALIZES DISTRIBUTION DATA
C     FROM THE MAIN PROGRAM AND SUBROUTINES PARA AND SERIES WITH
C     THE B(O) AND B(1) FORM IN EACH OF 10 CLASSES THROUGH THE USE
C     OF SIMPLE LINEAR REGRESSION.
C
      XMODE = VALUE(L1,L2,2,3)
      XMEAN = VALUE(L1,L2,2,3)
      STD = ((VALUE(L1,L2,2,3)-XINT(L1,L2,1))/3.)
      XLMBDA = VALUE(L1,L2,2,3)-XINT(L1,L2,1)
      ALPHA = VALUE(L1,L2,2,3)
      BETA = VALUE(L1,L2,3,3)
      SIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) SIZE = (A(NCL+1)-A(1))/10.
      XINT(L1,L2,11) = XINT(L1,L2,2)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) XINT(L1,L2,11) = A(NCL+1)
      X = XINT(L1,L2,1)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) X = A(1)
      DO 5000 I = 1,10
      XINT(L1,L2,I) = X
      X = X+SIZE
 5000 CONTINUE
      DO 5050 I = 1,10
      X = XINT(L1,L2,I)
      SUMY = 0.
      SUMX = 0.
      SUMXY = 0.
      SUMSQ = 0.
```

```
C
C     W CONTROLS THE NUMBER OF DATA POINTS USED IN THE REGRESSION
C     COMPUTATIONS.
C
      W = 10.+IDINT(SIZE*3.)
      XSIZE = SIZE/W
      LASTJ = IDINT(W)
      DO 5040 J = 1,LASTJ
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 99)  GO TO 5030
      DO 5010 K3 = 1,NCL
      K  = 0
      IF ((X .GE. A(K3)).AND.(X .LE. A(K3+1))) K = K3
      IF (K .GE. 1) GO TO 5020
 5010 CONTINUE
C
C     SERIES OR PARA GENERATED DISTRIBUTIONS.
C
 5020 Y = ZVAL(K,1)+(ZVAL(K,2)*X)+(ZVAL(K,3)*(X**2))
     &+(ZVAL(K,4)*(X**3))
 5030 CONTINUE
C
C     TRIANGULAR DISTRIBUTION.
C
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 1) THEN
         IF (XINT(L1,L2,1) .LE. X .AND. X .LE. XMODE) THEN
         Y = (2.*(X-XINT(L1,L2,1)))/((XMODE-XINT(L1,L2,1))*10.*SIZE)
         ELSE
         Y = (2.*(XINT(L1,L2,11)-X))/((XINT(L1,L2,11)-XMODE)*10.*SIZE)
         END IF
C
C     NORMAL  DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 2) THEN
      Y = (1./(STD*2.506628275))*(DEXP((-1.0)*(((X-XMEAN)/STD)**2)/2.))
C
C     EXPONENTIAL DISTRIBUTION  (SHIFTED).
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 3) THEN
      Y = (1./XLMBDA)*(DEXP((-1.0)*((X-XINT(L1,L2,1))/XLMBDA)))
C
C     GAMMA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 4) THEN
      Y = (1./(DGAMMA(ALPHA)*(BETA**ALPHA)))*DEXP(-X/BETA)*(X**(ALPHA-1.
     &))
C
C     BETA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 5) THEN
      Y = (DGAMMA(ALPHA+BETA)/(DGAMMA(ALPHA)*DGAMMA(BETA)))*
     &(1./(10.*SIZE)**(ALPHA+BETA-2.))*
     &((X-XINT(L1,L2,1))**(ALPHA-1.))*
     &((XINT(L1,L2,11)-X)**(BETA-1.))
      END IF
      IF (Y .LT. 0) Y = 0
      SUMX = SUMX+X
      SUMY = SUMY+Y
      SUMXY = SUMXY+(X*Y)
      SUMSQ = SUMSQ+(X**2)
      X = X+XSIZE
 5040 CONTINUE
      VALUE(L1,L2,I,2) = (SUMXY-((SUMX*SUMY)/W))/(SUMSQ-((SUMX**2)/W))
      VALUE(L1,L2,I,1) = (SUMY/W)-(VALUE(L1,L2,I,2)*(SUMX/W))
 5050 CONTINUE
C
C     DO 5060 CALCULATES THE AREA UNDER THE APPROXIMATED DISTRIBUTION.
C     AN ADJUSTMENT FACTOR FOR THE AMOUNT THAT THIS AREA HAS BEEN
C     UNDERESTIMATED OR OVERESTIMATED IS THEREBY DETERMINED.
C
      DO 5060 I = 1,10
      Q  = XINT(L1,L2,I+1)-XINT(L1,L2,I)
      Q1 = (XINT(L1,L2,I)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      Q2 = (XINT(L1,L2,I+1)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      IF (Q1 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q1*(-1.0))
      IF (Q2 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q2*(-1.0))
      IF (Q1 .LT. 0.) Q1 = 0.
      IF (Q2 .LT. 0.) Q2 = 0.
      AREA = AREA+((Q1+Q2)*Q*0.5)
 5060 CONTINUE
      FACT = 1.0/AREA
C
C     DO 5070 ADJUSTS THE COEFFICIENTS OF ALL THE LINEAR POLYNOMIAL
C     PIECES BY THE FACTOR COMPUTED IN DO 5060 IN ORDER TO NORMALIZE
C     THE AREA BACK TO ONE.  THIS ACTS TO REDUCE ACCUMULATING ERRORS
C     DURING PROGRAM COMPUTATIONS.
C
      DO 5070 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)*FACT
      VALUE(L1,L2,I,2) = VALUE(L1,L2,I,2)*FACT
 5070 CONTINUE
      AREA = 0
      DO 5080  I = 1,130
      A(I)  = 0
      ZVAL(I,1) = 0
      ZVAL(I,2) = 0
      ZVAL(I,3) = 0
      ZVAL(I,4) = 0
 5080 CONTINUE
      RETURN
      END
C     END SUBROUTINE LINEAR
C
C     ************************************************************
C
C               S U B R O U T I N E          S O R T
C
C     ************************************************************
C
      SUBROUTINE SORT (NCL)
      REAL*8 A(130),B
      INTEGER NCL
      INTEGER I,K1
      COMMON/PARA3/A
C
C     SUBROUTINE SORT IS USED TO CONDUCT AN ALGEBRAIC SORT OF DATA
C     CREATED IN THE SERIES AND PARA SUBROUTINES.
C
 6000 K1  = 0
      DO 6010 I = 1,NCL
      IF ((A(I) .LT. (A(I+1)+.01)).AND.(A(I) .GT. (A(I+1)-.01)))
     &GO TO 6020
      IF (A(I) .LT. A(I+1)) GO TO 6010
      IF (A(I) .GT. A(I+1)) B = A(I)
      A(I) = A(I+1)
      A(I+1) = B
      K1=K1+1
 6010 CONTINUE
      IF (K1 .GE. 1) GO TO 6000
      GO TO 6040
 6020 NCL = NCL-1
      LASTJ = NCL+1
      DO 6030 J = I,LASTJ
      A(J) = A(J+1)
      A(J+1) = 0
 6030 CONTINUE
      GO TO 6000
 6040 RETURN
      END
C     END SUBROUTINE SORT
C
C     ************************************************************
C
C                 S U B R O U T I N E       C O M P A R
C
C     ************************************************************
C
      SUBROUTINE COMPAR(L1,L2,L3,L4,PR1GE2)
      REAL*8 XINT(104,500,12),VALUE(104,500,10,3),
     *       B0X,B1X,B0Y,B1Y,C1,C2,C3,C4,CELL,PR1GE2,VOLUME
     *       XLOWER,XUPPER,YLOWER,YUPPER
      INTEGER I,J,L1,L2,L3,L4
      COMMON/PARA1/XINT,VALUE
C
C     SUBROUTINE COMPAR COMPUTES P(X > OR = Y), WHERE X IS THE DISTRIBU-
C     TION OF NODE L1, ACTIVITY L2, AND Y IS THE DISTRIBUTION OF NODE
L3,
C     ACTIVITY L4.
C
C     IF Y(11) < OR = X(1), THEN P(X > OR = Y) = 1.  RETURN PR1GE2 =
1.0.
C
      IF (XINT(L3,L4,11) .LE. XINT(L1,L2,1)) THEN
      PR1GE2 = 1.0
      RETURN
C
C     IF X(11) < OR = Y(1), THEN P(X > OR = Y) = 0.  RETURN PRIGE2 =
0.0.
C
```

```
      ELSE IF (XINT(L1,L2,11) .LE. XINT(L3,L4,1)) THEN
      PR1GE2 = 0.0
      RETURN
      END IF
C
C     INITIALIZE P(X > OR = Y) [PR1GE2].
C
      PR1GE2 = 0.0
C
C     DO 9010 COMPUTES THE CONTRIBUTION TO P(X > OR = Y) IN EACH CELL
C     [X(I),X(I+1)] x [Y(J),Y(J+1)] OF THE JOINT DISTRIBUTION OF X AND Y
C     AND SUMS THESE CONTRIBUTIONS IN PR1GE2.
C
      DO 9010 I = 1,10
      DO 9000 J = 1,10
C
C     IF X(I+1) < OR = Y(J), THE CELL LIES COMPLETELY ABOVE THE LINE X =
Y,
C     SO THE CELL'S CONTRIBUTION TO P(X > OR = Y) IS 0.
C
      IF (XINT(L1,L2,I+1) .LE. XINT(L3,L4,J)) GO TO 9000
      B0X = VALUE(L1,L2,I,1)
      B1X = VALUE(L1,L2,I,2)
      B0Y = VALUE(L3,L4,J,1)
      B1Y = VALUE(L3,L4,J,2)
      XLOWER = XINT(L1,L2,I)
      XUPPER = XINT(L1,L2,I+1)
      YLOWER = XINT(L3,L4,J)
      YUPPER = XINT(L3,L4,J+1)
C
C     IF Y(J+1) < OR = X(I), THE CELL LIES COMPLETELY BELOW THE LINE X =
Y.
C
      IF (YUPPER .LE. XLOWER) THEN
      CELL =  (B0X*(XUPPER-XLOWER)+(B1X/2.0)*(XUPPER**2-XLOWER**2))
     *       *(B0Y*(YUPPER-YLOWER)+(B1Y/2.0)*(YUPPER**2-YLOWER**2))
      PR1GE2 = PR1GE2+CELL
      GO TO 9000
      ELSE
      C1 = -(B0X*(B0Y*YLOWER+(B1Y/2.0)*YLOWER**2))
      C2 = (B0X*B0Y/2.0)-(B1X/2.0)*(B0Y*YLOWER+(B1Y/2.0)*YLOWER**2)
      C3 = (B1X*B0Y/3.0)+(B0X*B1Y/6.0)
      C4 = B1X*B1Y/8.0
      END IF
C
C     IF Y(J) < X(I) < Y(J+1) < X(I+1), THE LINE X = Y PASSES THROUGH
THE
C     LEFT SIDE AND THE TOP OF THE CELL.
C
      IF ((YLOWER .LT. XLOWER) .AND. (XLOWER. LT. YUPPER) .AND.
     *    (YUPPER .LT. XUPPER)) THEN
      CELL =  C1*(YUPPER-XLOWER)
     *       +C2*(YUPPER**2-XLOWER**2)
     *       +C3*(YUPPER**3-XLOWER**3)
     *       +C4*(YUPPER**4-XLOWER**4)
     *       +(B0X*(XUPPER-YUPPER)+(B1X/2.0)*(XUPPER**2-YUPPER**2))
     *       *(B0Y*(YUPPER-YLOWER)+(B1Y/2.0)*(YUPPER**2-YLOWER**2))
      PR1GE2 = PR1GE2+CELL
      GO TO 9000
C
C     IF X(I) < OR = Y(J) AND Y(J+1) < X(I+1),THE LINE X = Y PASSES
C     THROUGH THE BOTTOM AND THE TOP OF THE CELL.
C
      ELSE IF ((XLOWER .LE. YLOWER) .AND. (YUPPER .LT. XUPPER)) THEN
      CELL =  C1*(YUPPER-YLOWER)
     *       +C2*(YUPPER**2-YLOWER**2)
     *       +C3*(YUPPER**3-YLOWER**3)
     *       +C4*(YUPPER**4-YLOWER**4)
     *       +(B0X*(XUPPER-YUPPER)+(B1X/2.0)*(XUPPER**2-YUPPER**2))
     *       *(B0Y*(YUPPER-YLOWER)+(B1Y/2.0)*(YUPPER**2-YLOWER**2))
      PR1GE2 = PR1GE2+CELL
      GO TO 9000
C
C     IF X(I) < OR = Y(J) < X(I+1) < OR = Y(J+1), THE LINE X = Y PASSES
C     THROUGH THE BOTTOM AND THE RIGHT SIDE OF THE CELL OR THROUGH THE
C     LOWER-LEFT AND UPPER-RIGHT CORNERS OF THE CELL.
C
      ELSE IF ((XLOWER .LE. YLOWER) .AND. (YLOWER .LT. XUPPER) .AND.
     *         (XUPPER .LE. YUPPER)) THEN
      CELL =  C1*(XUPPER-YLOWER)
     *       +C2*(XUPPER**2-YLOWER**2)
     *       +C3*(XUPPER**3-YLOWER**3)
     *       +C4*(XUPPER**4-YLOWER**4)
      PR1GE2 = PR1GE2+CELL
      GO TO 9000
C
C     IF Y(J) < X(I) AND X(I+1) < Y(J+1), THE LINE X = Y PASSES THROUGH
C     BOTH SIDES OF THE CELL.
C
      ELSE IF ((YLOWER .LT. XLOWER) .AND. (XUPPER .LT. YUPPER)) THEN
      CELL =  C1*(XUPPER-XLOWER)
     *       +C2*(XUPPER**2-XLOWER**2)
     *       +C3*(XUPPER**3-XLOWER**3)
     *       +C4*(XUPPER**4-XLOWER**4)
      PR1GE2 = PR1GE2+CELL
      GO TO 9000
      END IF
 9000 CONTINUE
 9010 CONTINUE
C
C     DO 9030 COMPUTES THE VOLUME UNDER THE APPROXIMATED JOINT DISTRIBU-
C     TION OF X AND Y.  AN ADJUSTMENT FOR THE AMOUNT THAT THIS VOLUME
C     HAS BEEN UNDERESTIMATED OR OVERESTIMATED IS THEN MADE TO PR1GE2.
C
      VOLUME = 0.0
      DO 9030 I = 1,10
      DO 9020 J = 1,10
      B0X = VALUE(L1,L2,I,1)
      B1X = VALUE(L1,L2,I,2)
      B0Y = VALUE(L3,L4,J,1)
      B1Y = VALUE(L3,L4,J,2)
      XLOWER = XINT(L1,L2,I)
      XUPPER = XINT(L1,L2,I+1)
      YLOWER = XINT(L3,L4,J)
      YUPPER = XINT(L3,L4,J+1)
      CELL =  (B0X*(XUPPER-XLOWER)+(B1X/2.0)*(XUPPER**2-XLOWER**2))
     *       *(B0Y*(YUPPER-YLOWER)+(B1Y/2.0)*(YUPPER**2-YLOWER**2))
      VOLUME = VOLUME+CELL
 9020 CONTINUE
 9030 CONTINUE
      PR1GE2 = PR1GE2/VOLUME
      RETURN
      END
C     END SUBROUTINE COMPAR
C
C     *************************************************************
C
C               S U B R O U T I N E        D O M P T H
C
C     *************************************************************
C
      SUBROUTINE DOMPTH(N,NPATHS)
      REAL*8 XINT(104,500,12),VALUE(104,500,10,3),
     *       PR1GE2
      DIMENSION NET(100,103),IPRE(100,99,2),NPA(500,101),
     *          NPPA(100,5),INP(500),NP(5),NPP(100),NPK1(500),
     *          NPK2(500),NPR1(5)
      INTEGER I,IACT,INP,ISLAST,ISNODE,ISNOD1,ISNOD2,
     *        J,JJ,J1,J2,J3,
     *        K,KK,
     *        N,NET,NNN,NOPAT,NP,NPA,NPATHS,NPK,NPK1,NPK2,NPP,NPPA,
     *        NPR1,NSS
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA4/NET,IPRE
C
C     SUBROUTINE DOMPTH DETERMINES THE K MOST STOCHASTICALLY DOMINATING
C     PATHS IN THE NETWORK.  THE FOLLOWING ARRAYS AND VARIABLES ARE USED
C     IN THIS SUBROUTINE:
C       NOPAT     : NUMBER OF PATHS IN THE MAIN PATH LIST
C       NPATHS    : DESIRED NUMBER OF PATHS IN THE SET OF K MOST
C                     STOCHASTICALLY DOMINATING PATHS (MAXIMUM 5)
C       NPK       : NUMBER OF CANDIDATE PATHS THROUGH NODE I
C       NP(K)     : RANK IN THE MAIN PATH LIST OF THE Kth MOST
C                     STOCHASTICALLY DOMINATING PATH ENDING AT NODE I
C       NPA(I,J)  : NODES FORMING THE Ith RANK PATH IN THE MAIN PATH LIST
C       NPP(I)    : NUMBER OF DOMINATING PATHS ENDING AT NODE I
C       NPPA(I,J) : RANK IN THE MAIN PATH LIST OF THE Jth MOST DOMINATING
C                     PATH ENDING AT NODE I
C       NPK1(K)   : RANK IN THE MAIN PATH LIST OF THE Kth CANDIDATE PATH
C                     ENDING AT NODE I
C       NPK2(K)   : NUMBER OF THE PREDECESSOR NODE TO NODE I OF THE Kth
C                     CANDIDATE PATH ENDING AT NODE I
C       NPR1(K)   : RANK OF THE Kth MOST DOMINATING PATH AMONG THE
CANDIDATE
C                     PATHS ENDING AT NODE I
C       INP(J)    : INDICATOR OF PATH J IN THE MAIN PATH LIST
C                     = 1 IF PATH J IS AMONG THE K MOST DOMINATING PATHS
AT
```

```
C                         A PREDECESSOR NODE TO NODE I
C                     = 0 WHEN PATH J IS DETERMINED TO BE ONE OF THE K
C                         MOST DOMINATING PATHS AND IS REMOVED FROM
FURTHER
C                         CONSIDERATION AT NODE I
C     THE DISTRIBUTIONS OF THE K MOST STOCHASTICALLY DOMINATING PATHS
THROUGH
C     EACH NODE IN THE MAIN PATH LIST ARE IN XINT(104,1 TO 500,-)
TOGETHER
C     WITH VALUE(104,1 TO 500,-,-).  THE DISTRIBUTIONS OF THE CANDIDATE
C     PATHS AT NODE I ARE IN XINT(103,1 TO 500,-), VALUE(103,1 TO 500,-
,-).
C
      DO 9110 I = 1,500
      DO 9100 J = 1,10
      XINT(103,I,J) = 0.0
      XINT(104,I,J) = 0.0
      VALUE(103,I,J,1) = 0.0
      VALUE(103,I,J,2) = 0.0
      VALUE(104,I,J,1) = 0.0
      VALUE(104,I,J,2) = 0.0
 9100 CONTINUE
      XINT(103,I,11) = 0.0
      XINT(104,I,11) = 0.0
 9110 CONTINUE
C
C     INITIALIZE THE MAIN PATH LIST AT THE STARTING NODE.
C
      NOPAT = 1
      NPP(1) = 1
      NPPA(1,1) = 1
      INP(1) = 1
      NPA(1,1) = 1
      NPA(1,2) = 1
C
C     DO 9250 DETERMINES THE K MOST STOCHASTICALLY DOMINATING PATHS
THROUGH
C     EACH NODE.
C
      DO 9250 I = 2,N
      NNN = NPATHS
      NPK = 0
C
C     DO 9160 DETERMINES THE DISTRIBUTIONS OF THE (NO. OF PREDECESSORS
OF
C     NODE I)*NPATHS CANDIDATE PATHS AT NODE I.
C
      DO 9160 J = 1,NET(I,102)
      ISNODE = IPRE(I,J,1)
      IACT = IPRE(I,J,2)
C
C     LOAD THE DISTRIBUTION OF THE ACTIVITY FROM NODE ISNODE TO NODE I
C     INTO TEMPORARY LOCATION 1.
C
      DO 9120 KK = 1,10
      XINT(101,1,KK) = XINT(ISNODE,IACT,KK)
      VALUE(101,1,KK,1) = VALUE(ISNODE,IACT,KK,1)
      VALUE(101,1,KK,2) = VALUE(ISNODE,IACT,KK,2)
 9120 CONTINUE
      XINT(101,1,11) = XINT(ISNODE,IACT,11)
      DO 9150 K = 1,NPP(ISNODE)
      NSS = NPPA(ISNODE,K)
      NPK = NPK+1
C
C     LOAD THE DISTRIBUTION OF THE RANK NSS PATH THROUGH NODE ISNODE IN
THE
C     MAIN PATH LIST INTO THE DISTRIBUTION OF THE RANK NPK PATH THROUGH
C     NODE I IN THE CANDIDATE PATH LIST.
C
      DO 9130 KK = 1,10
      XINT(103,NPK,KK) = XINT(104,NSS,KK)
      VALUE(103,NPK,KK,1) = VALUE(104,NSS,KK,1)
      VALUE(103,NPK,KK,2) = VALUE(104,NSS,KK,2)
 9130 CONTINUE
      XINT(103,NPK,11) = XINT(104,NSS,11)
C
C     CONVOLVE THE DISTRIBUTION OF THE RANK NSS PATH THROUGH NODE ISNODE
C     IN THE MAIN PATH LIST AND ACTIVITY IACT AND PLACE THE CONVOLUTION
IN
C     THE DISTRIBUTION OF THE RANK NPK PATH THROUGH NODE I IN THE MAIN
C     PATH LIST.
C
      IF (ISNODE .EQ. 1) THEN
      DO 9140 KK = 1,10
      XINT(103,NPK,KK) = XINT(101,1,KK)
      VALUE(103,NPK,KK,1) = VALUE(101,1,KK,1)
      VALUE(103,NPK,KK,2) = VALUE(101,1,KK,2)
 9140 CONTINUE
      XINT(103,NPK,11) = XINT(101,1,11)
      ELSE
      CALL SERIES(103,NPK,101,1)
      END IF
      NPK1(NPK) = NSS
      NPK2(NPK) = ISNODE
 9150 CONTINUE
 9160 CONTINUE
      IF (NPK .LT. NPATHS) NNN = NPK
C
C     DO 9210 DETERMINES THE K MOST STOCHASTICALLY DOMINATING PATHS
THROUGH
C     NODE I FROM AMONG THE NPK CANDIDATE PATHS.
C
      DO 9210 K = 1,NNN
      ISNOD2 = 0
      J = 1
C
C     THE Jth CANDIDATE PATH IS DESIGNATED THE CONTENDER FOR THE Kth
MOST
C     STOCHASTICALLY DOMINATING PATH THROUGH NODE I.
C
 9170 J2 = J
      J1 = NPK1(J)
```

```
C
C     IF THE Jth CANDIDATE PATH IS ONE OF THE K MOST STOCHASTICALLY
DOMI-
C     NATING PATHS THROUGH NODE I WHICH HAVE ALREADY BEEN DETERMINED
C     [INP(NPK1(J)) = 0], IT IS NOT FURTHER CONSIDERED.
C
      IF (INP(J1).EQ. 0) GO TO 9190
      NP(K) = J1
      NPR1(K) = J
      ISNOD1 = NPK2(J)
C
C     THE J2th CANDIDATE PATH IS TESTED AGAINST THE Jth CANDIDATE PATH.
C
 9180 J2 = J2+1
C
C     IF J2 > NPK, THE Jth CANDIDATE PATH IS THE Kth MOST STOCHASTICALLY
C     DOMINATING PATH THROUGH NODE I.
C
      IF (J2 .GT. NPK) GO TO 9200
      J3 = NPK1(J2)
C
C     IF THE J2th CANDIDATE PATH IS ONE OF THE K MOST STOCHASTICALLY
DOMI-
C     NATING PATHS THROUGH NODE I WHICH HAVE ALREADY BEEN DETERMINED
C     [INP(NPK1(J2)) = 0], IT IS NOT FURTHER CONSIDERED.
C
      IF (INP(J3) .EQ. 0) GO TO 9180
      ISLAST = ISNOD2
      ISNOD2 = NPK2(J2)
C
C     IF THE Jth AND J2th CANDIDATE PATHS HAVE THE SAME PREDECESSOR NODE
TO
C     NODE I, THE J2th CANDIDATE PATH IS NOT FURTHER CONSIDERED, SINCE
THE
C     K MOST STOCHASTICALLY DOMINATING PATHS THROUGH THAT PREDECESSOR
NODE
C     ARE RANK-ORDERED, AND THE Jth PATH STOCHASTICALLY DOMINATES THE
J2th
C     PATH.  IF THE J2th AND THE (J2-1)th CANDIDATE PATHS HAVE THE SAME
PRE-
C     DECESSOR NODE TO NODE I, THE J2th CANDIDATE PATH IS AGAIN NOT
FURTHER
C     CONSIDERED, SINCE THE K MOST STOCHASTICALLY DOMINATING PATHS
THROUGH
C     THAT PREDECESSOR NODE ARE RANK-ORDERED, AND THE (J2-1)th PATH
STOCHAS-
C     TICALLY DOMINATES THE J2th PATH, AND HENCE THE Jth PATH, WHICH
STO-
C     CASTICALLY DOMINATES THE (J2-1)th PATH, STOCHASTICALLY DOMINATES
THE
C     J2th PATH BY TRANSITIVITY.
C
      IF ((ISNOD2 .EQ. ISNOD1) .OR. (ISNOD2 .EQ. ISLAST)) GO TO 9180
C
C     COMPUTE THE PROBABILITY THAT THE Jth CANDIDATE PATH STOCHASTICALLY
```

```
C     DOMINATES (DOMINATES IN PROBABILITY) THE J2th CANDIDATE PATH, I.E.
C     P(DISTRIBUTION OF Jth PATH DURATION > OR = DISTRIBUTION OF J2th
PATH
C     DURATION).
C
      CALL COMPAR(103,J,103,J2,PR1GE2)
C
C     IF THE Jth CANDIDATE PATH STOCHASTICALLY DOMINATES THE J2th
CANDIDATE
C     PATH, THE Jth CANDIDATE PATH REMAINS THE CONTENDER FOR THE Kth
MOST
C     STOCHASTICALLY DOMINATING PATH THROUGH NODE I, AND THE (J2+1)th
CANDI-
C     DATE PATH IS TESTED NEXT.
C
      IF (PR1GE2 .GE. 0.5) GO TO 9180
C
C     IF THE J2th CANDIDATE PATH STOCHASTICALLY DOMINATES THE Jth
CANDIDATE
C     PATH, THE J2th CANDIDATE PATH IS DESIGNATED THE CONTENDER FOR THE
Kth
C     MOST STOCHASTICALLY DOMINATING PATH THROUGH NODE I, AND THE
(J2+1)th
C     CANDIDATE IS TESTED NEXT AGAINST THE CONTENDER.
C
      NP(K) = J3
      NPR1(K) = J2
      J = J2
      ISNOD1 = ISNOD2
      GO TO 9180
 9190 J = J+1
      IF (J .LE. NPK) GO TO 9170
C
C     WHEN THE Kth MOST STOCHASTICALLY DOMINATING PATH HAS BEEN
DETERMINED,
C     SET THE INDICATOR OF ITS PATH NUMBER IN THE MAIN PATH LIST = 0, SO
C     THAT THE PATH IS NOT FURTHER CONSIDERED AT NODE I.
C
 9200 INP(NP(K)) = 0
 9210 CONTINUE
C
C     DO 9240 UPDATES THE MAIN PATH LIST AND PATH PARAMETERS.
C
      NPP(I) = NNN
      DO 9240 K = 1,NNN
C
C     THE Jth PATH IN THE MAIN PATH LIST WAS THE Kth MOST STOCHASTICALLY
C     DOMINATING PATH ENDING AT NODE I.  RESET THE INDICATOR OF THIS
PATH = 1.
C
      J = NP(K)
      INP(J) = 1
C
C     THE Kth MOST STOCHASTICALLY DOMINATING PATH ENDING AT NODE I IS
NOW THE
C     (NOPAT+K)th PATH IN THE MAIN PATH LIST.  SET THE INDICATOR OF THIS
```

```
C     PATH = 1.
C
      JJ = NOPAT+K
      INP(JJ) = 1
C
C     LOAD THIS PATH RANK INTO NPPA(I,K).
C
      NPPA(I,K) = JJ
C
C     LOAD THE NODES OF THIS PATH INTO THE NPA ARRAY.  NPA(I,1) IS THE
LENGTH
C     OF THE Ith PATH IN THE MAIN PATH LIST.
C
      NPA(JJ,1) = NPA(J,1)+1
      NPA(JJ,NPA(J,1)+2) = I
      DO 9220 KK = 2,NPA(J,1)+1
      NPA(JJ,KK) = NPA(J,KK)
 9220 CONTINUE
C
C     THE DISTRIBUTION OF THE Kth MOST STOCHASTICALLY DOMINATING PATH
ENDING
C     AT NODE I IS THE DISTRIBUTION OF THE NPR1(K)th CANDIDATE PATH.
LOAD
C     THIS DISTRIBUTION INTO THE DISTRIBUTION OF THE (NOPAT+K)th PATH IN
THE
C     MAIN PATH LIST.
C
      J1 = NPR1(K)
      DO 9230 KK = 1,10
      XINT(104,JJ,KK) = XINT(103,J1,KK)
      VALUE(104,JJ,KK,1) = VALUE(103,J1,KK,1)
      VALUE(104,JJ,KK,2) = VALUE(103,J1,KK,2)
 9230 CONTINUE
      XINT(104,JJ,11) = XINT(103,J1,11)
 9240 CONTINUE
C
C     THE NUMBER OF PATHS IN THE MAIN PATH LIST IS NOW NOPAT+NNN.
C
      NOPAT = NOPAT+NNN
 9250 CONTINUE
      PRINT 9270
      PRINT 9280,NPATHS
      DO 9260 K = 1,NPATHS
      JJ = NPPA(N,K)
      PRINT 9290,K,NPA(JJ,1),(NPA(JJ,KK),KK = 2,NPA(JJ,1)+1)
 9260 CONTINUE
 9270 FORMAT (1H1)
 9280 FORMAT (1X,'THE ',I1,' MOST STOCHASTICALLY DOMINATING PATHS ',
     *'THROUGH THE NETWORK ARE:')
 9290 FORMAT (/ 1X, 'THE RANK ',I1,' PATH WITH ',I3,' NODES:' /
     *(1X,20I4 /))
      RETURN
      END
C     END SUBROUTINE DOMPTH
```

APPENDIX D

VALIDATION VERSION OF PART PROGRAM WITH "INDEPENDENT MULTIPLE ARCS" APPROXIMATION

```
C
C
C                         VALIDATION VERSION
C                                 OF
C            POLYGONAL APPROXIMATION AND REDUCTION TECHNIQUE
C                               (PART)
C                              ALGORITHM
C                                 FOR
C                     ACYCLIC, DIRECTED NETWORKS
C                                USING
C                "INDEPENDENT MULTIPLE ARCS" NETWORKS
C                           TO APPROXIMATE
C                         NONSEPARABLE NETWORKS
C
C
C      THIS PROGRAM GENERATES "STRONGLY RANDOMIZED NETWORKS," REDUCES
C      THEM WITH THE PART ALGORITHM USING "INDEPENDENT MULTIPLE ARCS"
C      NETWORKS TO APPROXIMATE NONSEPARABLE NETWORKS, SIMULATES THEM,
C      AND OUTPUTS STATISTICAL COMPARISONS OF THE PART-APPROXIMATED
C      AND SIMULATED NETWORK THROUGHPUT DISTRIBUTIONS.  THE PROGRAM IS
C      IS WRITTEN IN FORTRAN 77 AND IS PRESENTLY DESIGNED TO BE OPERATED
C      IN A TIME SHARING MODE WITH ALL DATA INPUT FROM TWO (2) DATA
C      FILES.  THE PROGRAM DIRECTS OUTPUT IN NINE (9) OPTIONAL FORMATS
C      TO A TIME SHARING TERMINAL.  IF DESIRED, THE READ STATEMENTS AT
C      THE BEGINNING OF THE MAIN PROGRAM CAN BE MODIFIED TO ALLOW DATA
C      INPUT DIRECTLY FROM THE TIME SHARING TERMINAL.
C
C      THE CURRENT DIMENSIONS OF THE PROGRAM ALLOW A NETWORK WITH A
C      MAXIMUM OF 100 NODES AND A MAXIMUM OF 99 ACTIVITIES BEGINNING
C      AT EACH NODE.  THESE LIMITS CAN BE EXPANDED BY CHANGING THE
C      DIMENSIONS OF THE XINT AND VALUE ARRAYS.
C
C
C      ************************************************************
C
C               O P E R A T I N G   I N S T R U C T I O N S
C
C      ************************************************************
C
C      INSTRUCTIONS FOR BUILDING DATA FILES
C      ------------------------------------
C
C            DATA FILE DATAH.VAL
C
C      THIS DATA FILE CONTAINS DESCRIPTIONS OF THE PRECODED DISTRIBUTIONS
C      OF ACTIVITY DURATION.
C
C      THERE ARE 5 FIELDS OF DATA.
C      FIELD 1 IS THE CODE FOR THE TYPE OF DISTRIBUTION.
C           1 = TRIANGULAR DISTRIBUTION
C           2 = NORMAL DISTRIBUTION
C           3 = EXPONENTIAL DISTRIBUTION
C           4 = GAMMA DISTRIBUTION
C           5 = BETA DISTRIBUTION
C           6 = UNIFORM DISTRIBUTION
C      FIELD 2 IS
```

```
C         MODE FOR A TRIANGULAR DISTRIBUTION.
C         MEAN FOR A NORMAL DISTRIBUTION.
C         MEAN FOR AN EXPONENTIAL DISTRIBUTION.
C         ALPHA FOR A GAMMA OR A BETA DISTRIBUTION.
C         1/(B-A) FOR A UNIFORM DISTRIBUTION.
C     FIELD 3 IS BETA FOR A GAMMA OR A BETA DISTRIBUTION.
C     FIELD 4 IS THE MINIMUM VALUE OF THE DISTRIBUTION.
C     FIELD 5 IS THE MAXIMUM VALUE OF THE DISTRIBUTION.
C
C
C           DATA FILE CONTROL.VAL
C
C     THIS IS A SINGLE LINE DATA FILE WHICH CONTAINS CONTROL
C     PARAMETERS FOR INPUT, OUTPUT, AND MONTE CARLO SIMULATION.
C
C     THERE ARE 6 FIELDS OF DATA.
C     FIELD 1 IS THE NUMBER OF NETWORKS TO BE GENERATED (MAXIMUM = 100).
C     FIELD 2 IS THE NUMBER OF NODES IN THE NETWORK.
C         0 = NUMBER OF NODES IS TO BE RANDOMLY GENERATED.
C     FIELD 3 IS THE NUMBER OF ACTIVITIES IN THE NETWORK.
C         0 = NUMBER OF ACTIVITIES IS TO BE RANDOMLY GENERATED.
C     FIELD 4 IS THE OUTPUT OPTION DESIRED FOR THE PART RESULTS.
C         1 = A DESCRIPTION OF EACH OF THE 10 CLASSES OF THE
C             FINAL DISTRIBUTION IN THE FORM OF Y = B(O) + B(1) X.
C         2 = A CUMULATIVE DISTRIBUTION FUNCTION OF THE FINAL
C             DISTRIBUTION.
C         3 = A DISCRETE PROBABILITY DENSITY FUNCTION AND A
C             SIMULATION FREQUENCY HISTOGRAM IN GRAPHICAL FORMAT.
C         4 = A COMBINATION OF 1 AND 2 ABOVE.
C         5 = A COMBINATION OF 1 AND 3 ABOVE.
C         6 = A COMBINATION OF 2 AND 3 ABOVE.
C         7 = A COMBINATION OF 1, 2, AND 3 ABOVE.
C         8 = ONLY THE EXPECTED VALUE AND STANDARD DEVIATION.
C         9 = ONLY STATISTICAL COMPARISONS.
C     FIELD 5 IS THE NUMBER OF ITERATIONS OF THE MONTE CARLO
C     SIMULATION REQUESTED (MAXIMUM = 10,000).
C         0 = NO MONTE CARLO SIMULATION IS REQUESTED.
C     FIELD 6 IS THE NUMBER OF PRECODED DISTRIBUTIONS (MAXIMUM = 20).
C
C
C     NOTE
C
C     ALL UNUSED FIELDS MUST BE ZEROED OUT.
C
C
C
C     **********************************************************
C
C                   M A I N     P R O G R A M
C
C     **********************************************************
C
      REAL*8 XINT(200,99,12),VALUE(200,99,10,3),A(130)
      REAL*8 ZVAL(130,5),XX(100,2),TOTAAR(51),COMPAR(200,4)
      REAL*8 SIMT(100,10000),SIMTOT(51),DIST(20,5)
      REAL*8 AREA,AVG,COUNT,SIG,SIZE
```

```
     *                ,X,XSIZE
     *                ,DIFF
      REAL*4 HIGH,HLOW,PERCNT,KSCR20,KSCR10,KSCR05,KSCR02,KSCR01,DMAX
     *    ,AMEAN,ASTD,RN,STEP,UP,XT
      INTEGER I,IDUAL,IEDN,IPRINT,ISTN,ISTNP,ISEED,ICODED,IFLAG
     *       ,MM
     *       ,N,NACT,NACTS,NAN,NCL,NET,NETT,NSIM,NSTART,NGEN,NGENCT
     *       ,NCODED,NCROSS,NNEW,NNODES,NACTSS
     *       ,J,J1,J3
     *       ,K,KK
     *       ,L,L1,L2,L3,L4,LASTK,LA,L3COUNT
     *       ,UA
      REAL*4 DELTA,TOTTIM
      DIMENSION NET(200,103),NNODES(100),NCROSS(100)
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      COMMON/PARA4/XX
      COMMON/PARA5/NET
      COMMON/PARA6/SIMT
      CHARACTER*1 KBL,KBM
      DATA KBL/' '/,KBM/'*'/
      DATA NCL/0/
      DATA TOTTIM/0.0/
      EXTERNAL RNSET,RNNOR,RNUN
C
C     INITIALIZE RANDOM NUMBER GENERATOR.
C
      ISEED = 123456789
      CALL RNSET(ISEED)
C
C     OPEN INPUT AND OUTPUT FILES
C
      OPEN (UNIT = 12, FILE = 'datah.val')
      OPEN (UNIT = 13, FILE = 'control.val')
C
C     READ CONTROL INFORMATION.
C
      READ (13,1900) NGEN,N,NACTS,NAN,NSIM,NCODED
      NSTART = N
      NACTSS = NACTS
C
C     DO 0900 READS DISTRIBUTION DATA FROM DATAH.VAL AND LOADS IT
C     INTO THE DIST ARRAY.
C
      DO 0900 I = 1,NCODED
      READ (12,1902) (DIST(I,J), J=1,5)
 0900 CONTINUE
      STEP = 1.0/REAL(NCODED)
C
C     DO 1530 GENERATES, REDUCES, SIMULATES, AND STATISTICALLY COMPARES
C     NGEN "STRONGLY RANDOMIZED NETWORKS."
C
      DO 1530 NGENCT = 1,NGEN
      CALL TIMER(DELTA)
      N = NSTART
      NACTS = NACTSS
      NCROSS(NGENCT) = 0
C
C     RANDOMLY GENERATE THE NUMBER OF NODES (N), IF NECESSARY.
C
      IF (N .GT. 0) GO TO 0910
      CALL RNUN(1,RN)
      XT = 2.0+(99.0*RN)
      N = INT(XT)
      IF (N .GT. 100) N = 100
C
C     RANDOMLY GENERATE THE NUMBER OF ACTIVITIES (NACTS), IF NECESSARY.
C
 0910 IF (NACTS .GT. 0) GO TO 0920
      LA = N-1
      UA = N*(N-1)/2
      AMEAN = ((REAL(LA+UA))/2.0)-(((REAL(UA-LA))**2)/500.0)
      ASTD = (REAL(UA-LA))/2.5
      CALL RNNOR(1,RN)
      XT = (RN*ASTD)+AMEAN
      NACTS = INT(XT)
      IF (NACTS .LT. LA) NACTS = LA
      IF (NACTS .GT. UA) NACTS = UA
C
C     RANDOMLY GENERATE THE NETWORK (NET ARRAY).
C
 0920 CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
      CALL GENRAN(N,NACTS)
      CALL TIMER(DELTA)
C
C     DO 0970 RANDOMLY SELECTS ONE OF THE PRECODED DISTRIBUTIONS
C     FOR EACH ACTIVITY AND LOADS THE DISTRIBUTION'S DATA INTO
C     THE VALUE AND XINT ARRAYS.  THIS DO ALSO DETERMINES IF
C     THE ACTIVITY DISTRIBUTION IS OTHER THAN UNIFORM, AND,
C     AND, IF SO, CALLS LINEAR TO APPROXIMATE IT WITH A
C     PIECEWISE POLYGONAL FUNCTION.
C
      DO 0970 I = 1,N-1
      L1 = I
      DO 0960 J = 1,NET(I,103)
      L2 = J
      CALL RNUN(1,RN)
      UP = 0.0
      DO 0930 K = 1,NCODED
      UP = UP+STEP
      IF (RN .GT. UP) GO TO 0930
      ICODED = K
      GO TO 0940
 0930 CONTINUE
 0940 VALUE(L1,L2,1,3) = DIST(ICODED,1)
      VALUE(L1,L2,2,3) = DIST(ICODED,2)
      VALUE(L1,L2,3,3) = DIST(ICODED,3)
      XINT(L1,L2,1) = DIST(ICODED,4)
      XINT(L1,L2,2) = DIST(ICODED,5)
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 6) THEN
      CALL LINEAR(L1,L2,NCL)
      GO TO 0960
C
C     DO 0950 CONVERTS DATA FOR UNIFORM DISTRIBUTIONS INTO A USABLE
C     FORM FOR SUBROUTINES SERIES AND PARA.
C
      ELSE
      XINT(L1,L2,11) = XINT(L1,L2,2)
      X = XINT(L1,L2,1)
      XSIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      DO 0950 K = 1,10
      VALUE(L1,L2,K,1) = VALUE(L1,L2,2,3)
      VALUE(L1,L2,K,2) = 0.0
      XINT(L1,L2,K) = X
      X = X+XSIZE
 0950 CONTINUE
      END IF
 0960 CONTINUE
 0970 CONTINUE
C
C     MONTE CARLO SIMULATION OF THE NETWORK.
C
      IF (NSIM .EQ. 0) GO TO 1030
      CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
      CALL SIMULT(N,NSIM)
      CALL TIMER(DELTA)
C
C     REDUCTION OF THE NETWORK BEGINS.
C
C     DO 1040 CHECKS IF A CONVOLUTION (SERIES-REDUCTION) OPERATION
C     IS POSSIBLE, i.e., IF THERE EXISTS A NODE I NOT ON THE OUTPUT
C     CRITICAL LIST SUCH THAT
C         IN-DEGREE NODE I = OUT-DEGREE NODE I = 1.
C
 1030 L3COUNT = 2
 1035 DO 1040 I=L3COUNT,N-1
      L3 = I
      IF ((NET(I,102)+NET(I,103)) .EQ. 2) GO TO 1050
 1040 CONTINUE
C
C     IF (IN-DEGREE NODE I + OUT-DEGREE NODE I) > 2 FOR ALL I NOT = 1
C     OR N, NETWORK IS NONSEPARABLE, SO PROCEED TO "INDEPENDENT
C     MULTIPLE ARCS" APPROXIMATION.
C
      IF (L3COUNT .EQ. 2) GO TO 1145
      GO TO 1080
C
C     A CONVOLUTION IS POSSIBLE WITH THE TWO ACTIVITIES, ONE OF WHICH
C     TERMINATES AT NODE L3 AND THE OTHER OF WHICH STARTS AT NODE L3.
C     DO 1060 IDENTIFIES THE STARTING NODE NUMBER AND THE ACTIVITY
C     NUMBER OF THE ACTIVITY TERMINATING AT NODE L3.  THEN THE SERIES
C     SUBNETWORK CONSISTING OF THESE TWO ACTIVITIES IS CONVOLUTED INTO
C     AN EQUIVALENT ACTIVITY.
C
 1050 DO 1060 I=1,L3-1
```

```
      DO 1060 J=2,NET(I,103)+1
      L1 = I
      L2 = J-1
      IF (NET(I,J) .EQ. L3) GO TO 1070
 1060 CONTINUE
 1070 CALL SERIES(L1,L2,L3,1)
      NET(L1,L2+1) = NET(L3,2)
      NET(L3,2) = 0
      NET(L3,101) = 0
      NET(L3,102) = 0
      NET(L3,103) = 0
      L3COUNT = L3+1
      IF (L3COUNT .EQ. N) GO TO 1080
      GO TO 1035
C
C     DO 1140 CHECKS IF A MAXIMUM (PARALLEL-REDUCTION) OPERATION IS
C     POSSIBLE, i.e., IF THERE EXIST TWO DIFFERENT ACTIVITIES, A1 AND
C     A2, SUCH THAT
C         STARTING NODE (A1) = STARTING NODE (A2), AND
C           ENDING NODE (A1) = ENDING NODE (A2).
C     THEN THE PARALLEL SUBNETWORK CONSISTING OF THESE TWO ACTIVITIES
C     IS PARALLEL-REDUCED WITH A MAXIMUM OPERATION INTO AN EQUIVALENT
C     ACTIVITY.
C
 1080 DO 1140 I=1,N-1
      L1 = I
 1085 DO 1090 J=2,NET(L1,103)
      L2 = J-1
      IF (NET(L1,J) .EQ. 0) GO TO 1140
      DO 1090 K=J+1,NET(L1,103)+1
      L3 = K-1
      IF (NET(L1,J) .EQ. NET(L1,K)) THEN
      IEDN = NET(L1,J)
      GO TO 1110
      ELSE
      GO TO 1090
      END IF
 1090 CONTINUE
      GO TO 1140
 1110 CALL PARA(L1,L2,L3)
      NET(L1,103) = NET(L1,103)-1
      NET(IEDN,102) = NET(IEDN,102)-1
      DO 1120 K=L3,NET(L1,103)
      NET(L1,K+1) = NET(L1,K+2)
      DO 1115 L=1,10
      XINT(L1,K,L)= XINT(L1,K+1,L)
      VALUE(L1,K,L,1) = VALUE(L1,K+1,L,1)
      VALUE(L1,K,L,2) = VALUE(L1,K+1,L,2)
 1115 CONTINUE
      XINT(L1,K,11) = XINT(L1,K+1,11)
 1120 CONTINUE
      K = NET(L1,103)+1
      NET(L1,K+1) = 0
      DO 1130 L=1,10
      XINT(L1,K,L) = 0.
      VALUE(L1,K,L,1) = 0.
      VALUE(L1,K,L,2) = 0.
 1130 CONTINUE
      XINT(L1,K,11) = 0.
      GO TO 1085
 1140 CONTINUE
      GO TO 1030
C
C     THROUGH 1146 CHECKS IF THE NETWORK HAS BEEN SERIES-PARALLEL
C     REDUCED TO A SINGLE EQUIVALENT ACTIVITY.
C
 1145 IF (NET(1,103) .NE. 1) GO TO 1147
      IF (NET(N,102) .NE. 1) GO TO 1147
      DO 1146 I=2,N-1
      IF (NET(I,102) + NET(I,103)) 1550,1146,1147
 1146 CONTINUE
C
C     THE NETWORK HAS BEEN SERIES-PARALLEL REDUCED TO A SINGLE EQUIVA-
C     LENT ACTIVITY.
C
      NNODES(NGENCT) = N
      GO TO 1240
C
C     TO 1240 REDUCES THE NONSEPARABLE NETWORK USING "INDEPENDENT MULTI-
C     PLE ARCS" APPROXIMATION WITH THE "FIRST AVAILABLE ARC WITH
PROPERTY
C     1" METHOD.
C
 1147 NCROSS(NGENCT) = NCROSS(NGENCT)+1
C
C     DO 1148 IDENTIFIES THE START NODE ISTN OF THE FIRST AVAILABLE
C     CROSS-CONNECTION IN THE NONSEPARABLE NETWORK.
C
      DO 1148 I=2,N-2
      IF ((NET(I,102) .EQ. 1) .AND. (NET(I,103) .GT. 1)) THEN
      ISTN = I
      GO TO 1149
      ELSE
      GO TO 1148
      END IF
 1148 CONTINUE
C
C     THROUGH 1151 IDENTIFIES THE START NODE ISTNP AND THE ACTIVITY
C     NUMBER IDUAL OF THE SINGLE ACTIVITY TERMINATING AT NODE ISTN.
C     THIS ACTIVITY IS THE "A" ACTIVITY CONNECTING NODE ISTNP AND NODE
C     ISTN WHICH MUST BE "INDEPENDENTLY MULTIPLIED."
C
 1149 DO 1151 I=1,N-3
      DO 1150 J=2,NET(I,103)+1
      IF (NET(I,J) .EQ. ISTN) THEN
      ISTNP = I
      IDUAL = J-1
      GO TO 1152
      ELSE
      GO TO 1150
      END IF
 1150 CONTINUE
 1151 CONTINUE
C
C     NDUAL IS THE NUMBER OF "INDEPENDENT MULTIPLES" OF ACTIVITY IDUAL
C     WHICH MUST BE INSERTED INTO THE NETWORK.
C
 1152 NDUAL = NET(ISTN,103)-1
      NNEW = N+NDUAL
C
C     DO 1153 INCREASES NODE NUMBERS ABOVE ISTN BY NDUAL.
C
      DO 1153 J=1,N
      DO 1153 J1=1,NET(J,103)+1
      IF (NET(J,J1) .GT. ISTN) NET(J,J1) = NET(J,J1)+NDUAL
 1153 CONTINUE
C
C     DO 1154 SHIFTS ROWS OF NET ARRAY, FROM N DOWN TO ISTN+1, AHEAD
C     NDUAL ROWS.
C
      DO 1154 J=NNEW,ISTN+NDUAL+1,-1
      DO 1154 J1=1,103
      NET(J,J1) = NET(J-NDUAL,J1)
      NET(J-NDUAL,J1)=0
 1154 CONTINUE
C
C     DO 1155 INSERTS NDUAL NODES AFTER NODE ISTN FOR NDUAL "INDEPENDENT
C     MULTIPLES" OF THE "PROPERTY 1" ACTIVITY IN THE NET ARRAY.
C
      DO 1155 J=ISTN+1,ISTN+NDUAL
      NET(J,1) = J
      NET(J,2) = NET(ISTN,J-ISTN+2)
      NET(ISTN,J-ISTN+2) = 0
      NET(J,101) = 1
      NET(J,102) = 1
      NET(J,103) = 1
 1155 CONTINUE
      NET(ISTN,103) = 1
C
C     DO 1156 SHIFTS THE TERMINATING NODE NUMBERS OF ACTIVITIES STARTING
C     AT NODE ISTNP AFTER ISTN AHEAD NDUAL POSITIONS IN THE NET ARRAY.
C
      DO 1156 J=NET(ISTNP,103)+NDUAL+1,IDUAL+NDUAL+2,-1
      NET(ISTNP,J) = NET(ISTNP,J-NDUAL)
      NET(ISTNP,J-NDUAL) = 0
 1156 CONTINUE
C
C     DO 1157 INSERTS NDUAL NODES - ISTN+1,...,ISTN+NDUAL - AS THE
C     TERMINATING NODES OF THE "INDEPENDENTLY MULTIPLIED" "A" ACTIVITY
C     FROM NODE ISTNP IN THE NET ARRAY.
C
      DO 1157 J=1,NDUAL
      NET(ISTNP,IDUAL+J+1) = ISTN+J
 1157 CONTINUE
      NET(ISTNP,103) = NET(ISTNP,103)+NDUAL
C
C     DO 1158 SHIFTS ROWS OF XINT ARRAY, FROM N DOWN TO ISTN+1, AHEAD
C     NDUAL ROWS.
```

```
C
      DO 1158 J=NNEW,ISTN+NDUAL+1,-1
      DO 1158 J1=1,99
      DO 1158 J2=1,12
      XINT(J,J1,J2) = XINT(J-NDUAL,J1,J2)
      XINT(J-NDUAL,J1,J2) = 0.
 1158 CONTINUE
C
C     DO 1159 SHIFTS ACTIVITY LINEARIZATION POINTS FOR ACTIVITIES
C     2,...,NDUAL+1 FROM NODE ISTN TO THE FIRST AND ONLY ACTIVITIES
C     FROM THE NEW NODES - ISTN+1,...,ISTN+NDUAL - IN THE XINT ARRAY.
C
      DO 1159 J1=1,NDUAL
      DO 1159 J2=1,12
      XINT(ISTN+J1,1,J2) = XINT(ISTN,J1+1,J2)
      XINT(ISTN,J1+1,J2) = 0.
 1159 CONTINUE
C
C     DO 1160 SHIFTS ACTIVITY LINEARIZATION POINTS FOR ACTIVITIES
C     AFTER IDUAL FROM NODE ISTN AHEAD NDUAL POSITIONS IN THE XINT
ARRAY.
C
      DO 1160 J1=NET(ISTNP,103),IDUAL+NDUAL+1,-1
      DO 1160 J2=1,12
      XINT(ISTNP,J1,J2) = XINT(ISTNP,J1-NDUAL,J2)
      XINT(ISTNP,J1-NDUAL,J2) = 0.
 1160 CONTINUE
C
C     DO 1161 INSERTS NDUAL COPIES OF THE ACTIVITY LINEARIZATION POINTS
C     OF ACTIVITY IDUAL FROM NODE ISTNP TO THE NEW NODES - ISTN+1,...,
C     ISTN+NDUAL - IN THE XINT ARRAY.
C
      DO 1161 J1=1,NDUAL
      DO 1161 J2=1,12
      XINT(ISTNP,IDUAL+J1,J2) = XINT(ISTNP,IDUAL,J2)
 1161 CONTINUE
C
C     DO 1162 SHIFTS ROWS OF VALUE ARRAY, FROM N DOWN TO ISTN+1, AHEAD
C     NDUAL ROWS.
C
      DO 1162 J=NNEW,ISTN+NDUAL+1,-1
      DO 1162 J1=1,99
      DO 1162 J2=1,10
      DO 1162 J3=1,3
      VALUE(J,J1,J2,J3) = VALUE(J-NDUAL,J1,J2,J3)
      VALUE(J-NDUAL,J1,J2,J3) = 0.
 1162 CONTINUE
C
C     DO 1163 SHIFTS ACTIVITY POLYGONAL APPROXIMATION COEFFICIENTS FOR
C     ACTIVITIES 2,...NDUAL+1 FROM NODE ISTN TO THE FIRST AND ONLY
C     ACTIVITIES FROM THE NEW NODES - ISTN+1,...,ISTN+NDUAL - IN THE
C     VALUE ARRAY.
C
      DO 1163 J1=1,NDUAL
      DO 1163 J2=1,10
      DO 1163 J3=1,3
      VALUE(ISTN+J1,1,J2,J3) = VALUE(ISTN,J1+1,J2,J3)
      VALUE(ISTN,J1+1,J2,J3) = 0.
 1163 CONTINUE
C
C     DO 1164 SHIFTS ACTIVITY POLYGONAL APPROXIMATION COEFFICIENTS FOR
C     ACTIVITIES AFTER IDUAL FROM NODE ISTNP AHEAD NDUAL POSITIONS IN
THE
C     VALUE ARRAY.
C
      DO 1164 J1=NET(ISTNP,103),IDUAL+NDUAL+1,-1
      DO 1164 J2=1,10
      DO 1164 J3=1,3
      VALUE(ISTNP,J1,J2,J3) = VALUE(ISTNP,J1-NDUAL,J2,J3)
      VALUE(ISTNP,J1-NDUAL,J2,J3) = 0.
 1164 CONTINUE
C
C     DO 1165 INSERTS NDUAL COPIES OF THE ACTIVITY POLYGONAL APPROXIMA-
C     TION COEFFICIENTS OF ACTIVITY IDUAL FROM NODE ISTNP TO THE NEW
C     NODES - ISTN+1,...,ISTN+NDUAL - IN THE VALUE ARRAY.
C
      DO 1165 J1=1,NDUAL
      DO 1165 J2=1,10
      DO 1165 J3=1,3
      VALUE(ISTNP,IDUAL+J1,J2,J3) = VALUE(ISTNP,IDUAL,J2,J3)
 1165 CONTINUE
      N = NNEW
      GO TO 1030
C
C     AT THIS POINT IN THE MAIN PROGRAM ALL CALCULATIONS HAVE
C     BEEN COMPLETED AND DATA IS PREPARED FOR FINAL OUTPUT.
C
 1240 IF (NAN .NE. 9) THEN
      PRINT 1910
      PRINT 1915,NGENCT,NGEN
      ELSE
      GO TO 1245
      END IF
 1245 L = 1
      KK = 0
      DO 1270 I = 1,10
C
C     THE XX ARRAY IS USED FOR HISTOGRAM AND CDF CALCULATIONS.
C
      XX(1,1) = XINT(1,1,I)
      SIZE = (XINT(1,1,2)-XINT(1,1,1))/5.
      LASTK = L+4
      DO 1250 K = L,LASTK
      KK = KK+1
      XX(K,2) = VALUE(1,1,I,1)+(VALUE(1,1,I,2)*XX(K,1))
      IF ((KK .LE. 1).AND.(L .GT. 4)) XX(K,2) = (((VALUE(1,1,I,2)
     &*XX(K,1))+VALUE(1,1,I,1))+(VALUE(1,1,I-1,2)*XX(K,1))+
     &VALUE(1,1,I-1,1))/2.
      XX(K+1,1) = XX(K,1)+SIZE
1250  CONTINUE
      KK = 0
      L = L+5
      IF ((NAN .EQ. 1).OR.(NAN .EQ. 4).OR.(NAN .EQ. 5).OR.
     &(NAN .EQ. 7)) GO TO 1260
      GO TO 1270
 1260 IF (I .EQ. 1) PRINT 1920
      PRINT 1930,I,XINT(1,1,I),XINT(1,1,I+1)
      PRINT 1940,VALUE(1,1,I,1),VALUE(1,1,I,2)
 1270 CONTINUE
      XX(51,2)=VALUE(1,1,10,1)+VALUE(1,1,10,2)*XX(51,1)
C
C     TOTAAR IS USED FOR CDF CALCULATIONS.
C
      AREA = 0.0
      DO 1280 I = 1,50
      AREA = AREA+((XX(I,2)+XX(I+1,2))*SIZE*.5)
      TOTAAR(I) = AREA
 1280 CONTINUE
      AREA = 1.0/AREA
      DO 1290 I = 1,50
      TOTAAR(I) = TOTAAR(I)*AREA
 1290 CONTINUE
      DO 1295 I = 51,2,-1
      TOTAAR(I) = TOTAAR(I-1)
 1295 CONTINUE
      TOTAAR(1) = 0.0
      XX(1,1) = XINT(1,1,1)
      IF (NAN .EQ. 9) GO TO 1350
      IF ((NAN .EQ. 2).OR.(NAN .EQ. 4).OR.(NAN .EQ. 6).OR.
     &(NAN .EQ. 7)) GO TO 1300
      GO TO 1320
 1300 PRINT 1910
      PRINT 1950
      DO 1310 I = 1,51
      PRINT 1960,XX(I,1),TOTAAR(I)
 1310 CONTINUE
 1320 IF ((NAN .EQ. 3).OR.(NAN .EQ. 5).OR.(NAN .EQ. 6).OR.
     &(NAN .EQ. 7)) GO TO 1330
      GO TO 1340
 1330 IPRINT = 51
      IFLAG = 0
      CALL PLOT(IPRINT,KBL,KBM,IFLAG)
 1340 CONTINUE
C
C     DO 1360 COMPUTES AN APPROXIMATED EXPECTED VALUE USING
C     GROUPED DATA.  DO 1370 COMPUTES AN APPROXIMATED STANDARD
C     DEVIATION.
C
 1350 AVG = 0.0
      SIG = 0.0
      DO 1360 MM = 1,50
      AVG = AVG+((XX(MM,1)+(SIZE/2.))*(TOTAAR(MM+1)-TOTAAR(MM)))
 1360 CONTINUE
      COMPAR(NGENCT,3) = AVG
      DO 1370 MM = 1,50
      SIG = SIG+(((XX(MM,1)+(SIZE/2.)-AVG)**2)*(TOTAAR(MM+1)-
     &TOTAAR(MM)))
 1370 CONTINUE
```

```
      SIG = DSQRT(SIG)
      COMPAR(NGENCT,4) = SIG
      IF (NAN .EQ. 9) GO TO 1390
      PRINT 1910
      PRINT 1970,AVG,SIG
      DO 1380 MM = 1,4
      BLOW = AVG-(FLOAT(MM)*SIG)
      BIGH = AVG+(FLOAT(MM)*SIG)
C
C     THESE ARE FIXED PERCENTAGES.  THE ASSUMPTION IS MADE THAT
C     THE FINAL PRODUCT WILL RESEMBLE A NORMAL DISTRIBUTION.  THEY
C     CORRESPOND TO 1, 2, 3, AND 4 STANDARD DEVIATIONS RESPECTIVELY.
C
      IF (MM .EQ. 1) PERCNT = 68.24
      IF (MM .EQ. 2) PERCNT = 95.44
      IF (MM .EQ. 3) PERCNT = 99.73
      IF (MM .EQ. 4) PERCNT = 99.99
      PRINT 1980,BLOW,HIGH,PERCNT
 1380 CONTINUE
 1390 IF (NSIM .EQ. 0) GO TO 1530
C
C     THROUGH 1500 COMPILES OUTPUT FROM THE MONTE CARLO SIMULATION.
C
C     DO 1410 COMPILES THE CUMULATIVE DISTRIBUTION FUNCTION.
C
      COUNT = 0.0
      SIMTOT(1) = 0.0
      DO 1410 J = 1,50
      DO 1400 K = 1,NSIM
      IF((XX(J,1) .LE. SIMT(NSTART,K)) .AND. (SIMT(NSTART,K) .LT.
     &XX(J+1,1))) COUNT = COUNT+1.
 1400 CONTINUE
      XX(J+1,2) = COUNT/DFLOAT(NSIM)
      SIMTOT(J+1) = SIMTOT(J)+XX(J+1,2)
      COUNT = 0.0
 1410 CONTINUE
      IF (NAN .EQ. 9) GO TO 1450
      PRINT 1910
      PRINT 1925
      IF ((NAN .EQ. 2) .OR. (NAN .EQ. 4) .OR. (NAN .EQ. 6) .OR.
     &(NAN .EQ. 7)) GO TO 1420
      GO TO 1440
 1420 PRINT 1950
      DO 1430 J = 1,51
      PRINT 1960,XX(J,1),SIMTOT(J)
 1430 CONTINUE
 1440 IF ((NAN .EQ. 3) .OR. (NAN .EQ. 5) .OR. (NAN .EQ. 6) .OR.
     &(NAN .EQ. 7)) GO TO 1450
      GO TO 1470
 1450 DO 1460 J = 1,50
      XX(J,1) = XX(J,1)+(SIZE/2.)
      XX(J,2) = XX(J+1,2)
 1460 CONTINUE
      IF (NAN .EQ. 9) GO TO 1475
      IPRINT = 50
      IFLAG = 1
      CALL PLOT(IPRINT,KBL,KBM,IFLAG)
 1470 CONTINUE
C
C     DO 1480 COMPUTES AN APPROXIMATED EXPECTED VALUE AND
C     DO 1490 COMPUTES AN APPROXIMATED STANDARD DEVIATION
C     USING GROUPED DATA.
C
 1475 AVG = 0.0
      SIG = 0.0
      DO 1480 J = 1,50
      AVG = AVG+(XX(J,1)*(SIMTOT(J+1)-SIMTOT(J)))
 1480 CONTINUE
      DO 1490 J = 1,50
      SIG = SIG+(((XX(J,1)-AVG)**2)*(SIMTOT(J+1)-SIMTOT(J)))
 1490 CONTINUE
      SIG = DSQRT(SIG)
      IF (NAN .EQ. 9) GO TO 1505
      PRINT 1910
      PRINT 1970,AVG,SIG
      DO 1500 MM = 1,4
      HLOW = AVG-(FLOAT(MM)*SIG)
      HIGH = AVG+(FLOAT(MM)*SIG)
C
C     IT IS ASSUMED THAT THE DISTRIBUTION THROUGH NODE I RESEMBLES
C     A NORMAL DISTRIBUTION.  THE FIXED PERCENTAGES CORRESPOND TO
C     1, 2, 3, AND 4 STANDARD DEVIATIONS, RESPECTIVELY, FROM THE
C     EXPECTED VALUE.
C
      IF (MM .EQ. 1) PERCNT = 68.24
      IF (MM .EQ. 2) PERCNT = 95.44
      IF (MM .EQ. 3) PERCNT = 99.73
      IF (MM .EQ. 4) PERCNT = 99.99
      PRINT 1980,HLOW,HIGH,PERCNT
 1500 CONTINUE
C
C     COMPUTE RELATIVE ERROR OF APPROXIMATED MEAN AND STANDARD DEVIATION
C     OF NETWORK THROUGHPUT DISTRIBUTION.
C
 1505 COMPAR(NGENCT,3) = ((COMPAR(NGENCT,3)-AVG)/AVG)*100.0
      COMPAR(NGENCT,4) = ((COMPAR(NGENCT,4)-SIG)/SIG)*100.0
C
C     COMPARE POLYGONAL APPROXIMATION OF THROUGHPUT DISTRIBUTION
C     WITH SIMULATED THROUGHPUT DISTRIBUTION USING THE KOLMOGOROV-
C     SMIRNOV ONE-SAMPLE TEST.
C
      KSCR20 = 1.0730/SQRT(50.)
      KSCR10 = 1.2239/SQRT(50.)
      KSCR05 = 1.3581/SQRT(50.)
      KSCR02 = 1.5174/SQRT(50.)
      KSCR01 = 1.6276/SQRT(50.)
C
C     COMPUTE THE K-S TEST STATISTIC D-MAX.
C
      DMAX = 0.0
      DO 1520 I = 2,51
      DIFF = DABS(SIMTOT(I)-TOTAAR(I))
      IF (DIFF .GT. DMAX) DMAX = DIFF
 1520 CONTINUE
      IF (NAN .NE. 9) THEN
      PRINT 1910
      PRINT 1991,DMAX
      PRINT 1992,KSCR20,KSCR10,KSCR05,KSCR02,KSCR01
      IF (DMAX .LE. KSCR05) PRINT 1993
      ELSE
      COMPAR(NGENCT,1) = DMAX
      COMPAR(NGENCT,2) = 0.0
      IF (DMAX .LE. KSCR20) COMPAR(NGENCT,2) = 20.0
      IF ((DMAX .GT. KSCR20) .AND. (DMAX .LE. KSCR10))
     *COMPAR(NGENCT,2) = 10.0
      IF ((DMAX .GT. KSCR10) .AND. (DMAX .LE. KSCR05))
     *COMPAR(NGENCT,2) = 5.0
      IF ((DMAX .GT. KSCR05) .AND. (DMAX .LE. KSCR02))
     *COMPAR(NGENCT,2) = 2.0
      IF ((DMAX .GT. KSCR02) .AND. (DMAX .LE. KSCR01))
     *COMPAR(NGENCT,2) = 1.0
      END IF
      CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
 1530 CONTINUE
      IF (NAN .NE. 9) THEN
      PRINT 1996,TOTTIM
      STOP
      ELSE
      PRINT 1994,NGEN,NSTART,NACTSS
      PRINT 1992,KSCR20,KSCR10,KSCR05,KSCR02,KSCR01
      PRINT 1995
      CALL TIMER(DELTA)
      DO 1540 I = 1,NGEN
      IF (COMPAR(I,2) .EQ. 0.0)  PRINT 1981,COMPAR(I,1),COMPAR(I,3),
     *COMPAR(I,4),NNODES(I),NCROSS(I)
      IF (COMPAR(I,2) .EQ. 1.0)  PRINT 1982,COMPAR(I,1),COMPAR(I,3),
     *COMPAR(I,4),NNODES(I),NCROSS(I)
      IF (COMPAR(I,2) .EQ. 2.0)  PRINT 1983,COMPAR(I,1),COMPAR(I,3),
     *COMPAR(I,4),NNODES(I),NCROSS(I)
      IF (COMPAR(I,2) .EQ. 5.0)  PRINT 1984,COMPAR(I,1),COMPAR(I,3),
     *COMPAR(I,4),NNODES(I),NCROSS(I)
      IF (COMPAR(I,2) .EQ. 10.0) PRINT 1985,COMPAR(I,1),COMPAR(I,3),
     *COMPAR(I,4),NNODES(I),NCROSS(I)
      IF (COMPAR(I,2) .EQ. 20.0) PRINT 1986,COMPAR(I,1),COMPAR(I,3),
     *COMPAR(I,4),NNODES(I),NCROSS(I)
 1540 CONTINUE
      CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
      PRINT 1996,TOTTIM
      END IF
      STOP
 1550 PRINT 1990
      STOP
C
C     FORMAT STATEMENTS
C
 1900 FORMAT (2(I3,1X),I4,1X,I1,1X,I5,1X,I2)
 1901 FORMAT (3(I2,25(1X,I2)/),I2,21(1X,I2),1X,I1,2(1X,I2))
 1902 FORMAT (F1.0,4(1X,F8.2))
 1910 FORMAT (1H1)
 1915 FORMAT (1X,'THE RESULTS FOR NETWORK NUMBER ',I3,' OF ',I3,
     &' NETWORKS GENERATED ARE:' //)
 1920 FORMAT (1X,'THE POLYGONAL APPROXIMATION OF THE TIME DISTRIBUTION',
     &' THROUGH THE PROJECT IS:' //)
 1925 FORMAT (1X,'THE SIMULATED TIME DISTRIBUTION',
     &' THROUGH THE PROJECT IS:' //)
 1930 FORMAT (1X,'INTERVAL',I3,4X,'LOWER LIMIT =',F8.2,3X,
     &'UPPER LIMIT =',F8.2 //)
 1940 FORMAT (15X,'X = (',F12.8,') + (',F12.8,') T' //)
 1950 FORMAT (14X,'CUMULATIVE DISTRIBUTION FUNCTION' //
     &21X,'T',13X,'F(T)')
 1960 FORMAT (16X,F9.3,F17.8)
 1970 FORMAT (12X,'EXPECTED VALUE OF T     =',F13.8 /
     &12X,'STANDARD DEVIATION OF T =',F13.8 //)
 1980 FORMAT (1X,'THE PROBABILITY OF THE PROJECT THROUGHPUT TIME',
     &' FALLING BETWEEN' / 1X,F8.3,' TIME UNITS AND ',F8.3,
     &' TIME UNITS IS ABOUT ',F5.2,'%.'//)
 1981 FORMAT (1X,F6.4,6X,'   <1%',14X,F6.2,'%',9X,F6.2,'%',7X,I3,6X,I3)
 1982 FORMAT (1X,F6.4,6X,' 1% - 2%',12X,F6.2,'%',9X,F6.2,'%',7X,I3,6X,
     &I3)
 1983 FORMAT (1X,F6.4,6X,' 2% - 5%',12X,F6.2,'%',9X,F6.2,'%',7X,I3,6X,
     &I3)
 1984 FORMAT (1X,F6.4,6X,' 5% - 10%',11X,F6.2,'%',9X,F6.2,'%',7X,I3,6X,
     &I3)
 1985 FORMAT (1X,F6.4,6X,'10% - 20%',11X,F6.2,'%',9X,F6.2,'%',7X,I3,6X,
     &I3)
 1986 FORMAT (1X,F6.4,6X,'   >20%',13X,F6.2,'%',9X,F6.2,'%',7X,I3,6X,I3)
 1990 FORMAT (1X,'PROGRAM STOPPED' / 1X,'IMPROPER NODE NUMBER(S) '
     &,'ENCOUNTERED')
 1991 FORMAT (1X,'KOLMOGOROV-SMIRNOV ONE-SAMPLE TEST COMPARISON OF ',
     &'POLYGONAL APPROXIMATION' / 1X,'OF NETWORK THROUGHPUT DISTRIBUTION
     & AND SIMULATED NETWORK THROUGHPUT' / 1X,'DISTRIBUTION:' //
     &1X,'K-S TEST STATISTIC D-MAX = ', F6.4 /)
 1992 FORMAT (1X,'K-S CRITICAL VALUES:' / 15X,'20 PERCENT = ',F6.4 /
     &15X,'10 PERCENT = ',F6.4 / 16X,'5 PERCENT = ',F6.4 /
     &16X,'2 PERCENT = ',F6.4 / 16X,'1 PERCENT = ',F6.4 /)
 1993 FORMAT (1X,'FAIL TO REJECT THE NULL HYPOTHESIS THAT THE ',
     &'DISTRIBUTIONS ARE THE SAME' / 1X,'AT THE 5% LEVEL OF ',
     &'STATISTICAL SIGNIFICANCE.')
 1994 FORMAT (1X,'STATISTICAL COMPARISONS FOR THE ',I3,' NETWORKS ',
     &'GENERATED' / 1X,'WITH ',I3,' NODES AND ',I4,' ACTIVITIES ARE:' //
     &)
 1995 FORMAT (10X,'PROBABILITY VALUE',3X,'RELATIVE ERROR',2X,'RELATIVE '
     &,'ERROR',2X,'TOTAL',2X,'NO. CROSS' / 2X,'D-MAX',2X,'(TYPE 1',
     &' ERROR PROB)',5X,'OF MEAN',8X,'OF STN DEV',4X,'NODES',2X,
     &'-CONNECTS' /)
 1996 FORMAT (// 1X,'CPU TIME FOR PART PROCESSING IS ',F8.3,' SECONDS'
     &//)
      END
C     END MAIN PROGRAM
C
C     ****************************************************************
C
```

```
C               S U B R O U T I N E          P A R A
C
C     ***********************************************************
C
      SUBROUTINE PARA (L1,L2,L3)
      REAL*8 VALUE(200,99,10,3),XINT(200,99,12)
      REAL*8 XVAL,ZVAL(130,5),PAR(2,15,6),FACT,B(130)
      REAL*4 Z
      INTEGER L1,L2,L3,NV1,NV2
      INTEGER K4(2,30)
      INTEGER I,IINT,N,NCL,J,K,K3,L6,LASTJ,LASTK
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/B
C
C     SUBROUTINE PARA IS USED TO REDUCE PARALLEL ARCS INTO A SINGLE
C     EQUIVALENT ARC.  IT FINDS THE MAX OPERATOR BY MULTIPLYING CAP
C     F(X) AGAINST CAP G(X) OVER THE INTERVALS OF VALIDITY.
C
      NV1 = 10
      NV2 = 10
      DO 2020 N = 1,2
      L6 = L2
      IF (N .EQ. 2) L6 = L3
      FACT = 0
      DO 2010 J = 1,10
      B(1) = XINT(L1,L6,J)
C
C     DO 2000 CONVERTS EACH LINEAR POLYNOMIAL PIECE OF LITTLE F(X)
C     INTO THE CORRESPONDING QUADRATIC POLYNOMIAL PIECE OF ITS
C     CUMULATIVE DISTRIBUTION CAP F(X).
C
      DO 2000 I = 1,2
      XVAL = VALUE(L1,L6,J,I)
      Z = FLOAT(I)
      PAR(N,J,I+1) = XVAL/Z
      PAR(N,J,1) = PAR(N,J,1)+((-1.0)*(XVAL/Z)*(B(1)**I))
      K4(N,J) = I+1
 2000 CONTINUE
      IF (J .GT. 1) PAR(N,J,1) = PAR(N,J,1)+FACT
      FACT = PAR(N,J,1)+(PAR(N,J,2)*XINT(L1,L6,J+1))+(PAR(N,J,3)
     &*(XINT(L1,L6,J+1)**2))
 2010 CONTINUE
 2020 CONTINUE
C
C     DO 2040 ASSIGNS INTERVAL BOUNDARY VALUES TO THE B ARRAY.
C
      DO 2040 I = 1,22
      IF (I .GT. 11) GO TO 2030
      B(I) = XINT(L1,L2,I)
      GO TO 2040
 2030 B(I) = XINT(L1,L3,I-11)
 2040 CONTINUE
      NCL = 21
      CALL SORT(NCL)
C
```

```
C     DO 2080 DETERMINES THE POINT AT WHICH THE DISTRIBUTION DOMAINS
C     OF THE TWO ARCS BEING COMBINED OVERLAP.  ONCE THIS POINT IS
C     DETERMINED, THE B ARRAY IS ADJUSTED TO REFLECT THE OVERLAP
C     (ALL VALUES LESS THAN THIS POINT OF FIRST OVERLAP NEED NOT BE
C     CONSIDERED, BECAUSE ONE OF THE DISTRIBUTIONS EQUALS ZERO AT
C     THESE VALUES). IF THE DOMAINS ARE DISJOINT OR OVERLAP AT ONLY.
C     ONE BOUNDARY POINT, THE RESULT OF THE APPLICATION OF THE
C     MAXIMUM OPERATOR IS JUST THE UNCHANGED APPROXIMATED PROBABILITY
C     DENSITY FUNCTION OF THE DISTRIBUTION DEFINED ON THE HIGHER-
C     VALUED DOMAIN.  GO TO 2180 OR GO TO 2160 RETURNS THIS FUNCTION
C     DIRECTLY WITHOUT FURTHER PROCESSING.
C
      IINT = 0
      LASTJ = NCL+1
      DO 2080 J = 1,LASTJ
      IF ((XINT(L1,L2,1) .GE. XINT(L1,L3,1)-0.001) .AND.
     &(XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001)) GO TO 2080
      IF (IINT .GE. 1) GO TO 2060
      IF (XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001) GO TO 2050
      IF (XINT(L1,L3,J+1) .GE. XINT(L1,L2,1)-0.001) IINT = J
      IF ((XINT(L1,L3,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L2,1) .GE. XINT(L1,L3,J+1)-0.001)
     &.AND. (XINT(L1,L2,1) .LE. XINT(L1,L3,J+1)+0.001)
     &.AND. (XINT(L1,L3,J+2) .LE. 0.001))) GO TO 2180
      GO TO 2080
 2050 IF (XINT(L1,L2,J+1) .GE. XINT(L1,L3,1)-0.001) IINT = J
      IF ((XINT(L1,L2,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L3,1) .GE. XINT(L1,L2,J+1)-0.001)
     &.AND. (XINT(L1,L3,1) .LE. XINT(L1,L2,J+1)+0.001)
     &.AND. (XINT(L1,L2,J+2) .LE. 0.001))) GO TO 2160
      GO TO 2080
 2060 LASTK = NCL-(IINT-1)
      DO 2070 K = 1,LASTK
      B(K) = B(K+IINT)
      B(K+IINT) = 0
 2070 CONTINUE
      GO TO 2090
 2080 CONTINUE
 2090 NCL = NCL-IINT
C
C     DO 2150 IS THE OUTER LOOP FOR THE PROCESS OF CREATING THE
C     EQUIVALENT ARC.  NCL IS THE NUMBER OF CLASSES INVOLVED
C     BETWEEN THE TWO ARCS.
C
      N1 = 0
      N2 = 0
      DO 2150 I = 1,NCL
      DO 2110 J = 1,11
C
C     DO 2110 DETERMINES THE APPROPRIATE INTERVALS OF EACH DISTRIBUTION
C     THAT ARE VALID FOR THE B(I) VALUE BEING CONSIDERED.  N1 AND
C     N2 ARE THE CONTROLS FOR UPPER AND LOWER ARCS RESPECTIVELY.
C
      IF (N1 .GE. 1) GO TO 2100
      IF (((B(I) .GE. XINT(L1,L2,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L2,J+1)+0.001)) .OR. (XINT(L1,L2,J+1) .LE. 0.001))
     &N1 = J
 2100 CONTINUE
      IF (N2 .GE. 1) GO TO 2110
      IF (((B(I) .GE. XINT(L1,L3,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L3,J+1)+0.001)) .OR. (XINT(L1,L3,J+1) .LE. 0.001))
     &N2 = J
 2110 CONTINUE
      IF (N2 .GT. NV2) K4(2,N2) = 1
      IF (N1 .GT. NV1) K4(1,N1) = 1
C
C     DO 2130 AND DO 2120 PERFORM THE POLYGONAL MULTIPLICATION FOR
C     CAP F(X) AND CAP G(X).
C
      LASTJ = K4(2,N2)
      LASTK = K4(1,N1)
      DO 2130 J = 1,LASTJ
      DO 2120 K = 1,LASTK
      IF (N2 .GT. NV2) PAR(2,N2,J) =  1
      IF (N1 .GT. NV1) PAR(1,N1,K) =  1
      K3 = J+K-1
      ZVAL(I,K3) = ZVAL(I,K3)+(PAR(1,N1,K)*PAR(2,N2,J))
 2120 CONTINUE
 2130 CONTINUE
C
C     DO 2140 OBTAINS THE FIRST DERIVATIVE OF THE RESULT OF THE
C     MULTIPLICATION OF CAP F(X) AND CAP G(X) IN THE FORM OF A
C     LITTLE H(X) FOR THAT PRODUCT.
C
      DO 2140 J = 1,4
      ZVAL(I,J) = ZVAL(I,J+1)*FLOAT(J)
      ZVAL(I,J+1) = 0
 2140 CONTINUE
      N1 = 0
      N2 = 0
 2150 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE RESULTS OF THE
C     PARALLEL REDUCTION WITH THE B(0) AND B(1) FORM IN EACH OF 10
C     CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR(L1,L2,NCL)
      GO TO 2180
 2160 DO 2170 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L3,I,1)
      VALUE(L1,L2,I,2) = VALUE(L1,L3,I,2)
      XINT(L1,L2,I) = XINT(L1,L3,I)
 2170 CONTINUE
      XINT(L1,L2,11) = XINT(L1,L3,11)
 2180 VALUE(L1,L2,1,3) = 0
      DO 2210 I = 1,2
      DO 2200 J = 1,10
      DO 2190 K = 1,3
      PAR(I,J,K) = 0
 2190 CONTINUE
 2200 CONTINUE
 2210 CONTINUE
      RETURN
      END
C     END SUBROUTINE PARA
C
C     ***********************************************************
C
C                S U B R O U T I N E      S E R I E S
C
C     ************************************************************
C
      SUBROUTINE SERIES (L1,L2,L3,L4)
      REAL*8 VALUE(200,99,10,3),XINT(200,99,12)
      REAL*8 ZVAL(130,5),XLIM(2),A(130)
      REAL*8 F0,F1,G0,G1,XL
      INTEGER L1,L2,L3,L4
      INTEGER ISEL(2)
      INTEGER I,IK,J,K,NCL,NCL1,NE
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
C
C     SUBROUTINE SERIES PERFORMS THE CONVOLUTION OF TWO PROBABILITY
C     DISTRIBUTIONS BY INTEGRATING THE PRODUCT OF THEIR PIECEWISE
C     POLYGONAL APPROXIMATIONS IN THE FORMS OF F(X) AND G(T-X).
C
C     THIS SECTION DETERMINES THE INTERVALS OF VALIDITY FOR THE
C     CONVOLUTION.
C
C     THE A ARRAY IS USED FOR THE SAME PURPOSE AS THE B ARRAY IN PARA.
C
      K = 0
C
C     DO 3010 CREATES ALL POSSIBLE INTERVALS OF THE NEW DISTRIBUTION
C     BY ADDING THE INTERVALS OF THE TWO DISTRIBUTIONS BEING WORKED.
C
      DO 3010 I = 1,12
      IF ((XINT(L3,L4,I) .LE. 0).AND.(I .GT. 1)) GO TO 3020
      DO 3000 J = 1,12
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) NCL1 = J-2
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) GO TO 3010
      K = K+1
      A(K) = XINT(L1,L2,J)+XINT(L3,L4,I)
 3000 CONTINUE
 3010 CONTINUE
 3020 NINT = I-2
      NCL = K-1
C
C     DO 3120 IS CONTROLLED BY THE NUMBER OF CLASSES IN THE F(X)
C     DISTRIBUTION.  DO 3110 IS CONTROLLED BY THE NUMBER OF CLASSES
C     CREATED BY COMBINING F(X) AND G(T-X).  DO 3100 IS CONTROLLED
C     BY THE NUMBER OF CLASSES IN THE G(T-X) DISTRIBUTION.  THIS
C     ALLOWS THE EVALUATION OF ALL OF THE CREATED CLASSES FOR EVERY
C     CLASS IN BOTH DISTRIBUTIONS.
C
      CALL SORT(NCL)
      DO 3120 K = 1,NCL1
      DO 3110 I = 1,NCL
      DO 3100 J = 1,NINT
      IK = 0
C
C     THIS IF STATEMENT DETERMINES WHICH INTERVALS ARE VALID FOR THE
C     INTERVAL END POINT A(I) BEING EVALUATED AND FOR THE VALUE OF K
C     BEING CONTROLLED BY DO 3120.
C
      IF ((A(I) .GE. XINT(L1,L2,K)+XINT(L3,L4,J)-0.001) .AND. (A(I+1)
     &.LE. XINT(L1,L2,K+1)+XINT(L3,L4,J+1)+0.001)) IK = J
      IF (IK .GE. 1) GO TO 3030
      GO TO 3100
 3030 ISEL(1) = 0
      ISEL(2) = 0
C
C     THE IF STATEMENTS INVOLVING XLIM ARE USED TO DETERMINE THE
C     UPPER AND LOWER LIMITS OF INTEGRATION.  IT IS DETERMINED WHETHER
C     THE LIMIT COMES FROM THE F(X) OR THE G(T-X) DISTRIBUTION.  ISEL
C     IS USED TO DESIGNATE VALUES FROM THE G(T-X) DISTRIBUTION.
C
      IF (XINT(L1,L2,K) .GE. (A(I+1)-XINT(L3,L4,J+1)-0.001)) GO TO 3040
      XLIM(1) = XINT(L3,L4,J+1)
      ISEL(1) = 999
      GO TO 3050
 3040 XLIM(1) = XINT(L1,L2,K)
 3050 IF (XINT(L1,L2,K+1) .LE. (A(I)-XINT(L3,L4,J)+0.001)) GO TO 3060
      XLIM(2) = XINT(L3,L4,J)
      ISEL(2) = 999
      GO TO 3070
 3060 XLIM(2) = XINT(L1,L2,K+1)
 3070 CONTINUE
      DO 3090 NE = 1,2
      F0 = VALUE(L1,L2,K,1)
      F1 = VALUE(L1,L2,K,2)
      G0 = VALUE(L3,L4,IK,1)
      G1 = VALUE(L3,L4,IK,2)
      XL = XLIM(NE)
      Z = 1.0
      IF (NE .EQ. 1) Z = -1.0
      IF (ISEL(NE) .EQ. 999) GO TO 3080
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL AT A FINITE
C     LIMIT.  THE INTEGRATION IS BROKEN DOWN INTO ITS COMPONENT PARTS
C     BY THE POWER OF THE COEFFICIENT THAT RESULTS.  Z CONTROLS THE
C     SIGN OF THE INTEGRAL BASED ON WHETHER THE LOWER OR UPPER LIMIT
C     IS BEING EVALUATED.
C
      ZVAL(I,1) = ZVAL(I,1)+((F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((-1.0*F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+(((F1*G1*XL**2)/2.)+(F0*G1*XL))*Z
      ZVAL(I,3) = ZVAL(I,3)+((-1.0*F0*G1)/2.)*Z
      GO TO 3090
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL FOR LIMITS.
C     IN THE FORM OF (T-X).  THE FORMULAS ARE DIFFERENT BECAUSE
```

```
C     OF THE DIFFERENT POLYNOMIAL CREATED WHEN THE INTEGRATION
C     INVOLVES LIMITS IN THE FORM OF (T-X).
C
 3080 ZVAL(I,1) = ZVAL(I,1)+((-1.0*F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+((-1.0*F1*G0*XL)+(F0*G0)-
     &((F1*G1*XL**2)/2.))*Z
      ZVAL(I,3) = ZVAL(I,3)+((F1*G0)/2.)*Z
      ZVAL(I,4) = ZVAL(I,4)+((F1*G1)/6.)*Z
 3090 CONTINUE
 3100 CONTINUE
 3110 CONTINUE
 3120 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE CONVOLUTION
C     RESULTS WITH THE B(0) AND B(1) FORM IN EACH OF 10 CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR(L1,L2,NCL)
      RETURN
      END
C     END SUBROUTINE SERIES
C
C     ************************************************************
C
C                S U B R O U T I N E           P L O T
C
C     ************************************************************
C
      SUBROUTINE PLOT (IPRINT,KBL,KBM,IFLAG)
      REAL*8 XX(100,2),SORT
      CHARACTER*1 KBL,KBM,LINE(101)
      INTEGER I,IFLAG,IPRINT,J,JPLOT,K,NN
      COMMON/PARA4/XX
C
C     PLOT IS USED TO CREATE THE HISTOGRAM FOR FINAL OUTPUT.
C     THE VARIABLE SORT IN THIS SUBROUTINE IS NOT RELATED TO
C     THE SUBROUTINE SORT.
C
      SORT = XX(1,2)
      DO 4000 I = 2,IPRINT
      IF (SORT .LE. XX(I,2)) SORT = XX(I,2)
 4000 CONTINUE
      PRINT 4900
      IF (IFLAG .EQ. 0) THEN
      PRINT 4910
      ELSE
      PRINT 4915
      END IF
      IF (SORT .GT. 0.5) PRINT 4920
      IF ((SORT .GT. 0.25).AND.(SORT .LE. 0.50)) PRINT 4930
      IF ((SORT .GT. 0.10).AND.(SORT .LE. 0.25)) PRINT 4940
      IF ((SORT .GT. 0.05).AND.(SORT .LE. 0.10)) PRINT 4950
      IF (SORT .LE. 0.05) PRINT 4960
      PRINT 4970
      DO 4030 I = 1,IPRINT
      DO 4010 J = 1,51
      LINE(J) = KBL
 4010 CONTINUE
      IF (SORT .GT. 0.5) JPLOT = (INT((XX(I,2)*50.)+0.5))+1
      IF ((SORT .GT. 0.25).AND.(SORT .LE. 0.50))
     &JPLOT = (INT((XX(I,2)*100.)+0.5))+1
      IF ((SORT .GT. 0.10).AND.(SORT .LE. 0.25))
     &JPLOT = (INT((XX(I,2)*200.)+0.5))+1
      IF ((SORT .GT. 0.05).AND.(SORT .LE. 0.10))
     &JPLOT = (INT((XX(I,2)*500.)+0.5))+1
      IF (SORT .LE. 0.05) JPLOT = (INT((XX(I,2)*1000.0)+0.5))+1
      IF (JPLOT .LE. 0) JPLOT = 1
      IF (JPLOT .GT. 51) JPLOT = 51
      DO 4020 NN = 1,JPLOT
      LINE(NN) = KBM
 4020 CONTINUE
      PRINT 4980,XX(I,1),(LINE(K), K = 1,JPLOT)
 4030 CONTINUE
C
C     FORMAT STATEMENTS
C
 4900 FORMAT (1H1)
 4910 FORMAT (15X,'PROBABILITY DENSITY FUNCTION' //)
 4915 FORMAT (15X,'SIMULATION FREQUENCY HISTOGRAM' //)
 4920 FORMAT (9X,'0        .20       .40       .60       .80       1.0')
 4930 FORMAT (9X,'0        .10       .20       .30       .40       .50')
 4940 FORMAT (9X,'0        .05       .10       .15       .20       .25')
 4950 FORMAT (9X,'0        .02       .04       .06       .08       .10')
 4960 FORMAT (9X,'0        .01       .02       .03       .04       .05')
 4970 FORMAT (9X,'I----+----I----+----I----+----I----+----I----+----I')
 4980 FORMAT (1X,F8.3,2X,51A1)
      RETURN
      END
C     END SUBROUTINE PLOT
C
C     *************************************************************
C
C                S U B R O U T I N E        L I N E A R
C
C     *************************************************************
C
      SUBROUTINE LINEAR (L1,L2,NCL)
      REAL*8 VALUE(200,99,10,3),XINT(200,99,12),ZVAL(130,5),A(130)
      REAL*8 Q,Q1,Q2,STD,SUMX,SUMY,SUMXY,SUMSQ
      REAL*8 ALPHA,AREA,BETA,FACT,SIZE,W,X,XLMBDA,XMEAN
      REAL*8 XMODE,XSIZE,Y
      INTEGER L1,L2
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      EXTERNAL DGAMMA
C
C     SUBROUTINE LINEAR PIECEWISE POLYGONALIZES DISTRIBUTION DATA
C     FROM THE MAIN PROGRAM AND SUBROUTINES PARA AND SERIES WITH
C     THE B(O) AND B(1) FORM IN EACH OF 10 CLASSES THROUGH THE USE
C     OF SIMPLE LINEAR REGRESSION.
```

```
C
      XMODE = VALUE(L1,L2,2,3)
      XMEAN = VALUE(L1,L2,2,3)
      STD = ((VALUE(L1,L2,2,3)-XINT(L1,L2,1))/3.)
      XLMBDA = VALUE(L1,L2,2,3)-XINT(L1,L2,1)
      ALPHA = VALUE(L1,L2,2,3)
      BETA = VALUE(L1,L2,3,3)
      SIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) SIZE = (A(NCL+1)-A(1))/10.
      XINT(L1,L2,11) = XINT(L1,L2,2)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) XINT(L1,L2,11) = A(NCL+1)
      X = XINT(L1,L2,1)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) X = A(1)
      DO 5000 I = 1,10
      XINT(L1,L2,I) = X
      X = X+SIZE
 5000 CONTINUE
      DO 5050 I = 1,10
      X = XINT(L1,L2,I)
      SUMY = 0.
      SUMX = 0.
      SUMXY = 0.
      SUMSQ = 0.
C
C     W CONTROLS THE NUMBER OF DATA POINTS USED IN THE REGRESSION
C     COMPUTATIONS.
C
      W = 10.+IDINT(SIZE*3.)
      XSIZE = SIZE/W
      LASTJ = IDINT(W)
      DO 5040 J = 1,LASTJ
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 99)  GO TO 5030
      DO 5010 K3 = 1,NCL
      K  = 0
      IF ((X .GE. A(K3)).AND.(X .LE. A(K3+1))) K = K3
      IF (K .GE. 1) GO TO 5020
 5010 CONTINUE
C
C     SERIES OR PARA GENERATED DISTRIBUTIONS.
C
 5020 Y = ZVAL(K,1)+(ZVAL(K,2)*X)+(ZVAL(K,3)*(X**2))
     &+(ZVAL(K,4)*(X**3))
 5030 CONTINUE
C
C     TRIANGULAR DISTRIBUTION.
C
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 1) THEN
         IF (XINT(L1,L2,1) .LE. X .AND. X .LE. XMODE) THEN
         Y = (2.*(X-XINT(L1,L2,1)))/((XMODE-XINT(L1,L2,1))*10.*SIZE)
         ELSE
         Y = (2.*(XINT(L1,L2,11)-X))/((XINT(L1,L2,11)-XMODE)*10.*SIZE)
         END IF
C
C     NORMAL  DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 2) THEN
      Y = (1./(STD*2.506628275))*(DEXP((-1.0)*(((X-XMEAN)/STD)**2)/2.))
C
C     EXPONENTIAL DISTRIBUTION  (SHIFTED).
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 3) THEN
      Y = (1./XLMBDA)*(DEXP((-1.0)*((X-XINT(L1,L2,1))/XLMBDA)))
C
C     GAMMA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 4) THEN
      Y = (1./(DGAMMA(ALPHA)*(BETA**ALPHA)))*DEXP(-X/BETA)*(X**(ALPHA-1.
     &))
C
C     BETA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 5) THEN
      Y = (DGAMMA(ALPHA+BETA)/(DGAMMA(ALPHA)*DGAMMA(BETA)))*
     &(1./(10.*SIZE)**(ALPHA+BETA-2.))*
     &((X-XINT(L1,L2,1))**(ALPHA-1.))*
     &((XINT(L1,L2,11)-X)**(BETA-1.))
      END IF
      IF (Y .LT. 0) Y = 0
      SUMX = SUMX+X
      SUMY = SUMY+Y
      SUMXY = SUMXY+(X*Y)
      SUMSQ = SUMSQ+(X**2)
      X = X+XSIZE
 5040 CONTINUE
      VALUE(L1,L2,I,2) = (SUMXY-((SUMX*SUMY)/W))/(SUMSQ-((SUMX**2)/W))
      VALUE(L1,L2,I,1) = (SUMY/W)-(VALUE(L1,L2,I,2)*(SUMX/W))
 5050 CONTINUE
C
C     DO 5060 CALCULATES THE AREA UNDER THE APPROXIMATED DISTRIBUTION.
C     AN ADJUSTMENT FACTOR FOR THE AMOUNT THAT THIS AREA HAS BEEN
C     UNDERESTIMATED OR OVERESTIMATED IS THEREBY DETERMINED.
C
      DO 5060 I = 1,10
      Q  = XINT(L1,L2,I+1)-XINT(L1,L2,I)
      Q1 = (XINT(L1,L2,I)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      Q2 = (XINT(L1,L2,I+1)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      IF (Q1 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q1*(-1.0))
      IF (Q2 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q2*(-1.0))
      IF (Q1 .LT. 0.) Q1 = 0.
      IF (Q2 .LT. 0.) Q2 = 0.
      AREA = AREA+((Q1+Q2)*Q*0.5)
 5060 CONTINUE
      FACT = 1.0/AREA
C
C     DO 5070 ADJUSTS THE COEFFICIENTS OF ALL THE LINEAR POLYNOMIAL
C     PIECES BY THE FACTOR COMPUTED IN DO 5060 IN ORDER TO NORMALIZE
C     THE AREA BACK TO ONE.  THIS ACTS TO REDUCE ACCUMULATING ERRORS
C     DURING PROGRAM COMPUTATIONS.
C
      DO 5070 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)*FACT
      VALUE(L1,L2,I,2) = VALUE(L1,L2,I,2)*FACT
 5070 CONTINUE
      AREA = 0
      DO 5080  I = 1,130
      A(I)  = 0
      ZVAL(I,1) = 0
      ZVAL(I,2) = 0
      ZVAL(I,3) = 0
      ZVAL(I,4) = 0
 5080 CONTINUE
      RETURN
      END
C     END SUBROUTINE LINEAR
C
C     ************************************************************
C
C                 S U B R O U T I N E         S O R T
C
C     ************************************************************
C
      SUBROUTINE SORT (NCL)
      REAL*8 A(130),B
      INTEGER NCL
      INTEGER I,K1
      COMMON/PARA3/A
C
C     SUBROUTINE SORT IS USED TO CONDUCT AN ALGEBRAIC SORT OF DATA
C     CREATED IN THE SERIES AND PARA SUBROUTINES.
C
 6000 K1  = 0
      DO 6010 I = 1,NCL
      IF ((A(I) .LT. (A(I+1)+.01)).AND.(A(I) .GT. (A(I+1)-.01)))
     &GO TO 6020
      IF (A(I) .LT. A(I+1)) GO TO 6010
      IF (A(I) .GT. A(I+1)) B = A(I)
      A(I) = A(I+1)
      A(I+1) = B
      K1=K1+1
 6010 CONTINUE
      IF (K1 .GE. 1) GO TO 6000
      GO TO 6040
 6020 NCL = NCL-1
      LASTJ = NCL+1
      DO 6030 J = I,LASTJ
      A(J) = A(J+1)
      A(J+1) = 0
 6030 CONTINUE
      GO TO 6000
 6040 RETURN
      END
C     END SUBROUTINE SORT
C
C     ************************************************************
C
C                 S U B R O U T I N E       S I M U L T
C
C     ************************************************************
```

```
C
      SUBROUTINE SIMULT (N,NSIM)
      REAL*8 XINT(200,99,12),VALUE(200,99,10,3)
      REAL*8 T(100,99),SIMT(100,10000)
      REAL*8 ALPHA,BETA,X,XLNGTH,XLMBDA,XMAX,XMEAN,XMIN,XMODE
      REAL*8 RN,STD,TTEMP,TMAX
      DIMENSION NET(200,103)
      INTEGER ISIM,N,NDIST,NSIM
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA5/NET
      COMMON/PARA6/SIMT
      EXTERNAL DRNUN,DRNNOR,DRNEXP,DRNGAM,DRNBET,RNSET
C
C     DO 7130 GENERATES A SIMULATED NETWORK THROUGHPUT FOR EACH OF
C     NSIM ITERATIONS OF THE MONTE CARLO SIMULATION OF THE NETWORK.
C
      DO 7130 ISIM=1,NSIM
C
C     DO 7080 GENERATES A RANDOM VALUE FROM THE ACTIVITY RESOURCE
C     CONSUMPTION (ACTIVITY TIME) DISTRIBUTION OF EACH ACTIVITY.
C
      DO 7080 I=1,N-1
      DO 7070 J=1,NET(I,103)
      NDIST = IDINT(VALUE(I,J,1,3))
      XMIN = XINT(I,J,1)
      XMAX = XINT(I,J,11)
      XLNGTH = XMAX-XMIN
      GO TO (7010,7020,7030,7040,7050,7060) NDIST
C
C     TRIANGULAR DISTRIBUTION.
C
 7010 CALL DRNUN(1,RN)
      XMODE = VALUE(I,J,2,3)
      X = (XMODE-XMIN)/XLNGTH
      IF (RN .GT. X) GO TO 7015
      T(I,J) = XMIN+DSQRT(RN*XLNGTH*(XMODE-XMIN))
      GO TO 7070
 7015 T(I,J) = XMAX-DSQRT(XLNGTH*(XMAX-XMODE)*(1.-RN))
      GO TO 7070
C
C     NORMAL DISTRIBUTION.
C
 7020 CALL DRNNOR(1,RN)
      XMEAN = VALUE(I,J,2,3)
      STD = (XMEAN-XMIN)/3.
      T(I,J) = (RN*STD)+XMEAN
      IF ((T(I,J) .LT. XMIN) .OR. (T(I,J) .GT. XMAX)) GO TO 7020
      GO TO 7070
C
C     EXPONENTIAL DISTRIBUTION.
C
 7030 CALL DRNEXP(1,RN)
      XLMBDA = VALUE(I,J,2,3)-XMIN
      T(I,J) = (XLMBDA*RN)+XMIN
      IF (T(I,J) .GT. XMAX) GO TO 7030
      GO TO 7070
```

```
C
C     GAMMA DISTRIBUTION.
C
 7040 ALPHA = VALUE(I,J,2,3)
      BETA = VALUE(I,J,3,3)
      CALL DRNGAM(1,ALPHA,RN)
      T(I,J) = BETA*RN
      IF (T(I,J) .GT. XMAX) GO TO 7040
      GO TO 7070
C
C     BETA DISTRIBUTION.
C
 7050 ALPHA = VALUE(I,J,2,3)
      BETA = VALUE(I,J,3,3)
      CALL DRNBET(1,ALPHA,BETA,RN)
      T(I,J) = XMIN+(XLNGTH*RN)
      GO TO 7070
C
C     UNIFORM DISTRIBUTION.
C
 7060 CALL DRNUN(1,RN)
      T(I,J) = XMIN+(XLNGTH*RN)
 7070 CONTINUE
 7080 CONTINUE
C
C     DO 7120 GENERATES THE CRITICAL PATH TO EACH NODE.  THE
C     SIMULATED TIME THROUGH NODE I FROM SIMULATION ITERATION L
C     IS STORED IN SIMT(I,L).
C
      SIMT(1,ISIM) = 0.0
      DO 7120 I=2,N
      TMAX = 0.0
C
C     DO 7110 DETERMINES THE STARTING NODES AND THE ACTIVITIES WHICH
C     TERMINATE AT NODE I > STARTING NODES, AND COMPUTES THE SIMULATED
C     THROUGHPUT VALUE THROUGH NODE I AS THE MAXIMUM OF THE
C         [(THROUGHPUT VALUE THROUGH STARTING NODE) +
C          (ACTIVITY VALUE FROM STARTING NODE TO NODE I)].
C
      DO 7110 J=1,I-1
      DO 7100 J1=2,NET(J,103)+1
      IF (NET(J,J1)-I) 7100,7090,7100
 7090 TTEMP = SIMT(J,ISIM)+T(J,J1-1)
      IF (TTEMP .LT. TMAX) GO TO 7100
      TMAX = TTEMP
 7100 CONTINUE
 7110 CONTINUE
      SIMT(I,ISIM) = TMAX
 7120 CONTINUE
 7130 CONTINUE
      RETURN
      END
C     END SUBROUTINE SIMULT
C
C     *************************************************************
C
```

```
C                  S U B R O U T I N E       G E N R A N
C
C     *************************************************************
C
      SUBROUTINE GENRAN (N,NACTS)
      DIMENSION NET(200,103),NAF(99),NBE(99)
      REAL*4 DEN,DL,DN,DN2,DN3,UP,X,Y,Y1
      INTEGER I,IJ,J,K,KK,L,N,NI,NO,NN,NACTS,NARCS,NDEL,NDIFF,NFREE,NEM,
     &        NRC
      COMMON/PARA5/NET
      EXTERNAL RNUN
C
C     THIS SUBROUTINE GENERATES A RANDOM ACYCLIC, DIRECTED ACTIVITY
C     NETWORK WITH N NODES AND NACTS ACTIVITIES WITH THE METHOD OF
C     DEMEULEMEESTER, DODIN AND HERROELEN (1993).
C
 8000 DO 8010 I = 1,200
      DO 8005 J = 1,103
      NET(I,J) = 0
 8005 CONTINUE
      NET(I,101) = 1
 8010 CONTINUE
C
C     COMPUTE NUMBER OF ACTIVITIES TO DELETE WITH THE DELETION METHOD.
C
      L = N*(N-1)/2
      NDEL = L-NACTS
C
C     COMPUTE NDIFF SUCH THAT
C         INITIAL NUMBER OF FREE ACTIVITIES = NACTS - NDIFF
C     FOR THE ADDITION METHOD.
C
      NDIFF = (2*N)-4
      DN = REAL(N)
      DL = (REAL(L)/2.0)+1.0
      DN2 = DN*(DN-1.0)
      DN3 = DN+0.5
C
C     IF [N(N-1)/4]+1 < OR = NACTS, CHOOSE THE DELETION METHOD.
C     IF [N(N-1)/4]+1 > NACTS, CHOOSE THE ADDITION METHOD.
C
      IF (DL-NACTS) 8020,8020,8110
C
C     THE DELETION METHOD.
C
 8020 DO 8040 I = 1,N-1
      NET(I,1) = I
      DO 8030 J = I+1,N
      NET(I,J) = J
 8030 CONTINUE
      NET(I,102) = I-1
      NET(I,103) = N-I
 8040 CONTINUE
      NET(N,1) = N
      NET(N,102) = N-1
C
```

```
C     DO 8100 DELETES NDEL RANDOMLY SELECTED ACTIVITIES.
C
 8050 DO 8100 I = 1,NDEL
C
C     CHECK THAT THERE IS AT LEAST ONE ACTIVITY FEASIBLE FOR
C     ACTIVITY DELETION.  IF NOT, RESTART NETWORK GENERATION.
C
      DO 8055 J = 1,N-1
      NO = J
      IF (NET(NO,103) .LT. 2) GO TO 8055
      DO 8054 K = NO+1,N
      IF (NET(K,102) .GE. 2 .AND. NET(NO,K) .EQ. K) GO TO 8060
 8054 CONTINUE
 8055 CONTINUE
      GO TO 8000
C
C     RANDOMLY SELECT THE STARTING NODE (NO) OF THE ACTIVITY TO BE
C     DELETED FROM AMONG THE NODES FEASIBLE FOR ACTIVITY-DELETION.
C
 8060 CALL RNUN(1,Y)
      Y1 = (Y*DN2)+0.25
      X = DN3-SQRT(Y1)
      NO = INT(X)
      IF (NO .GT. X) NO = NO-1
      IF (NET(NO,103) .LT. 2) GO TO 8060
C
C     RANDOMLY SELECT THE ENDING NODE (NI) OF THE ACTIVITY TO BE
C     DELETED FROM AMONG THE NODES FEASIBLE FOR ACTIVITY-DELETION.
C
      K = 0
      DO 8070 J = NO+1,N
      IF (NET(J,102) .LT. 2) GO TO 8070
      IF (NET(NO,J) .EQ. 0) GO TO 8070
      K = K+1
      NAF(K) = J
 8070 CONTINUE
      IF (K .EQ. 0) GO TO 8060
      DEN = 1.0/REAL(K)
      CALL RNUN(1,X)
      DO 8080 J = 1,K
      UP = DEN*REAL(J)
      IF (X .GT. UP) GO TO 8080
      NI = NAF(J)
      GO TO 8090
 8080 CONTINUE
C
C     DELETE THE ACTIVITY FROM NODE NO TO NODE NI.
C
 8090 NET(NO,NI) = 0
      NET(NO,103) = NET(NO,103)-1
      NET(NI,102) = NET(NI,102)-1
 8100 CONTINUE
      GO TO 8250
C
C     THE ADDITION METHOD.
C
```

```
 8110 DO 8120 I = 1,N
      NET(I,1) = I
 8120 CONTINUE
C
C     INITIALIZE NUMBER OF NONRECEIVING NODES (NRC) AND NUMBER OF
C     NONEMITTING NODES (NEM).
C
      NRC = N-3
      NEM = N-3
C
C     ADD ACTIVITIES FROM NODE 1 TO NODE 2 AND FROM NODE N-1 TO NODE N.
C
      NET(1,2) = 2
      NET(N-1,N) = N
      NET(1,103) = 1
      NET(2,102) = 1
      NET(N-1,103) = 1
      NET(N,102) = 1
C
C     INITIALIZE NUMBER OF ACTIVITIES ADDED SO FAR (NARCS).
C
      NARCS = 2
C
C     IF INITIAL NUMBER OF FREE ACTIVITIES
C         [NACTS - (2N-4) = NACTS - NDIFF] IS < OR = 0,
C     THEN ALL NACTS ACTIVITIES TO BE ADDED ARE SUBJECT TO FEASIBILITY
C     CONDITIONS AND CANNOT BE RANDOMLY SELECTED.
C
      IF (NDIFF .GE. NACTS) GO TO 8170
C
C     SET FLAG (KK = 0) THAT INITIAL NUMBER OF FREE ACTIVITIES IS > 0.
C
      KK = 0
C
C     RANDOMLY SELECT THE START NODE (NO) OF THE ACTIVITY TO BE ADDED
C     FROM AMONG THE FEASIBLE NODES.
C
 8130 CALL RNUN(1,Y)
      Y1 = (Y*DN2)+0.25
      X = DN3-SQRT(Y1)
      NO = INT(X)
      IF (NO .GT. X) NO = NO-1
      NN = N-NO
      IF (NET(NO,103) .GE. NN) GO TO 8130
C
C     RANDOMLY SELECT THE END NODE (NI) OF THE ACTIVITY TO BE ADDED
C     FROM AMONG THE FEASIBLE NODES.
C
      K = 0
      DO 8140 J = NO+1,N
      IF (NET(NO,J) .NE. 0) GO TO 8140
      K = K+1
      NAF(K) = J
 8140 CONTINUE
      DEN = 1.0/REAL(K)
      CALL RNUN(1,X)
      UP = 0.0
      DO 8150 J = 1,K
      UP = UP+DEN
      IF (X .GT. UP) GO TO 8150
      NI = NAF(J)
      GO TO 8160
 8150 CONTINUE
C
C     ADD THE ACTIVITY FROM NODE NO TO NODE NI.
C
 8160 NET(NO,NI) = NI
      NARCS = NARCS+1
      IF (NET(NO,103) .EQ. 0) NEM = NEM-1
      IF (NET(NI,102) .EQ. 0) NRC = NRC-1
      NET(NO,103) = NET(NO,103)+1
      NET(NI,102) = NET(NI,102)+1
C
C     IF
C         NUMBER OF ACTIVITIES ADDED SO FAR (NARCS) IS > OR =
C         NUMBER OF ACTIVITIES REQUIRED (NACTS),
C     THEN THE NETWORK IS COMPLETE.
C
      IF (NARCS .GE. NACTS) GO TO 8250
C
C     IF THE FLAG (KK) INDICATES THAT
C         NUMBER OF NONRECEIVING NODES (NRC) = 0, AND
C         NUMBER OF NONEMITTING NODES (NEM) = 0, I.E.
C     FEASIBILITY REQUIREMENTS ARE MET, THEN THE REMAINING ACTIVITIES
C     TO BE ADDED ARE FREE ACTIVITIES AND ARE TO BE RANDOMLY SELECTED.
C
      IF (KK .EQ. 1) GO TO 8130
C
C     IF
C         NUMBER OF FREE ACTIVITIES (NFREE) IS > 0,
C     THEN THE NEXT ACTIVITY TO BE ADDED IS A FREE ACTIVITY AND IS TO BE
C     RANDOMLY SELECTED.
C
      NFREE = NACTS-NARCS-NRC-NEM
      IF (NFREE .GT. 0) GO TO 8130
C
C     IF
C         NUMBER OF FREE ACTIVITIES (NFREE) = 0, AND
C         NUMBER OF NONRECEIVING NODES (NRC) = 0,
C     THEN CHECK THE NUMBER OF NONEMITTING NODES (NEM).
C
 8170 IF (NRC .EQ. 0) GO TO 8200
C
C     IF NOT, ADD ACTIVITIES SO AS TO REDUCE THE NUMBER OF NONRECEIVING
C     NODES (NRC) TO 0.
C
      K = 0
      DO 8180 I = 3,N-1
      IF (NET(I,102) .GT. 0) GO TO 8180
      K = K+1
      NAF(K) = I
 8180 CONTINUE
```

```
      IF (K .EQ. 0) GO TO 8200
      DO 8190 I =1,K
      IJ = K+1-I
      NI = NAF(IJ)
      CALL RNUN(1,Y)
      X = 1.0+(REAL(NI-1)*Y)
      NO = INT(X)
      IF (NO .GT. X) NO = NO-1
      NET(NO,NI) = NI
      NARCS = NARCS+1
      NET(NO,103) = NET(NO,103)+1
      NET(NI,102) = NET(NI,102)+1
 8190 CONTINUE
C
C     IF
C         NUMBER OF NONEMITTING NODES (NEM) = 0,
C     THEN THE FEASIBILITY REQUIREMENTS ARE MET.
C
 8200 IF (NEM .EQ. 0) GO TO 8230
C
C     IF NOT, ADD ACTIVITIES SO AS TO REDUCE THE NUMBER OF NONEMITTING
C     NODES (NEM) TO 0.
C
      K = 0
      DO 8210 I = 2,N-2
      IF (NET(I,103) .GT. 0) GO TO 8210
      K = K+1
      NBE(K) = I
 8210 CONTINUE
      IF (K .EQ. 0) GO TO 8230
      DO 8220 I = 1,K
      NO = NBE(I)
      CALL RNUN(1,X)
      Y = REAL(NO+1)+(REAL(N-NO)*X)
      NI = INT(Y)
      IF (NI .GT. Y) NI = NI-1
      NET(NO,NI) = NI
      NARCS = NARCS+1
      NET(NO,103) = NET(NO,103)+1
      NET(NI,102) = NET(NI,102)+1
 8220 CONTINUE
C
C     SET FLAG (KK = 1) THAT FEASIBILITY REQUIREMENTS HAVE BEEN MET.
C
 8230 KK = 1
C
C     IF NUMBER OF ACTIVITIES ADDED SO FAR (NARCS) IS
C         < NUMBER OF ACTIVITIES REQUIRED (NACTS), THEN RANDOMLY SELECT
C           THE NEXT ACTIVITY TO BE ADDED,
C         = NACTS, THEN THE NETWORK IS COMPLETE,
C         > NACTS, THEN USE THE DELETION METHOD TO DELETE EXCESS
C           ACTIVITIES.
C
      IF (NARCS-NACTS) 8130,8250,8240
 8240 NDEL = NARCS-NACTS
      GO TO 8050
```

```
C
C     RECONFIGURE NET ARRAY.
C
 8250 DO 8270 I = 1,N-1
      K = 2
      DO 8260 J = I+1,N
      IF (NET(I,J) .EQ. 0) GO TO 8260
      NET(I,K) = NET(I,J)
      IF (K .LT. J) NET(I,J) = 0
      K = K+1
 8260 CONTINUE
 8270 CONTINUE
      RETURN
      END
C     END SUBROUTINE GENRAN
C
C     **************************************************************
C
C               S U B R O U T I N E        C P U T I M E
C
C     **************************************************************
C
      SUBROUTINE CPUTIME(CPTIME)
      REAL*4 CPTIME
      TYPE TB_TYPE
        SEQUENCE
          REAL*4 USRTIME
          REAL*4 SYSTIME
      END TYPE
      TYPE (TB_TYPE) DTIME_SRC
      CPTIME = DTIME_(DTIME_SRC)
      RETURN
      END
C     END SUBROUTINE CPUTIME
C
C     **************************************************************
C
C               S U B R O U T I N E         T I M E R
C
C     **************************************************************
C
      SUBROUTINE TIMER(DELTA)
      REAL*4  DELTA,CPU2
      CALL CPUTIME(CPU2)
      DELTA = CPU2
      RETURN
      END
C     END SUBROUTINE TIMER
```

APPENDIX E

VALIDATION VERSION OF PART PROGRAM WITH SEQUENTIAL APPROXIMATION

```
C
C
C                           VALIDATION VERSION
C                                 OF
C             POLYGONAL APPROXIMATION AND REDUCTION TECHNIQUE
C                                 (PART)
C                               ALGORITHM
C                                  FOR
C                       ACYCLIC, DIRECTED NETWORKS
C                                 USING
C                   "SEQUENTIAL APPROXIMATION" METHOD
C
C
C     THIS PROGRAM GENERATES "STRONGLY RANDOMIZED NETWORKS," REDUCES
C     THEM WITH THE PART ALGORITHM USING "SEQUENTIAL APPROXIMATION,"
C     SIMULATES THEM, AND OUTPUTS STATISTICAL COMPARISONS OF THE
C     PART-APPROXIMATED AND SIMULATED NETWORK THROUGHPUT DISTRIBUTIONS.
C     THE PROGRAM IS WRITTEN IN FORTRAN 77 AND IS PRESENTLY DESIGNED
C     TO BE OPERATED IN A TIME SHARING MODE WITH ALL DATA INPUT FROM
C     TWO (2) DATA FILES.  THE PROGRAM DIRECTS OUTPUT IN NINE (9)
C     OPTIONAL FORMATS TO A TIME SHARING TERMINAL.  IF DESIRED, THE
C     READ STATEMENTS AT THE BEGINNING OF THE MAIN PROGRAM CAN BE
C     MODIFIED TO ALLOW DATA INPUT DIRECTLY FROM THE TIME SHARING
C     TERMINAL.
C
C     THE CURRENT DIMENSIONS OF THE PROGRAM ALLOW A NETWORK WITH A
C     MAXIMUM OF 100 NODES AND A MAXIMUM OF 99 ACTIVITIES BEGINNING
C     AT EACH NODE.  THESE LIMITS CAN BE EXPANDED BY CHANGING THE
C     DIMENSIONS OF THE XINT AND VALUE ARRAYS.
C
C
C     ************************************************************
C
C                  O P E R A T I N G   I N S T R U C T I O N S
C
C     ************************************************************
C
C     INSTRUCTIONS FOR BUILDING DATA FILES
C     ------------------------------------
C
C           DATA FILE DATAH.VAL
C
C     THIS DATA FILE CONTAINS DESCRIPTIONS OF THE PRECODED DISTRIBUTIONS
C     OF ACTIVITY DURATION.
C
C     THERE ARE 5 FIELDS OF DATA.
C     FIELD 1 IS THE CODE FOR THE TYPE OF DISTRIBUTION.
C         1 = TRIANGULAR DISTRIBUTION
C         2 = NORMAL DISTRIBUTION
C         3 = EXPONENTIAL DISTRIBUTION
C         4 = GAMMA DISTRIBUTION
C         5 = BETA DISTRIBUTION
C         6 = UNIFORM DISTRIBUTION
C     FIELD 2 IS
C         MODE FOR A TRIANGULAR DISTRIBUTION.
C         MEAN FOR A NORMAL DISTRIBUTION.
```

```
C         MEAN FOR AN EXPONENTIAL DISTRIBUTION.
C         ALPHA FOR A GAMMA OR A BETA DISTRIBUTION.
C         1/(B-A) FOR A UNIFORM DISTRIBUTION.
C     FIELD 3 IS BETA FOR A GAMMA OR A BETA DISTRIBUTION.
C     FIELD 4 IS THE MINIMUM VALUE OF THE DISTRIBUTION.
C     FIELD 5 IS THE MAXIMUM VALUE OF THE DISTRIBUTION.
C
C
C           DATA FILE CONTROL.VAL
C
C     THIS IS A SINGLE LINE DATA FILE WHICH CONTAINS CONTROL
C     PARAMETERS FOR INPUT, OUTPUT, AND MONTE CARLO SIMULATION.
C
C     THERE ARE 7 FIELDS OF DATA.
C     FIELD 1 IS THE NUMBER OF NETWORKS TO BE GENERATED (MAXIMUM = 100).
C     FIELD 2 IS THE NUMBER OF NODES IN THE NETWORK.
C         0 = NUMBER OF NODES IS TO BE RANDOMLY GENERATED.
C     FIELD 3 IS THE NUMBER OF ACTIVITIES IN THE NETWORK.
C         0 = NUMBER OF ACTIVITIES IS TO BE RANDOMLY GENERATED.
C     FIELD 4 IS THE OUTPUT OPTION DESIRED FOR THE PART RESULTS.
C         1 = A DESCRIPTION OF EACH OF THE 10 CLASSES OF THE
C             FINAL DISTRIBUTION IN THE FORM OF Y = B(O) + B(1) X.
C         2 = A CUMULATIVE DISTRIBUTION FUNCTION OF THE FINAL
C             DISTRIBUTION.
C         3 = A DISCRETE PROBABILITY DENSITY FUNCTION AND A
C             SIMULATION FREQUENCY HISTOGRAM IN GRAPHICAL FORMAT.
C         4 = A COMBINATION OF 1 AND 2 ABOVE.
C         5 = A COMBINATION OF 1 AND 3 ABOVE.
C         6 = A COMBINATION OF 2 AND 3 ABOVE.
C         7 = A COMBINATION OF 1, 2, AND 3 ABOVE.
C         8 = ONLY THE EXPECTED VALUE AND STANDARD DEVIATION.
C         9 = ONLY STATISTICAL COMPARISONS.
C     FIELD 5 IS THE NUMBER OF ITERATIONS OF THE MONTE CARLO
C     SIMULATION REQUESTED (MAXIMUM = 10,000).
C         0 = NO MONTE CARLO SIMULATION IS REQUESTED.
C     FIELD 6 IS THE NUMBER OF PRECODED DISTRIBUTIONS (MAXIMUM = 20).
C     FIELD 7 IS THE PROBABILITY THAT A NODE IS ON THE OUTPUT CRITICAL
C        LIST.
C
C
C     NOTE
C
C     ALL UNUSED FIELDS MUST BE ZEROED OUT.
C
C
C
C     **********************************************************
C
C                  M A I N     P R O G R A M
C
C     **********************************************************
C
      REAL*8 XINT(101,100,12),VALUE(101,100,10,3),A(130)
      REAL*8 ZVAL(130,5),XX(100,2),TOTAAR(51),COMPAR(100,4)
      REAL*8 SIMT(100,10000),SIMTOT(51),DIST(20,5)
      REAL*8 AREA,AVG,COUNT,SIG,SIZE
```

```
     *                ,X,XSIZE
     *                ,DIFF
      REAL*4 HIGH,HLOW,PERCNT,KSCR20,KSCR10,KSCR05,KSCR02,KSCR01,DMAX
     *    ,AMEAN,ASTD,PROB,RN,STEP,UP,XT
      INTEGER I,IEDN,ICOUNT,IACT,IOCL,IPRE,IPRINT,ISNODE,ISEED
     *       ,ICODED,IFLAG,IFLAG1
     *       ,MM
     *       ,N,NACTS,NAN,NCL,NET,NSIM,NGEN,NGENCT,NCODED,NACTSS,NSTART
     *       ,J,J1
     *       ,K,KK
     *       ,L,L1,L2,L3,LASTK,L3COUNT,LA
     *       ,UA
      REAL*4 DELTA,TOTTIM
      DIMENSION NET(100,103),IOCL(100),IPRE(99,2)
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      COMMON/PARA4/XX
      COMMON/PARA5/NET
      COMMON/PARA6/SIMT
      CHARACTER*1 KBL,KBM
      DATA KBL/' '/,KBM/'*'/
      DATA NCL/0/
      DATA TOTTIM/0.0/
      EXTERNAL RNSET,RNNOR,RNUN
C
C     INITIALIZE RANDOM NUMBER GENERATOR.
C

      ISEED = 123456789
      CALL RNSET(ISEED)
C
C     OPEN INPUT AND OUTPUT FILES
C
      OPEN (UNIT = 12, FILE = 'datah.val')
      OPEN (UNIT = 13, FILE = 'control.val')
C
C     READ CONTROL INFORMATION.
C
      READ (13,1900) NGEN,N,NACTS,NAN,NSIM,NCODED,PROB
      NSTART = N
      NACTSS = NACTS
C
C     DO 0900 READS DISTRIBUTION DATA FROM DATAH.VAL AND LOADS IT
C     INTO THE DIST ARRAY.
C
      DO 0900 I = 1,NCODED
      READ (12,1902) (DIST(I,J), J=1,5)
 0900 CONTINUE
      STEP = 1.0/REAL(NCODED)
C
C     DO 1540 GENERATES, REDUCES, SIMULATES, AND STATISTICALLY
C     COMPARES NGEN "STRONGLY RANDOMIZED NETWORKS."
C
      DO 1540 NGENCT = 1,NGEN
      CALL TIMER(DELTA)
      N = NSTART
      NACTS = NACTSS
C
C     RANDOMLY GENERATE THE NUMBER OF NODES (N), IF NECESSARY.
C
      IF (N .GT. 0) GO TO 0910
      CALL RNUN(1,RN)
      XT = 2.0+(99.0*RN)
      N = INT(XT)
      IF (N .GT. 100) N = 100
C
C     RANDOMLY GENERATE THE NUMBER OF ACTIVITIES (NACTS), IF NECESSARY.
C
 0910 IF (NACTS .GT. 0) GO TO 0920
      LA = N-1
      UA = N*(N-1)/2
      AMEAN = ((REAL(LA+UA))/2.)-(((REAL(UA-LA))**2)/500.0)
      ASTD = (REAL(UA-LA))/2.5
      CALL RNNOR(1,RN)
      XT = (RN*ASTD)+AMEAN
      NACTS = INT(XT)
      IF (NACTS .LT. LA) NACTS = LA
      IF (NACTS .GT. UA) NACTS = UA
C
C     RANDOMLY GENERATE THE NETWORK (NET ARRAY).
C
 0920 CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
      CALL GENRAN(N,NACTS)
      CALL TIMER(DELTA)
C
C     RANDOMLY GENERATE THE OUTPUT CRITICAL LIST (IOCL).
C
      DO 0930 I = 1,N-1
      IOCL(I) = 0
      IF (PROB .EQ. 0.0) GO TO 0930
      CALL RNUN(1,RN)
      IF (RN .LT. PROB) IOCL(I) = 1
 0930 CONTINUE
      IOCL(N) = 1
C
C     DO 1025 RANDOMLY SELECTS ONE OF THE PRECODED DISTRIBUTIONS
C     FOR EACH ACTIVITY AND LOADS THE DISTRIBUTION'S DATA INTO
C     THE VALUE AND XINT ARRAYS.  THIS DO ALSO DETERMINES IF
C     THE ACTIVITY DISTRIBUTION IS OTHER THAN UNIFORM, AND,
C     IF SO, CALLS LINEAR TO APPROXIMATE IT WITH A
C     PIECEWISE POLYGONAL FUNCTION.
C
      DO 1025 I = 1,N-1
      L1 = I
      DO 1020 J = 1,NET(I,103)
      L2 = J
      CALL RNUN(1,RN)
      UP = 0.0
      DO 1000 K = 1,NCODED
      UP = UP+STEP
      IF (RN .GT. UP ) GO TO 1000
      ICODED = K
      GO TO 1010
 1000 CONTINUE
 1010 VALUE(L1,L2,1,3) = DIST(ICODED,1)
      VALUE(L1,L2,2,3) = DIST(ICODED,2)
      VALUE(L1,L2,3,3) = DIST(ICODED,3)
      XINT(L1,L2,1) = DIST(ICODED,4)
      XINT(L1,L2,2) = DIST(ICODED,5)
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 6) THEN
      CALL LINEAR(L1,L2,NCL)
      GO TO 1020
C
C     DO 1015 CONVERTS DATA FOR UNIFORM DISTRIBUTIONS INTO A USABLE
C     FORM FOR SUBROUTINES SERIES AND PARA.
C
      ELSE
      XINT(L1,L2,11) = XINT(L1,L2,2)
      X = XINT(L1,L2,1)
      XSIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      DO 1015 K = 1,10
      VALUE(L1,L2,K,1) = VALUE(L1,L2,2,3)
      VALUE(L1,L2,K,2) = 0.0
      XINT(L1,L2,K) = X
      X = X+XSIZE
 1015 CONTINUE
      END IF
 1020 CONTINUE
 1025 CONTINUE
C
C     MONTE CARLO SIMULATION OF THE NETWORK.
C
      IF (NSIM .EQ. 0) GO TO 1030
      CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
      CALL SIMULT(N,NSIM)
      CALL TIMER(DELTA)
C
C     REDUCTION OF THE NETWORK BEGINS.
C
C     DO 1040 CHECKS IF A CONVOLUTION (SERIES-REDUCTION) OPERATION
C     IS POSSIBLE, i.e., IF THERE EXISTS A NODE I NOT ON THE OUTPUT
C     CRITICAL LIST SUCH THAT
C         IN-DEGREE NODE I = OUT-DEGREE NODE I = 1.
C
 1030 L3COUNT = 2
 1035 DO 1040 I=L3COUNT,N-1
      L3 = I
      IF ((NET(I,102)+NET(I,103)) .EQ. 2 .AND. IOCL(I) .EQ. 0)
     &GO TO 1050
 1040 CONTINUE
C
C     IF (IN-DEGREE NODE I + OUT-DEGREE NODE I) > 2 FOR ALL I NOT = 1
C     OR N, NETWORK IS NONSEPARABLE, SO PROCEED TO "SEQUENTIAL
C     APPROXIMATION"
C
```

```
      IF (L3COUNT .EQ. 2) GO TO 1145
      GO TO 1080
C
C     A CONVOLUTION IS POSSIBLE WITH THE TWO ACTIVITIES, ONE OF WHICH
C     TERMINATES AT NODE L3 AND THE OTHER OF WHICH STARTS AT NODE L3.
C     DO 1060 IDENTIFIES THE STARTING NODE NUMBER AND THE ACTIVITY
C     NUMBER OF THE ACTIVITY TERMINATING AT NODE L3.  THEN THE SERIES
C     SUBNETWORK CONSISTING OF THESE TWO ACTIVITIES IS CONVOLUTED INTO
C     AN EQUIVALENT ACTIVITY.
C
 1050 DO 1060 I=1,L3-1
      DO 1060 J=2,NET(I,103)+1
      L1 = I
      L2 = J-1
      IF (NET(I,J) .EQ. L3) GO TO 1070
 1060 CONTINUE
 1070 CALL SERIES(L1,L2,L3,1)
      NET(L1,L2+1) = NET(L3,2)
      NET(L3,2) = 0
      NET(L3,101) = 0
      NET(L3,102) = 0
      NET(L3,103) = 0
      L3COUNT = L3+1
      IF (L3COUNT .EQ. N) GO TO 1080
      GO TO 1035
C
C     DO 1140 CHECKS IF A MAXIMUM (PARALLEL-REDUCTION) OPERATION IS
C     POSSIBLE, i.e., IF THERE EXIST TWO DIFFERENT ACTIVITIES, A1 AND
C     A2, SUCH THAT
C         STARTING NODE (A1) = STARTING NODE (A2), AND
C           ENDING NODE (A1) = ENDING NODE (A2).
C     THEN THE PARALLEL SUBNETWORK CONSISTING OF THESE TWO ACTIVITIES
C     IS PARALLEL-REDUCED WITH A MAXIMUM OPERATION INTO AN EQUIVALENT
C     ACTIVITY.
C
 1080 DO 1140 I=1,N-1
      L1 = I
 1085 DO 1090 J=2,NET(L1,103)
      L2 = J-1
      IF (NET(L1,J) .EQ. 0) GO TO 1140
      DO 1090 K=J+1,NET(L1,103)+1
      L3 = K-1
      IF (NET(L1,J) .EQ. NET(L1,K)) THEN
      IEDN = NET(L1,J)
      GO TO 1110
      ELSE
      GO TO 1090
      END IF
 1090 CONTINUE
      GO TO 1140
 1110 CALL PARA(L1,L2,L3)
      NET(L1,103) = NET(L1,103)-1
      NET(IEDN,102) = NET(IEDN,102)-1
      DO 1120 K=L3,NET(L1,103)
      NET(L1,K+1) = NET(L1,K+2)
      DO 1115 L=1,10
      XINT(L1,K,L)= XINT(L1,K+1,L)
      VALUE(L1,K,L,1) = VALUE(L1,K+1,L,1)
      VALUE(L1,K,L,2) = VALUE(L1,K+1,L,2)
 1115 CONTINUE
      XINT(L1,K,11) = XINT(L1,K+1,11)
 1120 CONTINUE
      K = NET(L1,103)+1
      NET(L1,K+1) = 0
      DO 1130 L=1,10
      XINT(L1,K,L) = 0.
      VALUE(L1,K,L,1) = 0.
      VALUE(L1,K,L,2) = 0.
 1130 CONTINUE
      XINT(L1,K,11) = 0.
      GO TO 1085
 1140 CONTINUE
      GO TO 1030
C
C     THROUGH 1146 CHECKS IF THE NETWORK HAS BEEN SERIES-PARALLEL
C     REDUCED TO A SINGLE EQUIVALENT ACTIVITY.
C
 1145 IF (NET(1,103) .NE. 1) GO TO 1150
      IF (NET(N,102) .NE. 1) GO TO 1150
      DO 1146 I=2,N-1
      IF (NET(I,102) + NET(I,103)) 1560,1146,1150
 1146 CONTINUE
C
C     THE NETWORK HAS BEEN SERIES-PARALLEL REDUCED TO A SINGLE EQUIVA-
C     LENT ACTIVITY.  DO 1147 LOADS THE DISTRIBUTION OF THIS ACTIVITY
C     INTO THE 100TH ACTIVITY POSITION OF NODE N.
C
      DO 1147 J=1,10
      XINT(N,100,J) = XINT(1,1,J)
      VALUE(N,100,J,1) = VALUE(1,1,J,1)
      VALUE(N,100,J,2) = VALUE(1,1,J,2)
 1147 CONTINUE
      XINT(N,100,11) = XINT(1,1,11)
      GO TO 1240
C
C     DO 1220 REDUCES THE NONSEPARABLE NETWORK USING THE "SEQUENTIAL
C     APPROXIMATION" METHOD.
C
 1150 DO 1220 I=2,N
      ICOUNT = 0
      IF (NET(I,101)) 1560,1220,1155
C
C     DO 1170 DETERMINES THE STARTING NODE NUMBER AND THE ACTIVITY
C     NUMBER OF ALL THE ACTIVITIES WHICH TERMINATE AT NODE I > STARTING
C     NODE.
C
 1155 DO 1170 J=1,I-1
      IF (NET(J,101)) 1560,1170,1160
 1160 DO 1169 J1=2,NET(J,103)+1
      IF (NET(J,J1) - I) 1169,1165,1169
 1165 ICOUNT = ICOUNT+1
      IPRE(ICOUNT,1) = J
      IPRE(ICOUNT,2) = J1-1
 1169 CONTINUE
 1170 CONTINUE
C
C     THROUGH 1220 CONVOLVES THE RESOURCE CONSUMPTION DISTRIBUTION
C     THROUGH THE STARTING NODE OF THE ACTIVITY AND THE RESOURCE
C     CONSUMPTION DISTRIBUTION OF EACH ACTIVITY WHICH TERMINATES
C     AT NODE I AND THEN FINDS THE MAXIMUM OF THESE CONVOLUTIONS.
C
C     IF THE FIRST STARTING NODE = NODE 1, THE CONVOLUTION IS EQUAL TO
C     THE DISTRIBUTION OF THE ACTIVITY WHICH TERMINATES AT NODE I.
C
      IF (IPRE(1,1) .EQ. 1) THEN
      IACT = IPRE(1,2)
      DO 1175 J=1,10
      XINT(101,1,J) = XINT(1,IACT,J)
      VALUE(101,1,J,1) = VALUE(1,IACT,J,1)
      VALUE(101,1,J,2) = VALUE(1,IACT,J,2)
 1175 CONTINUE
      XINT(101,1,11) = XINT(1,IACT,11)
C
C     OTHERWISE, LOAD THE DISTRIBUTION THROUGH THE FIRST STARTING NODE
C     INTO TEMPORARY LOCATION 1.
C
      ELSE
      ISNODE = IPRE(1,1)
      IACT = IPRE(1,2)
      DO 1180 J=1,10
      XINT(101,1,J) = XINT(ISNODE,100,J)
      VALUE(101,1,J,1) = VALUE(ISNODE,100,J,1)
      VALUE(101,1,J,2) = VALUE(ISNODE,100,J,2)
 1180 CONTINUE
      XINT(101,1,11) = XINT(ISNODE,100,11)
C
C     LOAD THE DISTRIBUTION OF THE FIRST ACTIVITY TERMINATING AT NODE I
C     IN TEMPORARY LOCATION 2.
C
      DO 1185 J=1,10
      XINT(101,2,J) = XINT(ISNODE,IACT,J)
      VALUE(101,2,J,1) = VALUE(ISNODE,IACT,J,1)
      VALUE(101,2,J,2) = VALUE(ISNODE,IACT,J,2)
 1185 CONTINUE
      XINT(101,2,11) = XINT(ISNODE,IACT,11)
C
C     CONVOLVE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2 AND
C     PLACE THE CONVOLUTION IN TEMPORARY LOCATION 1.
C
      CALL SERIES(101,1,101,2)
      END IF
C
C     IF THERE IS ONLY ONE ACTIVITY TERMINATING AT NODE I, THE
DISTRIBUTION
C     THROUGH NODE I IS THE CONVOLUTION IN TEMPORARY LOCATION 1.  LOAD
THIS
C     INTO THE 100TH ACTIVITY POSITION OF NODE I.
C
```

```
      IF (ICOUNT .EQ. 1) THEN
      DO 1190 J=1,10
      XINT(I,100,J) = XINT(101,1,J)
      VALUE(I,100,J,1) = VALUE(101,1,J,1)
      VALUE(I,100,J,2) = VALUE(101,1,J,2)
 1190 CONTINUE
      XINT(I,100,11) = XINT(101,1,11)
C
C     IF THERE ARE TWO OR MORE ACTIVITIES TERMINATING AT NODE I, LOAD
THE
C     DISTRIBUTION THROUGH THE STARTING NODE OF THE NEXT ACTIVITY INTO
C     TEMPORARY LOCATION 3.
C
      ELSE
      DO 1205 K=2,ICOUNT
      ISNODE = IPRE(K,1)
      IACT = IPRE(K,2)
      DO 1195 J=1,10
      XINT(101,3,J) = XINT(ISNODE,100,J)
      VALUE(101,3,J,1) = VALUE(ISNODE,100,J,1)
      VALUE(101,3,J,2) = VALUE(ISNODE,100,J,2)
 1195 CONTINUE
      XINT(101,3,11) = XINT(ISNODE,100,11)
C
C     THEN LOAD THE DISTRIBUTION OF THE NEXT ACTIVITY INTO TEMPORARY
C     LOCATION 4.
C
      DO 1200 J=1,10
      XINT(101,4,J) = XINT(ISNODE,IACT,J)
      VALUE(101,4,J,1) = VALUE(ISNODE,IACT,J,1)
      VALUE(101,4,J,2) = VALUE(ISNODE,IACT,J,2)
 1200 CONTINUE
      XINT(101,4,11) = XINT(ISNODE,IACT,11)
C
C     CONVOLUTE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 3 AND 4 AND
C     LOAD THE CONVOLUTION INTO TEMPORARY LOCATION 3.
C
      CALL SERIES(101,3,101,4)
C
C     PARALLEL-REDUCE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 3
AND
C     LOAD THE MAXIMUM INTO TEMPORARY LOCATION 1.
C
      CALL PARA(101,1,3)
 1205 CONTINUE
C
C     THE DISTRIBUTION THROUGH NODE I IS THE MAXIMUM IN TEMPORARY
LOCATION 1.
C     LOAD THIS INTO THE 100TH ACTIVITY POSITION OF NODE I.
C
      DO 1210 J=1,10
      XINT(I,100,J) = XINT(101,1,J)
      VALUE(I,100,J,1) = VALUE(101,1,J,1)
      VALUE(I,100,J,2) = VALUE(101,1,J,2)
 1210 CONTINUE
      XINT(I,100,11) = XINT(101,1,11)
      END IF
 1220 CONTINUE
C
C     THE DISTRIBUTION THROUGH NODE N IS THE FINAL EQUIVALENT ACTIVITY
OF THE
C     NETWORK.
C
 1240 CONTINUE
      IF (NAN .NE. 9) THEN
      PRINT 1910
      PRINT 1915,NGENCT,NGEN
      ELSE
      GO TO 1245
      END IF
C
C     IF NODE I IS ON THE OUTPUT CRITICAL LIST, DO 1385 PRESENTS THE
C     DISTRIBUTION THROUGH NODE I IN THE OUTPUT.
C
 1245 IFLAG = 0
      DO 1385 I = 2,N
      IF (IOCL(I) .EQ. 0) GO TO 1385
      IF ((IFLAG .EQ. 1) .AND. (NAN .NE. 9)) PRINT 1910
      IFLAG = 1
      L = 1
      KK = 0
      DO 1270 J = 1,10
C
C     THE XX ARRAY IS USED FOR HISTOGRAM AND CDF CALCULATIONS.
C
      XX(1,1) = XINT(I,100,J)
      SIZE = (XINT(I,100,2)-XINT(I,100,1))/5.
      LASTK = L+4
      DO 1250 K = L,LASTK
      KK = KK+1
      XX(K,2) = VALUE(I,100,J,1)+(VALUE(I,100,J,2)*XX(K,1))
      IF ((KK .LE. 1).AND.(L .GT. 4)) XX(K,2) = (((VALUE(I,100,J,2)
     &*XX(K,1))+VALUE(I,100,J,1))+(VALUE(I,100,J-1,2)*XX(K,1))+
     &VALUE(I,100,J-1,1))/2.
      XX(K+1,1) = XX(K,1)+SIZE
1250  CONTINUE
      KK = 0
      L = L+5
      IF ((NAN .EQ. 1).OR.(NAN .EQ. 4).OR.(NAN .EQ. 5).OR.
     &(NAN .EQ. 7)) GO TO 1260
      GO TO 1270
 1260 IF (I .NE. N .AND. J .EQ. 1) THEN
      PRINT 1920,I
      ELSE IF (I .EQ. N .AND. J .EQ. 1) THEN
      PRINT 1925
      END IF
      PRINT 1930,J,XINT(I,100,J),XINT(I,100,J+1)
      PRINT 1940,VALUE(I,100,J,1),VALUE(I,100,J,2)
 1270 CONTINUE
      XX(51,2)=VALUE(I,100,10,1)+VALUE(I,100,10,2)*XX(51,1)
C
C     TOTAAR IS USED FOR CDF CALCULATIONS.
```

```
C
      AREA = 0.0
      DO 1280 J = 1,50
      AREA = AREA+((XX(J,2)+XX(J+1,2))*SIZE*.5)
      TOTAAR(J) = AREA
 1280 CONTINUE
      AREA = 1.0/AREA
      DO 1290 J = 1,50
      TOTAAR(J) = TOTAAR(J)*AREA
 1290 CONTINUE
      DO 1295 J = 51,2,-1
      TOTAAR(J) = TOTAAR(J-1)
 1295 CONTINUE
      TOTAAR(1) = 0.0
      XX(1,1) = XINT(I,100,1)
      IF (NAN .EQ. 9) GO TO 1350
      IF ((NAN .EQ. 2).OR.(NAN .EQ. 4).OR.(NAN .EQ. 6).OR.
     &(NAN .EQ. 7)) GO TO 1300
      GO TO 1320
 1300 PRINT 1910
      PRINT 1950
      DO 1310 J = 1,51
      PRINT 1960,XX(J,1),TOTAAR(J)
 1310 CONTINUE
 1320 IF ((NAN .EQ. 3).OR.(NAN .EQ. 5).OR.(NAN .EQ. 6).OR.
     &(NAN .EQ. 7)) GO TO 1330
      GO TO 1340
 1330 IPRINT = 51
      IFLAG1 = 0
      CALL PLOT(IPRINT,KBL,KBM,IFLAG1)
 1340 CONTINUE
C
C     DO 1360 COMPUTES AN APPROXIMATED EXPECTED VALUE AND
C     DO 1370 COMPUTES AN APPROXIMATED STANDARD DEVIATION
C     USING GROUPED DATA.
C
 1350 AVG = 0.0
      SIG = 0.0
      DO 1360 MM = 1,50
      AVG = AVG+((XX(MM,1)+(SIZE/2.))*(TOTAAR(MM+1)-TOTAAR(MM)))
 1360 CONTINUE
      COMPAR(NGENCT,3) = AVG
      DO 1370 MM = 1,50
      SIG = SIG+(((XX(MM,1)+(SIZE/2.)-AVG)**2)*(TOTAAR(MM+1)-
     &TOTAAR(MM)))
 1370 CONTINUE
      SIG = DSQRT(SIG)
      COMPAR(NGENCT,4) = SIG
      IF (NAN .EQ. 9) GO TO 1385
      PRINT 1910
      PRINT 1970,AVG,SIG
      DO 1380 MM = 1,4
      BLOW = AVG-(REAL(MM)*SIG)
      BIGH = AVG+(REAL(MM)*SIG)
C
C     IT IS ASSUMED THAT THE DISTRIBUTION THROUGH NODE I RESEMBLES
```

```
C     A NORMAL DISTRIBUTION.  THE FIXED PERCENTAGES CORRESPOND TO
C     1, 2, 3, AND 4 STANDARD DEVIATIONS, RESPECTIVELY, FROM THE
C     EXPECTED VALUE.
C
      IF (MM .EQ. 1) PERCNT = 68.24
      IF (MM .EQ. 2) PERCNT = 95.44
      IF (MM .EQ. 3) PERCNT = 99.73
      IF (MM .EQ. 4) PERCNT = 99.99
      IF (I .NE. N) THEN
      PRINT 1980,I,HLOW,HIGH,PERCNT
      ELSE
      PRINT 1987,HLOW,HIGH,PERCNT
      END IF
 1380 CONTINUE
 1385 CONTINUE
      IF (NSIM .EQ. 0) GO TO 1540
C
C     DO 1510 COMPILES OUTPUT FROM THE MONTE CARLO SIMULATION FOR EACH
C     NODE ON THE OUTPUT CRITICAL LIST.
C
      DO 1510 I=2,N
      IF (IOCL(I) .EQ. 0) GO TO 1510
C
C     DO 1390 COMPUTES THE PARTITION OF THE INTERVAL OVER WHICH THE
C     THROUGHPUT DISTRIBUTION THROUGH NODE I IS DEFINED.
C
      XX(1,1) = XINT(I,100,1)
      SIZE = (XINT(I,100,2)-XINT(I,100,1))/5.
      DO 1390 J=2,51
      XX(J,1) = XX(J-1,1)+SIZE
 1390 CONTINUE
C
C     DO 1410 COMPILES THE CUMULATIVE DISTRIBUTION FUNCTION.
C
      COUNT = 0.0
      SIMTOT(1) = 0.0
      DO 1410 J=1,50
      DO 1400 K=1,NSIM
      IF ((XX(J,1) .LE. SIMT(I,K)) .AND. (SIMT(I,K) .LT. XX(J+1,1)))
     &COUNT = COUNT+1.
 1400 CONTINUE
      XX(J+1,2) = COUNT/REAL(NSIM)
      SIMTOT(J+1) = SIMTOT(J)+XX(J+1,2)
      COUNT = 0.0
 1410 CONTINUE
      IF (NAN .EQ. 9) GO TO 1450
      PRINT 1910
      IF (I .NE. N) THEN
      PRINT 1921,I
      ELSE
      PRINT 1926
      END IF
      IF ((NAN .EQ. 2) .OR. (NAN .EQ. 4) .OR. (NAN .EQ. 6) .OR.
     &(NAN .EQ. 7)) GO TO 1420
      GO TO 1440
 1420 PRINT 1950
```

```
      DO 1430 J=1,51
      PRINT 1960,XX(J,1),SIMTOT(J)
 1430 CONTINUE
 1440 IF ((NAN .EQ. 3) .OR. (NAN .EQ. 5) .OR. (NAN .EQ. 6) .OR.
     &(NAN .EQ. 7)) GO TO 1450
      GO TO 1470
 1450 DO 1460 J=1,50
      XX(J,1) = XX(J,1)+(SIZE/2.)
      XX(J,2) = XX(J+1,2)
 1460 CONTINUE
      IF (NAN .EQ. 9) GO TO 1475
      IPRINT = 50
      IFLAG1 = 1
      CALL PLOT(IPRINT,KBL,KBM,IFLAG1)
 1470 CONTINUE
C
C     DO 1480 COMPUTES AN APPROXIMATED EXPECTED VALUE AND
C     DO 1490 COMPUTES AN APPROXIMATED STANDARD DEVIATION
C     USING GROUPED DATA.
C
 1475 AVG = 0.0
      SIG = 0.0
      DO 1480 J=1,50
      AVG = AVG+(XX(J,1)*(SIMTOT(J+1)-SIMTOT(J)))
 1480 CONTINUE
      DO 1490 J=1,50
      SIG = SIG+(((XX(J,1)-AVG)**2)*(SIMTOT(J+1)-SIMTOT(J)))
 1490 CONTINUE
      SIG = DSQRT(SIG)
      IF (NAN .EQ. 9) GO TO 1510
      PRINT 1910
      PRINT 1970,AVG,SIG
      DO 1500 MM=1,4
      HLOW = AVG-(REAL(MM)*SIG)
      HIGH = AVG+(REAL(MM)*SIG)
C
C     IT IS ASSUMED THAT THE DISTRIBUTION THROUGH NODE I RESEMBLES
C     A NORMAL DISTRIBUTION.  THE FIXED PERCENTAGES CORRESPOND TO
C     1, 2, 3, AND 4 STANDARD DEVIATIONS, RESPECTIVELY, FROM THE
C     EXPECTED VALUE.
C
      IF (MM .EQ. 1) PERCNT = 68.24
      IF (MM .EQ. 2) PERCNT = 95.44
      IF (MM .EQ. 3) PERCNT = 99.73
      IF (MM .EQ. 4) PERCNT = 99.99
      IF (I .NE. N) THEN
      PRINT 1980,I,HLOW,HIGH,PERCNT
      ELSE
      PRINT 1987,HLOW,HIGH,PERCNT
      END IF
 1500 CONTINUE
 1510 CONTINUE
C
C     COMPUTE RELATIVE ERROR OF APPROXIMATED MEAN AND STANDARD DEVIATION
C     OF NETWORK THROUGHPUT DISTRIBUTION.
C
```

```
      COMPAR(NGENCT,3) = ((COMPAR(NGENCT,3)-AVG)/AVG)*100.0
      COMPAR(NGENCT,4) = ((COMPAR(NGENCT,4)-SIG)/SIG)*100.0
C
C     COMPARE POLYGONAL APPROXIMATION OF THROUGHPUT DISTRIBUTION
C     WITH SIMULATED THROUGHPUT DISTRIBUTION USING THE KOLMOGOROV-
C     SMIRNOV ONE-SAMPLE TEST.
C
      KSCR20 = 1.0730/SQRT(50.)
      KSCR10 = 1.2239/SQRT(50.)
      KSCR05 = 1.3581/SQRT(50.)
      KSCR02 = 1.5174/SQRT(50.)
      KSCR01 = 1.6276/SQRT(50.)
C
C     COMPUTE THE K-S TEST STATISTIC D-MAX.
C
      DMAX = 0.0
      DO 1530 I = 2,51
      DIFF = DABS(SIMTOT(I)-TOTAAR(I))
      IF (DIFF .GT. DMAX) DMAX = DIFF
 1530 CONTINUE
      IF (NAN .NE. 9) THEN
      PRINT 1910
      PRINT 1991,DMAX
      PRINT 1992,KSCR20,KSCR10,KSCR05,KSCR02,KSCR01
      IF (DMAX .LE. KSCR05) PRINT 1993
      ELSE
      COMPAR(NGENCT,1) = DMAX
      COMPAR(NGENCT,2) = 0.0
      IF (DMAX .LE. KSCR20) COMPAR(NGENCT,2) = 20.0
      IF ((DMAX .GT. KSCR20) .AND. (DMAX .LE. KSCR10))
     *COMPAR(NGENCT,2) = 10.0
      IF ((DMAX .GT. KSCR10) .AND. (DMAX .LE. KSCR05))
     *COMPAR(NGENCT,2) = 5.0
      IF ((DMAX .GT. KSCR05) .AND. (DMAX .LE. KSCR02))
     *COMPAR(NGENCT,2) = 2.0
      IF ((DMAX .GT. KSCR02) .AND. (DMAX .LE. KSCR01))
     *COMPAR(NGENCT,2) = 1.0
      END IF
      CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
 1540 CONTINUE
      IF (NAN .NE. 9) THEN
      PRINT 1996,TOTTIM
      STOP
      ELSE
      PRINT 1994,NGEN,N,NACTS
      PRINT 1992,KSCR20,KSCR10,KSCR05,KSCR02,KSCR01
      PRINT 1995
      CALL TIMER(DELTA)
      DO 1550 I = 1,NGEN
      IF (COMPAR(I,2) .EQ. 0.0)  PRINT 1981,COMPAR(I,1),COMPAR(I,3),
     &COMPAR(I,4)
      IF (COMPAR(I,2) .EQ. 1.0)  PRINT 1982,COMPAR(I,1),COMPAR(I,3),
     &COMPAR(I,4)
      IF (COMPAR(I,2) .EQ. 2.0)  PRINT 1983,COMPAR(I,1),COMPAR(I,3),
     &COMPAR(I,4)
```

```
      IF (COMPAR(I,2) .EQ. 5.0)  PRINT 1984,COMPAR(I,1),COMPAR(I,3),
     &COMPAR(I,4)
      IF (COMPAR(I,2) .EQ. 10.0) PRINT 1985,COMPAR(I,1),COMPAR(I,3),
     &COMPAR(I,4)
      IF (COMPAR(I,2) .EQ. 20.0) PRINT 1986,COMPAR(I,1),COMPAR(I,3),
     &COMPAR(I,4)
 1550 CONTINUE
      CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
      PRINT 1996,TOTTIM
      END IF
      STOP
 1560 PRINT 1990
      STOP
C
C     FORMAT STATEMENTS
C
 1900 FORMAT (2(I3,1X),I4,1X,I1,1X,I5,1X,I2,1X,F4.2)
 1901 FORMAT (3(I2,25(1X,I2)/),I2,21(1X,I2),1X,I1,2(1X,I2))
 1902 FORMAT (F1.0,4(1X,F8.2))
 1910 FORMAT (1H1)
 1915 FORMAT (1X,'THE RESULTS FOR NETWORK NUMBER ',I3,' OF ',I3,
     &' NETWORKS GENERATED ARE:' //)
 1920 FORMAT (1X,'THE POLYGONAL APPROXIMATION OF THE TIME DISTRIBUTION',
     &' THROUGH NODE',1X,I2,1X,'IS:' //)
 1921 FORMAT (1X,'THE SIMULATED TIME DISTRIBUTION',
     &' THROUGH NODE',1X,I2,1X,'IS:' //)
 1925 FORMAT (1X,'THE POLYGONAL APPROXIMATION OF THE TIME DISTRIBUTION',
     &' THROUGH THE PROJECT IS:' //)
 1926 FORMAT (1X,'THE SIMULATED TIME DISTRIBUTION',
     &' THROUGH THE PROJECT IS:' //)
 1930 FORMAT (1X,'INTERVAL',I3,4X,'LOWER LIMIT =',F8.2,3X,
     &'UPPER LIMIT =',F8.2 //)
 1940 FORMAT (15X,'X = (',F12.8,') + (',F12.8,') T' //)
 1950 FORMAT (14X,'CUMULATIVE DISTRIBUTION FUNCTION' //
     &21X,'T',14X,'F(T)')
 1960 FORMAT (16X,F9.3,F17.8)
 1970 FORMAT (12X,'EXPECTED VALUE OF T     =',F13.8 /
     &12X,'STANDARD DEVIATION OF T =',F13.8 //)
 1980 FORMAT (1X,'THE PROBABILITY OF NODE ',I3,' THROUGHPUT TIME',
     &' FALLING BETWEEN' / 1X,F8.3,' TIME UNITS AND',F8.3,
     &' TIME UNITS IS ABOUT ',F5.2,' %.'//)
 1981 FORMAT (1X,F6.4,6X,'   <1%',14X,F6.2,'%',9X,F6.2,'%')
 1982 FORMAT (1X,F6.4,6X,' 1% - 2%',12X,F6.2,'%',9X,F6.2,'%')
 1983 FORMAT (1X,F6.4,6X,' 2% - 5%',12X,F6.2,'%',9X,F6.2,'%')
 1984 FORMAT (1X,F6.4,6X,' 5% - 10%',11X,F6.2,'%',9X,F6.2,'%')
 1985 FORMAT (1X,F6.4,6X,'10% - 20%',11X,F6.2,'%',9X,F6.2,'%')
 1986 FORMAT (1X,F6.4,6X,'   >20%',13X,F6.2,'%',9X,F6.2,'%')
 1987 FORMAT (1X,'THE PROBABILITY OF THE PROJECT THROUGHPUT TIME',
     &' FALLING BETWEEN' / 1X,F8.3,' TIME UNITS AND ',F8.3,
     &' TIME UNITS IS ABOUT ',F5.2,' %.'//)
 1990 FORMAT (1X,'PROGRAM STOPPED' / 1X,'IMPROPER NODE NUMBER(S) '
     &,'ENCOUNTERED')
 1991 FORMAT (1X,'KOLMOGOROV-SMIRNOV ONE-SAMPLE TEST COMPARISON OF ',
     &'POLYGONAL APPROXIMATION' / 1X,'OF NETWORK THROUGHPUT DISTRIBUTION
     & AND SIMULATED NETWORK THROUGHPUT' / 1X,'DISTRIBUTION:' //
     &1X,'K-S TEST STATISTIC D-MAX = ', F6.4 /)
 1992 FORMAT (1X,'K-S CRITICAL VALUES:' / 15X,'20 PERCENT = ',F6.4 /
     &15X,'10 PERCENT = ',F6.4 / 16X,'5 PERCENT = ',F6.4 /
     &16X,'2 PERCENT = ',F6.4 / 16X,'1 PERCENT = ',F6.4 /)
 1993 FORMAT (1X,'FAIL TO REJECT THE NULL HYPOTHESIS THAT THE ',
     &'DISTRIBUTIONS ARE THE SAME' / 1X,'AT THE 5% LEVEL OF ',
     &'STATISTICAL SIGNIFICANCE.')
 1994 FORMAT (1X,'STATISTICAL COMPARISONS FOR THE ',I3,' NETWORKS ',
     &'GENERATED' / 1X,'WITH ',I3,' NODES AND ',I4,' ACTIVITIES ARE:' //
     &)
 1995 FORMAT (10X,'PROBABILITY VALUE',3X,'RELATIVE ERROR',2X,'RELATIVE '
     &,'ERROR' / 2X,'D-MAX',2X,'(TYPE 1 ERROR PROB)',5X,'OF MEAN',8X,
     &'OF STN DEV' /)
 1996 FORMAT (// 1X,'CPU TIME FOR PART PROCESSING IS ',F8.3,' SECONDS'
     &//)
      END
C     END MAIN PROGRAM
C
C     ************************************************************
C
C                SUBROUTINE            PARA
C
C     ************************************************************
C
      SUBROUTINE PARA (L1,L2,L3)
      REAL*8 VALUE(101,100,10,3),XINT(101,100,12)
      REAL*8 XVAL,ZVAL(130,5),PAR(2,15,6),FACT,B(130)
      REAL*4 Z
      INTEGER L1,L2,L3,NV1,NV2
      INTEGER K4(2,30)
      INTEGER I,IINT,N,NCL,J,K,K3,L6,LASTJ,LASTK
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/B
C
C     SUBROUTINE PARA IS USED TO REDUCE PARALLEL ARCS INTO A SINGLE
C     EQUIVALENT ARC.  IT FINDS THE MAX OPERATOR BY MULTIPLYING CAP
C     F(X) AGAINST CAP G(X) OVER THE INTERVALS OF VALIDITY.
C
      NV1 = 10
      NV2 = 10
      DO 2020 N = 1,2
      L6 = L2
      IF (N .EQ. 2) L6 = L3
      FACT = 0
      DO 2010 J = 1,10
      B(1) = XINT(L1,L6,J)
C
C     DO 2000 CONVERTS EACH LINEAR POLYNOMIAL PIECE OF LITTLE F(X)
C     INTO THE CORRESPONDING QUADRATIC POLYNOMIAL PIECE OF ITS
C     CUMULATIVE DISTRIBUTION CAP F(X).
C
      DO 2000 I = 1,2
      XVAL = VALUE(L1,L6,J,I)
      Z = FLOAT(I)
      PAR(N,J,I+1) = XVAL/Z
      PAR(N,J,1) = PAR(N,J,1)+((-1.0)*(XVAL/Z)*(B(1)**I))
      K4(N,J) = I+1
 2000 CONTINUE
      IF (J .GT. 1) PAR(N,J,1) = PAR(N,J,1)+FACT
      FACT = PAR(N,J,1)+(PAR(N,J,2)*XINT(L1,L6,J+1))+(PAR(N,J,3)
     &*(XINT(L1,L6,J+1)**2))
 2010 CONTINUE
 2020 CONTINUE
C
C     DO 2040 ASSIGNS INTERVAL BOUNDARY VALUES TO THE B ARRAY.
C
      DO 2040 I = 1,22
      IF (I .GT. 11) GO TO 2030
      B(I) = XINT(L1,L2,I)
      GO TO 2040
 2030 B(I) = XINT(L1,L3,I-11)
 2040 CONTINUE
      NCL = 21
      CALL SORT(NCL)
C
C     DO 2080 DETERMINES THE POINT AT WHICH THE DISTRIBUTION DOMAINS
C     OF THE TWO ARCS BEING COMBINED OVERLAP.  ONCE THIS POINT IS
C     DETERMINED, THE B ARRAY IS ADJUSTED TO REFLECT THE OVERLAP
C     (ALL VALUES LESS THAN THIS POINT OF FIRST OVERLAP NEED NOT BE
C     CONSIDERED, BECAUSE ONE OF THE DISTRIBUTIONS EQUALS ZERO AT
C     THESE VALUES). IF THE DOMAINS ARE DISJOINT OR OVERLAP AT ONLY.
C     ONE BOUNDARY POINT, THE RESULT OF THE APPLICATION OF THE
C     MAXIMUM OPERATOR IS JUST THE UNCHANGED APPROXIMATED PROBABILITY
C     DENSITY FUNCTION OF THE DISTRIBUTION DEFINED ON THE HIGHER-
C     VALUED DOMAIN.  GO TO 2180 OR GO TO 2160 RETURNS THIS FUNCTION
C     DIRECTLY WITHOUT FURTHER PROCESSING.
C
      IINT = 0
      LASTJ = NCL+1
      DO 2080 J = 1,LASTJ
      IF ((XINT(L1,L2,1) .GE. XINT(L1,L3,1)-0.001) .AND.
     &(XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001)) GO TO 2080
      IF (IINT .GE. 1) GO TO 2060
      IF (XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001) GO TO 2050
      IF (XINT(L1,L3,J+1) .GE. XINT(L1,L2,1)-0.001) IINT = J
      IF ((XINT(L1,L3,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L2,1) .GE. XINT(L1,L3,J+1)-0.001)
     &.AND. (XINT(L1,L2,1) .LE. XINT(L1,L3,J+1)+0.001)
     &.AND. (XINT(L1,L3,J+2) .LE. 0.001))) GO TO 2180
      GO TO 2080
 2050 IF (XINT(L1,L2,J+1) .GE. XINT(L1,L3,1)-0.001) IINT = J
      IF ((XINT(L1,L2,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L3,1) .GE. XINT(L1,L2,J+1)-0.001)
     &.AND. (XINT(L1,L3,1) .LE. XINT(L1,L2,J+1)+0.001)
     &.AND. (XINT(L1,L2,J+2) .LE. 0.001))) GO TO 2160
      GO TO 2080
 2060 LASTK = NCL-(IINT-1)
      DO 2070 K = 1,LASTK
      B(K) = B(K+IINT)
      B(K+IINT) = 0
 2070 CONTINUE
```

```
      GO TO 2090
 2080 CONTINUE
 2090 NCL = NCL-IINT
C
C     DO 2150 IS THE OUTER LOOP FOR THE PROCESS OF CREATING THE
C     EQUIVALENT ARC.  NCL IS THE NUMBER OF CLASSES INVOLVED
C     BETWEEN THE TWO ARCS.
C
      N1 = 0
      N2 = 0
      DO 2150 I = 1,NCL
      DO 2110 J = 1,11
C
C     DO 2110 DETERMINES THE APPROPRIATE INTERVALS OF EACH DISTRIBUTION
C     THAT ARE VALID FOR THE B(I) VALUE BEING CONSIDERED.  N1 AND
C     N2 ARE THE CONTROLS FOR UPPER AND LOWER ARCS RESPECTIVELY.
C
      IF (N1 .GE. 1) GO TO 2100
      IF (((B(I) .GE. XINT(L1,L2,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L2,J+1)+0.001)) .OR. (XINT(L1,L2,J+1) .LE. 0.001))
     &N1 = J
 2100 CONTINUE
      IF (N2 .GE. 1) GO TO 2110
      IF (((B(I) .GE. XINT(L1,L3,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L3,J+1)+0.001)) .OR. (XINT(L1,L3,J+1) .LE. 0.001))
     &N2 = J
 2110 CONTINUE
      IF (N2 .GT. NV2) K4(2,N2) = 1
      IF (N1 .GT. NV1) K4(1,N1) = 1
C
C     DO 2130 AND DO 2120 PERFORM THE POLYGONAL MULTIPLICATION FOR
C     CAP F(X) AND CAP G(X).
C
      LASTJ = K4(2,N2)
      LASTK = K4(1,N1)
      DO 2130 J = 1,LASTJ
      DO 2120 K = 1,LASTK
      IF (N2 .GT. NV2) PAR(2,N2,J) =  1
      IF (N1 .GT. NV1) PAR(1,N1,K) =  1
      K3 = J+K-1
      ZVAL(I,K3) = ZVAL(I,K3)+(PAR(1,N1,K)*PAR(2,N2,J))
 2120 CONTINUE
 2130 CONTINUE
C
C     DO 2140 OBTAINS THE FIRST DERIVATIVE OF THE RESULT OF THE
C     MULTIPLICATION OF CAP F(X) AND CAP G(X) IN THE FORM OF A
C     LITTLE H(X) FOR THAT PRODUCT.
C
      DO 2140 J = 1,4
      ZVAL(I,J) = ZVAL(I,J+1)*FLOAT(J)
      ZVAL(I,J+1) = 0
 2140 CONTINUE
      N1 = 0
      N2 = 0
 2150 CONTINUE
C
```

```
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE RESULTS OF THE
C     PARALLEL REDUCTION WITH THE B(O) AND B(1) FORM IN EACH OF 10
C     CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR(L1,L2,NCL)
      GO TO 2180
 2160 DO 2170 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L3,I,1)
      VALUE(L1,L2,I,2) = VALUE(L1,L3,I,2)
      XINT(L1,L2,I) = XINT(L1,L3,I)
 2170 CONTINUE
      XINT(L1,L2,11) = XINT(L1,L3,11)
 2180 VALUE(L1,L2,1,3) = 0
      DO 2210 I = 1,2
      DO 2200 J = 1,10
      DO 2190 K = 1,3
      PAR(I,J,K) = 0
 2190 CONTINUE
 2200 CONTINUE
 2210 CONTINUE
      RETURN
      END
C     END SUBROUTINE PARA
C
C     *************************************************************
C
C                S U B R O U T I N E        S E R I E S
C
C     **************************************************************
C
      SUBROUTINE SERIES (L1,L2,L3,L4)
      REAL*8 VALUE(101,100,10,3),XINT(101,100,12)
      REAL*8 ZVAL(130,5),XLIM(2),A(130)
      REAL*8 F0,F1,G0,G1,XL
      INTEGER L1,L2,L3,L4
      INTEGER ISEL(2)
      INTEGER I,IK,J,K,NCL,NCL1,NE
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
C
C     SUBROUTINE SERIES PERFORMS THE CONVOLUTION OF TWO PROBABILITY
C     DISTRIBUTIONS BY INTEGRATING THE PRODUCT OF THEIR PIECEWISE
C     POLYGONAL APPROXIMATIONS IN THE FORMS OF F(X) AND G(T-X).
C
C     THIS SECTION DETERMINES THE INTERVALS OF VALIDITY FOR THE
C     CONVOLUTION.
C
C     THE A ARRAY IS USED FOR THE SAME PURPOSE AS THE B ARRAY IN PARA.
C
      K = 0
C
C     DO 3010 CREATES ALL POSSIBLE INTERVALS OF THE NEW DISTRIBUTION
C     BY ADDING THE INTERVALS OF THE TWO DISTRIBUTIONS BEING WORKED.
C
```

```
      DO 3010 I = 1,12
      IF ((XINT(L3,L4,I) .LE. 0).AND.(I .GT. 1)) GO TO 3020
      DO 3000 J = 1,12
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) NCL1 = J-2
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) GO TO 3010
      K = K+1
      A(K) = XINT(L1,L2,J)+XINT(L3,L4,I)
 3000 CONTINUE
 3010 CONTINUE
 3020 NINT = I-2
      NCL = K-1
C
C     DO 3120 IS CONTROLLED BY THE NUMBER OF CLASSES IN THE F(X)
C     DISTRIBUTION.  DO 3110 IS CONTROLLED BY THE NUMBER OF CLASSES
C     CREATED BY COMBINING F(X) AND G(T-X).  DO 3100 IS CONTROLLED
C     BY THE NUMBER OF CLASSES IN THE G(T-X) DISTRIBUTION.  THIS
C     ALLOWS THE EVALUATION OF ALL OF THE CREATED CLASSES FOR EVERY
C     CLASS IN BOTH DISTRIBUTIONS.
C
      CALL SORT(NCL)
      DO 3120 K = 1,NCL1
      DO 3110 I = 1,NCL
      DO 3100 J = 1,NINT
      IK = 0
C
C     THIS IF STATEMENT DETERMINES WHICH INTERVALS ARE VALID FOR THE
C     INTERVAL END POINT A(I) BEING EVALUATED AND FOR THE VALUE OF K
C     BEING CONTROLLED BY DO 3120.
C
      IF ((A(I) .GE. XINT(L1,L2,K)+XINT(L3,L4,J)-0.001) .AND. (A(I+1)
     &.LE. XINT(L1,L2,K+1)+XINT(L3,L4,J+1)+0.001)) IK = J
      IF (IK .GE. 1) GO TO 3030
      GO TO 3100
 3030 ISEL(1) = 0
      ISEL(2) = 0
C
C     THE IF STATEMENTS INVOLVING XLIM ARE USED TO DETERMINE THE
C     UPPER AND LOWER LIMITS OF INTEGRATION.  IT IS DETERMINED WHETHER
C     THE LIMIT COMES FROM THE F(X) OR THE G(T-X) DISTRIBUTION.  ISEL
C     IS USED TO DESIGNATE VALUES FROM THE G(T-X) DISTRIBUTION.
C
      IF (XINT(L1,L2,K) .GE. (A(I+1)-XINT(L3,L4,J+1)-0.001)) GO TO 3040
      XLIM(1) = XINT(L3,L4,J+1)
      ISEL(1) = 999
      GO TO 3050
 3040 XLIM(1) = XINT(L1,L2,K)
 3050 IF (XINT(L1,L2,K+1) .LE. (A(I)-XINT(L3,L4,J)+0.001)) GO TO 3060
      XLIM(2) = XINT(L3,L4,J)
      ISEL(2) = 999
      GO TO 3070
 3060 XLIM(2) = XINT(L1,L2,K+1)
 3070 CONTINUE
      DO 3090 NE = 1,2
      F0 = VALUE(L1,L2,K,1)
      F1 = VALUE(L1,L2,K,2)
      G0 = VALUE(L3,L4,IK,1)
      G1 = VALUE(L3,L4,IK,2)
      XL = XLIM(NE)
      Z = 1.0
      IF (NE .EQ. 1) Z = -1.0
      IF (ISEL(NE) .EQ. 999) GO TO 3080
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL AT A FINITE
C     LIMIT.  THE INTEGRATION IS BROKEN DOWN INTO ITS COMPONENT PARTS
C     BY THE POWER OF THE COEFFICIENT THAT RESULTS.  Z CONTROLS THE
C     SIGN OF THE INTEGRAL BASED ON WHETHER THE LOWER OR UPPER LIMIT
C     IS BEING EVALUATED.
C
      ZVAL(I,1) = ZVAL(I,1)+((F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((-1.0*F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+(((F1*G1*XL**2)/2.)+(F0*G1*XL))*Z
      ZVAL(I,3) = ZVAL(I,3)+((-1.0*F0*G1)/2.)*Z
      GO TO 3090
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL FOR LIMITS.
C     IN THE FORM OF (T-X).  THE FORMULAS ARE DIFFERENT BECAUSE
C     OF THE DIFFERENT POLYNOMIAL CREATED WHEN THE INTEGRATION
C     INVOLVES LIMITS IN THE FORM OF (T-X).
C
 3080 ZVAL(I,1) = ZVAL(I,1)+((-1.0*F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+((-1.0*F1*G0*XL)+(F0*G0)-
     &((F1*G1*XL**2)/2.))*Z
      ZVAL(I,3) = ZVAL(I,3)+((F1*G0)/2.)*Z
      ZVAL(I,4) = ZVAL(I,4)+((F1*G1)/6.)*Z
 3090 CONTINUE
 3100 CONTINUE
 3110 CONTINUE
 3120 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE CONVOLUTION
C     RESULTS WITH THE B(0) AND B(1) FORM IN EACH OF 10 CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR(L1,L2,NCL)
      RETURN
      END
C     END SUBROUTINE SERIES
C
C     ***********************************************************
C
C                 S U B R O U T I N E         P L O T
C
C     ***********************************************************
C
      SUBROUTINE PLOT (IPRINT,KBL,KBM,IFLAG)
      REAL*8 XX(100,2),SORT
      CHARACTER*1 KBL,KBM,LINE(101)
      INTEGER I,IFLAG,IPRINT,J,JPLOT,K,NN
      COMMON/PARA4/XX
C
C     PLOT IS USED TO CREATE THE HISTOGRAM FOR FINAL OUTPUT.
```

```
C     THE VARIABLE SORT IN THIS SUBROUTINE IS NOT RELATED TO
C     THE SUBROUTINE SORT.
C
      SORT = XX(1,2)
      DO 4000 I = 2,IPRINT
      IF (SORT .LE. XX(I,2)) SORT = XX(I,2)
 4000 CONTINUE
      PRINT 4900
      IF (IFLAG .EQ. 0) THEN
      PRINT 4910
      ELSE
      PRINT 4915
      END IF
      IF (SORT .GT. 0.5) PRINT 4920
      IF ((SORT .GT. 0.25).AND.(SORT .LE. 0.50)) PRINT 4930
      IF ((SORT .GT. 0.10).AND.(SORT .LE. 0.25)) PRINT 4940
      IF ((SORT .GT. 0.05).AND.(SORT .LE. 0.10)) PRINT 4950
      IF (SORT .LE. 0.05) PRINT 4960
      PRINT 4970
      DO 4030 I = 1,IPRINT
      DO 4010 J = 1,51
      LINE(J) = KBL
 4010 CONTINUE
      IF (SORT .GT. 0.5) JPLOT = (INT((XX(I,2)*50.)+0.5))+1
      IF ((SORT .GT. 0.25).AND.(SORT .LE. 0.50))
     &JPLOT = (INT((XX(I,2)*100.)+0.5))+1
      IF ((SORT .GT. 0.10).AND.(SORT .LE. 0.25))
     &JPLOT = (INT((XX(I,2)*200.)+0.5))+1
      IF ((SORT .GT. 0.05).AND.(SORT .LE. 0.10))
     &JPLOT = (INT((XX(I,2)*500.)+0.5))+1
      IF (SORT .LE. 0.05) JPLOT = (INT((XX(I,2)*1000.0)+0.5))+1
      IF (JPLOT .LE. 0) JPLOT = 1
      IF (JPLOT .GT. 51) JPLOT = 51
      DO 4020 NN = 1,JPLOT
      LINE(NN) = KBM
 4020 CONTINUE
      PRINT 4980,XX(I,1),(LINE(K), K = 1,JPLOT)
 4030 CONTINUE
C
C     FORMAT STATEMENTS
C
 4900 FORMAT (1H1)
 4910 FORMAT (15X,'PROBABILITY DENSITY FUNCTION' //)
 4915 FORMAT (15X,'SIMULATION FREQUENCY HISTOGRAM' //)
 4920 FORMAT (9X,'0       .20       .40       .60       .80       1.0')
 4930 FORMAT (9X,'0       .10       .20       .30       .40       .50')
 4940 FORMAT (9X,'0       .05       .10       .15       .20       .25')
 4950 FORMAT (9X,'0       .02       .04       .06       .08       .10')
 4960 FORMAT (9X,'0       .01       .02       .03       .04       .05')
 4970 FORMAT (9X,'I----+----I----+----I----+----I----+----I----+----I')
 4980 FORMAT (1X,F8.3,2X,51A1)
      RETURN
      END
C     END SUBROUTINE PLOT
C
C     ************************************************************
```

```
C
C                S U B R O U T I N E        L I N E A R
C
C     *************************************************************
C
      SUBROUTINE LINEAR (L1,L2,NCL)
      REAL*8 VALUE(101,100,10,3),XINT(101,100,12),ZVAL(130,5),A(130)
      REAL*8 Q,Q1,Q2,STD,SUMX,SUMY,SUMXY,SUMSQ
      REAL*8 ALPHA,AREA,BETA,FACT,SIZE,W,X,XLMBDA,XMEAN
      REAL*8 XMODE,XSIZE,Y
      INTEGER L1,L2
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      EXTERNAL DGAMMA
C
C     SUBROUTINE LINEAR PIECEWISE POLYGONALIZES DISTRIBUTION DATA
C     FROM THE MAIN PROGRAM AND SUBROUTINES PARA AND SERIES WITH
C     THE B(O) AND B(1) FORM IN EACH OF 10 CLASSES THROUGH THE USE
C     OF SIMPLE LINEAR REGRESSION.
C
      XMODE = VALUE(L1,L2,2,3)
      XMEAN = VALUE(L1,L2,2,3)
      STD = ((VALUE(L1,L2,2,3)-XINT(L1,L2,1))/3.)
      XLMBDA = VALUE(L1,L2,2,3)-XINT(L1,L2,1)
      ALPHA = VALUE(L1,L2,2,3)
      BETA = VALUE(L1,L2,3,3)
      SIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) SIZE = (A(NCL+1)-A(1))/10.
      XINT(L1,L2,11) = XINT(L1,L2,2)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) XINT(L1,L2,11) = A(NCL+1)
      X = XINT(L1,L2,1)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) X = A(1)
      DO 5000 I = 1,10
      XINT(L1,L2,I) = X
      X = X+SIZE
 5000 CONTINUE
      DO 5050 I = 1,10
      X = XINT(L1,L2,I)
      SUMY = 0.
      SUMX = 0.
      SUMXY = 0.
      SUMSQ = 0.
C
C     W CONTROLS THE NUMBER OF DATA POINTS USED IN THE REGRESSION
C     COMPUTATIONS.
C
      W = 10.+IDINT(SIZE*3.)
      XSIZE = SIZE/W
      LASTJ = IDINT(W)
      DO 5040 J = 1,LASTJ
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 99)  GO TO 5030
      DO 5010 K3 = 1,NCL
      K  = 0
      IF ((X .GE. A(K3)).AND.(X .LE. A(K3+1))) K = K3
      IF (K .GE. 1) GO TO 5020
 5010 CONTINUE
C
C     SERIES OR PARA GENERATED DISTRIBUTIONS.
C
 5020 Y = ZVAL(K,1)+(ZVAL(K,2)*X)+(ZVAL(K,3)*(X**2))
     &+(ZVAL(K,4)*(X**3))
 5030 CONTINUE
C
C     TRIANGULAR DISTRIBUTION.
C
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 1) THEN
         IF (XINT(L1,L2,1) .LE. X .AND. X .LE. XMODE) THEN
         Y = (2.*(X-XINT(L1,L2,1)))/((XMODE-XINT(L1,L2,1))*10.*SIZE)
         ELSE
         Y = (2.*(XINT(L1,L2,11)-X))/((XINT(L1,L2,11)-XMODE)*10.*SIZE)
         END IF
C
C     NORMAL  DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 2) THEN
      Y = (1./(STD*2.506628275))*(DEXP((-1.0)*(((X-XMEAN)/STD)**2)/2.))
C
C     EXPONENTIAL DISTRIBUTION  (SHIFTED).
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 3) THEN
      Y = (1./XLMBDA)*(DEXP((-1.0)*((X-XINT(L1,L2,1))/XLMBDA)))
C
C     GAMMA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 4) THEN
      Y = (1./(DGAMMA(ALPHA)*(BETA**ALPHA)))*DEXP(-X/BETA)*(X**(ALPHA-1.
     &))
C
C     BETA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 5) THEN
      Y = (DGAMMA(ALPHA+BETA)/(DGAMMA(ALPHA)*DGAMMA(BETA)))*
     &(1./(10.*SIZE)**(ALPHA+BETA-2.))*
     &((X-XINT(L1,L2,1))**(ALPHA-1.))*
     &((XINT(L1,L2,11)-X)**(BETA-1.))
      END IF
      IF (Y .LT. 0) Y = 0
      SUMX = SUMX+X
      SUMY = SUMY+Y
      SUMXY = SUMXY+(X*Y)
      SUMSQ = SUMSQ+(X**2)
      X = X+XSIZE
 5040 CONTINUE
      VALUE(L1,L2,I,2) = (SUMXY-((SUMX*SUMY)/W))/(SUMSQ-((SUMX**2)/W))
      VALUE(L1,L2,I,1) = (SUMY/W)-(VALUE(L1,L2,I,2)*(SUMX/W))
 5050 CONTINUE
C
C     DO 5060 CALCULATES THE AREA UNDER THE APPROXIMATED DISTRIBUTION.
C     AN ADJUSTMENT FACTOR FOR THE AMOUNT THAT THIS AREA HAS BEEN
C     UNDERESTIMATED OR OVERESTIMATED IS THEREBY DETERMINED.
C
```

```
      DO 5060 I = 1,10
      Q  = XINT(L1,L2,I+1)-XINT(L1,L2,I)
      Q1 = (XINT(L1,L2,I)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      Q2 = (XINT(L1,L2,I+1)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      IF (Q1 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q1*(-1.0))
      IF (Q2 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q2*(-1.0))
      IF (Q1 .LT. 0.) Q1 = 0.
      IF (Q2 .LT. 0.) Q2 = 0.
      AREA = AREA+((Q1+Q2)*Q*0.5)
 5060 CONTINUE
      FACT = 1.0/AREA
C
C     DO 5070 ADJUSTS THE COEFFICIENTS OF ALL THE LINEAR POLYNOMIAL
C     PIECES BY THE FACTOR COMPUTED IN DO 5060 IN ORDER TO NORMALIZE
C     THE AREA BACK TO ONE.  THIS ACTS TO REDUCE ACCUMULATING ERRORS
C     DURING PROGRAM COMPUTATIONS.
C
      DO 5070 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)*FACT
      VALUE(L1,L2,I,2) = VALUE(L1,L2,I,2)*FACT
 5070 CONTINUE
      AREA = 0
      DO 5080  I = 1,130
      A(I)  = 0
      ZVAL(I,1) = 0
      ZVAL(I,2) = 0
      ZVAL(I,3) = 0
      ZVAL(I,4) = 0
 5080 CONTINUE
      RETURN
      END
C     END SUBROUTINE LINEAR
C
C     ***********************************************************
C
C                S U B R O U T I N E          S O R T
C
C     ***********************************************************
C
      SUBROUTINE SORT (NCL)
      REAL*8 A(130),B
      INTEGER NCL
      INTEGER I,K1
      COMMON/PARA3/A
C
C     SUBROUTINE SORT IS USED TO CONDUCT AN ALGEBRAIC SORT OF DATA
C     CREATED IN THE SERIES AND PARA SUBROUTINES.
C
 6000 K1  = 0
      DO 6010 I = 1,NCL
      IF ((A(I) .LT. (A(I+1)+.01)).AND.(A(I) .GT. (A(I+1)-.01)))
     &GO TO 6020
      IF (A(I) .LT. A(I+1)) GO TO 6010
      IF (A(I) .GT. A(I+1)) B = A(I)
      A(I) = A(I+1)
      A(I+1) = B
      K1=K1+1
 6010 CONTINUE
      IF (K1 .GE. 1) GO TO 6000
      GO TO 6040
 6020 NCL = NCL-1
      LASTJ = NCL+1
      DO 6030 J = I,LASTJ
      A(J) = A(J+1)
      A(J+1) = 0
 6030 CONTINUE
      GO TO 6000
 6040 RETURN
      END
C     END SUBROUTINE SORT
C
C     ***********************************************************
C
C                S U B R O U T I N E       S I M U L T
C
C     ***********************************************************
C
      SUBROUTINE SIMULT (N,NSIM)
      REAL*8 XINT(101,100,12),VALUE(101,100,10,3)
      REAL*8 T(100,99),SIMT(100,10000)
      REAL*8 ALPHA,BETA,X,XLNGTH,XLMBDA,XMAX,XMEAN,XMIN,XMODE
      REAL*8 RN,STD,TTEMP,TMAX
      DIMENSION NET(100,103)
      INTEGER ISIM,N,NDIST,NSIM
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA5/NET
      COMMON/PARA6/SIMT
      EXTERNAL DRNUN,DRNNOR,DRNEXP,DRNGAM,DRNBET,RNSET
C
C     DO 7130 GENERATES A SIMULATED NETWORK THROUGHPUT FOR EACH OF
C     NSIM ITERATIONS OF THE MONTE CARLO SIMULATION OF THE NETWORK.
C
      DO 7130 ISIM=1,NSIM
C
C     DO 7080 GENERATES A RANDOM VALUE FROM THE ACTIVITY RESOURCE
C     CONSUMPTION (ACTIVITY TIME) DISTRIBUTION OF EACH ACTIVITY.
C
      DO 7080 I=1,N-1
      DO 7070 J=1,NET(I,103)
      NDIST = IDINT(VALUE(I,J,1,3))
      XMIN = XINT(I,J,1)
      XMAX = XINT(I,J,11)
      XLNGTH = XMAX-XMIN
      GO TO (7010,7020,7030,7040,7050,7060) NDIST
C
C     TRIANGULAR DISTRIBUTION.
C
 7010 CALL DRNUN(1,RN)
      XMODE = VALUE(I,J,2,3)
      X = (XMODE-XMIN)/XLNGTH
      IF (RN .GT. X) GO TO 7015
      T(I,J) = XMIN+DSQRT(RN*XLNGTH*(XMODE-XMIN))
      GO TO 7070
 7015 T(I,J) = XMAX-DSQRT(XLNGTH*(XMAX-XMODE)*(1.-RN))
      GO TO 7070
C
C     NORMAL DISTRIBUTION.
C
 7020 CALL DRNNOR(1,RN)
      XMEAN = VALUE(I,J,2,3)
      STD = (XMEAN-XMIN)/3.
      T(I,J) = (RN*STD)+XMEAN
      IF ((T(I,J) .LT. XMIN) .OR. (T(I,J) .GT. XMAX)) GO TO 7020
      GO TO 7070
C
C     EXPONENTIAL DISTRIBUTION.
C
 7030 CALL DRNEXP(1,RN)
      XLMBDA = VALUE(I,J,2,3)-XMIN
      T(I,J) = (XLMBDA*RN)+XMIN
      IF (T(I,J) .GT. XMAX) GO TO 7030
      GO TO 7070
C
C     GAMMA DISTRIBUTION.
C
 7040 ALPHA = VALUE(I,J,2,3)
      BETA = VALUE(I,J,3,3)
      CALL DRNGAM(1,ALPHA,RN)
      T(I,J) = BETA*RN
      IF (T(I,J) .GT. XMAX) GO TO 7040
      GO TO 7070
C
C     BETA DISTRIBUTION.
C
 7050 ALPHA = VALUE(I,J,2,3)
      BETA = VALUE(I,J,3,3)
      CALL DRNBET(1,ALPHA,BETA,RN)
      T(I,J) = XMIN+(XLNGTH*RN)
      GO TO 7070
C
C     UNIFORM DISTRIBUTION.
C
 7060 CALL DRNUN(1,RN)
      T(I,J) = XMIN+(XLNGTH*RN)
 7070 CONTINUE
 7080 CONTINUE
C
C     DO 7120 GENERATES THE CRITICAL PATH TO EACH NODE.  THE
C     SIMULATED TIME THROUGH NODE I FROM SIMULATION ITERATION L
C     IS STORED IN SIMT(I,L).
C
      SIMT(1,ISIM) = 0.0
      DO 7120 I=2,N
      TMAX = 0.0
C
C     DO 7110 DETERMINES THE STARTING NODES AND THE ACTIVITIES WHICH
C     TERMINATE AT NODE I > STARTING NODES, AND COMPUTES THE SIMULATED
C     THROUGHPUT VALUE THROUGH NODE I AS THE MAXIMUM OF THE
```

```
C         [(THROUGHPUT VALUE THROUGH STARTING NODE) +
C          (ACTIVITY VALUE FROM STARTING NODE TO NODE I)].
C
      DO 7110 J=1,I-1
      DO 7100 J1=2,NET(J,103)+1
      IF (NET(J,J1)-I) 7100,7090,7100
 7090 TTEMP = SIMT(J,ISIM)+T(J,J1-1)
      IF (TTEMP .LT. TMAX) GO TO 7100
      TMAX = TTEMP
 7100 CONTINUE
 7110 CONTINUE
      SIMT(I,ISIM) = TMAX
 7120 CONTINUE
 7130 CONTINUE
      RETURN
      END
C     END SUBROUTINE SIMULT
C
C     ************************************************************
C
C                S U B R O U T I N E        G E N R A N
C
C     ************************************************************
C
      SUBROUTINE GENRAN (N,NACTS)
      DIMENSION NET(100,103),NAF(99),NBE(99)
      REAL*4 DEN,DL,DN,DN2,DN3,UP,X,Y,Y1
      INTEGER I,IJ,J,K,KK,L,N,NI,NO,NN,NACTS,NARCS,NDEL,NDIFF,NFREE,NEM,
     &        NRC
      COMMON/PARA5/NET
      EXTERNAL RNUN
C
C     THIS SUBROUTINE GENERATES A RANDOM ACYCLIC, DIRECTED ACTIVITY
C     NETWORK WITH N NODES AND NACTS ACTIVITIES WITH THE METHOD OF
C     DEMEULEMEESTER, DODIN AND HERROELEN (1993).
C
      DO 8010 I = 1,100
      DO 8000 J = 1,103
      NET(I,J) = 0
 8000 CONTINUE
      NET(I,101) = 1
 8010 CONTINUE
C
C     COMPUTE NUMBER OF ACTIVITIES TO DELETE WITH THE DELETION METHOD.
C
      L = N*(N-1)/2
      NDEL = L-NACTS
C
C     COMPUTE NDIFF SUCH THAT
C         INITIAL NUMBER OF FREE ACTIVITIES = NACTS - NDIFF
C     FOR THE ADDITION METHOD.
C
      NDIFF = (2*N)-4
      DN = REAL(N)
      DL = (REAL(L)/2.0)+1.0
      DN2 = DN*(DN-1.0)
      DN3 = DN+0.5
C
C     IF [N(N-1)/4]+1 < OR = NACTS, CHOOSE THE DELETION METHOD.
C     IF [N(N-1)/4]+1 > NACTS, CHOOSE THE ADDITION METHOD.
C
      IF (DL-NACTS) 8020,8020,8110
C
C     THE DELETION METHOD.
C
 8020 DO 8040 I = 1,N-1
      NET(I,1) = I
      DO 8030 J = I+1,N
      NET(I,J) = J
 8030 CONTINUE
      NET(I,102) = I-1
      NET(I,103) = N-I
 8040 CONTINUE
      NET(N,1) = N
      NET(N,102) = N-1
C
C     DO 8100 DELETES NDEL RANDOMLY SELECTED ACTIVITIES.
C
 8050 DO 8100 I = 1,NDEL
C
C     CHECK THAT THERE IS AT LEAST ONE ACTIVITY FEASIBLE FOR
C     ACTIVITY DELETION.  IF NOT, RESTART NETWORK GENERATION.
C
      DO 8055 J = 1,N-1
      NO = J
      IF (NET(NO,103) .LT. 2) GO TO 8055
      DO 8054 K = NO+1,N
      IF (NET(K,102) .GE. 2 .AND. NET(NO,K) .EQ. K) GO TO 8060
 8054 CONTINUE
 8055 CONTINUE
      GO TO 8000
C
C     RANDOMLY SELECT THE STARTING NODE (NO) OF THE ACTIVITY TO BE
C     DELETED FROM AMONG THE NODES FEASIBLE FOR ACTIVITY-DELETION.
C
 8060 CALL RNUN(1,Y)
      Y1 = (Y*DN2)+0.25
      X = DN3-SQRT(Y1)
      NO = INT(X)
      IF (NO .GT. X) NO = NO-1
      IF (NET(NO,103) .LT. 2) GO TO 8060
C
C     RANDOMLY SELECT THE ENDING NODE (NI) OF THE ACTIVITY TO BE
C     DELETED FROM AMONG THE NODES FEASIBLE FOR ACTIVITY-DELETION.
C
      K = 0
      DO 8070 J = NO+1,N
      IF (NET(J,102) .LT. 2) GO TO 8070
      IF (NET(NO,J) .EQ. 0) GO TO 8070
      K = K+1
      NAF(K) = J
 8070 CONTINUE
```

```
      IF (K .EQ. 0) GO TO 8060
      DEN = 1.0/REAL(K)
      CALL RNUN(1,X)
      DO 8080 J = 1,K
      UP = DEN*REAL(J)
      IF (X .GT. UP) GO TO 8080
      NI = NAF(J)
      GO TO 8090
 8080 CONTINUE
C
C     DELETE THE ACTIVITY FROM NODE NO TO NODE NI.
C
 8090 NET(NO,NI) = 0
      NET(NO,103) = NET(NO,103)-1
      NET(NI,102) = NET(NI,102)-1
 8100 CONTINUE
      GO TO 8250
C
C     THE ADDITION METHOD.
C
 8110 DO 8120 I = 1,N
      NET(I,1) = I
 8120 CONTINUE
C
C     INITIALIZE NUMBER OF NONRECEIVING NODES (NRC) AND NUMBER OF
C     NONEMITTING NODES (NEM).
C
      NRC = N-3
      NEM = N-3
C
C     ADD ACTIVITIES FROM NODE 1 TO NODE 2 AND FROM NODE N-1 TO NODE N.
C
      NET(1,2) = 2
      NET(N-1,N) = N
      NET(1,103) = 1
      NET(2,102) = 1
      NET(N-1,103) = 1
      NET(N,102) = 1
C
C     INITIALIZE NUMBER OF ACTIVITIES ADDED SO FAR (NARCS).
C
      NARCS = 2
C
C     IF INITIAL NUMBER OF FREE ACTIVITIES
C         [NACTS - (2N-4) = NACTS - NDIFF] IS < OR = 0,
C     THEN ALL NACTS ACTIVITIES TO BE ADDED ARE SUBJECT TO FEASIBILITY
C     CONDITIONS AND CANNOT BE RANDOMLY SELECTED.
C
      IF (NDIFF .GE. NACTS) GO TO 8170
C
C     SET FLAG (KK = 0) THAT INITIAL NUMBER OF FREE ACTIVITIES IS > 0.
C
      KK = 0
C
C     RANDOMLY SELECT THE START NODE (NO) OF THE ACTIVITY TO BE ADDED
C     FROM AMONG THE FEASIBLE NODES.
```

```
C
 8130 CALL RNUN(1,Y)
      Y1 = (Y*DN2)+0.25
      X = DN3-SQRT(Y1)
      NO = INT(X)
      IF (NO .GT. X) NO = NO-1
      NN = N-NO
      IF (NET(NO,103) .GE. NN) GO TO 8130
C
C     RANDOMLY SELECT THE END NODE (NI) OF THE ACTIVITY TO BE ADDED
C     FROM AMONG THE FEASIBLE NODES.
C
      K = 0
      DO 8140 J = NO+1,N
      IF (NET(NO,J) .NE. 0) GO TO 8140
      K = K+1
      NAF(K) = J
 8140 CONTINUE
      DEN = 1.0/REAL(K)
      CALL RNUN(1,X)
      UP = 0.0
      DO 8150 J = 1,K
      UP = UP+DEN
      IF (X .GT. UP) GO TO 8150
      NI = NAF(J)
      GO TO 8160
 8150 CONTINUE
C
C     ADD THE ACTIVITY FROM NODE NO TO NODE NI.
C
 8160 NET(NO,NI) = NI
      NARCS = NARCS+1
      IF (NET(NO,103) .EQ. 0) NEM = NEM-1
      IF (NET(NI,102) .EQ. 0) NRC = NRC-1
      NET(NO,103) = NET(NO,103)+1
      NET(NI,102) = NET(NI,102)+1
C
C     IF
C         NUMBER OF ACTIVITIES ADDED SO FAR (NARCS) IS > OR =
C         NUMBER OF ACTIVITIES REQUIRED (NACTS),
C     THEN THE NETWORK IS COMPLETE.
C
      IF (NARCS .GE. NACTS) GO TO 8250
C
C     IF THE FLAG (KK) INDICATES THAT
C         NUMBER OF NONRECEIVING NODES (NRC) = 0, AND
C         NUMBER OF NONEMITTING NODES (NEM) = 0, I.E.
C     FEASIBILITY REQUIREMENTS ARE MET, THEN THE REMAINING ACTIVITIES
C     TO BE ADDED ARE FREE ACTIVITIES AND ARE TO BE RANDOMLY SELECTED.
C
      IF (KK .EQ. 1) GO TO 8130
C
C     IF
C         NUMBER OF FREE ACTIVITIES (NFREE) IS > 0,
C     THEN THE NEXT ACTIVITY TO BE ADDED IS A FREE ACTIVITY AND IS TO BE
C     RANDOMLY SELECTED.
```

```
C
      NFREE = NACTS-NARCS-NRC-NEM
      IF (NFREE .GT. 0) GO TO 8130
C
C     IF
C         NUMBER OF FREE ACTIVITIES (NFREE) = 0, AND
C         NUMBER OF NONRECEIVING NODES (NRC) = 0,
C     THEN CHECK THE NUMBER OF NONEMITTING NODES (NEM).
C
 8170 IF (NRC .EQ. 0) GO TO 8200
C
C     IF NOT, ADD ACTIVITIES SO AS TO REDUCE THE NUMBER OF NONRECEIVING
C     NODES (NRC) TO 0.
C
      K = 0
      DO 8180 I = 3,N-1
      IF (NET(I,102) .GT. 0) GO TO 8180
      K = K+1
      NAF(K) = I
 8180 CONTINUE
      IF (K .EQ. 0) GO TO 8200
      DO 8190 I =1,K
      IJ = K+1-I
      NI = NAF(IJ)
      CALL RNUN(1,Y)
      X = 1.0+(REAL(NI-1)*Y)
      NO = INT(X)
      IF (NO .GT. X) NO = NO-1
      NET(NO,NI) = NI
      NARCS = NARCS+1
      NET(NO,103) = NET(NO,103)+1
      NET(NI,102) = NET(NI,102)+1
 8190 CONTINUE
C
C     IF
C         NUMBER OF NONEMITTING NODES (NEM) = 0,
C     THEN THE FEASIBILITY REQUIREMENTS ARE MET.
C
 8200 IF (NEM .EQ. 0) GO TO 8230
C
C     IF NOT, ADD ACTIVITIES SO AS TO REDUCE THE NUMBER OF NONEMITTING
C     NODES (NEM) TO 0.
C
      K = 0
      DO 8210 I = 2,N-2
      IF (NET(I,103) .GT. 0) GO TO 8210
      K = K+1
      NBE(K) = I
 8210 CONTINUE
      IF (K .EQ. 0) GO TO 8230
      DO 8220 I = 1,K
      NO = NBE(I)
      CALL RNUN(1,X)
      Y = REAL(NO+1)+(REAL(N-NO)*X)
      NI = INT(Y)
      IF (NI .GT. Y) NI = NI-1
      NET(NO,NI) = NI
      NARCS = NARCS+1
      NET(NO,103) = NET(NO,103)+1
      NET(NI,102) = NET(NI,102)+1
 8220 CONTINUE
C
C     SET FLAG (KK = 1) THAT FEASIBILITY REQUIREMENTS HAVE BEEN MET.
C
 8230 KK = 1
C
C     IF NUMBER OF ACTIVITIES ADDED SO FAR (NARCS) IS
C         < NUMBER OF ACTIVITIES REQUIRED (NACTS), THEN RANDOMLY SELECT
C           THE NEXT ACTIVITY TO BE ADDED,
C         = NACTS, THEN THE NETWORK IS COMPLETE,
C         > NACTS, THEN USE THE DELETION METHOD TO DELETE EXCESS
C           ACTIVITIES.
C
      IF (NARCS-NACTS) 8130,8250,8240
 8240 NDEL = NARCS-NACTS
      GO TO 8050
C
C     RECONFIGURE NET ARRAY.
C
 8250 DO 8270 I = 1,N-1
      K = 2
      DO 8260 J = I+1,N
      IF (NET(I,J) .EQ. 0) GO TO 8260
      NET(I,K) = NET(I,J)
      IF (K .LT. J) NET(I,J) = 0
      K = K+1
 8260 CONTINUE
 8270 CONTINUE
      RETURN
      END
C     END SUBROUTINE GENRAN
C
C     ************************************************************
C
C               S U B R O U T I N E         C P U T I M E
C
C     ************************************************************
C
      SUBROUTINE CPUTIME(CPTIME)
      REAL*4 CPTIME
      TYPE TB_TYPE
        SEQUENCE
          REAL*4 USRTIME
          REAL*4 SYSTIME
      END TYPE
      TYPE (TB_TYPE) DTIME_SRC
      CPTIME = DTIME_(DTIME_SRC)
      RETURN
      END
C     END SUBROUTINE CPUTIME
C
C     ************************************************************
```

```
C
C                S U B R O U T I N E          T I M E R
C
C     *************************************************************
C
      SUBROUTINE TIMER(DELTA)
      REAL*4  DELTA,CPU2
      CALL CPUTIME(CPU2)
      DELTA = CPU2
      RETURN
      END
C     END SUBROUTINE TIMER
```

APPENDIX F

VALIDATION VERSION OF PART PROGRAM FOR LARGE NETWORKS

```
C
C
C                           VALIDATION VERSION
C                                   OF
C              POLYGONAL APPROXIMATION AND REDUCTION TECHNIQUE
C                                 (PART)
C                                ALGORITHM
C                    TO APPROXIMATE CRITICALITY INDICES
C                                   OF
C                          ACTIVITIES AND NODES
C                                   AND
C                          TO IDENTIFY THE K MOST
C                     STOCHASTICALLY DOMINATING PATHS
C                                   OF
C                      ACYCLIC, DIRECTED NETWORKS
C                                  USING
C                    "SEQUENTIAL APPROXIMATION" METHOD
C
C
C      THIS PROGRAM GENERATES "STRONGLY RANDOMIZED NETWORKS," APPROXI-
C      MATES THE CRITICALITY INDICES OF THEIR ACTIVITIES AND NODES, AND
C      IDENTIFIES THEIR K MOST STOCHASTICALLY DOMINATING PATHS WITH THE
C      PART ALGORITHM USING "SEQUENTIAL APPROXIMATION," SIMULATES THEM,
C      AND OUTPUTS STATISTICAL COMPARISONS OF THE PART-APPROXIMATED
C      AND SIMULATED INDICES AND CRITICAL PATH SETS.  THE PROGRAM IS
C      WRITTEN IN FORTRAN 77 AND IS PRESENTLY DESIGNED TO BE OPERATED IN
C      A TIME SHARING MODE WITH ALL DATA INPUT FROM TWO (2) DATA FILES.
C      THE PROGRAM DIRECTS OUTPUT TO A TIME SHARING TERMINAL.  IF
DESIRED,
C      IF DESIRED, THE READ STATEMENTS AT THE BEGINNING OF THE MAIN PRO-
C      GRAM CAN BE MODIFIED TO ALLOW DATA INPUT DIRECTLY FROM THE TIME
C      SHARING TERMINAL.
C
C      THE CURRENT DIMENSIONS OF THE PROGRAM ALLOW A NETWORK WITH A
C      MAXIMUM OF 100 NODES AND A MAXIMUM OF 99 ACTIVITIES BEGINNING
C      AT EACH NODE.  THESE LIMITS CAN BE EXPANDED BY CHANGING THE
C      DIMENSIONS OF THE XINT AND VALUE ARRAYS.
C
C
C      **********************************************************
C
C                  O P E R A T I N G   I N S T R U C T I O N S
C
C      **********************************************************
C
C      INSTRUCTIONS FOR BUILDING DATA FILES
C      ------------------------------------
C
C            DATA FILE DATAH.PATHS-RNETGEN
C
C      THIS DATA FILE CONTAINS DESCRIPTIONS OF THE PRECODED DISTRIBUTIONS
C      OF ACTIVITY DURATION.
C
C      THERE ARE 5 FIELDS OF DATA.
C      FIELD 1 IS THE CODE FOR THE TYPE OF DISTRIBUTION.
C          1 = TRIANGULAR DISTRIBUTION
```

```
C         2 = NORMAL DISTRIBUTION
C         3 = EXPONENTIAL DISTRIBUTION
C         4 = GAMMA DISTRIBUTION
C         5 = BETA DISTRIBUTION
C         6 = UNIFORM DISTRIBUTION
C     FIELD 2 IS
C         MODE FOR A TRIANGULAR DISTRIBUTION.
C         MEAN FOR A NORMAL DISTRIBUTION.
C         MEAN FOR AN EXPONENTIAL DISTRIBUTION.
C         ALPHA FOR A GAMMA OR A BETA DISTRIBUTION.
C         1/(B-A) FOR A UNIFORM DISTRIBUTION.
C     FIELD 3 IS BETA FOR A GAMMA OR A BETA DISTRIBUTION.
C     FIELD 4 IS THE MINIMUM VALUE OF THE DISTRIBUTION.
C     FIELD 5 IS THE MAXIMUM VALUE OF THE DISTRIBUTION.
C
C
C           DATA FILE CONTROL.PATHS-RNETGEN
C
C     THIS IS A SINGLE LINE DATA FILE WHICH CONTAINS CONTROL
C     PARAMETERS FOR INPUT, OUTPUT, AND MONTE CARLO SIMULATION.
C
C     THERE ARE 6 FIELDS OF DATA.
C     FIELD 1 IS THE NUMBER OF NETWORKS TO BE GENERATED.
C     FIELD 2 IS THE NUMBER OF NODES IN THE NETWORK.
C         0 = NUMBER OF NODES IS TO BE RANDOMLY GENERATED.
C     FIELD 3 IS THE NUMBER OF ACTIVITIES IN THE NETWORK.
C         0 = NUMBER OF ACTIVITIES IS TO BE RANDOMLY GENERATED.
C     FIELD 4 IS THE DESIRED NUMBER OF PATHS IN THE SET OF K-MOST
C        STOCHASTICALLY DOMINATING PATHS (MAXIMUM = 5).
C     FIELD 5 IS THE NUMBER OF ITERATIONS OF THE MONTE CARLO
C     SIMULATION REQUESTED.
C         0 = NO MONTE CARLO SIMULATION IS REQUESTED.
C     FIELD 6 IS THE NUMBER OF PRECODED DISTRIBUTIONS (MAXIMUM = 20).
C
C
C     NOTE
C
C     ALL UNUSED FIELDS MUST BE ZEROED OUT.
C
C
C
C     ************************************************************
C
C                   M A I N       P R O G R A M
C
C     ************************************************************
C
      REAL*8 XINT(104,500,12),VALUE(104,500,10,3),A(130),
     *       ZVAL(130,5),DIST(20,5),
     *       CRTA(100,99),CRTNA(100,99),CRTN(100),
     *       CRTAS(100,99),CRTNAS(100,99),
     *       X,XSIZE
      REAL*4 AMEAN,ASTD,RN,STEP,UP,XT,DELTA,TOTTIM
      REAL*8 CONST,CRTNN,CUMCRT,RELERR,RELMAX,PR1GE2,TEMPA,TEMPS
      INTEGER I,II,IACT,ICODED,ICOUNT,IENODE,IFLAG,IPATH,IPRE,ISEED,
     *        ISNODE,
     *        J,J1,J2,JJ,
     *        K,KK,KOUNT,
     *        L,L1,L2,L3,LASTK,LA,LPATH,
     *        MM,
     *        N,NACTS,NCL,NCODED,NET,NGEN,NGENCT,NPA,NPPA,
     *        NPATH,NPATHS,NSIM,NSS,NACTSS,NSTART,
     *        UA
      DIMENSION NET(100,103),IPRE(100,99,2),IPATH(100,99,2),NPATH(100),
     *          LPATH(6,101),NPA(500,101),NPPA(100,5)
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      COMMON/PARA4/NET
      COMMON/PARA5/IPRE
      COMMON/PARA6/CRTAS,CRTNAS
      COMMON/PARA7/LPATH
      COMMON/PARA8/NPA,NPPA
      DATA NCL/0/,TOTTIM/0.0/
      EXTERNAL RNSET,RNNOR,RNUN
C
C     INITIALIZE RANDOM NUMBER GENERATOR.
C

      ISEED = 123456789
      CALL RNSET(ISEED)
C
C     OPEN INPUT AND OUTPUT FILES
C
      OPEN (UNIT = 12, FILE = 'datah.paths-rnetgen')
      OPEN (UNIT = 13, FILE = 'control.paths-rnetgen')
C
C     READ CONTROL INFORMATION.
C
      READ (13,1900) NGEN,N,NACTS,NPATHS,NSIM,NCODED
      NSTART = N
      NACTSS = NACTS
C
C     DO 0900 READS DISTRIBUTION DATA FROM DATAH.VAL AND LOADS IT
C     INTO THE DIST ARRAY.
C
      DO 0900 I = 1,NCODED
      READ (12,1902) (DIST(I,J), J=1,5)
 0900 CONTINUE
      STEP = 1.0/REAL(NCODED)
C
C     DO 1540 GENERATES, ANALYZES, SIMULATES, AND STATISTICALLY
C     COMPARES NGEN "STRONGLY RANDOMIZED NETWORKS."
C
      DO 1540 NGENCT = 1,NGEN
      CALL TIMER(DELTA)
      N = NSTART
      NACTS = NACTSS
      PRINT 1910
      PRINT 1915,NGENCT,NGEN
      PRINT 1916
C
```

```
C     RANDOMLY GENERATE THE NUMBER OF NODES (N), IF NECESSARY.
C
      IF (N .GT. 0) GO TO 0910
      CALL RNUN(1,RN)
      XT = 2.0+(99.0*RN)
      N = INT(XT)
      IF (N .GT. 100) N = 100
C
C     RANDOMLY GENERATE THE NUMBER OF ACTIVITIES (NACTS), IF NECESSARY.
C
 0910 IF (NACTS .GT. 0) GO TO 0920
      LA = N-1
      UA = N*(N-1)/2
      AMEAN = ((REAL(LA+UA))/2.)-(((REAL(UA-LA))**2)/500.0)
      ASTD = (REAL(UA-LA))/2.5
      CALL RNNOR(1,RN)
      XT = (RN*ASTD)+AMEAN
      NACTS = INT(XT)
      IF (NACTS .GT. UA) NACTS = UA
C
C     RANDOMLY GENERATE THE NETWORK (NET ARRAY).
C
 0920 CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
      CALL GENRAN(N,NACTS)
      CALL TIMER(DELTA)
C
C     DO 1025 RANDOMLY SELECTS ONE OF THE PRECODED DISTRIBUTIONS
C     FOR EACH ACTIVITY AND LOADS THE DISTRIBUTION'S DATA INTO
C     THE VALUE AND XINT ARRAYS.  THIS DO ALSO DETERMINES IF
C     THE ACTIVITY DISTRIBUTION IS OTHER THAN UNIFORM, AND,
C     IF SO, CALLS LINEAR TO APPROXIMATE IT WITH A
C     PIECEWISE POLYGONAL FUNCTION.
C
      DO 1025 I = 1,N-1
      L1 = I
      DO 1020 J = 1,NET(I,103)
      L2 = J
      CALL RNUN(1,RN)
      UP = 0.0
      DO 1000 K = 1,NCODED
      UP = UP+STEP
      IF (RN .GT. UP ) GO TO 1000
      ICODED = K
      GO TO 1010
 1000 CONTINUE
 1010 VALUE(L1,L2,1,3) = DIST(ICODED,1)
      VALUE(L1,L2,2,3) = DIST(ICODED,2)
      VALUE(L1,L2,3,3) = DIST(ICODED,3)
      XINT(L1,L2,1) = DIST(ICODED,4)
      XINT(L1,L2,2) = DIST(ICODED,5)
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 6) THEN
      CALL LINEAR(L1,L2,NCL)
      GO TO 1020
C
C     DO 1015 CONVERTS DATA FOR UNIFORM DISTRIBUTIONS INTO A USABLE
```

```
C     FORM FOR SUBROUTINES SERIES AND PARA.
C
      ELSE
      XINT(L1,L2,11) = XINT(L1,L2,2)
      X = XINT(L1,L2,1)
      XSIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      DO 1015 K = 1,10
      VALUE(L1,L2,K,1) = VALUE(L1,L2,2,3)
      VALUE(L1,L2,K,2) = 0.0
      XINT(L1,L2,K) = X
      X = X+XSIZE
 1015 CONTINUE
      END IF
 1020 CONTINUE
 1025 CONTINUE
C
C     ANALYSIS OF THE NETWORK BEGINS.
C
C     DO 1070 DETERMINES THE SET OF PREDECESSOR ACTIVITIES, I.E. THE
C     STARTING NODE NUMBER AND THE ACTIVITY NUMBER OF EACH ACTIVITY
WHICH
C     TERMINATES AT NODE I.
C
      DO 1070 I = 2,N
      ICOUNT = 0
      DO 1060 J = 1,I-1
      IF (NET(J,101)) 1550,1060,1030
 1030 DO 1050 J1 = 2,NET(J,103)+1
      IF (NET(J,J1) - I) 1050,1040,1050
 1040 ICOUNT = ICOUNT+1
      IPRE(I,ICOUNT,1) = J
      IPRE(I,ICOUNT,2) = J1-1
 1050 CONTINUE
 1060 CONTINUE
 1070 CONTINUE
C
C     DO 1220 DETERMINES THE DISTRIBUTION THROUGH EACH NODE IN THE
C     FORWARD DIRECTION IN THE NETWORK.
C
      DO 1220 I = 2,N
C
C     THROUGH 1220 CONVOLVES THE RESOURCE CONSUMPTION DISTRIBUTION
C     THROUGH THE STARTING NODE OF THE ACTIVITY AND THE RESOURCE
C     CONSUMPTION DISTRIBUTION OF EACH ACTIVITY WHICH TERMINATES
C     AT NODE I AND THEN FINDS THE MAXIMUM OF THESE CONVOLUTIONS.
C
C     IF THE FIRST STARTING NODE = NODE 1, THE CONVOLUTION IS EQUAL TO
C     THE DISTRIBUTION OF THE ACTIVITY WHICH TERMINATES AT NODE I.
C
      IF (IPRE(I,1,1) .EQ. 1) THEN
      IACT = IPRE(I,1,2)
      DO 1175 J = 1,10
      XINT(101,1,J) = XINT(1,IACT,J)
      VALUE(101,1,J,1) = VALUE(1,IACT,J,1)
      VALUE(101,1,J,2) = VALUE(1,IACT,J,2)
 1175 CONTINUE
```

```
      XINT(101,1,11) = XINT(1,IACT,11)
C
C     OTHERWISE, LOAD THE DISTRIBUTION THROUGH THE FIRST STARTING NODE
C     INTO TEMPORARY LOCATION 1.
C
      ELSE
      ISNODE = IPRE(I,1,1)
      IACT = IPRE(I,1,2)
      DO 1180 J = 1,10
      XINT(101,1,J) = XINT(ISNODE,100,J)
      VALUE(101,1,J,1) = VALUE(ISNODE,100,J,1)
      VALUE(101,1,J,2) = VALUE(ISNODE,100,J,2)
 1180 CONTINUE
      XINT(101,1,11) = XINT(ISNODE,100,11)
C
C     LOAD THE DISTRIBUTION OF THE FIRST ACTIVITY TERMINATING AT NODE I
C     IN TEMPORARY LOCATION 2.
C
      DO 1185 J = 1,10
      XINT(101,2,J) = XINT(ISNODE,IACT,J)
      VALUE(101,2,J,1) = VALUE(ISNODE,IACT,J,1)
      VALUE(101,2,J,2) = VALUE(ISNODE,IACT,J,2)
 1185 CONTINUE
      XINT(101,2,11) = XINT(ISNODE,IACT,11)
C
C     CONVOLVE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2 AND
C     PLACE THE CONVOLUTION IN TEMPORARY LOCATION 1.
C
      CALL SERIES(101,1,101,2)
      END IF
      CONTINUE
C
C     IF THERE IS ONLY ONE ACTIVITY TERMINATING AT NODE I, THE
DISTRIBUTION
C     THROUGH NODE I IS THE CONVOLUTION IN TEMPORARY LOCATION 1.  LOAD
THIS
C     INTO THE 100TH ACTIVITY POSITION OF NODE I.
C
      IF (NET(I,102) .EQ. 1) THEN
      DO 1190 J = 1,10
      XINT(I,100,J) = XINT(101,1,J)
      VALUE(I,100,J,1) = VALUE(101,1,J,1)
      VALUE(I,100,J,2) = VALUE(101,1,J,2)
 1190 CONTINUE
      XINT(I,100,11) = XINT(101,1,11)
C
C     IF THERE ARE TWO OR MORE ACTIVITIES TERMINATING AT NODE I, LOAD
THE
C     DISTRIBUTION THROUGH THE STARTING NODE OF THE NEXT ACTIVITY INTO
C     TEMPORARY LOCATION 3.
C
      ELSE
      DO 1205 K = 2,NET(I,102)
      ISNODE = IPRE(I,K,1)
      IACT = IPRE(I,K,2)
      DO 1195 J = 1,10
      XINT(101,3,J) = XINT(ISNODE,100,J)
      VALUE(101,3,J,1) = VALUE(ISNODE,100,J,1)
      VALUE(101,3,J,2) = VALUE(ISNODE,100,J,2)
 1195 CONTINUE
      XINT(101,3,11) = XINT(ISNODE,100,11)
C
C     THEN LOAD THE DISTRIBUTION OF THE NEXT ACTIVITY INTO TEMPORARY
C     LOCATION 4.
C
      DO 1200 J = 1,10
      XINT(101,4,J) = XINT(ISNODE,IACT,J)
      VALUE(101,4,J,1) = VALUE(ISNODE,IACT,J,1)
      VALUE(101,4,J,2) = VALUE(ISNODE,IACT,J,2)
 1200 CONTINUE
      XINT(101,4,11) = XINT(ISNODE,IACT,11)
C
C     CONVOLUTE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 3 AND 4 AND
C     LOAD THE CONVOLUTION INTO TEMPORARY LOCATION 3.
C
      CALL SERIES(101,3,101,4)
C
C     PARALLEL-REDUCE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 3
AND
C     LOAD THE MAXIMUM INTO TEMPORARY LOCATION 1.
C
      CALL PARA(101,1,3)
 1205 CONTINUE
C
C     THE FORWARD DISTRIBUTION THROUGH NODE I IS THE MAXIMUM IN
TEMPORARY
C     LOCATION 1.  LOAD THIS INTO THE 100TH ACTIVITY POSITION OF NODE I.
C
      DO 1210 J = 1,10
      XINT(I,100,J) = XINT(101,1,J)
      VALUE(I,100,J,1) = VALUE(101,1,J,1)
      VALUE(I,100,J,2) = VALUE(101,1,J,2)
 1210 CONTINUE
      XINT(I,100,11) = XINT(101,1,11)
      END IF
 1220 CONTINUE
C
C     DO 1270 DETERMINES THE DISTRIBUTION THROUGH EACH NODE IN THE
C     BACKWARD DIRECTION IN THE NETWORK.
C
      DO 1270 I = N-1,1,-1
C
C     THROUGH 1270 CONVOLVES THE RESOURCE CONSUMPTION DISTRIBUTION
C     THROUGH THE ENDING NODE OF THE ACTIVITY AND THE RESOURCE
C     CONSUMPTION DISTRIBUTION OF EACH ACTIVITY WHICH STARTS
C     AT NODE I AND THEN FINDS THE MAXIMUM OF THESE CONVOLUTIONS.
C
C     IF THE LAST ENDING NODE = NODE N, THE CONVOLUTION IS EQUAL TO
C     THE DISTRIBUTION OF THE ACTIVITY WHICH TERMINATES AT NODE N.
C
      IACT = NET(I,103)
      IENODE = NET(I,IACT+1)
      IF (IENODE .EQ. N) THEN
      DO 1225 J = 1,10
      XINT(101,1,J) = XINT(I,IACT,J)
      VALUE(101,1,J,1) = VALUE(I,IACT,J,1)
      VALUE(101,1,J,2) = VALUE(I,IACT,J,2)
 1225 CONTINUE
      XINT(101,1,11) = XINT(I,IACT,11)
C
C     OTHERWISE, LOAD THE DISTRIBUTION THROUGH THE LAST ENDING NODE
C     INTO TEMPORARY LOCATION 1.
C
      ELSE
      DO 1230 J = 1,10
      XINT(101,1,J) = XINT(IENODE,101,J)
      VALUE(101,1,J,1) = VALUE(IENODE,101,J,1)
      VALUE(101,1,J,2) = VALUE(IENODE,101,J,2)
 1230 CONTINUE
      XINT(101,1,11) = XINT(IENODE,101,11)
C
C     LOAD THE DISTRIBUTION OF THE LAST ACTIVITY STARTING AT NODE I
C     IN TEMPORARY LOCATION 2.
C
      DO 1235 J = 1,10
      XINT(101,2,J) = XINT(I,IACT,J)
      VALUE(101,2,J,1) = VALUE(I,IACT,J,1)
      VALUE(101,2,J,2) = VALUE(I,IACT,J,2)
 1235 CONTINUE
      XINT(101,2,11) = XINT(I,IACT,11)
C
C     CONVOLVE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2 AND
C     PLACE THE CONVOLUTION IN TEMPORARY LOCATION 1.
C
      CALL SERIES(101,1,101,2)
      END IF
      CONTINUE
C
C     IF THERE IS ONLY ONE ACTIVITY STARTING AT NODE I, THE DISTRIBUTION
C     THROUGH NODE I IS THE CONVOLUTION IN TEMPORARY LOCATION 1.  LOAD
THIS
C     INTO THE 101ST ACTIVITY POSITION OF NODE I.
C
      IF (NET(I,103) .EQ. 1) THEN
      DO 1240 J = 1,10
      XINT(I,101,J) = XINT(101,1,J)
      VALUE(I,101,J,1) = VALUE(101,1,J,1)
      VALUE(I,101,J,2) = VALUE(101,1,J,2)
 1240 CONTINUE
      XINT(I,101,11) = XINT(101,1,11)
C
C     IF THERE ARE TWO OR MORE ACTIVITIES STARTING AT NODE I, LOAD THE
C     DISTRIBUTION THROUGH THE ENDING NODE OF THE NEXT ACTIVITY INTO
C     TEMPORARY LOCATION 3.
C
      ELSE
      DO 1255 K = NET(I,103)-1,1,-1
      IACT = K
      IENODE = NET(I,IACT)+1
      DO 1245 J = 1,10
      XINT(101,3,J) = XINT(IENODE,101,J)
      VALUE(101,3,J,1) = VALUE(IENODE,101,J,1)
      VALUE(101,3,J,2) = VALUE(IENODE,101,J,2)
 1245 CONTINUE
      XINT(101,3,11) = XINT(IENODE,101,11)
C
C     THEN LOAD THE DISTRIBUTION OF THE NEXT ACTIVITY INTO TEMPORARY
C     LOCATION 4.
C
      DO 1250 J = 1,10
      XINT(101,4,J) = XINT(I,IACT,J)
      VALUE(101,4,J,1) = VALUE(I,IACT,J,1)
      VALUE(101,4,J,2) = VALUE(I,IACT,J,2)
 1250 CONTINUE
      XINT(101,4,11) = XINT(I,IACT,11)
C
C     CONVOLUTE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 3 AND 4 AND
C     LOAD THE CONVOLUTION INTO TEMPORARY LOCATION 3.
C
      CALL SERIES(101,3,101,4)
C
C     PARALLEL-REDUCE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 3
AND
C     LOAD THE MAXIMUM INTO TEMPORARY LOCATION 1.
C
      CALL PARA(101,1,3)
 1255 CONTINUE
C
C     THE BACKWARD DISTRIBUTION THROUGH NODE I IS THE MAXIMUM IN
TEMPORARY
C     LOCATION 1.  LOAD THIS INTO THE 101ST ACTIVITY POSITION OF NODE I.
C
      DO 1260 J = 1,10
      XINT(I,101,J) = XINT(101,1,J)
      VALUE(I,101,J,1) = VALUE(101,1,J,1)
      VALUE(I,101,J,2) = VALUE(101,1,J,2)
 1260 CONTINUE
      XINT(I,101,11) = XINT(101,1,11)
      END IF
 1270 CONTINUE
C
C     DO 1275 INITIALIZES THE CRITICALITY INDICES OF THE NODES (CRTN).
C
      DO 1275 I = 1,N
      CRTN(I) = 0.0
 1275 CONTINUE
C
C     THROUGH 1415 CALCULATES THE CRITICALITY INDICES OF ALL THE
C     ACTIVITIES AND NODES IN THE NETWORK.
C
      DO 1415 I = N,2,-1
      IF (I .NE. N) GO TO 1280
      NPATH(I) = NET(I,102)
      KK = 1
      NSS = NET(I,102)
      GO TO 1295
C
C     AT NODE I, THE NUMBER OF PATHS TO BE CONSIDERED IS:
C          (NUMBER OF PATHS CONSIDERED AT NODE (I+1))
C       + (IN-DEGREE OF NODE I) - (OUT-DEGREE OF NODE I).
C
 1280 NPATH(I) = NPATH(I+1)+NET(I,102)-NET(I,103)
      NSS = NPATH(I+1)
      K = 0
C
C     DO 1290 SHIFTS TO NODE I ALL THE PATHS CONSIDERED AT NODE (I+1)
C     EXCEPT THOSE PATHS WHOSE PREDECESSOR ACTIVITIES START AT NODE I.
C
      DO 1290 JJ = 1,NSS
      ISNODE = IPATH(I+1,JJ,1)
      IACT = IPATH(I+1,JJ,2)
      IF (ISNODE .EQ. I) GO TO 1290
      K = K+1
      IPATH(I,K,1) = ISNODE
      IPATH(I,K,2) = IACT
      DO 1285 J = 1,10
      XINT(102,K,J) = XINT(102,JJ,J)
      VALUE(102,K,J,1) = VALUE(102,JJ,J,1)
      VALUE(102,K,J,2) = VALUE(102,JJ,J,2)
 1285 CONTINUE
      XINT(102,K,11) = XINT(102,JJ,11)
 1290 CONTINUE
      KK = K+1
      NSS = NPATH(I)
 1295 CONTINUE
C
C     DO 1320 DETERMINES THE DISTRIBUTIONS OF ALL PATHS WHICH INCLUDE
THE
C     (IN-DEGREE - OF - NODE I) PREDECESSOR ACTIVITIES OF NODE I.
C
      DO 1320 J = KK,NSS
      JJ = J-KK+1
      IPATH(I,J,1) = IPRE(I,JJ,1)
      IPATH(I,J,2) = IPRE(I,JJ,2)
C
C     LOAD THE FORWARD DISTRIBUTION THROUGH THE STARTING NODE OF THE
JJth
C     PREDECESSOR ACTIVITY OF NODE I INTO TEMPORARY LOCATION 1.
C
      ISNODE = IPATH(I,J,1)
      IACT = IPATH(I,J,2)
      DO 1300 K = 1,10
      XINT(101,1,K) = XINT(ISNODE,100,K)
      VALUE(101,1,K,1) = VALUE(ISNODE,100,K,1)
      VALUE(101,1,K,2) = VALUE(ISNODE,100,K,2)
 1300 CONTINUE
      XINT(101,1,11) = XINT(ISNODE,100,11)
C
C     LOAD THE DISTRIBUTION OF THE JJth PREDECESSOR ACTIVITY OF NODE I
C     INTO TEMPORARY LOCATION 2.
```

```
C
      DO 1305 K = 1,10
      XINT(101,2,K) = XINT(ISNODE,IACT,K)
      VALUE(101,2,K,1) = VALUE(ISNODE,IACT,K,1)
      VALUE(101,2,K,2) = VALUE(ISNODE,IACT,K,2)
 1305 CONTINUE
      XINT(101,2,11) = XINT(ISNODE,IACT,11)
C
C     CONVOLVE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2 AND
PLACE
C     THE CONVOLUTION IN TEMPORARY LOCATION 1.
C
C     IF THE STARTING NODE OF THE JJth PREDECESSOR ACTIVITY OF NODE I IS
C     THE STARTING NODE OF THE NETWORK, NODE 1, THE CONVOLUTION IS THE
C     DISTRIBUTION OF THE JJth PREDECESSOR ACTIVITY.
C
      IF (ISNODE .EQ.1) THEN
      DO 1306 K = 1,10
      XINT(101,1,K) = XINT(101,2,K)
      VALUE(101,1,K,1) = VALUE(101,2,K,1)
      VALUE(101,1,K,2) = VALUE(101,2,K,2)
 1306 CONTINUE
      XINT(101,1,11) = XINT(101,2,11)
      ELSE
      CALL SERIES(101,1,101,2)
      END IF
      CONTINUE
      IF (I .EQ. N) GO TO 1311
C
C     LOAD THE BACKWARD DISTRIBUTION THROUGH NODE I, THE ENDING NODE OF
C     THE JJth PREDECESSOR ACTIVITY TO NODE I, INTO TEMPORARY LOCATION
2.
C
      DO 1310 K = 1,10
      XINT(101,2,K) = XINT(I,101,K)
      VALUE(101,2,K,1) = VALUE(I,101,K,1)
      VALUE(101,2,K,2) = VALUE(I,101,K,2)
 1310 CONTINUE
      XINT(101,2,11) = XINT(I,101,11)
C
C     CONVOLVE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2 AND
PLACE
C     THE CONVOLUTION IN TEMPORARY LOCATION 1.
C
      CALL SERIES(101,1,101,2)
C
C     THE CONVOLUTION IS THE DISTRIBUTION OF ALL PATHS WHICH INCLUDE THE
C     JJth PREDECESSOR ACTIVITY OF NODE I.  LOAD THIS DISTRIBUTION INTO
THE
C     Jth ACTIVITY POSITION OF NODE 102.
C
 1311 DO 1315 K = 1,10
      XINT(102,J,K) = XINT(101,1,K)
      VALUE(102,J,K,1) = VALUE(101,1,K,1)
      VALUE(102,J,K,2) = VALUE(101,1,K,2)
 1315 CONTINUE
```

```
      XINT(102,J,11) = XINT(101,1,11)
 1320 CONTINUE
      IF (NET(I,102) .EQ. 1) GO TO 1390
C
C     TO 1360 DETERMINES THE MAXIMUM OF THE DISTRIBUTIONS OF ALL THE
C     PATHS CONSIDERED AT NODE (I+1) EXCEPT THOSE PATHS WHOSE
PREDECESSOR
C     ACTIVITIES START AT NODE I.  DO 1290 SHIFTED THESE PATHS TO NODE
I.
C
      IFLAG = 0
      IF (NPATH(I)-NET(I,102)-1) 1325,1335,1345
C
C     IF
C         NUMBER OF PATHS CONSIDERED AT NODE (I+1)
C       = NUMBER OF PREDECESSOR ACTIVITIES OF NODE I,
C     THE MAXIMUM IS THE 0 DISTRIBUTION.  SET A FLAG (IFLAG = 1).
C
 1325 IFLAG = 1
      GO TO 1360
C
C     IF
C         NUMBER OF PATHS CONSIDERED AT NODE (I+1)
C       = (NUMBER OF PREDECESSOR ACTIVITIES OF NODE I) + 1,
C     THE MAXIMUM IS THE DISTRIBUTION OF THE ONE PATH WHOSE PREDECESSOR
C     ACTIVITY DOES NOT START AT NODE I.  LOAD THIS DISTRIBUTION INTO
C     TEMPORARY LOCATION 3.
C
 1335 DO 1340 K = 1,10
      XINT(101,3,K) = XINT(102,1,K)
      VALUE(101,3,K,1) = VALUE(102,1,K,1)
      VALUE(101,3,K,2) = VALUE(102,1,K,2)
 1340 CONTINUE
      XINT(101,3,11) = XINT(102,1,11)
      GO TO 1360
C
C     IF
C         NUMBER OF PATHS CONSIDERED AT NODE (I+1)
C       > (NUMBER OF PREDECESSOR ACTIVITIES OF NODE I) + 1,
C     THERE ARE TWO OR MORE PATHS WHOSE PREDECESSOR ACTIVITIES DO NOT
START
C     AT NODE I.  DO 1350 DETERMINES THE MAXIMUM OF THE DISTRIBUTIONS OF
C     THESE PATHS.
C
C     DO 1347 LOADS THE 1st PATH WHOSE PREDECESSOR ACTIVITIES DO NOT
START
C     AT NODE I INTO TEMPORARY LOCATION 4.
C
 1345 DO 1347 K = 1,10
      XINT(101,4,K) = XINT(102,1,K)
      VALUE(101,4,K,1) = VALUE(102,1,K,1)
      VALUE(101,4,K,2) = VALUE(102,1,K,2)
 1347 CONTINUE
      XINT(101,4,11) = XINT(102,1,11)
      DO 1350 K = 2,NPATH(I)-NET(I,102)
      CALL PARA(102,1,K)
 1350 CONTINUE
C
C     LOAD THIS MAXIMUM DISTRIBUTION INTO TEMPORARY LOCATION 3.
C
      DO 1355 K = 1,10
      XINT(101,3,K) = XINT(102,1,K)
      VALUE(101,3,K,1) = VALUE(102,1,K,1)
      VALUE(101,3,K,2) = VALUE(102,1,K,2)
 1355 CONTINUE
      XINT(101,3,11) = XINT(102,1,11)
C
C     RELOAD THE 1st PATH WHOSE PREDECESSOR ACTIVITIES DO NOT START AT
C     NODE I BACK INTO THE 1st ACTIVITY POSITION OF NODE 102.
C
      DO 1357 K = 1,10
      XINT(102,1,K) = XINT(101,4,K)
      VALUE(102,1,K,1) = VALUE(101,4,K,1)
      VALUE(102,1,K,2) = VALUE(101,4,K,2)
 1357 CONTINUE
      XINT(102,1,11) = XINT(101,4,11)
C
C     DO 1380 DETERMINES
C         P(ALL PATHS WHICH INCLUDE THE Jth PREDECESSOR ACTIVITY >
C           ALL OTHER PATHS)
C     FOR EACH PREDECESSOR ACTIVITY (IACT) OF NODE I.  THIS PROBABILITY
C     IS THE CRITICALITY INDEX OF THE ACTIVITY (CRTA(I,IACT)).
C
 1360 DO 1380 J = NPATH(I)-NET(I,102)+1,NPATH(I)
C
C     LOAD THE MAXIMUM DISTRIBUTION OF ALL PATHS WHOSE PREDECESSOR
C     ACTIVITIES DO NOT START AT NODE I INTO TEMPORARY LOCATION 1.
C
      DO 1365 K = 1,10
      XINT(101,1,K) = XINT(101,3,K)
      VALUE(101,1,K,1) = VALUE(101,3,K,1)
      VALUE(101,1,K,2) = VALUE(101,3,K,2)
 1365 CONTINUE
      XINT(101,1,11) = XINT(101,3,11)
      DO 1375 K = NPATH(I)-NET(I,102)+1,NPATH(I)
      IF (K .EQ. J) GO TO 1375
C
C     LOAD THE DISTRIBUTIONS OF ALL PATHS WHICH INCLUDE THE Kth
PREDECES-
C     SOR ACTIVITY OF NODE I INTO TEMPORARY LOCATION 2, WHERE K IS DIF-
C     FERENT FROM J.
C
      DO 1370 KK = 1,10
      XINT(101,2,KK) = XINT(102,K,KK)
      VALUE(101,2,KK,1) = VALUE(102,K,KK,1)
      VALUE(101,2,KK,2) = VALUE(102,K,KK,2)
 1370 CONTINUE
      XINT(101,2,11) = XINT(102,K,11)
C
C     PARALLEL-REDUCE THE DISTRIBUTIONS IN TEMPORARY LOCATIONS 1 AND 2
AND
C     LOAD THE MAXIMUM INTO TEMPORARY LOCATION 1.
```

```
C
C     IF THE MAXIMUM DISTRIBUTION OF ALL PATHS WHOSE PREDECESSOR ACTIVI-
C     TIES DO NOT START AT NODE I IS THE 0 DISTRIBUTION, THE PARALLEL-
C     REDUCTION WITH THE DISTRIBUTION OF ALL PATHS WHICH INCLUDE THE 1st
C     PREDECESSOR ACTIVITY OF NODE I IS THE DISTRIBUTION OF THE LATTER.
C
      IF ((K .EQ. NPATH(I)-NET(I,102)+1) .AND. (IFLAG .EQ. 1)) THEN
      DO 1372 KK = 1,10
      XINT(101,1,KK) = XINT(101,2,KK)
      VALUE(101,1,KK,1) = VALUE(101,2,KK,1)
      VALUE(101,1,KK,2) = VALUE(101,2,KK,2)
 1372 CONTINUE
      XINT(101,1,11) = XINT(101,2,11)
      ELSE
      CALL PARA(101,1,2)
      END IF
 1375 CONTINUE
      CALL COMPAR(102,J,101,1,PR1GE2)
      IACT = J-(NPATH(I)-NET(I,102))
      CRTA(I,IACT) = PR1GE2
 1380 CONTINUE
C
C     THROUGH 1410 COMPUTES NORMALIZED CRITICALITY INDICES OF THE Jth
C     PREDECESSOR ACTIVITIES OF NODE I (CRTNA(I,J)) AND THE CRITICALITY
C     INDEX OF NODE I (CRTN(I)).
C
      IF (I .NE. N) GO TO 1400
      CUMCRT = 0.0
      DO 1385 J = 1,NET(N,102)
      CUMCRT = CUMCRT + CRTA(I,J)
 1385 CONTINUE
      CRTN(N) = CUMCRT
      CONST = CUMCRT
      IF (CONST .EQ. 0.0) CONST = 1.0
      GO TO 1400
 1390 IF (I .EQ. N) GO TO 1395
      CRTA(I,1) = CRTN(I)
      CRTNA(I,1) = CRTA(I,1)/CONST
      GO TO 1400
 1395 CRTN(I) = 1.0
      CRTA(I,1) = CRTN(I)
      CRTNA(I,1) = CRTA(I,1)
      CONST = 1.0
 1400 PRINT 1920,I
      PRINT 1925
      DO 1405 J = 1,NET(I,102)
      CRTNA(I,J) = CRTA(I,J)/CONST
      PRINT 1930,IPRE(I,J,1),CRTA(I,J),CRTNA(I,J)
 1405 CONTINUE
      DO 1410 J = 1,NET(I,102)
      ISNODE = IPRE(I,J,1)
      CRTN(ISNODE) = CRTN(ISNODE)+CRTA(I,J)
 1410 CONTINUE
 1415 CONTINUE
C
C     DO 1420 COMPUTES NORMALIZED CRITICALITY INDICES OF THE NODES
```

```
C     (CRTNN).
C
      PRINT 1935
      DO 1420 I = 1,N
      CRTNN = CRTN(I)/CONST
      PRINT 1930,I,CRTN(I),CRTNN
 1420 CONTINUE
C
C     MONTE CARLO SIMULATION OF THE NETWORK.
C
      IF (NSIM .EQ. 0) GO TO 1540
      CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
      CALL SIMULC(N,NSIM,NPATHS)
      CALL TIMER(DELTA)

C
C     COMPUTE THE NUMBER OF OTHER ACTIVITIES WHICH HAVE LARGER
CRITICALITY
C     INDICES THAN EACH ACTIVITY AND THE RELATIVE ERROR OF EACH ACTIVITY
C     CRITICALITY INDEX.
C
      PRINT 1910
      PRINT 1940
      RELMAX = 0.0
      DO 1490 I = N,2,-1
      DO 1480 J = 1,NET(I,102)
      K = 0
      KK = 0
      ISNODE = IPRE(I,J,1)
      TEMPA = CRTA(I,J)
      TEMPS = CRTAS(I,J)
      DO 1470 II = N,2,-1
      DO 1460 JJ = 1,NET(II,102)
      IF ((CRTA(II,JJ)-TEMPA) .GT. 1.0D-06) GO TO 1430
      GO TO 1440
 1430 K = K+1
 1440 IF ((CRTAS(II,JJ)-TEMPS) .GT. 1.0D-06) GO TO 1450
      GO TO 1460
 1450 KK = KK+1
 1460 CONTINUE
 1470 CONTINUE
      IF ((CRTNAS(I,J) .EQ. 0.0) .AND. (CRTNA(I,J) .EQ. 0.0)) THEN
      RELERR = 0.0
      ELSE
      RELERR = ((CRTNA(I,J)-CRTNAS(I,J))/CRTNAS(I,J))*100.0
      END IF
      IF ((DABS(RELERR) .GT. DABS(RELMAX)) .AND. (CRTNAS(I,J) .NE. 0.0))
     *RELMAX = RELERR
      IF ((CRTNAS(I,J) .EQ. 0.0) .AND. (CRTNA(I,J) .NE. 0.0)) THEN
      PRINT 1944,ISNODE,I,CRTA(I,J),CRTAS(I,J),CRTNA(I,J),CRTNAS(I,J),
     *           K,KK
      ELSE
      PRINT 1945,ISNODE,I,CRTA(I,J),CRTAS(I,J),CRTNA(I,J),CRTNAS(I,J),
     *           K,KK,RELERR
      END IF
 1480 CONTINUE
 1490 CONTINUE
      PRINT 1950, RELMAX
C
C     DETERMINE THE NPATHS MOST STOCHASTICALLY DOMINATING PATHS THROUGH
THE
C     NETWORK.
C
      CALL DOMPTH(N,NPATHS)
      PRINT 1960
      DO 1500 K = 1,NPATHS
      PRINT 1970,K,LPATH(K,101),(LPATH(K,KK),KK = 1,LPATH(K,101))
 1500 CONTINUE
      PRINT 1980
C
C     DO 1530 COUNTS THE NUMBER OF NODES IN COMMON BETWEEN THE Kth MOST
C     STOCHASTICALLY DOMINATING PATH AND THE SIMULATION-APPROXIMATED Kth
MOST
C     CRITICAL PATH.
C
      DO 1530 K = 1,NPATHS
      KOUNT = 0
      JJ = NPPA(N,K)
      DO 1520 I = 1,LPATH(K,101)
      DO 1510 J = 2,NPA(JJ,1)+1
      IF (LPATH(K,I) .NE. NPA(JJ,J)) GO TO 1510
      KOUNT = KOUNT+1
      GO TO 1520
 1510 CONTINUE
 1520 CONTINUE
      PRINT 1990,K,NPA(JJ,1),LPATH(K,101),KOUNT
 1530 CONTINUE
 1540 CONTINUE
      CALL TIMER(DELTA)
      TOTTIM = TOTTIM+DELTA
      PRINT 1996,TOTTIM
      STOP
 1550 PRINT 1995
      STOP
C
C     FORMAT STATEMENTS
C
 1900 FORMAT (2(I3,1X),I4,1X,I1,1X,I5,1X,I2)
 1901 FORMAT (3(I2,25(1X,I2)/),I2,21(1X,I2),1X,I1,2(1X,I2))
 1902 FORMAT (F1.0,4(1X,F8.2))
 1910 FORMAT (1H1)
 1915 FORMAT (1X,'THE RESULTS FOR NETWORK NUMBER ',I3,' OF ',I3,
     &' NETWORKS GENERATED ARE:' //)
 1916 FORMAT (1X,'FROM THE POLYGONAL APPROXIMATION AND REDUCTION ',
     &'TECHNIQUE:')
 1920 FORMAT (/ 1X,'THE ACTIVITIES ENDING AT NODE ',I3,' AND THEIR ',
     &'CRITICALITY INDICES ARE:')
 1925 FORMAT (1X,24X,'NORMALIZED'/
     &        1X,'STARTING',2X,'CRITICALITY',2X,'CRITICALITY'/
     &        1X,2X,'NODE',7X,'INDEX',8X,'INDEX')
 1930 FORMAT (1X,2X,I3,7X,F7.5,6X,F7.5)
 1935 FORMAT (/ 1X,'THE CRITICALITY INDICES OF THE NODES ARE:'/
     &          1X,23X,'NORMALIZED'/
     &          1X,10X,'CRITICALITY',2X,'CRITICALITY'/
     &          1X,2X,'NODE',7X,'INDEX',8X,'INDEX')
 1940 FORMAT (1X,'COMPARISONS OF ACTIVITY CRITICALITY INDICES:'//
     &        1X,28X,'NORM.',3X,'NORM.',2X,'NO. ACTS.',1X,'NO. ACTS.',
     &           3X,'RELATIVE'/
     &        1X,12X,'CRIT.',3X,'CRIT.',3X,'CRIT.',3X,'CRIT.',3X,
     &           'GREATER',3X,'GREATER',5X,'ERROR'/
     &        1X,'START',2X,'END',2X,'INDEX',3X,'INDEX',3X,'INDEX',3X,
     &           'INDEX',2X,'CRIT. IN.',1X,'CRIT. IN.',3X,'OF NORM.'/
     &        1X,1X,'NODE',1X,'NODE',1X,'(APPR.)',2X,'(SIM.)',1X,
     &           '(APPR.)',2X,'(SIM.)',2X,'(APPR.)',4X,'(SIM.)',2X,
     &           'CRIT. INDEX')
 1944 FORMAT (1X,1X,I3,3X,I3,4(1X,F7.5),2(4X,I3,3X),2X,'UNDEFINED')
 1945 FORMAT (1X,1X,I3,3X,I3,4(1X,F7.5),2(4X,I3,3X),2X,F7.2,'%')
 1950 FORMAT (/ 1X,'THE MAXIMUM (ABSOLUTE) RELATIVE ERROR OF'/
     &          1X,'NORMALIZED ACTIVITY CRITICALITY INDEX IS: ',
     &            F7.2,'%')
 1960 FORMAT (/ 1X,'FROM MONTE CARLO SIMULATION:')
 1970 FORMAT(/ 1X,'THE RANK ',I1,' APPROXIMATED CRITICAL PATH WITH ',
     &           I3,' NODES:'/
     &         (1X,20I4/))
 1980 FORMAT (// 1X,'COMPARISONS OF PATHS:')
 1990 FORMAT (/ 1X,'RANK ',I1,' PATHS:'/
     &          1X,3('NO. NODES',2X)/
     &          1X,1X,'(APPR.)',5X,'(SIM.)',3X,'IN COMMON'/
     &          1X,3(3X,I3,5X))
 1995 FORMAT (1X,'PROGRAM STOPPED' / 1X,'IMPROPER NODE NUMBER(S) '
     &,'ENCOUNTERED')
 1996 FORMAT (// 1X,'CPU TIME FOR PART PROCESSING IS ',F8.3,' SECONDS'
     &//)
      END
C     END MAIN PROGRAM
C
C     ***************************************************************
C
C                S U B R O U T I N E           P A R A
C
C     ***************************************************************
C
      SUBROUTINE PARA (L1,L2,L3)
      REAL*8 VALUE(104,500,10,3),XINT(104,500,12)
      REAL*8 XVAL,ZVAL(130,5),PAR(2,15,6),FACT,B(130)
      REAL*4 Z
      INTEGER L1,L2,L3,NV1,NV2
      INTEGER K4(2,30)
      INTEGER I,IINT,N,NCL,J,K,K3,L6,LASTJ,LASTK
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/B
C
C     SUBROUTINE PARA IS USED TO REDUCE PARALLEL ARCS INTO A SINGLE
C     EQUIVALENT ARC.  IT FINDS THE MAX OPERATOR BY MULTIPLYING CAP
C     F(X) AGAINST CAP G(X) OVER THE INTERVALS OF VALIDITY.
C
```

```
      NV1 = 10
      NV2 = 10
      DO 2020 N = 1,2
      L6 = L2
      IF (N .EQ. 2) L6 = L3
      FACT = 0
      DO 2010 J = 1,10
      B(1) = XINT(L1,L6,J)
C
C     DO 2000 CONVERTS EACH LINEAR POLYNOMIAL PIECE OF LITTLE F(X)
C     INTO THE CORRESPONDING QUADRATIC POLYNOMIAL PIECE OF ITS
C     CUMULATIVE DISTRIBUTION CAP F(X).
C
      DO 2000 I = 1,2
      XVAL = VALUE(L1,L6,J,I)
      Z = FLOAT(I)
      PAR(N,J,I+1) = XVAL/Z
      PAR(N,J,1) = PAR(N,J,1)+((-1.0)*(XVAL/Z)*(B(1)**I))
      K4(N,J) = I+1
 2000 CONTINUE
      IF (J .GT. 1) PAR(N,J,1) = PAR(N,J,1)+FACT
      FACT = PAR(N,J,1)+(PAR(N,J,2)*XINT(L1,L6,J+1))+(PAR(N,J,3)
     &*(XINT(L1,L6,J+1)**2))
 2010 CONTINUE
 2020 CONTINUE
C
C     DO 2040 ASSIGNS INTERVAL BOUNDARY VALUES TO THE B ARRAY.
C
      DO 2040 I = 1,22
      IF (I .GT. 11) GO TO 2030
      B(I) = XINT(L1,L2,I)
      GO TO 2040
 2030 B(I) = XINT(L1,L3,I-11)
 2040 CONTINUE
      NCL = 21
      CALL SORT(NCL)
C
C     DO 2080 DETERMINES THE POINT AT WHICH THE DISTRIBUTION DOMAINS
C     OF THE TWO ARCS BEING COMBINED OVERLAP.  ONCE THIS POINT IS
C     DETERMINED, THE B ARRAY IS ADJUSTED TO REFLECT THE OVERLAP
C     (ALL VALUES LESS THAN THIS POINT OF FIRST OVERLAP NEED NOT BE
C     CONSIDERED, BECAUSE ONE OF THE DISTRIBUTIONS EQUALS ZERO AT
C     THESE VALUES). IF THE DOMAINS ARE DISJOINT OR OVERLAP AT ONLY.
C     ONE BOUNDARY POINT, THE RESULT OF THE APPLICATION OF THE
C     MAXIMUM OPERATOR IS JUST THE UNCHANGED APPROXIMATED PROBABILITY
C     DENSITY FUNCTION OF THE DISTRIBUTION DEFINED ON THE HIGHER-
C     VALUED DOMAIN.  GO TO 2180 OR GO TO 2160 RETURNS THIS FUNCTION
C     DIRECTLY WITHOUT FURTHER PROCESSING.
C
      IINT = 0
      LASTJ = NCL+1
      DO 2080 J = 1,LASTJ
      IF ((XINT(L1,L2,1) .GE. XINT(L1,L3,1)-0.001) .AND.
     &(XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001)) GO TO 2080
      IF (IINT .GE. 1) GO TO 2060
      IF (XINT(L1,L2,1) .LE. XINT(L1,L3,1)+0.001) GO TO 2050
      IF (XINT(L1,L3,J+1) .GE. XINT(L1,L2,1)-0.001) IINT = J
      IF ((XINT(L1,L3,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L2,1) .GE. XINT(L1,L3,J+1)-0.001)
     &.AND. (XINT(L1,L2,1) .LE. XINT(L1,L3,J+1)+0.001)
     &.AND. (XINT(L1,L3,J+2) .LE. 0.001))) GO TO 2180
      GO TO 2080
 2050 IF (XINT(L1,L2,J+1) .GE. XINT(L1,L3,1)-0.001) IINT = J
      IF ((XINT(L1,L2,J+1) .LE. 0.001)
     &.OR. ((XINT(L1,L3,1) .GE. XINT(L1,L2,J+1)-0.001)
     &.AND. (XINT(L1,L3,1) .LE. XINT(L1,L2,J+1)+0.001)
     &.AND. (XINT(L1,L2,J+2) .LE. 0.001))) GO TO 2160
      GO TO 2080
 2060 LASTK = NCL-(IINT-1)
      DO 2070 K = 1,LASTK
      B(K) = B(K+IINT)
      B(K+IINT) = 0
 2070 CONTINUE
      GO TO 2090
 2080 CONTINUE
 2090 NCL = NCL-IINT
C
C     DO 2150 IS THE OUTER LOOP FOR THE PROCESS OF CREATING THE
C     EQUIVALENT ARC.  NCL IS THE NUMBER OF CLASSES INVOLVED
C     BETWEEN THE TWO ARCS.
C
      N1 = 0
      N2 = 0
      DO 2150 I = 1,NCL
      DO 2110 J = 1,11
C
C     DO 2110 DETERMINES THE APPROPRIATE INTERVALS OF EACH DISTRIBUTION
C     THAT ARE VALID FOR THE B(I) VALUE BEING CONSIDERED.  N1 AND
C     N2 ARE THE CONTROLS FOR UPPER AND LOWER ARCS RESPECTIVELY.
C
      IF (N1 .GE. 1) GO TO 2100
      IF (((B(I) .GE. XINT(L1,L2,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L2,J+1)+0.001)) .OR. (XINT(L1,L2,J+1) .LE. 0.001))
     &N1 = J
 2100 CONTINUE
      IF (N2 .GE. 1) GO TO 2110
      IF (((B(I) .GE. XINT(L1,L3,J)-0.001) .AND. (B(I+1)
     &.LE. XINT(L1,L3,J+1)+0.001)) .OR. (XINT(L1,L3,J+1) .LE. 0.001))
     &N2 = J
 2110 CONTINUE
      IF (N2 .GT. NV2) K4(2,N2) = 1
      IF (N1 .GT. NV1) K4(1,N1) = 1
C
C     DO 2130 AND DO 2120 PERFORM THE POLYGONAL MULTIPLICATION FOR
C     CAP F(X) AND CAP G(X).
C
      LASTJ = K4(2,N2)
      LASTK = K4(1,N1)
      DO 2130 J = 1,LASTJ
      DO 2120 K = 1,LASTK
      IF (N2 .GT. NV2) PAR(2,N2,J) =  1
      IF (N1 .GT. NV1) PAR(1,N1,K) =  1
      K3 = J+K-1
      ZVAL(I,K3) = ZVAL(I,K3)+(PAR(1,N1,K)*PAR(2,N2,J))
 2120 CONTINUE
 2130 CONTINUE
C
C     DO 2140 OBTAINS THE FIRST DERIVATIVE OF THE RESULT OF THE
C     MULTIPLICATION OF CAP F(X) AND CAP G(X) IN THE FORM OF A
C     LITTLE H(X) FOR THAT PRODUCT.
C
      DO 2140 J = 1,4
      ZVAL(I,J) = ZVAL(I,J+1)*FLOAT(J)
      ZVAL(I,J+1) = 0
 2140 CONTINUE
      N1 = 0
      N2 = 0
 2150 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE RESULTS OF THE
C     PARALLEL REDUCTION WITH THE B(O) AND B(1) FORM IN EACH OF 10
C     CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR(L1,L2,NCL)
      GO TO 2180
 2160 DO 2170 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L3,I,1)
      VALUE(L1,L2,I,2) = VALUE(L1,L3,I,2)
      XINT(L1,L2,I) = XINT(L1,L3,I)
 2170 CONTINUE
      XINT(L1,L2,11) = XINT(L1,L3,11)
 2180 VALUE(L1,L2,1,3) = 0
      DO 2210 I = 1,2
      DO 2200 J = 1,10
      DO 2190 K = 1,3
      PAR(I,J,K) = 0
 2190 CONTINUE
 2200 CONTINUE
 2210 CONTINUE
      RETURN
      END
C     END SUBROUTINE PARA
C
C     *************************************************************
C
C                S U B R O U T I N E       S E R I E S
C
C     **************************************************************
C
      SUBROUTINE SERIES (L1,L2,L3,L4)
      REAL*8 VALUE(104,500,10,3),XINT(104,500,12)
      REAL*8 ZVAL(130,5),XLIM(2),A(130)
      REAL*8 F0,F1,G0,G1,XL
      INTEGER L1,L2,L3,L4
      INTEGER ISEL(2)
      INTEGER I,IK,J,K,NCL,NCL1,NE
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
C
C     SUBROUTINE SERIES PERFORMS THE CONVOLUTION OF TWO PROBABILITY
C     DISTRIBUTIONS BY INTEGRATING THE PRODUCT OF THEIR PIECEWISE
C     POLYGONAL APPROXIMATIONS IN THE FORMS OF F(X) AND G(T-X).
C
C     THIS SECTION DETERMINES THE INTERVALS OF VALIDITY FOR THE
C     CONVOLUTION.
C
C     THE A ARRAY IS USED FOR THE SAME PURPOSE AS THE B ARRAY IN PARA.
C
      K = 0
C
C     DO 3010 CREATES ALL POSSIBLE INTERVALS OF THE NEW DISTRIBUTION
C     BY ADDING THE INTERVALS OF THE TWO DISTRIBUTIONS BEING WORKED.
C
      DO 3010 I = 1,12
      IF ((XINT(L3,L4,I) .LE. 0).AND.(I .GT. 1)) GO TO 3020
      DO 3000 J = 1,12
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) NCL1 = J-2
      IF ((XINT(L1,L2,J) .LE. 0).AND.(J .GT. 1)) GO TO 3010
      K = K+1
      A(K) = XINT(L1,L2,J)+XINT(L3,L4,I)
 3000 CONTINUE
 3010 CONTINUE
 3020 NINT = I-2
      NCL = K-1
C
C     DO 3120 IS CONTROLLED BY THE NUMBER OF CLASSES IN THE F(X)
C     DISTRIBUTION.  DO 3110 IS CONTROLLED BY THE NUMBER OF CLASSES
C     CREATED BY COMBINING F(X) AND G(T-X).  DO 3100 IS CONTROLLED
C     BY THE NUMBER OF CLASSES IN THE G(T-X) DISTRIBUTION.  THIS
C     ALLOWS THE EVALUATION OF ALL OF THE CREATED CLASSES FOR EVERY
C     CLASS IN BOTH DISTRIBUTIONS.
C
      CALL SORT(NCL)
      DO 3120 K = 1,NCL1
      DO 3110 I = 1,NCL
      DO 3100 J = 1,NINT
      IK = 0
C
C     THIS IF STATEMENT DETERMINES WHICH INTERVALS ARE VALID FOR THE
C     INTERVAL END POINT A(I) BEING EVALUATED AND FOR THE VALUE OF K
C     BEING CONTROLLED BY DO 3120.
C
      IF ((A(I) .GE. XINT(L1,L2,K)+XINT(L3,L4,J)-0.001) .AND. (A(I+1)
     &.LE. XINT(L1,L2,K+1)+XINT(L3,L4,J+1)+0.001)) IK = J
      IF (IK .GE. 1) GO TO 3030
      GO TO 3100
 3030 ISEL(1) = 0
      ISEL(2) = 0
C
C     THE IF STATEMENTS INVOLVING XLIM ARE USED TO DETERMINE THE
C     UPPER AND LOWER LIMITS OF INTEGRATION.  IT IS DETERMINED WHETHER
C     THE LIMIT COMES FROM THE F(X) OR THE G(T-X) DISTRIBUTION.  ISEL
```

```
C     IS USED TO DESIGNATE VALUES FROM THE G(T-X) DISTRIBUTION.
C
      IF (XINT(L1,L2,K) .GE. (A(I+1)-XINT(L3,L4,J+1)-0.001)) GO TO 3040
      XLIM(1) = XINT(L3,L4,J+1)
      ISEL(1) = 999
      GO TO 3050
 3040 XLIM(1) = XINT(L1,L2,K)
 3050 IF (XINT(L1,L2,K+1) .LE. (A(I)-XINT(L3,L4,J)+0.001)) GO TO 3060
      XLIM(2) = XINT(L3,L4,J)
      ISEL(2) = 999
      GO TO 3070
 3060 XLIM(2) = XINT(L1,L2,K+1)
 3070 CONTINUE
      DO 3090 NE = 1,2
      F0 = VALUE(L1,L2,K,1)
      F1 = VALUE(L1,L2,K,2)
      G0 = VALUE(L3,L4,IK,1)
      G1 = VALUE(L3,L4,IK,2)
      XL = XLIM(NE)
      Z = 1.0
      IF (NE .EQ. 1) Z = -1.0
      IF (ISEL(NE) .EQ. 999) GO TO 3080
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL AT A FINITE
C     LIMIT.  THE INTEGRATION IS BROKEN DOWN INTO ITS COMPONENT PARTS
C     BY THE POWER OF THE COEFFICIENT THAT RESULTS.  Z CONTROLS THE
C     SIGN OF THE INTEGRAL BASED ON WHETHER THE LOWER OR UPPER LIMIT
C     IS BEING EVALUATED.
C
      ZVAL(I,1) = ZVAL(I,1)+((F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((-1.0*F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+(((F1*G1*XL**2)/2.)+(F0*G1*XL))*Z
      ZVAL(I,3) = ZVAL(I,3)+((-1.0*F0*G1)/2.)*Z
      GO TO 3090
C
C     THIS SECTION EVALUATES THE CONVOLUTION INTEGRAL FOR LIMITS.
C     IN THE FORM OF (T-X).  THE FORMULAS ARE DIFFERENT BECAUSE
C     OF THE DIFFERENT POLYNOMIAL CREATED WHEN THE INTEGRATION
C     INVOLVES LIMITS IN THE FORM OF (T-X).
C
 3080 ZVAL(I,1) = ZVAL(I,1)+((-1.0*F0*G0*XL)+((F1*G0*XL**2)/2.)
     &+((F1*G1*XL**3)/3.)+((-1.0*F0*G1*XL**2)/2.))*Z
      ZVAL(I,2) = ZVAL(I,2)+((-1.0*F1*G0*XL)+(F0*G0)-
     &((F1*G1*XL**2)/2.))*Z
      ZVAL(I,3) = ZVAL(I,3)+((F1*G0)/2.)*Z
      ZVAL(I,4) = ZVAL(I,4)+((F1*G1)/6.)*Z
 3090 CONTINUE
 3100 CONTINUE
 3110 CONTINUE
 3120 CONTINUE
C
C     LINEAR IS CALLED TO PIECEWISE POLYGONALIZE THE CONVOLUTION
C     RESULTS WITH THE B(0) AND B(1) FORM IN EACH OF 10 CLASSES.
C
      VALUE(L1,L2,1,3) = 99.
      CALL LINEAR(L1,L2,NCL)
      RETURN
      END
C     END SUBROUTINE SERIES
C
C     **************************************************************
C
C                S U B R O U T I N E        L I N E A R
C
C     **************************************************************
C
      SUBROUTINE LINEAR (L1,L2,NCL)
      REAL*8 VALUE(104,500,10,3),XINT(104,500,12),ZVAL(130,5),A(130)
      REAL*8 Q,Q1,Q2,STD,SUMX,SUMY,SUMXY,SUMSQ
      REAL*8 ALPHA,AREA,BETA,FACT,SIZE,W,X,XLMBDA,XMEAN
      REAL*8 XMODE,XSIZE,Y
      INTEGER L1,L2
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA2/ZVAL
      COMMON/PARA3/A
      EXTERNAL DGAMMA
C
C     SUBROUTINE LINEAR PIECEWISE POLYGONALIZES DISTRIBUTION DATA
C     FROM THE MAIN PROGRAM AND SUBROUTINES PARA AND SERIES WITH
C     THE B(O) AND B(1) FORM IN EACH OF 10 CLASSES THROUGH THE USE
C     OF SIMPLE LINEAR REGRESSION.
C
      XMODE = VALUE(L1,L2,2,3)
      XMEAN = VALUE(L1,L2,2,3)
      STD = ((VALUE(L1,L2,2,3)-XINT(L1,L2,1))/3.)
      XLMBDA = VALUE(L1,L2,2,3)-XINT(L1,L2,1)
      ALPHA = VALUE(L1,L2,2,3)
      BETA = VALUE(L1,L2,3,3)
      SIZE = (XINT(L1,L2,2)-XINT(L1,L2,1))/10.
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) SIZE = (A(NCL+1)-A(1))/10.
      XINT(L1,L2,11) = XINT(L1,L2,2)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) XINT(L1,L2,11) = A(NCL+1)
      X = XINT(L1,L2,1)
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 99) X = A(1)
      DO 5000 I = 1,10
      XINT(L1,L2,I) = X
      X = X+SIZE
 5000 CONTINUE
      DO 5050 I = 1,10
      X = XINT(L1,L2,I)
      SUMY = 0.
      SUMX = 0.
      SUMXY = 0.
      SUMSQ = 0.
C
C     W CONTROLS THE NUMBER OF DATA POINTS USED IN THE REGRESSION
C     COMPUTATIONS.
C
      W = 10.+IDINT(SIZE*3.)
      XSIZE = SIZE/W
      LASTJ = IDINT(W)
      DO 5040 J = 1,LASTJ
      IF (IDINT(VALUE(L1,L2,1,3)) .NE. 99)  GO TO 5030
      DO 5010 K3 = 1,NCL
      K  = 0
      IF ((X .GE. A(K3)).AND.(X .LE. A(K3+1))) K = K3
      IF (K .GE. 1) GO TO 5020
 5010 CONTINUE
C
C     SERIES OR PARA GENERATED DISTRIBUTIONS.
C
 5020 Y = ZVAL(K,1)+(ZVAL(K,2)*X)+(ZVAL(K,3)*(X**2))
     &+(ZVAL(K,4)*(X**3))
 5030 CONTINUE
C
C     TRIANGULAR DISTRIBUTION.
C
      IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 1) THEN
         IF (XINT(L1,L2,1) .LE. X .AND. X .LE. XMODE) THEN
         Y = (2.*(X-XINT(L1,L2,1)))/((XMODE-XINT(L1,L2,1))*10.*SIZE)
         ELSE
         Y = (2.*(XINT(L1,L2,11)-X))/((XINT(L1,L2,11)-XMODE)*10.*SIZE)
         END IF
C
C     NORMAL  DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 2) THEN
      Y = (1./(STD*2.506628275))*(DEXP((-1.0)*(((X-XMEAN)/STD)**2)/2.))
C
C     EXPONENTIAL DISTRIBUTION  (SHIFTED).
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 3) THEN
      Y = (1./XLMBDA)*(DEXP((-1.0)*((X-XINT(L1,L2,1))/XLMBDA)))
C
C     GAMMA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 4) THEN
      Y = (1./(DGAMMA(ALPHA)*(BETA**ALPHA)))*DEXP(-X/BETA)*(X**(ALPHA-1.
     &))
C
C     BETA DISTRIBUTION.
C
      ELSE IF (IDINT(VALUE(L1,L2,1,3)) .EQ. 5) THEN
      Y = (DGAMMA(ALPHA+BETA)/(DGAMMA(ALPHA)*DGAMMA(BETA)))*
     &(1./(10.*SIZE)**(ALPHA+BETA-2.))*
     &((X-XINT(L1,L2,1))**(ALPHA-1.))*
     &((XINT(L1,L2,11)-X)**(BETA-1.))
      END IF
      IF (Y .LT. 0) Y = 0
      SUMX = SUMX+X
      SUMY = SUMY+Y
      SUMXY = SUMXY+(X*Y)
      SUMSQ = SUMSQ+(X**2;
      X = X+XSIZE
 5040 CONTINUE
      VALUE(L1,L2,I,2) = (SUMXY-((SUMX*SUMY)/W))/(SUMSQ-((SUMX**2)/W))
      VALUE(L1,L2,I,1) = (SUMY/W)-(VALUE(L1,L2,I,2)*(SUMX/W))
 5050 CONTINUE
```

```
C
C     DO 5060 CALCULATES THE AREA UNDER THE APPROXIMATED DISTRIBUTION.
C     AN ADJUSTMENT FACTOR FOR THE AMOUNT THAT THIS AREA HAS BEEN
C     UNDERESTIMATED OR OVERESTIMATED IS THEREBY DETERMINED.
C
      DO 5060 I = 1,10
      Q  = XINT(L1,L2,I+1)-XINT(L1,L2,I)
      Q1 = (XINT(L1,L2,I)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      Q2 = (XINT(L1,L2,I+1)*VALUE(L1,L2,I,2))+VALUE(L1,L2,I,1)
      IF (Q1 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q1*(-1.0))
      IF (Q2 .LT. 0.) VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)+(Q2*(-1.0))
      IF (Q1 .LT. 0.) Q1 = 0.
      IF (Q2 .LT. 0.) Q2 = 0.
      AREA = AREA+((Q1+Q2)*Q*0.5)
 5060 CONTINUE
      FACT = 1.0/AREA
C
C     DO 5070 ADJUSTS THE COEFFICIENTS OF ALL THE LINEAR POLYNOMIAL
C     PIECES BY THE FACTOR COMPUTED IN DO 5060 IN ORDER TO NORMALIZE
C     THE AREA BACK TO ONE.  THIS ACTS TO REDUCE ACCUMULATING ERRORS
C     DURING PROGRAM COMPUTATIONS.
C
      DO 5070 I = 1,10
      VALUE(L1,L2,I,1) = VALUE(L1,L2,I,1)*FACT
      VALUE(L1,L2,I,2) = VALUE(L1,L2,I,2)*FACT
 5070 CONTINUE
      AREA = 0
      DO 5080  I = 1,130
      A(I)  = 0
      ZVAL(I,1) = 0
      ZVAL(I,2) = 0
      ZVAL(I,3) = 0
      ZVAL(I,4) = 0
 5080 CONTINUE
      RETURN
      END
C     END SUBROUTINE LINEAR
C
C     ************************************************************
C
C                SUBROUTINE            SORT
C
C     ************************************************************
C
      SUBROUTINE SORT (NCL)
      REAL*8 A(130),B
      INTEGER NCL
      INTEGER I,K1
      COMMON/PARA3/A
C
C     SUBROUTINE SORT IS USED TO CONDUCT AN ALGEBRAIC SORT OF DATA
C     CREATED IN THE SERIES AND PARA SUBROUTINES.
C
 6000 K1  = 0
      DO 6010 I = 1,NCL
      IF ((A(I) .LT. (A(I+1)+.01)).AND.(A(I) .GT. (A(I+1)-.01)))
```

```
     &GO TO 6020
      IF (A(I) .LT. A(I+1)) GO TO 6010
      IF (A(I) .GT. A(I+1)) B = A(I)
      A(I) = A(I+1)
      A(I+1) = B
      K1=K1+1
 6010 CONTINUE
      IF (K1 .GE. 1) GO TO 6000
      GO TO 6040
 6020 NCL = NCL-1
      LASTJ = NCL+1
      DO 6030 J = I,LASTJ
      A(J) = A(J+1)
      A(J+1) = 0
 6030 CONTINUE
      GO TO 6000
 6040 RETURN
      END
C     END SUBROUTINE SORT
C
C     ************************************************************
C
C                S U B R O U T I N E       S I M U L C
C
C     ************************************************************
C
      SUBROUTINE SIMULC (N,NSIM,NPATHS)
      REAL*8 XINT(104,500,12),VALUE(104,500,10,3),
     *       T(100,99),TEARLY(100),TLATE(100),
     *       CRTAS(100,99),CRTNAS(100,99),CRTNS(100),PTHCRT(6)
      REAL*8 ALPHA,BETA,X,XLNGTH,XLMBDA,XMAX,XMEAN,XMIN,XMODE,
     *       RN,STD,TDIFF,TFLOAT,TMAX,TMIN,TTOTAL,CRTNNS,CUMCRT
      DIMENSION NET(100,103),IPRE(100,99,2),
     *          NCA(100,99),NOCP(100),NOCS(100),LPATH(6,101)
      INTEGER I,IACT,ICOL,IENODE,IPRE,ISIM,ISNODE,ISTART,
     *        J,JJ,
     *        K,KOUNT,KTPATH,
     *        LASTP,LPATH,LSHIFT,
     *        N,NCA,NDIST,NET,NNPATH,NOCP,NOCS,NPATHS,NSIM
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA4/NET
      COMMON/PARA5/IPRE
      COMMON/PARA6/CRTAS,CRTNAS
      COMMON/PARA7/LPATH
      EXTERNAL DRNUN,DRNNOR,DRNEXP,DRNGAM,DRNBET,RNSET
C
C     SUBROUTINE SIMULC GENERATES NSIM MONTE CARLO SIMULATIONS OF THE
C     NETWORK, APPROXIMATES THE CRITICALITY INDICES OF THE ACTIVITIES,
C     AND IDENTIFIES THE TOP NPATHS CRITICAL PATHS.
C
C     DO 7005 INITIALIZES THE NCA ARRAY.  NCA(I,J) IS THE NUMBER OF
C     TIMES THE Jth PREDECESSOR OF NODE I APPEARS ON A CRITICAL PATH.
C
      DO 7005 I = 1,100
      DO 7000 J = 1,99
      NCA(I,J) = 0
 7000 CONTINUE
 7005 CONTINUE
C
C     DO 7180 GENERATES ACTIVITY RESOURCE CONSUMPTIONS (ACTIVITY TIMES)
C     FOR EACH OF THE NSIM ITERATIONS OF THE MONTE CARLO SIMULATION OF
C     NETWORK AND COUNTS THE NUMBER OF TIMES EACH ACTIVITY APPEARS ON A
C     CRITICAL PATH.
C
      DO 7180 ISIM = 1,NSIM
C
C     DO 7080 GENERATES A RANDOM VALUE FROM THE ACTIVITY RESOURCE
C     CONSUMPTION (ACTIVITY TIME) DISTRIBUTION OF EACH ACTIVITY.
C
      DO 7080 I = 1,N-1
      DO 7070 J = 1,NET(I,103)
      NDIST = IDINT(VALUE(I,J,1,3))
      XMIN = XINT(I,J,1)
      XMAX = XINT(I,J,11)
      XLNGTH = XMAX-XMIN
      GO TO (7010,7020,7030,7040,7050,7060) NDIST
C
C     TRIANGULAR DISTRIBUTION.
C
 7010 CALL DRNUN(1,RN)
      XMODE = VALUE(I,J,2,3)
      X = (XMODE-XMIN)/XLNGTH
      IF (RN .GT. X) GO TO 7015
      T(I,J) = XMIN+DSQRT(RN*XLNGTH*(XMODE-XMIN))
      GO TO 7070
 7015 T(I,J) = XMAX-DSQRT(XLNGTH*(XMAX-XMODE)*(1.-RN))
      GO TO 7070
C
C     NORMAL DISTRIBUTION.
C
 7020 CALL DRNNOR(1,RN)
      XMEAN = VALUE(I,J,2,3)
      STD = (XMEAN-XMIN)/3.
      T(I,J) = (RN*STD)+XMEAN
      IF ((T(I,J) .LT. XMIN) .OR. (T(I,J) .GT. XMAX)) GO TO 7020
      GO TO 7070
C
C     EXPONENTIAL DISTRIBUTION.
C
 7030 CALL DRNEXP(1,RN)
      XLMBDA = VALUE(I,J,2,3)-XMIN
      T(I,J) = (XLMBDA*RN)+XMIN
      IF (T(I,J) .GT. XMAX) GO TO 7030
      GO TO 7070
C
C     GAMMA DISTRIBUTION.
C
 7040 ALPHA = VALUE(I,J,2,3)
      BETA = VALUE(I,J,3,3)
      CALL DRNGAM(1,ALPHA,RN)
      T(I,J) = BETA*RN
      IF (T(I,J) .GT. XMAX) GO TO 7040
      GO TO 7070
C
C     BETA DISTRIBUTION.
C
 7050 ALPHA = VALUE(I,J,2,3)
      BETA = VALUE(I,J,3,3)
      CALL DRNBET(1,ALPHA,BETA,RN)
      T(I,J) = XMIN+(XLNGTH*RN)
      GO TO 7070
C
C     UNIFORM DISTRIBUTION.
C
 7060 CALL DRNUN(1,RN)
      T(I,J) = XMIN+(XLNGTH*RN)
 7070 CONTINUE
 7080 CONTINUE
C
C     DO 7100 COMPUTES THE EARLIEST COMPLETION TIME OF EACH NODE
(TEARLY).
C
      TEARLY(1) = 0.0
      DO 7110 I = 2,N
      TMAX = 0.0
      DO 7100 J = 1,NET(I,102)
      ISNODE = IPRE(I,J,1)
      IACT = IPRE(I,J,2)
      TTOTAL = TEARLY(ISNODE)+T(ISNODE,IACT)
      IF (TTOTAL .LE. TMAX) GO TO 7100
      TMAX = TTOTAL
 7100 CONTINUE
      TEARLY(I) = TMAX
 7110 CONTINUE
C
C     DO 7120 INITIALIZES NOCP(I), THE NUMBER OF CRITICAL NODES PRECED-
C     ING NODE I, AND NOCS(I), THE NUMBER OF CRITICAL NODES SUCCEEDING
C     NODE I.
C
      DO 7120 I = 1,N
      NOCP(I) = 0
      NOCS(I) = 0
 7120 CONTINUE
C
C     DO 7150 COMPUTES THE LATEST COMPLETION TIME OF EACH NODE (TLATE).
C
      TLATE(N) = TEARLY(N)
      NOCS(N) = 1
      DO 7150 I = N-1,1,-1
      TMIN = 10000.0D0
      DO 7140 J = 1,NET(I,103)
      IENODE = NET(I,J+1)
      TDIFF = TLATE(IENODE)-T(I,J)
      TFLOAT = TDIFF-TEARLY(I)
      IF (DABS(TFLOAT) .LE. 0.01D0) GO TO 7130
      IF (TDIFF .LT. TMIN) TMIN = TDIFF
      GO TO 7140
 7130 NOCS(I) = NOCS(I)+NOCS(IENODE)
```

```
      TMIN = TDIFF
 7140 CONTINUE
      TLATE(I) = TMIN
 7150 CONTINUE
C
C     DO 7180 DETERMINES THE NUMBER OF TIMES EACH ACTIVITY APPEARS ON A
C     CRITICAL PATH (NCA).
C
      NOCP(1) = 1
      DO 7170 I = 2,N
      DO 7160 J = 1,NET(I,102)
      ISNODE = IPRE(I,J,1)
      IACT = IPRE(I,J,2)
      TFLOAT = TLATE(I)-(TEARLY(ISNODE)+T(ISNODE,IACT))
      IF (DABS(TFLOAT) .GT. 0.01D0) GO TO 7160
      NCA(I,J) = NCA(I,J)+NOCS(I)*NOCP(ISNODE)
      NOCP(I) = NOCP(I)+NOCP(ISNODE)
 7160 CONTINUE
 7170 CONTINUE
 7180 CONTINUE
      PRINT 7910
      PRINT 7915
C
C     DO 7200 COMPUTES THE CRITICALITY INDICES OF THE PREDECESSOR
ACTIVI-
C     TIES OF EACH NODE I (I = 2,...,N) (CRTAS).
C
      DO 7200 I = 2,N
      DO 7190 J = 1,NET(I,102)
      CRTAS(I,J) = DBLE(NCA(I,J))/DBLE(NSIM)
 7190 CONTINUE
 7200 CONTINUE
C
C     DO 7230 COMPUTES NORMALIZED CRITICALITY INDICES OF THE PREDECESSOR
C     ACTIVITIES OF EACH NODE I (I = 2,...,N) (CRTNAS).
C
      CUMCRT = 0.0
      DO 7210 J = 1,NET(N,102)
      CUMCRT = CUMCRT+CRTAS(N,J)
 7210 CONTINUE
      DO 7230 I = N,2,-1
      PRINT 7920,I
      PRINT 7925
      DO 7220 J = 1,NET(I,102)
      CRTNAS(I,J) = CRTAS(I,J)/CUMCRT
      PRINT 7930,IPRE(I,J,1),NCA(I,J),CRTAS(I,J),CRTNAS(I,J)
 7220 CONTINUE
 7230 CONTINUE
C
C     DO 7260 COMPUTES THE CRITICALITY INDICES OF THE NODES (CRTNS).
C
      DO 7240 I = 1,N
      CRTNS(I) = 0.0
 7240 CONTINUE
      CRTNS(N) = CUMCRT
      DO 7260 I = 2,N
      DO 7250 J = 1,NET(I,102)
      ISNODE = IPRE(I,J,1)
      CRTNS(ISNODE) = CRTNS(ISNODE)+CRTAS(I,J)
 7250 CONTINUE
 7260 CONTINUE
C
C     DO 7270 COMPUTES NORMALIZED CRITICALITY INDICES OF THE NODES
C     (CRTNNS).
C
      PRINT 7935
      DO 7270 I = 1,N
      CRTNNS = CRTNS(I)/CUMCRT
      PRINT 7940,I,CRTNS(I),CRTNNS
 7270 CONTINUE
C
C     TO SUBROUTINE-END ENUMERATES ALL THE PATHS OF THE NETWORK WITH THE
C     METHOD OF ASANO AND SATO (1985) AND IDENTIFIES THE TOP NPATHS
C     CRITICAL PATHS.
C
      DO 7290 I = 1,6
      DO 7280 J = 1,101
      LPATH(I,J) = 0
 7280 CONTINUE
 7290 CONTINUE
C
C     IDENTIFY A FIRST PATH AND LOAD IT INTO THE PATH ARRAY (LPATH).
C     [STARTING AT THE SOURCE NODE 1, TRACE THE POINTER fs() TO THE
FIRST
C     SUCCESSOR OF EACH NODE, PERFORMING THE OPERATION v(i) = fs(v(i-1))
C     FOR i = 1,2,... UNTIL v(i) = TERMINAL NODE n.]
C
      LPATH(1,1) = 1
      KOUNT = 1
      DO 7300 J = 2,N
      ISNODE = LPATH(1,J-1)
      LPATH(1,J) = NET(ISNODE,2)
      KOUNT = KOUNT+1
      IF (LPATH(1,J) .EQ. N) GO TO 7310
 7300 CONTINUE
 7310 LPATH(1,101) = KOUNT
      NNPATH = 1
C
C     COMPUTE
C         CRITICALITY INDEX OF THE FIRST PATH (PTHCRT)
C       = AVERAGE OF THE CRITICALITY INDICES OF THE ACTIVITIES ON THE
PATH.
C
      PTHCRT(1) = 0.0
      DO 7340 J = 2,LPATH(1,101)
      ISNODE = LPATH(1,J-1)
      IENODE = LPATH(1,J)
      DO 7320 K = 1,NET(IENODE,102)
      IF (IPRE(IENODE,K,1) .EQ. ISNODE) THEN
      IACT = K
      GO TO 7330
      ELSE
      GO TO 7320
      END IF
 7320 CONTINUE
 7330 PTHCRT(1) = PTHCRT(1)+CRTAS(IENODE,IACT)
 7340 CONTINUE
      PTHCRT(1) = PTHCRT(1)/REAL(LPATH(1,101)-1)
C
C     IDENTIFY A NEW PATH AND LOAD IT INTO THE TEMPORARY LOCATION IN THE
C     PATH ARRAY (LPATH(6,-)).  [FIND THE LARGEST i SUCH THAT
C     next(v(i),v(i+1)) IS NOT 0.]
C
      LASTP = 1
 7350 DO 7390 J = LPATH(LASTP,101),2,-1
      ISNODE = LPATH(LASTP,J-1)
      IENODE = LPATH(LASTP,J)
      DO 7360 K = 2,NET(ISNODE,103)+1
      ICOL = K
      IF (NET(ISNODE,ICOL) .EQ. IENODE) GO TO 7370
 7360 CONTINUE
 7370 ICOL = ICOL+1
      IF (NET(ISNODE,ICOL) .NE. 0) THEN
      DO 7380 JJ = 1,J-1
      LPATH(6,JJ) = LPATH(LASTP,JJ)
 7380 CONTINUE
      LPATH(6,J) = NET(ISNODE,ICOL)
      KOUNT = J
      IF (LPATH(6,J) .EQ. N) GO TO 7420
      GO TO 7400
      ELSE
      GO TO 7390
      END IF
 7390 CONTINUE
C
C     [IF THERE EXISTS NO i SUCH THAT next(v(i),v(i+1)) IS NOT 0, THEN
ALL
C     THE POSSIBLE PATHS HAVE BEEN ENUMERATED.]
C
      RETURN
C
C     [OTHERWISE, SET v(i+1) = next(v(i),v(i+1)), AND THEN TRACE THE
POINT-
C     ER fs() FROM v(i+1) TO THE TERMINAL NODE n TO COMPLETE THE NEW
PATH.]
C
 7400 DO 7410 J = KOUNT+1,N
      ISNODE = LPATH(6,J-1)
      LPATH(6,J) = NET(ISNODE,2)
      KOUNT = KOUNT+1
      IF (LPATH(6,J) .EQ. N) GO TO 7420
 7410 CONTINUE
 7420 LPATH(6,101) = KOUNT
      NNPATH = NNPATH+1
      LASTP = 6
C
C     COMPUTE THE CRITICALITY INDEX OF THE NEW PATH (PTHCRT).
C
```

```
      PTHCRT(6) = 0.0
      DO 7450 J = 2,LPATH(6,101)
      ISNODE = LPATH(6,J-1)
      IENODE = LPATH(6,J)
      DO 7430 K = 1,NET(IENODE,102)
      IF (IPRE(IENODE,K,1) .EQ. ISNODE) THEN
      IACT = K
      GO TO 7440
      ELSE
      GO TO 7430
      END IF
 7430 CONTINUE
 7440 PTHCRT(6) = PTHCRT(6)+CRTAS(IENODE,IACT)
 7450 CONTINUE
      PTHCRT(6) = PTHCRT(6)/REAL(LPATH(6,101)-1)
C
C     INSERT THE NEW PATH INTO THE PATH ARRAY (LPATH) IN RANK ORDER OF
C     DESCENDING MAGNITUDE OF CRITICALITY INDICES.
C
      KTPATH = NNPATH-1
      IF (NNPATH .GT. NPATHS) KTPATH = NPATHS
      IF (PTHCRT(6) .LE. PTHCRT(KTPATH)) THEN
      LSHIFT = 0
      ELSE IF (PTHCRT(6) .GT. PTHCRT(1)) THEN
      LSHIFT = 1
      ELSE
      DO 7460 I = 1,KTPATH-1
      IF ((PTHCRT(I) .GE. PTHCRT(6)) .AND. (PTHCRT(6) .GT. PTHCRT(I+1)))
     *LSHIFT = I+1
 7460 CONTINUE
      END IF
C
C     IF THE NEW PATH'S CRITICALITY INDEX IS < OR = MINIMUM OF THE
CRITI-
C     CALITY INDICES OF THE PATHS IN THE PATH ARRAY AND THE PATH ARRAY
IS
C     FULL, DISCARD THE NEW PATH.
C
      IF ((LSHIFT .EQ. 0) .AND. (KTPATH .EQ. NPATHS)) GO TO 7350
C
C     IF THE NEW PATH'S CRITICALITY INDEX IS < OR = MINIMUM OF THE
CRITI-
C     CALITY INDICES OF THE PATHS IN THE PATH ARRAY AND THE PATH ARRAY
IS
C     NOT FULL, ADD THE NEW PATH TO THE BOTTOM OF THE PATH ARRAY.
C
      IF ((LSHIFT .EQ. 0) .AND. (KTPATH .LT. NPATHS)) GO TO 7470
      GO TO 7490
 7470 DO 7480 J = 1,LPATH(6,101)
      LPATH(KTPATH+1,J) = LPATH(6,J)
 7480 CONTINUE
      LPATH(KTPATH+1,101) = LPATH(6,101)
      PTHCRT(KTPATH+1) = PTHCRT(6)
      GO TO 7350
C
C     OTHERWISE, SHIFT THE PATH ARRAY DOWN ONE PATH BEGINNING WITH PATH
```

```
C     LSHIFT AND INSERT THE NEW PATH IN POSITION LSHIFT.
C
 7490 ISTART = MIN0(KTPATH+1,5)
      DO 7510 I = ISTART,LSHIFT+1,-1
      DO 7500 J = 1,LPATH(I-1,101)
      LPATH(I,J) = LPATH(I-1,J)
 7500 CONTINUE
      LPATH(I,101) = LPATH(I-1,101)
      PTHCRT(I) = PTHCRT(I-1)
 7510 CONTINUE
      DO 7520 J = 1,LPATH(6,101)
      LPATH(LSHIFT,J) = LPATH(6,J)
 7520 CONTINUE
      LPATH(LSHIFT,101) = LPATH(6,101)
      PTHCRT(LSHIFT) = PTHCRT(6)
      GO TO 7350
 7910 FORMAT (1H1)
 7915 FORMAT (1X,'FROM MONTE CARLO SIMULATION:')
 7920 FORMAT (/ 1X,'THE ACTIVITIES ENDING AT NODE ',I3,' AND THEIR ',
     &'CRITICALITY INDICES ARE:')
 7925 FORMAT (1X,10X,'NO. TIMES ON',15X,'NORMALIZED'/
     &        1X,'STARTING',3X,'A CRITICAL',3X,'CRITICALITY',2X,
     &           'CRITICALITY'/
     &        1X,2X,'NODE',8X,'PATH',9X,'INDEX',8X,'INDEX')
 7930 FORMAT (1X,2X,I3,8X,I6,7X,F7.5,6X,F7.5)
 7935 FORMAT (/ 1X,'THE CRITICALITY INDICES OF THE NODES ARE:'/
     &          1X,23X,'NORMALIZED'/
     &          1X,10X,'CRITICALITY',2X,'CRITICALITY'/
     &          1X,2X,'NODE',7X,'INDEX',8X,'INDEX')
 7940 FORMAT (1X,2X,I3,7X,F7.5,6X,F7.5)
      RETURN
      END
C     END SUBROUTINE SIMULC
C
C     ***************************************************************
C
C                S U B R O U T I N E      G E N R A N
C
C     ***************************************************************
C
      SUBROUTINE GENRAN (N,NACTS)
      DIMENSION NET(100,103),NAF(99),NBE(99)
      REAL*4 DEN,DL,DN,DN2,DN3,UP,X,Y,Y1
      INTEGER I,IJ,J,K,KK,L,N,NI,NO,NN,NACTS,NARCS,NDEL,NDIFF,NFREE,NEM,
     &        NRC
      COMMON/PARA4/NET
      EXTERNAL RNUN
C
C     THIS SUBROUTINE GENERATES A RANDOM ACYCLIC, DIRECTED ACTIVITY
C     NETWORK WITH N NODES AND NACTS ACTIVITIES WITH THE METHOD OF
C     DEMEULEMEESTER, DODIN AND HERROELEN (1993).
C
      DO 8010 I = 1,100
      DO 8000 J = 1,103
      NET(I,J) = 0
 8000 CONTINUE
```

```
      NET(I,101) = 1
 8010 CONTINUE
C
C     COMPUTE NUMBER OF ACTIVITIES TO DELETE WITH THE DELETION METHOD.
C
      L = N*(N-1)/2
      NDEL = L-NACTS
C
C     COMPUTE NDIFF SUCH THAT
C         INITIAL NUMBER OF FREE ACTIVITIES = NACTS - NDIFF
C     FOR THE ADDITION METHOD.
C
      NDIFF = (2*N)-4
      DN = REAL(N)
      DL = (REAL(L)/2.0)+1.0
      DN2 = DN*(DN-1.0)
      DN3 = DN+0.5
C
C     IF [N(N-1)/4]+1 < OR = NACTS, CHOOSE THE DELETION METHOD.
C     IF [N(N-1)/4]+1 > NACTS, CHOOSE THE ADDITION METHOD.
C
      IF (DL-NACTS) 8020,8020,8110
C
C     THE DELETION METHOD.
C
 8020 DO 8040 I = 1,N-1
      NET(I,1) = I
      DO 8030 J = I+1,N
      NET(I,J) = J
 8030 CONTINUE
      NET(I,102) = I-1
      NET(I,103) = N-I
 8040 CONTINUE
      NET(N,1) = N
      NET(N,102) = N-1
C
C     DO 8100 DELETES NDEL RANDOMLY SELECTED ACTIVITIES.
C
 8050 DO 8100 I = 1,NDEL
C
C     CHECK THAT THERE IS AT LEAST ONE ACTIVITY FEASIBLE FOR
C     ACTIVITY DELETION.  IF NOT, RESTART NETWORK GENERATION.
C
      DO 8055 J = 1,N-1
      NO = J
      IF (NET(NO,103) .LT. 2) GO TO 8055
      DO 8054 K = NO+1,N
      IF (NET(K,102) .GE. 2 .AND. NET(NO,K) .EQ. K) GO TO 8060
 8054 CONTINUE
 8055 CONTINUE
      GO TO 8000
C
C     RANDOMLY SELECT THE STARTING NODE (NO) OF THE ACTIVITY TO BE
C     DELETED FROM AMONG THE NODES FEASIBLE FOR ACTIVITY-DELETION.
C
 8060 CALL RNUN(1,Y)
```

```
      Y1 = (Y*DN2)+0.25
      X = DN3-SQRT(Y1)
      NO = INT(X)
      IF (NO .GT. X) NO = NO-1
      IF (NET(NO,103) .LT. 2) GO TO 8060
C
C     RANDOMLY SELECT THE ENDING NODE (NI) OF THE ACTIVITY TO BE
C     DELETED FROM AMONG THE NODES FEASIBLE FOR ACTIVITY-DELETION.
C
      K = 0
      DO 8070 J = NO+1,N
      IF (NET(J,102) .LT. 2) GO TO 8070
      IF (NET(NO,J) .EQ. 0) GO TO 8070
      K = K+1
      NAF(K) = J
 8070 CONTINUE
      IF (K .EQ. 0) GO TO 8060
      DEN = 1.0/REAL(K)
      CALL RNUN(1,X)
      DO 8080 J = 1,K
      UP = DEN*REAL(J)
      IF (X .GT. UP) GO TO 8080
      NI = NAF(J)
      GO TO 8090
 8080 CONTINUE
C
C     DELETE THE ACTIVITY FROM NODE NO TO NODE NI.
C
 8090 NET(NO,NI) = 0
      NET(NO,103) = NET(NO,103)-1
      NET(NI,102) = NET(NI,102)-1
 8100 CONTINUE
      GO TO 8250
C
C     THE ADDITION METHOD.
C
 8110 DO 8120 I = 1,N
      NET(I,1) = I
 8120 CONTINUE
C
C     INITIALIZE NUMBER OF NONRECEIVING NODES (NRC) AND NUMBER OF
C     NONEMITTING NODES (NEM).
C
      NRC = N-3
      NEM = N-3
C
C     ADD ACTIVITIES FROM NODE 1 TO NODE 2 AND FROM NODE N-1 TO NODE N.
C
      NET(1,2) = 2
      NET(N-1,N) = N
      NET(1,103) = 1
      NET(2,102) = 1
      NET(N-1,103) = 1
      NET(N,102) = 1
C
C     INITIALIZE NUMBER OF ACTIVITIES ADDED SO FAR (NARCS).
```

```
C
      NARCS = 2
C
C     IF INITIAL NUMBER OF FREE ACTIVITIES
C         [NACTS - (2N-4) = NACTS - NDIFF] IS < OR = 0,
C     THEN ALL NACTS ACTIVITIES TO BE ADDED ARE SUBJECT TO FEASIBILITY
C     CONDITIONS AND CANNOT BE RANDOMLY SELECTED.
C
      IF (NDIFF .GE. NACTS) GO TO 8170
C
C     SET FLAG (KK = 0) THAT INITIAL NUMBER OF FREE ACTIVITIES IS > 0.
C
      KK = 0
C
C     RANDOMLY SELECT THE START NODE (NO) OF THE ACTIVITY TO BE ADDED
C     FROM AMONG THE FEASIBLE NODES.
C
 8130 CALL RNUN(1,Y)
      Y1 = (Y*DN2)+0.25
      X = DN3-SQRT(Y1)
      NO = INT(X)
      IF (NO .GT. X) NO = NO-1
      NN = N-NO
      IF (NET(NO,103) .GE. NN) GO TO 8130
C
C     RANDOMLY SELECT THE END NODE (NI) OF THE ACTIVITY TO BE ADDED
C     FROM AMONG THE FEASIBLE NODES.
C
      K = 0
      DO 8140 J = NO+1,N
      IF (NET(NO,J) .NE. 0) GO TO 8140
      K = K+1
      NAF(K) = J
 8140 CONTINUE
      DEN = 1.0/REAL(K)
      CALL RNUN(1,X)
      UP = 0.0
      DO 8150 J = 1,K
      UP = UP+DEN
      IF (X .GT. UP) GO TO 8150
      NI = NAF(J)
      GO TO 8160
 8150 CONTINUE
C
C     ADD THE ACTIVITY FROM NODE NO TO NODE NI.
C
 8160 NET(NO,NI) = NI
      NARCS = NARCS+1
      IF (NET(NO,103) .EQ. 0) NEM = NEM-1
      IF (NET(NI,102) .EQ. 0) NRC = NRC-1
      NET(NO,103) = NET(NO,103)+1
      NET(NI,102) = NET(NI,102)+1
C
C     IF
C         NUMBER OF ACTIVITIES ADDED SO FAR (NARCS) IS > OR =
C         NUMBER OF ACTIVITIES REQUIRED (NACTS),
C     THEN THE NETWORK IS COMPLETE.
C
      IF (NARCS .GE. NACTS) GO TO 8250
C
C     IF THE FLAG (KK) INDICATES THAT
C         NUMBER OF NONRECEIVING NODES (NRC) = 0, AND
C         NUMBER OF NONEMITTING NODES (NEM) = 0, I.E.
C     FEASIBILITY REQUIREMENTS ARE MET, THEN THE REMAINING ACTIVITIES
C     TO BE ADDED ARE FREE ACTIVITIES AND ARE TO BE RANDOMLY SELECTED.
C
      IF (KK .EQ. 1) GO TO 8130
C
C     IF
C         NUMBER OF FREE ACTIVITIES (NFREE) IS > 0,
C     THEN THE NEXT ACTIVITY TO BE ADDED IS A FREE ACTIVITY AND IS TO BE
C     RANDOMLY SELECTED.
C
      NFREE = NACTS-NARCS-NRC-NEM
      IF (NFREE .GT. 0) GO TO 8130
C
C     IF
C         NUMBER OF FREE ACTIVITIES (NFREE) = 0, AND
C         NUMBER OF NONRECEIVING NODES (NRC) = 0,
C     THEN CHECK THE NUMBER OF NONEMITTING NODES (NEM).
C
 8170 IF (NRC .EQ. 0) GO TO 8200
C
C     IF NOT, ADD ACTIVITIES SO AS TO REDUCE THE NUMBER OF NONRECEIVING
C     NODES (NRC) TO 0.
C
      K = 0
      DO 8180 I = 3,N-1
      IF (NET(I,102) .GT. 0) GO TO 8180
      K = K+1
      NAF(K) = I
 8180 CONTINUE
      IF (K .EQ. 0) GO TO 8200
      DO 8190 I =1,K
      IJ = K+1-I
      NI = NAF(IJ)
      CALL RNUN(1,Y)
      X = 1.0+(REAL(NI-1)*Y)
      NO = INT(X)
      IF (NO .GT. X) NO = NO-1
      NET(NO,NI) = NI
      NARCS = NARCS+1
      NET(NO,103) = NET(NO,103)+1
      NET(NI,102) = NET(NI,102)+1
 8190 CONTINUE
C
C     IF
C         NUMBER OF NONEMITTING NODES (NEM) = 0,
C     THEN THE FEASIBILITY REQUIREMENTS ARE MET.
C
 8200 IF (NEM .EQ. 0) GO TO 8230
C
```

```
C     IF NOT, ADD ACTIVITIES SO AS TO REDUCE THE NUMBER OF NONEMITTING
C     NODES (NEM) TO 0.
C
      K = 0
      DO 8210 I = 2,N-2
      IF (NET(I,103) .GT. 0) GO TO 8210
      K = K+1
      NBE(K) = I
 8210 CONTINUE
      IF (K .EQ. 0) GO TO 8230
      DO 8220 I = 1,K
      NO = NBE(I)
      CALL RNUN(1,X)
      Y = REAL(NO+1)+(REAL(N-NO)*X)
      NI = INT(Y)
      IF (NI .GT. Y) NI = NI-1
      NET(NO,NI) = NI
      NARCS = NARCS+1
      NET(NO,103) = NET(NO,103)+1
      NET(NI,102) = NET(NI,102)+1
 8220 CONTINUE
C
C     SET FLAG (KK = 1) THAT FEASIBILITY REQUIREMENTS HAVE BEEN MET.
C
 8230 KK = 1
C
C     IF NUMBER OF ACTIVITIES ADDED SO FAR (NARCS) IS
C         < NUMBER OF ACTIVITIES REQUIRED (NACTS), THEN RANDOMLY SELECT
C           THE NEXT ACTIVITY TO BE ADDED,
C         = NACTS, THEN THE NETWORK IS COMPLETE,
C         > NACTS, THEN USE THE DELETION METHOD TO DELETE EXCESS
C           ACTIVITIES.
C
      IF (NARCS-NACTS) 8130,8250,8240
 8240 NDEL = NARCS-NACTS
      GO TO 8050
C
C     RECONFIGURE NET ARRAY.
C
 8250 DO 8270 I = 1,N-1
      K = 2
      DO 8260 J = I+1,N
      IF (NET(I,J) .EQ. 0) GO TO 8260
      NET(I,K) = NET(I,J)
      IF (K .LT. J) NET(I,J) = 0
      K = K+1
 8260 CONTINUE
 8270 CONTINUE
      RETURN
      END
C     END SUBROUTINE GENRAN
C
C     *************************************************************
C
C                 S U B R O U T I N E       C O M P A R
C
```

```
C     *************************************************************
C
      SUBROUTINE COMPAR(L1,L2,L3,L4,PR1GE2)
      REAL*8 XINT(104,500,12),VALUE(104,500,10,3),
     *       B0X,B1X,B0Y,B1Y,C1,C2,C3,C4,CELL,PR1GE2,VOLUME
     *       XLOWER,XUPPER,YLOWER,YUPPER
      INTEGER I,J,L1,L2,L3,L4
      COMMON/PARA1/XINT,VALUE
C
C     SUBROUTINE COMPAR COMPUTES P(X > OR = Y), WHERE X IS THE DISTRIBU-
C     TION OF NODE L1, ACTIVITY L2, AND Y IS THE DISTRIBUTION OF NODE
L3,
C     ACTIVITY L4.
C
C     IF Y(11) < OR = X(1), THEN P(X > OR = Y) = 1.  RETURN PR1GE2 =
1.0.
C
      IF (XINT(L3,L4,11) .LE. XINT(L1,L2,1)) THEN
      PR1GE2 = 1.0
      RETURN
C
C     IF X(11) < OR = Y(1), THEN P(X > OR = Y) = 0.  RETURN PRIGE2 =
0.0.
C
      ELSE IF (XINT(L1,L2,11) .LE. XINT(L3,L4,1)) THEN
      PR1GE2 = 0.0
      RETURN
      END IF
C
C     INITIALIZE P(X > OR = Y) [PR1GE2].
C
      PR1GE2 = 0.0
C
C     DO 9010 COMPUTES THE CONTRIBUTION TO P(X > OR = Y) IN EACH CELL
C     [X(I),X(I+1)] x [Y(J),Y(J+1)] OF THE JOINT DISTRIBUTION OF X AND Y
C     AND SUMS THESE CONTRIBUTIONS IN PR1GE2.
C
      DO 9010 I = 1,10
      DO 9000 J = 1,10
C
C     IF X(I+1) < OR = Y(J), THE CELL LIES COMPLETELY ABOVE THE LINE X =
Y,
C     SO THE CELL'S CONTRIBUTION TO P(X > OR = Y) IS 0.
C
      IF (XINT(L1,L2,I+1) .LE. XINT(L3,L4,J)) GO TO 9000
      B0X = VALUE(L1,L2,I,1)
      B1X = VALUE(L1,L2,I,2)
      B0Y = VALUE(L3,L4,J,1)
      B1Y = VALUE(L3,L4,J,2)
      XLOWER = XINT(L1,L2,I)
      XUPPER = XINT(L1,L2,I+1)
      YLOWER = XINT(L3,L4,J)
      YUPPER = XINT(L3,L4,J+1)
C
C     IF Y(J+1) < OR = X(I), THE CELL LIES COMPLETELY BELOW THE LINE X =
Y.
```

```
C
      IF (YUPPER .LE. XLOWER) THEN
      CELL =  (B0X*(XUPPER-XLOWER)+(B1X/2.0)*(XUPPER**2-XLOWER**2))
     *        *(B0Y*(YUPPER-YLOWER)+(B1Y/2.0)*(YUPPER**2-YLOWER**2))
      PR1GE2 = PR1GE2+CELL
      GO TO 9000
      ELSE
      C1 = -(B0X*(B0Y*YLOWER+(B1Y/2.0)*YLOWER**2))
      C2 = (B0X*B0Y/2.0)-(B1X/2.0)*(B0Y*YLOWER+(B1Y/2.0)*YLOWER**2)
      C3 = (B1X*B0Y/3.0)+(B0X*B1Y/6.0)
      C4 = B1X*B1Y/8.0
      END IF
C
C     IF Y(J) < X(I) < Y(J+1) < X(I+1), THE LINE X = Y PASSES THROUGH
THE
C     LEFT SIDE AND THE TOP OF THE CELL.
C
      IF ((YLOWER .LT. XLOWER) .AND. (XLOWER. LT. YUPPER) .AND.
     *    (YUPPER .LT. XUPPER)) THEN
      CELL =  C1*(YUPPER-XLOWER)
     *        +C2*(YUPPER**2-XLOWER**2)
     *        +C3*(YUPPER**3-XLOWER**3)
     *        +C4*(YUPPER**4-XLOWER**4)
     *        +(B0X*(XUPPER-YUPPER)+(B1X/2.0)*(XUPPER**2-YUPPER**2))
     *        *(B0Y*(YUPPER-YLOWER)+(B1Y/2.0)*(YUPPER**2-YLOWER**2))
      PR1GE2 = PR1GE2+CELL
      GO TO 9000
C
C     IF X(I) < OR = Y(J) AND Y(J+1) < X(I+1),THE LINE X = Y PASSES
C     THROUGH THE BOTTOM AND THE TOP OF THE CELL.
C
      ELSE IF ((XLOWER .LE. YLOWER) .AND. (YUPPER .LT. XUPPER)) THEN
      CELL =  C1*(YUPPER-YLOWER)
     *        +C2*(YUPPER**2-YLOWER**2)
     *        +C3*(YUPPER**3-YLOWER**3)
     *        +C4*(YUPPER**4-YLOWER**4)
     *        +(B0X*(XUPPER-YUPPER)+(B1X/2.0)*(XUPPER**2-YUPPER**2))
     *        *(B0Y*(YUPPER-YLOWER)+(B1Y/2.0)*(YUPPER**2-YLOWER**2))
      PR1GE2 = PR1GE2+CELL
      GO TO 9000
C
C     IF X(I) < OR = Y(J) < X(I+1) < OR = Y(J+1), THE LINE X = Y PASSES
C     THROUGH THE BOTTOM AND THE RIGHT SIDE OF THE CELL OR THROUGH THE
C     LOWER-LEFT AND UPPER-RIGHT CORNERS OF THE CELL.
C
      ELSE IF ((XLOWER .LE. YLOWER) .AND. (YLOWER .LT. XUPPER) .AND.
     *        (XUPPER .LE. YUPPER)) THEN
      CELL =  C1*(XUPPER-YLOWER)
     *        +C2*(XUPPER**2-YLOWER**2)
     *        +C3*(XUPPER**3-YLOWER**3)
     *        +C4*(XUPPER**4-YLOWER**4)
      PR1GE2 = PR1GE2+CELL
      GO TO 9000
C
C     IF Y(J) < X(I) AND X(I+1) < Y(J+1), THE LINE X = Y PASSES THROUGH
C     BOTH SIDES OF THE CELL.
```

```
C
      ELSE IF ((YLOWER .LT. XLOWER) .AND. (XUPPER .LT. YUPPER)) THEN
      CELL =  C1*(XUPPER-XLOWER)
     *       +C2*(XUPPER**2-XLOWER**2)
     *       +C3*(XUPPER**3-XLOWER**3)
     *       +C4*(XUPPER**4-XLOWER**4)
      PR1GE2 = PR1GE2+CELL
      GO TO 9000
      END IF
 9000 CONTINUE
 9010 CONTINUE
C
C     DO 9030 COMPUTES THE VOLUME UNDER THE APPROXIMATED JOINT DISTRIBU-
C     TION OF X AND Y.  AN ADJUSTMENT FOR THE AMOUNT THAT THIS VOLUME
C     HAS BEEN UNDERESTIMATED OR OVERESTIMATED IS THEN MADE TO PR1GE2.
C
      VOLUME = 0.0
      DO 9030 I = 1,10
      DO 9020 J = 1,10
      B0X = VALUE(L1,L2,I,1)
      B1X = VALUE(L1,L2,I,2)
      B0Y = VALUE(L3,L4,J,1)
      B1Y = VALUE(L3,L4,J,2)
      XLOWER = XINT(L1,L2,I)
      XUPPER = XINT(L1,L2,I+1)
      YLOWER = XINT(L3,L4,J)
      YUPPER = XINT(L3,L4,J+1)
      CELL =  (B0X*(XUPPER-XLOWER)+(B1X/2.0)*(XUPPER**2-XLOWER**2))
     *       *(B0Y*(YUPPER-YLOWER)+(B1Y/2.0)*(YUPPER**2-YLOWER**2))
      VOLUME = VOLUME+CELL
 9020 CONTINUE
 9030 CONTINUE
      PR1GE2 = PR1GE2/VOLUME
      RETURN
      END
C     END SUBROUTINE COMPAR
C
C     ************************************************************
C
C                S U B R O U T I N E       D O M P T H
C
C     ************************************************************
C
      SUBROUTINE DOMPTH(N,NPATHS)
      REAL*8 XINT(104,500,12),VALUE(104,500,10,3),
     *       PR1GE2
      DIMENSION NET(100,103),IPRE(100,99,2),NPA(500,101),
     *          NPPA(100,5),INP(500),NP(5),NPP(100),NPK1(500),
     *          NPK2(500),NPR1(5)
      INTEGER I,IACT,INP,ISLAST,ISNODE,ISNOD1,ISNOD2,
     *        J,JJ,J1,J2,J3,
     *        K,KK,
     *        N,NET,NNN,NOPAT,NP,NPA,NPATHS,NPK,NPK1,NPK2,NPP,NPPA,
     *        NPR1,NSS
      COMMON/PARA1/XINT,VALUE
      COMMON/PARA4/NET
      COMMON/PARA5/IPRE
      COMMON/PARA8/NPA,NPPA
C
C     SUBROUTINE DOMPTH DETERMINES THE K MOST STOCHASTICALLY DOMINATING
C     PATHS IN THE NETWORK.  THE FOLLOWING ARRAYS AND VARIABLES ARE USED
C     IN THIS SUBROUTINE:
C       NOPAT    : NUMBER OF PATHS IN THE MAIN PATH LIST
C       NPATHS   : DESIRED NUMBER OF PATHS IN THE SET OF K MOST
C                    STOCHASTICALLY DOMINATING PATHS (MAXIMUM 5)
C       NPK      : NUMBER OF CANDIDATE PATHS THROUGH NODE I
C       NP(K)    : RANK IN THE MAIN PATH LIST OF THE Kth MOST
C                    STOCHASTICALLY DOMINATING PATH ENDING AT NODE I
C       NPA(I,J) : NODES FORMING THE Ith RANK PATH IN THE MAIN PATH LIST
C       NPP(I)   : NUMBER OF DOMINATING PATHS ENDING AT NODE I
C       NPPA(I,J): RANK IN THE MAIN PATH LIST OF THE Jth MOST DOMINATING
C                    PATH ENDING AT NODE I
C       NPK1(K)  : RANK IN THE MAIN PATH LIST OF THE Kth CANDIDATE PATH
C                    ENDING AT NODE I
C       NPK2(K)  : NUMBER OF THE PREDECESSOR NODE TO NODE I OF THE Kth
C                    CANDIDATE PATH ENDING AT NODE I
C       NPR1(K)  : RANK OF THE Kth MOST DOMINATING PATH AMONG THE
CANDIDATE
C                    PATHS ENDING AT NODE I
C       INP(J)   : INDICATOR OF PATH J IN THE MAIN PATH LIST
C                    = 1 IF PATH J IS AMONG THE K MOST DOMINATING PATHS
AT
C                        A PREDECESSOR NODE TO NODE I
C                    = 0 WHEN PATH J IS DETERMINED TO BE ONE OF THE K
C                        MOST DOMINATING PATHS AND IS REMOVED FROM
FURTHER
C                        CONSIDERATION AT NODE I
C     THE DISTRIBUTIONS OF THE K MOST STOCHASTICALLY DOMINATING PATHS
THROUGH
C     EACH NODE IN THE MAIN PATH LIST ARE IN XINT(104,1 TO 500,-)
TOGETHER
C     WITH VALUE(104,1 TO 500,-,-).  THE DISTRIBUTIONS OF THE CANDIDATE
C     PATHS AT NODE I ARE IN XINT(103,1 TO 500,-), VALUE(103,1 TO 500,-
,-).
C
      DO 9110 I = 1,500
      DO 9100 J = 1,10
      XINT(103,I,J) = 0.0
      XINT(104,I,J) = 0.0
      VALUE(103,I,J,1) = 0.0
      VALUE(103,I,J,2) = 0.0
      VALUE(104,I,J,1) = 0.0
      VALUE(104,I,J,2) = 0.0
 9100 CONTINUE
      XINT(103,I,11) = 0.0
      XINT(104,I,11) = 0.0
 9110 CONTINUE
C
C     INITIALIZE THE MAIN PATH LIST AT THE STARTING NODE.
C
      NOPAT = 1
      NPP(1) = 1
      NPPA(1,1) = 1
      INP(1) = 1
      NPA(1,1) = 1
      NPA(1,2) = 1
C
C     DO 9250 DETERMINES THE K MOST STOCHASTICALLY DOMINATING PATHS
THROUGH
C     EACH NODE.
C
      DO 9250 I = 2,N
      NNN = NPATHS
      NPK = 0
C
C     DO 9160 DETERMINES THE DISTRIBUTIONS OF THE (NO. OF PREDECESSORS
OF
C     NODE I)*NPATHS CANDIDATE PATHS AT NODE I.
C
      DO 9160 J = 1,NET(I,102)
      ISNODE = IPRE(I,J,1)
      IACT = IPRE(I,J,2)
C
C     LOAD THE DISTRIBUTION OF THE ACTIVITY FROM NODE ISNODE TO NODE I
C     INTO TEMPORARY LOCATION 1.
C
      DO 9120 KK = 1,10
      XINT(101,1,KK) = XINT(ISNODE,IACT,KK)
      VALUE(101,1,KK,1) = VALUE(ISNODE,IACT,KK,1)
      VALUE(101,1,KK,2) = VALUE(ISNODE,IACT,KK,2)
 9120 CONTINUE
      XINT(101,1,11) = XINT(ISNODE,IACT,11)
      DO 9150 K = 1,NPP(ISNODE)
      NSS = NPPA(ISNODE,K)
      NPK = NPK+1
C
C     LOAD THE DISTRIBUTION OF THE RANK NSS PATH THROUGH NODE ISNODE IN
THE
C     MAIN PATH LIST INTO THE DISTRIBUTION OF THE RANK NPK PATH THROUGH
C     NODE I IN THE CANDIDATE PATH LIST.
C
      DO 9130 KK = 1,10
      XINT(103,NPK,KK) = XINT(104,NSS,KK)
      VALUE(103,NPK,KK,1) = VALUE(104,NSS,KK,1)
      VALUE(103,NPK,KK,2) = VALUE(104,NSS,KK,2)
 9130 CONTINUE
      XINT(103,NPK,11) = XINT(104,NSS,11)
C
C     CONVOLVE THE DISTRIBUTION OF THE RANK NSS PATH THROUGH NODE ISNODE
C     IN THE MAIN PATH LIST AND ACTIVITY IACT AND PLACE THE CONVOLUTION
IN
C     THE DISTRIBUTION OF THE RANK NPK PATH THROUGH NODE I IN THE MAIN
C     PATH LIST.
C
      IF (ISNODE .EQ. 1) THEN
      DO 9140 KK = 1,10
      XINT(103,NPK,KK) = XINT(101,1,KK)
      VALUE(103,NPK,KK,1) = VALUE(101,1,KK,1)
      VALUE(103,NPK,KK,2) = VALUE(101,1,KK,2)
 9140 CONTINUE
      XINT(103,NPK,11) = XINT(101,1,11)
      ELSE
      CALL SERIES(103,NPK,101,1)
      END IF
      NPK1(NPK) = NSS
      NPK2(NPK) = ISNODE
 9150 CONTINUE
 9160 CONTINUE
      IF (NPK .LT. NPATHS) NNN = NPK
C
C     DO 9210 DETERMINES THE K MOST STOCHASTICALLY DOMINATING PATHS
THROUGH
C     NODE I FROM AMONG THE NPK CANDIDATE PATHS.
C
      DO 9210 K = 1,NNN
      ISNOD2 = 0
      J = 1
C
C     THE Jth CANDIDATE PATH IS DESIGNATED THE CONTENDER FOR THE Kth
MOST
C     STOCHASTICALLY DOMINATING PATH THROUGH NODE I.
C
 9170 J2 = J
      J1 = NPK1(J)

C
C     IF THE Jth CANDIDATE PATH IS ONE OF THE K MOST STOCHASTICALLY
DOMI-
C     NATING PATHS THROUGH NODE I WHICH HAVE ALREADY BEEN DETERMINED
C     [INP(NPK1(J)) = 0], IT IS NOT FURTHER CONSIDERED.
C
      IF (INP(J1).EQ. 0) GO TO 9190
      NP(K) = J1
      NPR1(K) = J
      ISNOD1 = NPK2(J)
C
C     THE J2th CANDIDATE PATH IS TESTED AGAINST THE Jth CANDIDATE PATH.
C
 9180 J2 = J2+1
C
C     IF J2 > NPK, THE Jth CANDIDATE PATH IS THE Kth MOST STOCHASTICALLY
C     DOMINATING PATH THROUGH NODE I.
C
      IF (J2 .GT. NPK) GO TO 9200
      J3 = NPK1(J2)
C
C     IF THE J2th CANDIDATE PATH IS ONE OF THE K MOST STOCHASTICALLY
DOMI-
C     NATING PATHS THROUGH NODE I WHICH HAVE ALREADY BEEN DETERMINED
C     [INP(NPK1(J2)) = 0], IT IS NOT FURTHER CONSIDERED.
C
      IF (INP(J3) .EQ. 0) GO TO 9180
      ISLAST = ISNOD2
      ISNOD2 = NPK2(J2)
```

```
C
C     IF THE Jth AND J2th CANDIDATE PATHS HAVE THE SAME PREDECESSOR NODE
TO
C     NODE I, THE J2th CANDIDATE PATH IS NOT FURTHER CONSIDERED, SINCE
THE
C     K MOST STOCHASTICALLY DOMINATING PATHS THROUGH THAT PREDECESSOR
NODE
C     ARE RANK-ORDERED, AND THE Jth PATH STOCHASTICALLY DOMINATES THE
J2th
C     PATH.  IF THE J2th AND THE (J2-1)th CANDIDATE PATHS HAVE THE SAME
PRE-
C     DECESSOR NODE TO NODE I, THE J2th CANDIDATE PATH IS AGAIN NOT
FURTHER
C     CONSIDERED, SINCE THE K MOST STOCHASTICALLY DOMINATING PATHS
THROUGH
C     THAT PREDECESSOR NODE ARE RANK-ORDERED, AND THE (J2-1)th PATH
STOCHAS-
C     TICALLY DOMINATES THE J2th PATH, AND HENCE THE Jth PATH, WHICH
STO-
C     CASTICALLY DOMINATES THE (J2-1)th PATH, STOCHASTICALLY DOMINATES
THE
C     J2th PATH BY TRANSITIVITY.
C
      IF ((ISNOD2 .EQ. ISNOD1) .OR. (ISNOD2 .EQ. ISLAST)) GO TO 9180
C
C     COMPUTE THE PROBABILITY THAT THE Jth CANDIDATE PATH STOCHASTICALLY
C     DOMINATES (DOMINATES IN PROBABILITY) THE J2th CANDIDATE PATH, I.E.
C     P(DISTRIBUTION OF Jth PATH DURATION > OR = DISTRIBUTION OF J2th
PATH
C     DURATION).
C
      CALL COMPAR(103,J,103,J2,PR1GE2)
C
C     IF THE Jth CANDIDATE PATH STOCHASTICALLY DOMINATES THE J2th
CANDIDATE
C     PATH, THE Jth CANDIDATE PATH REMAINS THE CONTENDER FOR THE Kth
MOST
C     STOCHASTICALLY DOMINATING PATH THROUGH NODE I, AND THE (J2+1)th
CANDI-
C     DATE PATH IS TESTED NEXT.
C
      IF (PR1GE2 .GE. 0.5) GO TO 9180
C
C     IF THE J2th CANDIDATE PATH STOCHASTICALLY DOMINATES THE Jth
CANDIDATE
C     PATH, THE J2th CANDIDATE PATH IS DESIGNATED THE CONTENDER FOR THE
Kth
C     MOST STOCHASTICALLY DOMINATING PATH THROUGH NODE I, AND THE
(J2+1)th
C     CANDIDATE IS TESTED NEXT AGAINST THE CONTENDER.
C
      NP(K) = J3
      NPR1(K) = J2
      J = J2
      ISNOD1 = ISNOD2
      GO TO 9180
```

```
 9190 J = J+1
      IF (J .LE. NPK) GO TO 9170
C
C     WHEN THE Kth MOST STOCHASTICALLY DOMINATING PATH HAS BEEN
DETERMINED,
C     SET THE INDICATOR OF ITS PATH NUMBER IN THE MAIN PATH LIST = 0, SO
C     THAT THE PATH IS NOT FURTHER CONSIDERED AT NODE I.
C
 9200 INP(NP(K)) = 0
 9210 CONTINUE
C
C     DO 9240 UPDATES THE MAIN PATH LIST AND PATH PARAMETERS.
C
      NPP(I) = NNN
      DO 9240 K = 1,NNN
C
C     THE Jth PATH IN THE MAIN PATH LIST WAS THE Kth MOST STOCHASTICALLY
C     DOMINATING PATH ENDING AT NODE I.  RESET THE INDICATOR OF THIS
PATH = 1.
C
      J = NP(K)
      INP(J) = 1
C
C     THE Kth MOST STOCHASTICALLY DOMINATING PATH ENDING AT NODE I IS
NOW THE
C     (NOPAT+K)th PATH IN THE MAIN PATH LIST.  SET THE INDICATOR OF THIS
C     PATH = 1.
C
      JJ = NOPAT+K
      INP(JJ) = 1
C
C     LOAD THIS PATH RANK INTO NPPA(I,K).
C
      NPPA(I,K) = JJ
C
C     LOAD THE NODES OF THIS PATH INTO THE NPA ARRAY.  NPA(I,1) IS THE
LENGTH
C     OF THE Ith PATH IN THE MAIN PATH LIST.
C
      NPA(JJ,1) = NPA(J,1)+1
      NPA(JJ,NPA(J,1)+2) = I
      DO 9220 KK = 2,NPA(J,1)+1
      NPA(JJ,KK) = NPA(J,KK)
 9220 CONTINUE
C
C     THE DISTRIBUTION OF THE Kth MOST STOCHASTICALLY DOMINATING PATH
ENDING
C     AT NODE I IS THE DISTRIBUTION OF THE NPR1(K)th CANDIDATE PATH.
LOAD
C     THIS DISTRIBUTION INTO THE DISTRIBUTION OF THE (NOPAT+K)th PATH IN
THE
C     MAIN PATH LIST.
C
      J1 = NPR1(K)
      DO 9230 KK = 1,10
      XINT(104,JJ,KK) = XINT(103,J1,KK)
      VALUE(104,JJ,KK,1) = VALUE(103,J1,KK,1)
      VALUE(104,JJ,KK,2) = VALUE(103,J1,KK,2)
 9230 CONTINUE
      XINT(104,JJ,11) = XINT(103,J1,11)
 9240 CONTINUE
C
C     THE NUMBER OF PATHS IN THE MAIN PATH LIST IS NOW NOPAT+NNN.
C
      NOPAT = NOPAT+NNN
 9250 CONTINUE
      PRINT 9270
      PRINT 9280,NPATHS
      DO 9260 K = 1,NPATHS
      JJ = NPPA(N,K)
      PRINT 9290,K,NPA(JJ,1),(NPA(JJ,KK),KK = 2,NPA(JJ,1)+1)
 9260 CONTINUE
 9270 FORMAT (1H1)
 9280 FORMAT (1X,'THE ',I1,' MOST STOCHASTICALLY DOMINATING PATHS ',
     *'THROUGH THE NETWORK ARE:')
 9290 FORMAT (/ 1X, 'THE RANK ',I1,' PATH WITH ',I3,' NODES:' /
     *(1X,20I4 /))
      RETURN
      END
C     END SUBROUTINE DOMPTH
C
C     ************************************************************
C
C               S U B R O U T I N E       C P U T I M E
C
C     ************************************************************
C
      SUBROUTINE CPUTIME(CPTIME)
      REAL*4 CPTIME
      TYPE TB_TYPE
        SEQUENCE
          REAL*4 USRTIME
          REAL*4 SYSTIME
      END TYPE
      TYPE (TB_TYPE) DTIME_SRC
      CPTIME = DTIME_(DTIME_SRC)
      RETURN
      END
C     END SUBROUTINE CPUTIME
C
C     ************************************************************
C
C               S U B R O U T I N E        T I M E R
C
C     ************************************************************
C
      SUBROUTINE TIMER(DELTA)
      REAL*4  DELTA,CPU2
      CALL CPUTIME(CPU2)
      DELTA = CPU2
      RETURN
      END
```

```
C       END SUBROUTINE TIMER
```