



W&M ScholarWorks

Dissertations, Theses, and Masters Projects

Theses, Dissertations, & Master Projects

1989

A Structural Factoring Approach for Analyzing Probabilistic Networks

Kelly J. Hayhurst
College of William & Mary - Arts & Sciences

Follow this and additional works at: <https://scholarworks.wm.edu/etd>

 Part of the [Mathematics Commons](#)

Recommended Citation

Hayhurst, Kelly J., "A Structural Factoring Approach for Analyzing Probabilistic Networks" (1989).
Dissertations, Theses, and Masters Projects. Paper 1539625547.
<https://dx.doi.org/doi:10.21220/s2-9snk-x503>

This Thesis is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

A STRUCTURAL FACTORING APPROACH FOR ANALYZING PROBABILISTIC NETWORKS

A Thesis

Presented to

The Faculty of the Department of Mathematics

The College of William and Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Master of Arts

by

Kelly Hayhurst

1989

APPROVAL SHEET

This thesis is submitted in partial fulfillment of
the requirements for the degree of

Master of Arts


Author

Approved, December 1989


Douglas Shier


Rex Kincaid

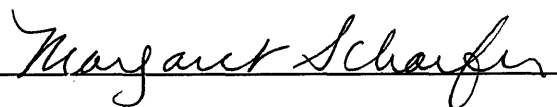

Margaret Schaefer

TABLE OF CONTENTS

	PAGE
ACKNOWLEDGEMENTS	iv
LIST OF TABLES	v
LIST OF FIGURES	vi
ABSTRACT	viii
SECTION 1. INTRODUCTION	2
SECTION 2. NOTATION	6
SECTION 3. LITERATURE SURVEY	8
SECTION 4. STRUCTURAL REDUCTION TECHNIQUES	11
4.1 Fundamental Reductions	12
4.2 Conditioned Reductions	18
4.3 Conditioned Reductions and Shortest Path Calculations	29
4.4 Total Factoring	33
4.5 Determining the Shortest Path Length Distribution	36
SECTION 5. IMPLEMENTATION OF THE REDUCTION ALGORITHM	38
SECTION 6. EXAMPLES	44
6.1 Kleindorfer Crossing Network	45
6.2 Fulkerson Longest Path Example	48
6.3 Shogan Example	50
6.4 Large Kleindorfer Network	52
SECTION 7. CONCLUSIONS	54
BIBLIOGRAPHY	55

ACKNOWLEDGEMENTS

The writer wishes to express her sincere appreciation to Professor Douglas Shier, under whose guidance this investigation was conducted, for his patient guidance and criticism throughout the investigation. The author is also indebted to Professors Rex Kincaid and Margaret Schaefer for their careful reading and criticism of this manuscript.

LIST OF TABLES

Table		Page
1.	Distribution of Shortest Path Length in the Network	32
2.	Storage of Arc and Node Data for a Sample Network	39
3.	Exact Distribution for the Shortest Path Through the Crossing Network	46
4.	Exact Distribution for the Critical Path Through the Crossing Network Compared with the Kleindorfer Cumulative Bounds	46
5.	Exact Distribution of the Longest Path for the Fulkerson Example	49
6.	Exact Distribution for the Shortest Path Through the Shogan Network	51

LIST OF FIGURES

Figure	Page
1.1. Elementary Shipping Network	2
2.1. Directed Graph	7
3.1. Series and Parallel Network Constructs	9
4.1. Series Structure	12
4.2. Reduced Series Structure	13
4.3. Parallel Structure	13
4.4. Reduced Parallel Structure	14
4.5. Series-Parallel Network	15
4.6. Reduced Series-Parallel Network	15
4.7. Example of a Figure-Eight Structure	16
4.8. Reduced Figure-Eight Structure	17
4.9. Partial Network Structure	18
4.10. Deterministic Fan Construct	20
4.11. Reduced Deterministic Fan Construct	20
4.12. Subnetworks for Fan Construct	22
4.13. Loop Construct	22
4.14. Reduced Loop Construct	23
4.15. Subnetworks for a Reduced Loop Construct.....	24
4.16. Examples of Octopus Constructs	25
4.17. Reduction of an Octopus Construct	25
4.18. Two Subnetworks of a Reduced Octopus Construct	26
4.19. Butterfly Construct with Indegree(B)= 2	27
4.20. Reduced Butterfly Construct	28

LIST OF FIGURES

Figure		Page
4.21.	Network with Loop Construct	30
4.22.	Reduced Subnetwork Structures	31
4.23.	Simplified Subnetworks	32
4.24.	Network with No Special Construct	33
4.25.	Subnetwork Structure After Factoring	34
4.26.	New Subnetwork Structure	35
4.27.	Final Subnetwork Structure	35
4.28.	Subnetwork Tree	37
5.1.	Bit Vector Used for Final Distribution Data	40
5.2.	Partial Subnetwork Tree	42
6.1.	Crossing Network	45
6.2.	Fulkerson Example	48
6.3.	Shogan Example	50
6.4.	Large Kleindorfer Network	52

ABSTRACT

In this paper, the problem of finding the shortest path through a stochastic network is analyzed by applying a structural factoring approach. A broad class of stochastic networks with the following characteristics is considered: (1) arc values (length, duration, cost, etc.) are discrete random variables; (2) these random variables are statistically independent; and (3) nodes do not fail. A general algorithm for determining the exact distribution of the shortest path length in such directed networks is presented. The algorithm is based on the concept of structural factoring, in which a stochastic network is decomposed into an equivalent set of smaller, generally less complex subnetworks. Several network constructs are identified and exploited to reduce significantly the computational effort required to solve a network problem relative to complete enumeration. This algorithm can be applied to two important classes of stochastic path problems: determining the critical path distribution for acyclic networks and the exact two-terminal reliability for probabilistic networks. Computational experience with the algorithm has been encouraging and has allowed the exact solution of networks previously analyzed only by approximation techniques.

**A STRUCTURAL FACTORING APPROACH FOR
ANALYZING PROBABILISTIC NETWORKS**

SECTION 1.
INTRODUCTION

In general, network analysis involves the study of systems that can be modeled in terms of nodes and arcs connecting certain of the nodes. Quantitative information such as length, time to failure, or time to completion may in addition be associated with the arcs. For example, Figure 1.1 shows a simple six-node network representing possible shipping routes from a factory (node 1) to a warehouse (node 6). The arcs of this network represent highway routes with their associated mileages given as the arc lengths. A commonly encountered problem in such a network is finding the shortest path from the factory to the warehouse. More generally, shortest path calculations are found to be valuable in analyzing the behavior of various large-scale distribution networks, such as those representing the flow of electricity, water, and vehicles.

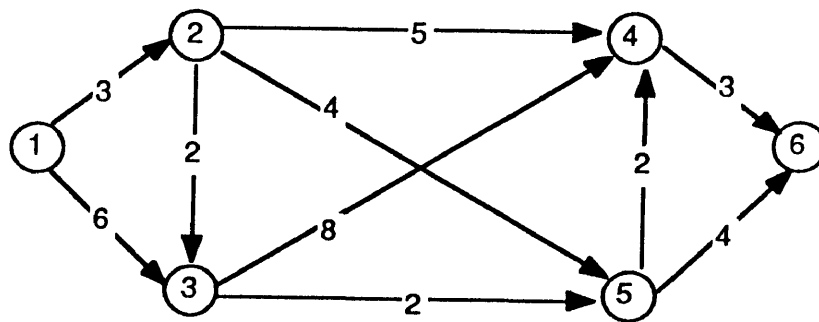


Figure 1.1. Elementary Shipping Network.

Historically, network models have played an important role in the management of complex projects as well as in the analysis of electronic and mechanical systems. The ubiquitous nature of networks has led to an expanding scope of interest in a variety of further areas. Network models are now applied in such areas as transportation problems, complex telecommunications processes, reliability analysis of distributed computer architectures, and even lifeline systems subject to seismic risk¹. The importance of these applications requires that efficient techniques and tools be developed to aid in the network analysis process.

Over the past 25 years, many efficient algorithms have been developed for analysis of networks where the behavior of the individual network components is assumed to be known. Such deterministic networks have been used to model a variety of transportation and scheduling problems. A prominent technique in this regard is the Critical Path Method⁷ (CPM) which is primarily used to determine time-cost tradeoffs in the scheduling and coordination of interrelated activities. Construction and maintenance projects are well suited for analysis by CPM. However, the assumption of deterministic components often fails to describe accurately many other real-world systems.

Progress in developing efficient algorithms to analyze stochastic networks, in which nodes or arcs or their lengths are governed by some random process, has been somewhat limited. One example of a technique used in the analysis of stochastic networks is the Program Evaluation and Review Technique⁷ (PERT) procedure. The PERT procedure expands the deterministic approach of CPM by taking into account the random nature of networks. However, the validity of this technique relies on various approximations and assumptions. Increasingly, these assumptions have been questioned and the accuracy of the resulting conclusions has been widely challenged.¹¹

Since the utility of a network model critically depends on the suitability of its assumptions, a more general model that does not impose such restrictions on the

distributional form of the network components is desirable. To address these concerns, this paper considers a broad class of stochastic networks with the following characteristics: (1) arc values (length, duration, cost, etc.) are discrete random variables, (2) these random variables are statistically independent, and (3) nodes do not fail. These assumptions greatly increase the tractability of the problem while preserving the model's realism by incorporating the random nature of the problem.⁹ Statistical independence is commonly assumed and it reduces the computational requirements associated with most solution approaches.

In this paper, the specific problem of finding the shortest path through such a stochastic network is considered. Recall that the arcs of the network now assume random lengths. These lengths could represent, for example, the duration in traversing a highway segment or the cost of completing a given function. In real-world settings, these values are most realistically viewed as random variables, rather than as fixed, deterministic parameters. Thus, the state of the network depends on the state assumed by each arc. It follows that the shortest path through the network is a function of the random arc lengths; hence, its length can be characterized by a probability distribution.

Although finding the distribution function for the length of the shortest path is conceptually simple, the computational burden of determining the probabilities associated with all possible path lengths is, in general, overwhelming. To illustrate this difficulty, consider the two-terminal network reliability problem which is actually a special case of the stochastic shortest path problem. The two-terminal reliability problem is defined relative to a network G , in which each arc i has a given probability p_i of functioning independent of all other arcs. The two-terminal reliability of the network is equal to the probability that there exists at least one path in G from a specified node s to a specified node t along which all arcs are functioning.

Now consider a stochastic network G' involving the same nodes and arcs as G . Each arc i in G' can only assume the lengths zero and one, with probabilities p_i and $1-p_i$, respectively. If there is a shortest s - t path through G' that has length 0, then there must be some path in G' on which all arc lengths are 0. This directly corresponds to a path in G composed of all functioning arcs. Thus, the probability that the shortest s - t path through G' has length zero is precisely the two-terminal reliability of the original network G .

Ball has shown, however, that network reliability problems are at least as difficult to solve as the set of computationally hard, NP-complete problems such as the infamous traveling salesman problem.¹ Hence, the stochastic shortest path problem is at least as difficult to solve and, in general, poses a formidable computational task. Consequently, most analysis techniques for the stochastic shortest path problem have focused on approximating or bounding the distribution function.²

The objective of this paper is to present a general algorithm for determining the exact distribution of the shortest path length in a directed, stochastic network. This algorithm will thus have applicability to two important classes of stochastic path problems: determining the critical path distribution for acyclic networks and the exact two-terminal reliability for probabilistic networks. The structure of the paper is as follows. In Section 2, the necessary notation and basic terminology will be defined. Previous approaches to the problem will be discussed in Section 3. Our structural decomposition technique will be described in Section 4, and implementation of the algorithm will be discussed in Section 5. In Section 6, the structural decomposition approach is applied to several network examples. A summary of our findings including the limitations of the approach will be presented in Section 7.

SECTION 2.

NOTATION

To facilitate the discussion of probabilistic networks, some basic terminology and notation are first introduced. Such a network is modeled using a directed graph $G=(N,A)$, where N is a set of nodes, representing warehouses or communication centers in the network, and A is a set of arcs, representing traffic routes or communication buses, connecting certain pairs of nodes. An arc (i, j) , where i, j are elements of N , is defined to be a directed link from the origin node i to the destination node j . If more than one arc connects a pair of nodes, the arcs will be denoted with different numbered superscripts. For example, the two different arcs spanning nodes 4 and 5 in Figure 2.1 are denoted $(4,5)^1$ and $(4,5)^2$. Since we are only concerned with the shortest path (or critical path) between two given nodes, we will only consider graphs with one source node s , which has only outgoing arcs, and one terminal node t , which has only entering arcs. Figure 2.1 shows a directed graph with source node 1 and terminal node 5. The indegree of node i , $\text{indegree}(i)$, is the number of arcs entering node i , and the outdegree of i , $\text{outdegree}(i)$, is the number of arcs leaving i . For each of the networks considered, $\text{indegree}(s)=0$ and $\text{outdegree}(t)=0$. In Figure 2.1, $\text{indegree}(3)=2$ and $\text{outdegree}(3)=1$.

A path in the graph from node i to node j , is defined as an ordered sequence of arcs connecting the two nodes. For example, one possible path from node 1 to node 5 in Figure 2.1 is through arcs $(1,2)$ and $(2,5)$. A cycle is a special type of path connecting a node to itself. In Figure 2.1, arcs $(3,2)$, $(2,4)$, and $(4,3)$ form a cycle. In general, the networks treated here will be allowed to contain cycles, except in the special case of critical path calculations for which the underlying graph must be acyclic.

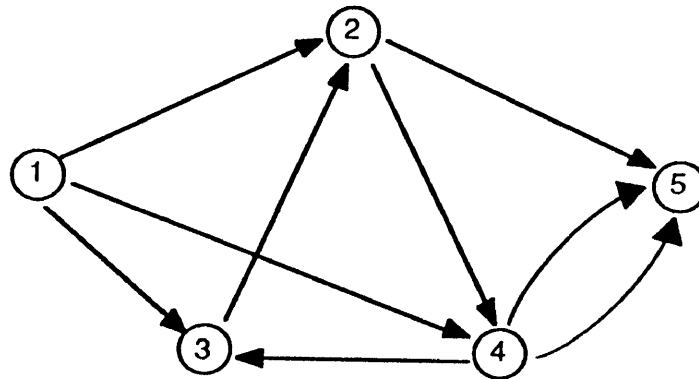


Figure 2.1. Directed Graph.

To incorporate information about the random behavior of the arcs, each arc will be assigned a finite set of integer values, indicating, for example, lengths or durations. The set of arc lengths for arc (i, j) is denoted $a_{(i,j)}$, and the number of lengths associated with arc (i, j) is denoted $s(i, j)$. A particular length assumed by arc (i, j) is denoted $L(i, j)$. For each arc (i, j) , there is a corresponding discrete probability distribution* for the arc lengths and this is denoted $p_{(i,j)}$. As an example, suppose the length of arc $(2,4)$ in Figure 2.1 is uniformly distributed on the interval $[6,9]$. Then the discrete set of arc lengths might be represented by the $s(2,4)=4$ lengths $a_{(2,4)}=(6,7,8,9)$ and the corresponding probability distribution by $p_{(2,4)}=(.25,.25,.25,.25)$.

* Using discrete distributions significantly increases the tractability of the problem in comparison with continuous distributions. Dodin² describes a process for discretizing a continuous distribution.

SECTION 3.

LITERATURE SURVEY

Although techniques for deterministic network analysis are widespread and applicable to large-scale networks, this is not the case for stochastic network analysis. For stochastic networks, current analysis methods are largely confined to approximation, simulation, and bounding techniques. This situation, as described in Section 1, is due to the computational difficulty inherent in these problems. Although exact methods have been developed, they are largely limited to rather small, acyclic networks. The following is a brief description of some of the more prominent techniques proposed for analyzing stochastic networks.

The most intuitive approach for finding the exact distribution of the shortest path length is to enumerate completely every conceivable combination of arc lengths that the network can attain. Although this approach is conceptually simple, it is infeasible as a general technique. Even for a relatively small network, this approach can require an extremely large number of calculations. In a network with 20 arcs and only 2 possible lengths per arc, for example, there would be $2^{20}=1,048,576$ different combinations of arc lengths to consider in determining the distribution of the shortest path length. (In general, if the network contains m arcs and each arc can take on k values, then there would be k^m combinations of arc values to consider.) Hence, this approach has limited value except for very small networks.

Martin¹¹ presented certain network reduction techniques, based on the structure of the network, that play a key role in the simplification of many complex

networks. Martin introduced the idea of simplifying series and parallel constructs, as shown in Figure 3.1, occurring within a network. These reduction techniques sometimes allow a complex network to be reduced to a network with only two nodes and one arc whose associated distribution function is exactly that of the shortest path length through the original network. These series and parallel reductions are basic building blocks of our structural factoring approach and will be discussed more fully in Section 4.

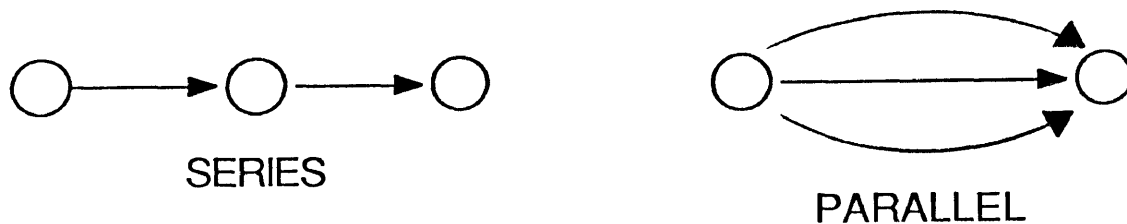


Figure 3.1. Series and Parallel Network Constructs.

Other methods have been developed to deal with more general networks, not reducible by series and parallel simplifications. One approach is to produce an analytic expression for the distribution function by conditioning on common arcs in the network and applying complete enumeration to produce the exact distribution.² Other exact approaches include the use of cutsets^{6,15} and complete path enumeration¹². Another analytic approach is to find an approximation for the probability distribution function. One such approach approximates the distribution function on each arc, in order to reduce the factor k appearing in the expression k^m above, and then applies complete enumeration to yield an approximate final distribution function.³ Even with such approximations, large networks can still present computational difficulty. Additionally, the error inherent in these approximations is not easily quantified.

To avoid the problems of complete enumeration, a number of bounding techniques have been introduced. One of the most widely used bounds was proposed by Malcolm, et al., in 1959.¹⁰ By replacing the distribution on each arc with its expected length, solving the resulting deterministic network produces a lower bound on expected project duration, often referred to as the PERT mean. Other approximations commonly used in many of these bounding techniques are applied while moving sequentially through the nodes of the network. For example, the method of Fulkerson iterates through a network using the expected value of the maximum length to a node as opposed to taking the maximum of the expected value.⁵ This method yields an improved lower bound on the expected completion time compared to the PERT mean and has since been refined by Elmaghraby³ and Dodin². The Fulkerson approach has been further extended by Kleindorfer⁸ and Shogan¹³ to provide lower and upper bounds on the distribution of the critical path. These bounds are largely limited, though, to acyclic networks.

Monte Carlo methods have also been developed to approximate the distribution of the shortest (or critical) path through a network. In 1969, Frank⁴ presented a basic approach to simulating a stochastic network:

- (1) generate a set of arc lengths according to each arc's probability distribution,
- (2) find the shortest path through the resulting deterministic network, and
- (3) repeat this process a large number of times to find the approximate distribution of the shortest path length.

This crude simulation approach can be substantially improved by applying statistical sampling procedures.³ The Monte Carlo approach can be both a practical and powerful approximation tool; however, this approach is prone to sampling errors and can require extensive computational resources to achieve the desired level of confidence¹³.

SECTION 4.

STRUCTURAL REDUCTION TECHNIQUES

In this section, a graph-theoretic procedure for determining the distribution of the shortest path through a stochastic network is described. The objective of the approach is to apply certain structural reductions to a given network until the network is reduced to an equivalent network having only two nodes. The arc connecting the two nodes in this reduced network will provide the distribution of the shortest path length in the original network.

In this paper, two different classes of structural reductions are applied to stochastic networks. The first class of reductions represents fundamental simplifications including the "series" and "parallel" reductions first employed by Martin¹¹. The second class of reductions are based on the concept of local "factoring". The basic difference between the two classes is that the fundamental reductions yield a smaller equivalent network. Local factoring, on the other hand, generates an equivalent set of subnetworks to be solved. Each type of reduction will be discussed more fully in the following subsections.

4.1 Fundamental Reductions

The fundamental reductions can be applied to three basic configurations present in a network: series, parallel, and figure-eight structures. Each of these reductions simplifies the given network by decreasing the number of nodes or the number of arcs (or both).

A series structure exists when there is a node B with $\text{indegree}(B)=1$ and $\text{outdegree}(B)=1$. Figure 4.1 gives an example of a series structure in which the discrete distribution function on arc (A,B) is uniform on [0,3] and the discrete distribution function on arc (B,C) is uniform on [2,6].

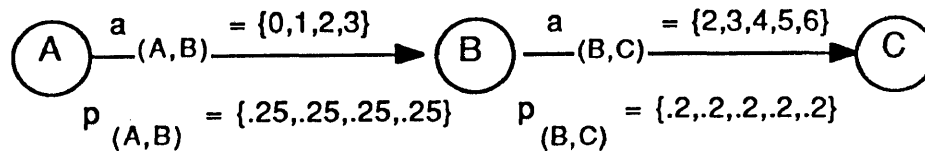


Figure 4.1. Series Structure.

Since any path that goes through node B must include arcs (A,B) and (B,C), these two arcs can be replaced by a single arc between nodes A and C. The set of lengths for the replacement arc (A,C) is the set of all possible additive combinations of arc lengths chosen from (A,B) and (B,C). For our example, $L(A,C)=3$ occurs since $L(A,B)=0$ and $L(B,C)=3$ are present. Moreover,

$$\begin{aligned} \Pr[L(A,C)=3] &= \Pr[L(A,B)=0] * \Pr[L(B,C)=3] + \Pr[L(A,B)=1] * \Pr[L(B,C)=2] \\ &= (.25) * (.2) + (.25) * (.2) = .1, \end{aligned}$$

using the independence of arc lengths associated with distinct arcs. More generally, for those lengths in the set $a_{(A,C)}$ that occur in several ways, their length is only denoted once in $a_{(A,C)}$ and the corresponding probability is the sum of the contributing probabilities. The reduced structure, which corresponds to the "discrete convolution" of the arc distributions on (A,B) and (B,C) , is shown in Figure 4.2.

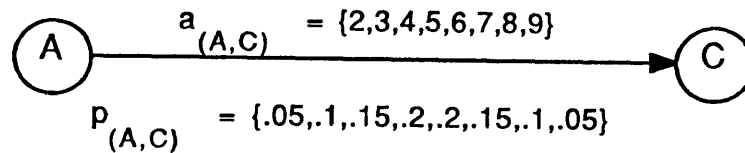


Figure 4.2. Reduced Series Structure.

Two or more arcs that connect the same pair of nodes constitute a parallel structure. Figure 4.3 shows a parallel structure where arc $a_{(A,B)1} = \{2,3,5\}$ and $p_{(A,B)1} = \{.25,.25,.5\}$, the distribution on arc $a_{(A,B)2}$ is uniform on $[0,3]$, and the distribution on $a_{(A,B)3}$ is uniform on $[2,3]$ **.

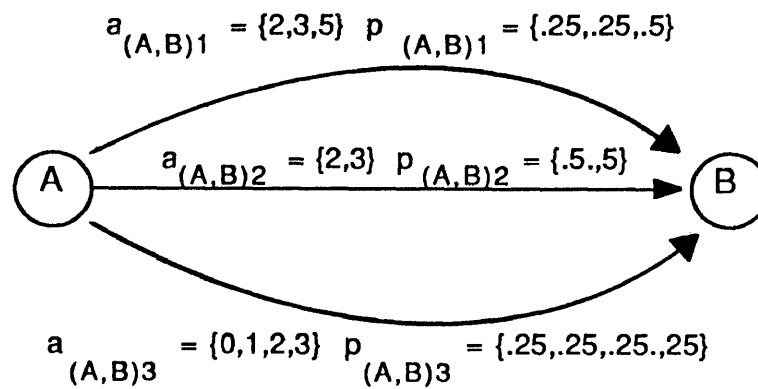


Figure 4.3. Parallel Structure.

** The uniform distribution was used in many examples for ease of illustration. In general, the same reduction techniques apply for any discrete distribution.

For any given realization of a stochastic network that contains this structure, the shortest path will contain at most one of the parallel arcs shown in Figure 4.3. When finding the shortest path through such a deterministic realization, only an arc having the minimum length among these parallel arcs would be considered for the shortest path. It follows that the set of parallel arcs can be replaced by a single arc (A,B), whose arc lengths represent the minimum lengths from all possible combinations of arc lengths of the parallel arcs. One possible combination of arc lengths in the example is $L(A,B)^1=3$, $L(A,B)^2=1$, and $L(A,B)^3=2$; hence, $\min(3,1,2)=1$ is included in $a_{(A,B)}$. For this particular combination, $\Pr[L(A,B)^1=3]*\Pr[L(A,B)^2=1]*\Pr[L(A,B)^3=2]$ contributes toward the overall probability $\Pr[L(A,B)=1]$. Specifically,

$$\Pr[L(A,B)=1] = \sum_{j,k} \Pr[L(A,B)^1=j]*\Pr[L(A,B)^2=1]*\Pr[L(A,B)^3=k] = .25$$

again using the independence of arc lengths.

More generally, any length in the set $a_{(A,B)}$ that arises from several combinations is represented only once in $a_{(A,B)}$ and its corresponding probability is the sum of the appropriate individual probabilities. By applying this strategy, the parallel structure in Figure 4.3 can be replaced by the single arc shown in Figure 4.4.

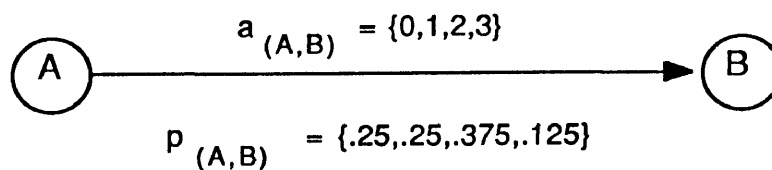


Figure 4.4. Reduced Parallel Structure.

The series and parallel reductions are well-known and have applications in many optimization problems. Martin¹¹ was the first to apply these reductions to the stochastic

shortest path problem. Using these two reduction steps, more complex networks can sometimes be reduced to a single arc. In fact, a series-parallel network is defined to be any network that can be simplified via series and parallel reductions to an equivalent two-node network. For example, consider the network G in Figure 4.5 where the distribution on each arc is uniform on $[1,2]$; that is, $a_{(i,j)}=[1,2]$ and $p_{(i,j)}=[.5,.5]$ for each (i,j) in G .

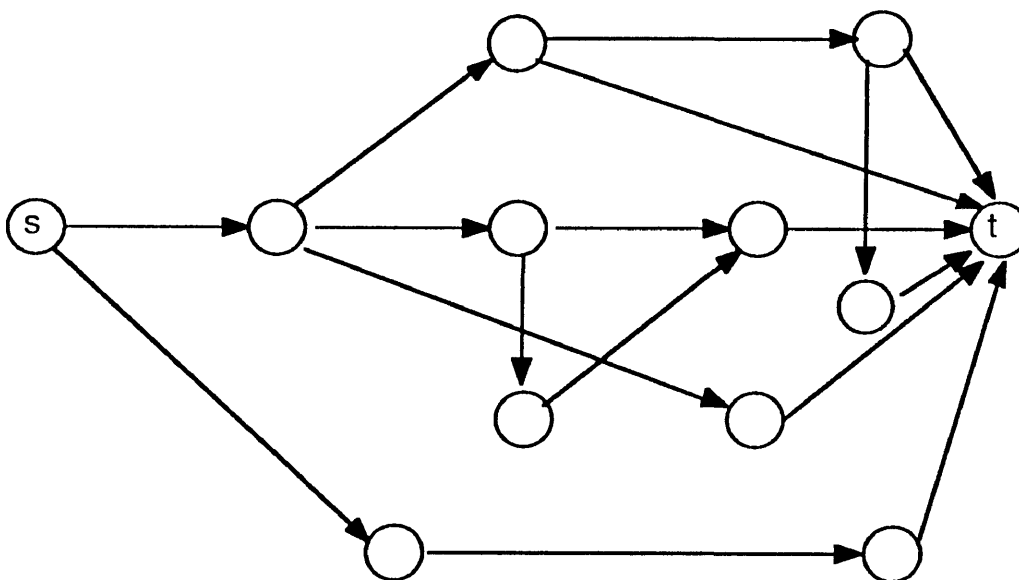


Figure 4.5. Series-Parallel Network.

By applying only series and parallel reductions, this network can be reduced to the equivalent network shown in Figure 4.6.

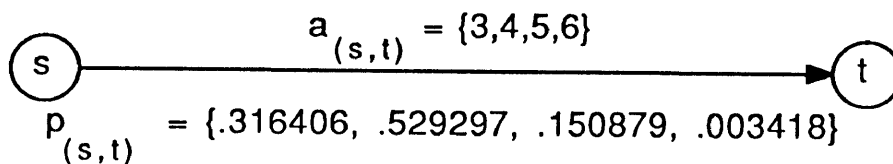


Figure 4.6. Reduced Series-Parallel Network.

The last fundamental reduction, called a figure-eight reduction, while less well-known than the series and parallel reductions, incorporates the same approach to simplifying a network structure into an equivalent, less complex, structure. The figure-eight structure, shown in Figure 4.7, is only found in cyclic networks. In this structure, the center node B in the structure must have exactly $\text{indegree}(B)=2$ and $\text{outdegree}(B)=2$ and must be connected to exactly two other nodes, labeled A and C in the figure. Specifically, B has one incoming arc from node A and one from C; both node A and node C have one incoming arc from B.

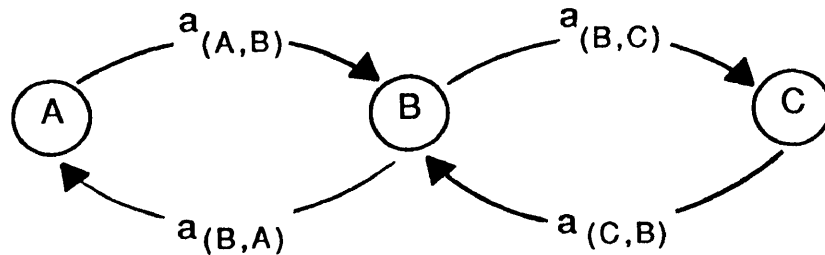


Figure 4.7. Example of a Figure-Eight Structure.

Since we are only considering the shortest path from the source node to the terminal node, the subpath $(A,B) \rightarrow (B,A)$ should never be included in a shortest path. Similarly, the subpath $(B,C) \rightarrow (C,B)$ can safely be ignored. By eliminating these two paths from the set of paths between nodes A and C that could possibly be included in the shortest path, the only remaining paths of interest are $(A,B) \rightarrow (B,C)$ and $(C,B) \rightarrow (B,A)$. The figure-eight structure can thus be simplified by replacing arcs (A,B) and (B,C) with a single arc (A,C) , and by replacing (C,B) and (B,A) with a single arc (C,A) . The lengths and probabilities of the new arcs are determined in the same manner as with series reductions.

For example, the set of lengths for (A,C) is the set of all possible additive combinations of lengths from (A,B) and (B,C). The simplified figure-eight assumes the form shown in Figure 4.8.

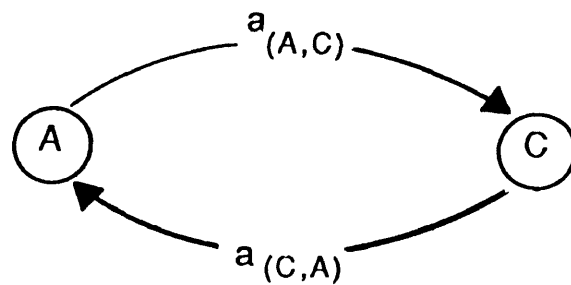


Figure 4.8. Reduced Figure-Eight Structure.

The advantage of using these fundamental reductions is that they yield a single equivalent network that has a smaller number of arcs and possibly fewer nodes. The contribution made to the stochastic shortest path problem through this investigation is the development of a new reduction technique, called conditional factoring, that, like the fundamental reductions, is based on the configuration of the nodes and arcs in the network. Conditional factoring offers additional possibilities for simplifying a given network.

4.2 Conditioned Reductions

Most exact methods for finding the distribution of the shortest path length through a stochastic network, whose architecture cannot be simplified by applying only series and parallel reductions, depend on complete enumeration. In simplifying a construct around a given node by complete enumeration, all possible combinations of arc lengths for all arcs connected to that node would be considered. For example, given the network construct centered at node B in figure 4.9, $3^4=81$ states of that construct would be considered in the reduction process. In general, complete enumeration is computationally infeasible except for very small networks where the arcs can assume only a minimal number of lengths.

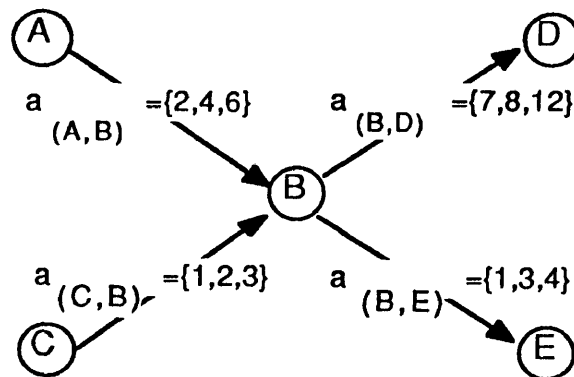


Figure 4.9. Partial Network Structure.

In this investigation, some constructs which are prevalent in many network architectures were identified that can be simplified without complete enumeration. The technique used to simplify these constructs is called conditional factoring, and it is based on the idea that it is sufficient to consider only a specific subset of arcs within certain network constructs in order to generate an equivalent, simplified representation. In general,

conditional factoring can be applied to any node, except the source or terminal nodes, within the network. However, conditional factoring reduces the computational effort, compared with complete enumeration, in simplifying a construct only when it is applied to the four special constructs described in this section. For all other constructs, conditional factoring is equivalent to complete enumeration.

The reductions are performed by identifying a specific construct centered around a node, called the central node, in the network. The structure of this construct is simplified by removing the central node and completing all possible paths through that central point of that construct. In contrast to the fundamental reductions, conditional factoring generates a set of subnetworks to be solved, where each subnetwork incorporates the new, reduced structure into the overall network architecture. These subnetworks are generally smaller in size (i.e. have fewer nodes) and, hopefully, involve reduced computational effort to solve. The lengths of the new arcs in the subnetworks are defined by conditioning on the arcs that are used more than once in defining new arcs in the reduced construct. These arcs that are used more than once in forming the new arcs of the reduced construct are called factoring arcs. A unique subnetwork is defined for each combination of lengths among the factoring arcs, and this eliminates any dependencies among the subnetworks. In this section, the concept of conditional factoring is illustrated by applying it to four special network constructs.

The first construct to be introduced is the fan construct. A deterministic fan construct centered at node B is shown in Figure 4.10. In general, a fan construct with central node i has the following characteristics: (1) i is any node in the network except the source or terminal node, and (2) either $\text{indegree}(i)=1$ or $\text{outdegree}(i)=1$ (if both are 1, then i is the center node in a series structure). If $\text{indegree}(i)=1$, the incoming arc to node i will be called the factoring arc and will be denoted $f(i)$. Similarly, if $\text{outdegree}(i)=1$, the outgoing arc from node i will be called the factoring arc and be denoted $f(i)$. For Figure 4.10, the factoring arc is $f(B)=(A,B)$.

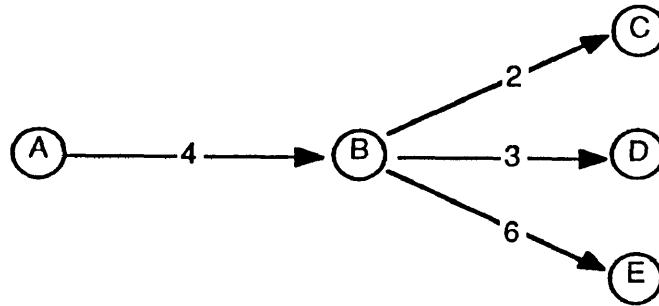


Figure 4.10. Deterministic Fan Construct.

A fan construct is graphically simplified by eliminating the central node and completing each possible path (relative to the structure) through that central connection point. In a deterministic fan construct, the length of the factoring arc is added to the lengths of each of the other arcs connected to the central node. For example, arc (A,C) in Figure 4.11 is formed from arcs (A,B) and (B,C), arc (A,D) is formed from arcs (A,B) and (B,D), and arc (A,E) is formed from arcs (A,B) and (B,E). The length of each new arc is accordingly the sum of lengths of its component arcs. For the network in Figure 4.10, this reduction process yields the network in Figure 4.11.

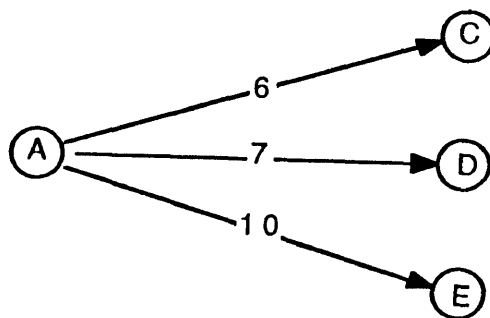


Figure 4.11. Reduced Deterministic Fan Construct.

The concept of conditional factoring is introduced when the length of each arc is assumed to be a random variable. A stochastic fan construct centered at node i is

decomposed based on the distribution of lengths on the factoring arc $f(i)$, and a distinct subnetwork is generated for each length in $a_{f(i)}$. The number of subnetworks created by factoring on node i will be denoted $\omega(i)$. Thus, for the stochastic fan construct, the number of subnetworks generated is equal to the number of possible lengths for the single factoring arc; i.e. $\omega(i)=s[f(i)]$.

In general, the arc lengths in the reduced structure are determined by adding the lengths of their component arcs. The probability that an arc assumes a constant length is 1. If two arcs with constant lengths are combined, the length of the resulting arc is also a constant and, thus, has corresponding probability 1. When an arc with a constant length is combined with an arc whose length is governed by a discrete distribution D , the length of the resulting arc is a random variable governed by D .

To generate the subnetwork S_j , for each $j=1, 2, \dots, \omega(i)$, the following steps are taken: (1) let $f(i)$ take on a constant value c_j , where c_j is the j^{th} length in $a_{f(i)}$; (2) eliminate node i from the original structure; (3) complete all possible paths through the central point; and (4) define the new arc lengths and the associated probabilities.

To demonstrate this process, consider the construct in Figure 4.10, and suppose each arc can take on a value in $\{1,2\}$ with corresponding probabilities $\{.4,.6\}$. This fan construct can be decomposed into an equivalent set of two independent subnetworks shown in Figure 4.12. The first subnetwork S_1 is constructed by assuming $L(A,B)=1$, and the second subnetwork S_2 is constructed for $L(A,B)=2$. The probability of obtaining subnetwork 1 is $\Pr(S_1)=\Pr[L(A,B)=1]=.4$, and, similarly, the probability of obtaining subnetwork 2 is $\Pr(S_2)=\Pr[L(A,B)=2]=.6$.

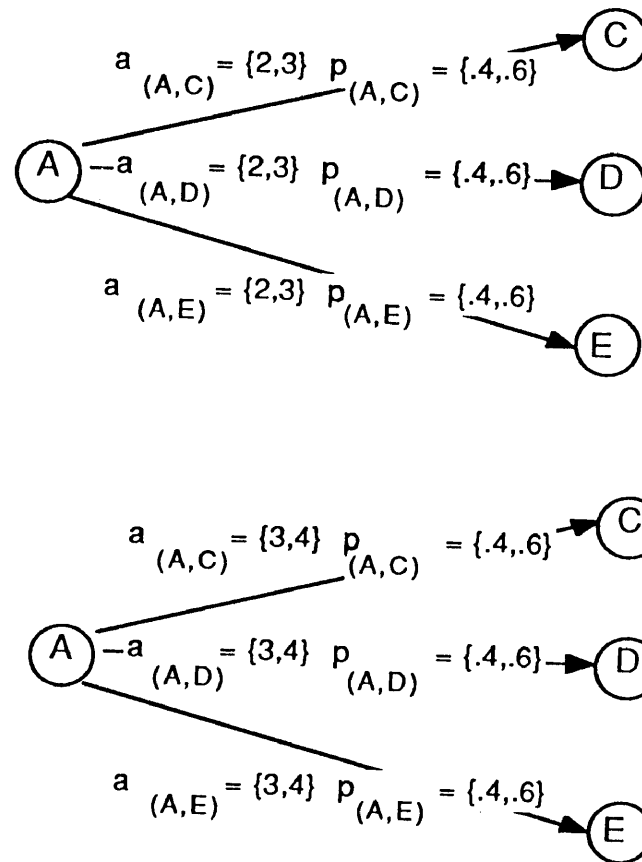


Figure 4.12. Subnetworks for Fan Construct.

The next three reductions focus on constructs with a central node i where either $\text{indegree}(i)=2$ or $\text{outdegree}(i)=2$. Additionally, each of these constructs contains at least one simple cycle of size two involving the central node i . The first of these constructs, shown in Figure 4.13, is called the loop construct with central node B.

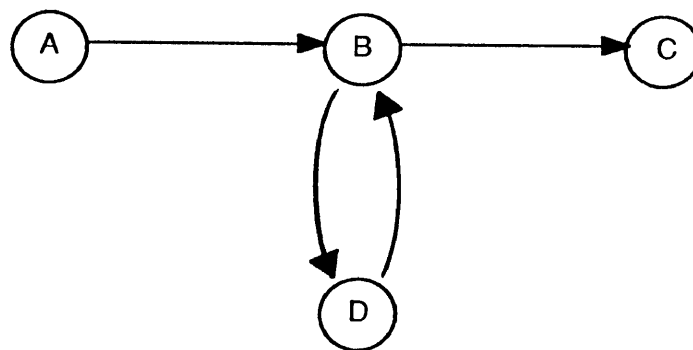


Figure 4.13. Loop Construct.

In general, a loop construct with central node B has the following characteristics: (1) $\text{indegree}(B)=2$, (2) $\text{outdegree}(B)=2$, and (3) B together with some node D form the unique simple cycle of the construct. The possible paths through this structure that would be considered for the shortest path are then $(A,B)\rightarrow(B,D)$, $(A,B)\rightarrow(B,C)$, and $(D,B)\rightarrow(B,C)$. The only other path through the structure, $(A,B)\rightarrow(B,D)\rightarrow(D,B)\rightarrow(B,C)$ would never be considered as part of a shortest path since a subset of that path, $(A,B)\rightarrow(B,C)$, would always be at least as short. As a result, it is possible to replace each of these three paths with an equivalent arc, as shown by the reduced construct shown in Figure 4.14.

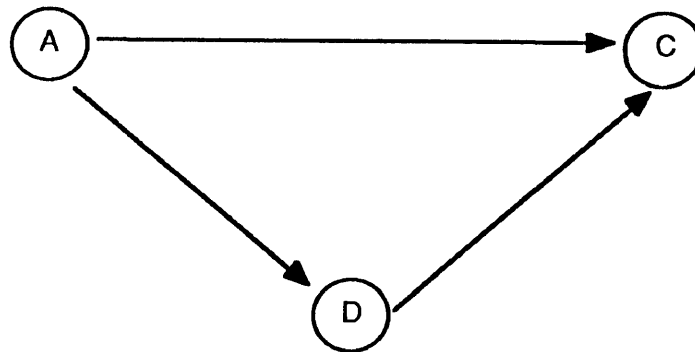


Figure 4.14. Reduced Loop Construct.

Since the arcs (A,B) and (B,C) are used more than once in defining the new arcs in the reduced construct, the lengths of both of these arcs must be considered when generating subnetworks. Hence, arcs (A,B) and (B,C) are factoring arcs for this structure and are denoted $f_1(B)$ and $f_2(B)$, respectively. In general, the number of subnetworks produced by any conditioned reduction is the product of the distribution sizes for each factoring arc. For a loop construct, a distinct subnetwork will be produced for each combination of lengths for the two factoring arcs, so $\omega(B)=s[f_1(B)]*s[f_2(B)]$. To generate the subnetwork S_k , where $k=1,2,\dots, \omega(B)$, the following steps are taken: (1) let each factoring arc $\alpha=f_j(B)$, $j=1,2$, take on

the constant value c_j , where c_j is a length in a_α ; (2) eliminate node B from the original structure; (3) complete all possible paths through the central point; and (4) appropriately define the new arc lengths and their probabilities.

To illustrate, suppose that in the loop construct of Figure 4.13 each arc can take on lengths in the set {1,2} with equal probability. From the first reduction step, we know that four distinct subnetworks will be generated where $L(A,B)=1$ and $L(B,C)=1$ in S_1 ; $L(A,B)=1$ and $L(B,C)=2$ in S_2 ; $L(A,B)=2$ and $L(B,C)=1$ in S_3 ; and, $L(A,B)=2$ and $L(B,C)=2$ in S_4 . These four subnetworks are shown in Figure 4.15. To simplify the notation in the following figures, the lengths and probabilities are displayed in an abbreviated form. In this notation, the lengths and corresponding probabilities for each arc will appear in the format

{length,probability: length,probability: ... : length,probability}

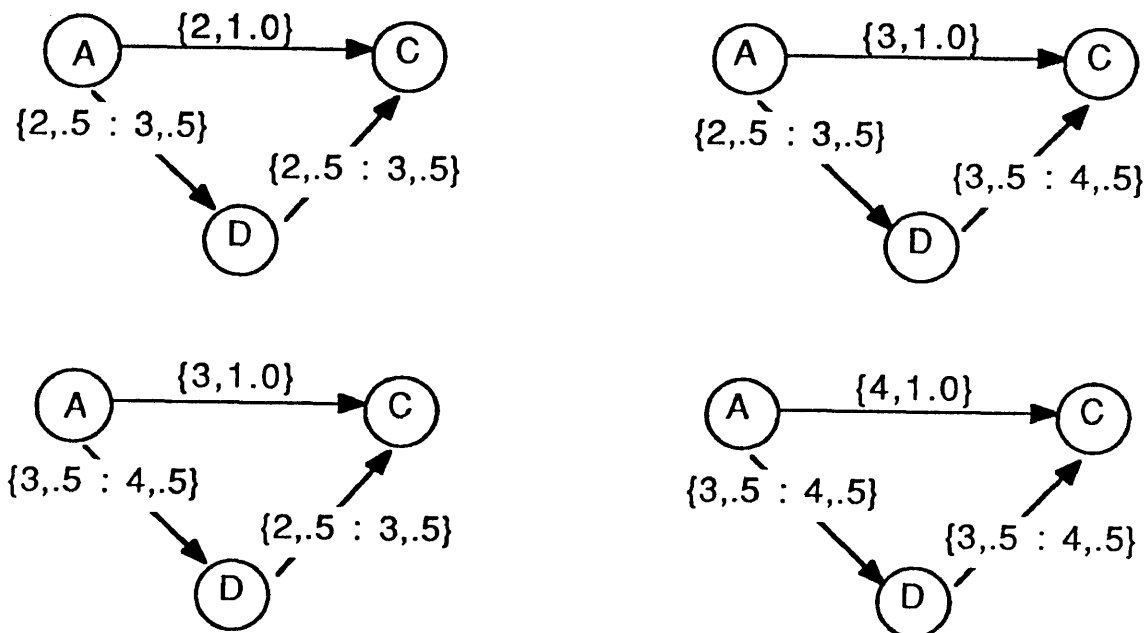


Figure 4.15. Subnetworks for a Reduced Loop Construct.

The third construct, called the octopus, is a generalization of the loop construct and is shown in Figure 4.16. In general, an octopus construct with central node B has the

following characteristics: (1) either $\text{indegree}(B)=2$ and $\text{outdegree}(B)>2$, or $\text{outdegree}(B)=2$ and $\text{indegree}(B)>2$; and (2) B together with some node A forms the unique two-node cycle of the structure.

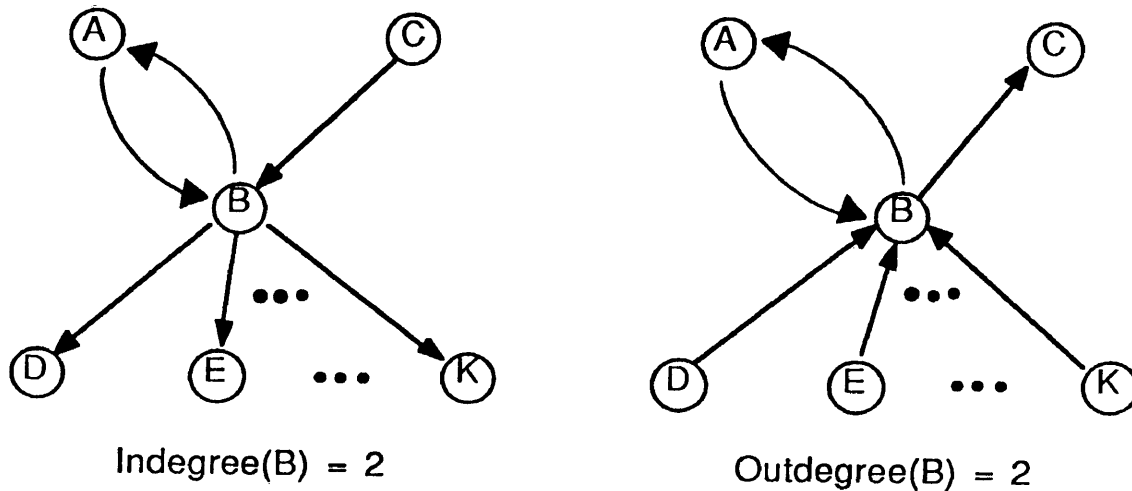


Figure 4.16. Examples of Octopus Constructs.

When the central node is removed from an octopus construct, the number of arcs in each resulting subnetwork is at least as great as the number of arcs in the original network. The structural reduction of an octopus structure is illustrated in Figure 4.17.

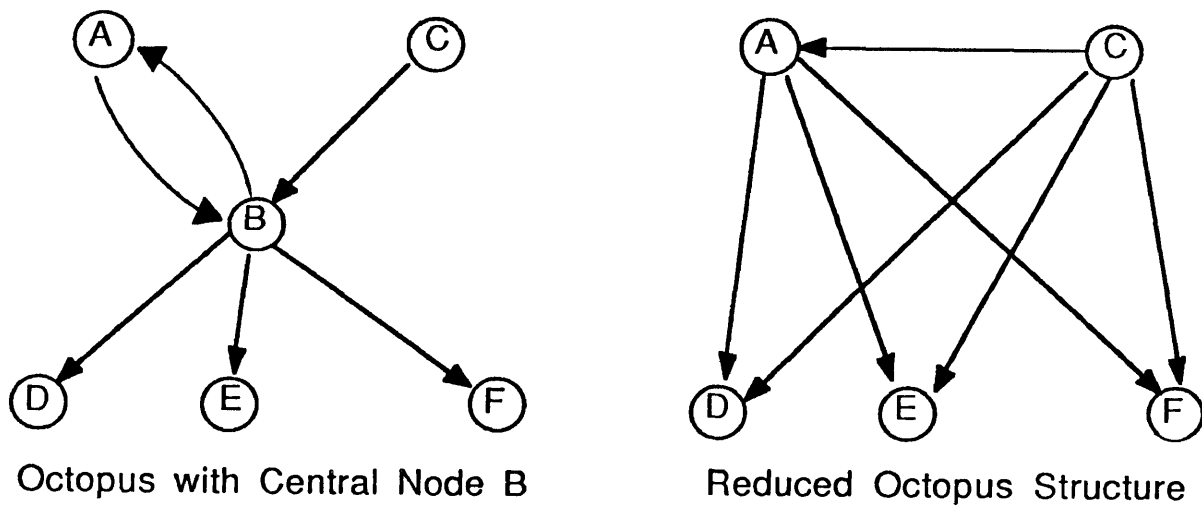


Figure 4.17. Reduction of an Octopus Construct.

Note that in reducing the octopus construct each arc in the original structure is used more than once except for arc (B,A). As a result of this commonality, each arc in the original octopus construct must be a factoring arc except for arc (B,A). The number of factoring arcs is thus $\gamma = \text{indegree}(B) + \text{outdegree}(B) - 1$. As in the fan and loop constructs, the same basic steps are applied to reduce an octopus construct: (1) let each factoring arc $\alpha = f_i(B)$, $i=1,2, \dots, \gamma$, take on a constant length in a_α ; (2) eliminate node B from the original structure; (3) complete all possible paths through the central point; and (4) appropriately define the new arc lengths and probabilities. Suppose each arc in the octopus structure of Figure 4.17 can assume an integer length in the set {1,2} with equal probability. Then, there would be $\gamma=5$ factoring arcs, and $\omega(B)=2^5=32$ subnetworks would be generated in reducing that structure. Two of these subnetworks are shown in Figure 4.18. In these subnetworks, only arc (C,A) has a nonconstant distribution. This is because (C,A) is formed by combining arcs (B,A) and (C,B), and arc (B,A) is the only arc in the structure that is not a factoring arc of node B.

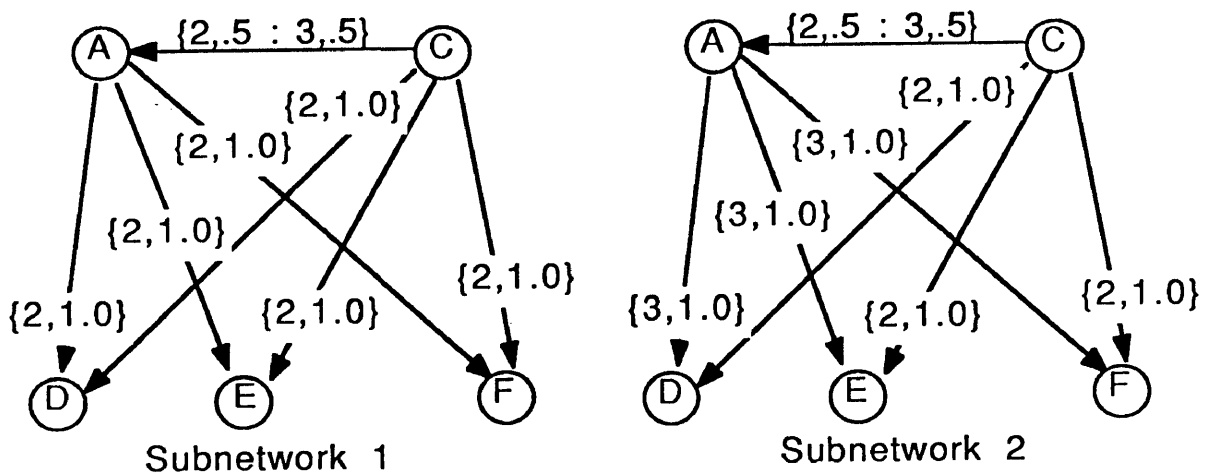


Figure 4.18. Two Subnetworks for a Reduced Octopus Construct.

The last conditioned reduction, shown in Figure 4.19, is called a butterfly construct. Like the octopus construct, either the indegree or the outdegree of the central node must be

2. The butterfly construct, though, must contain exactly 2 simple cycles connected to the central node. In general, a butterfly construct with central node B has the following characteristics: (1) either $\text{indegree}(B)=2$ and $\text{outdegree}(B)>2$, or $\text{outdegree}(B)=2$ and $\text{indegree}(B)>2$; and (2) B is involved with some node A in a simple cycle and B is also involved with some other node C in a simple cycle; see Figure 4.19.

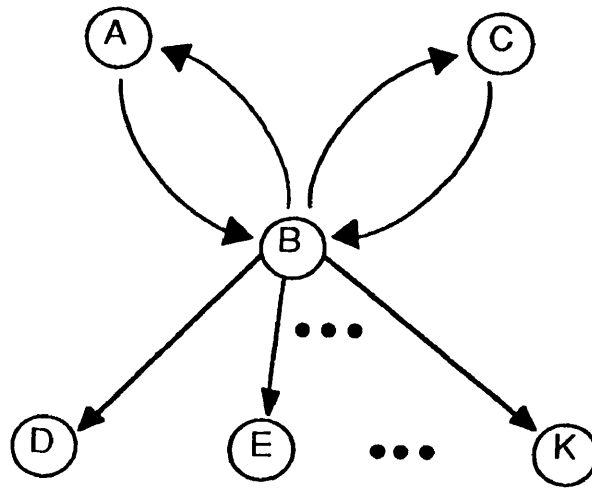


Figure 4.19. Butterfly Construct with $\text{Indegree}(B)=2$.

Removing the central node of a butterfly construct generally creates more arcs in each resulting subnetwork than present in the original structure. When the new arcs in the subnetworks of a butterfly construct are constructed, two of the arcs in the original network are only used one time, such as arcs (B,A) and (B,C) in Figure 4.19. An example of the graphical reduction of a butterfly construct is shown in Figure 4.20.

As with all conditioned reductions, each arc that is used more than once in creating the reduced construct will be a factoring arc. Hence, for any butterfly construct with central node B there will be $\gamma = \text{indegree}(B) + \text{outdegree}(B) - 2$ factoring arcs. The total number of subnetworks that would be generated is $\omega(B) = s[f_1(B)] * s[f_2(B)] * \dots * s[f_\gamma(B)]$.

The same steps are used to generate each of the subnetworks of a butterfly structure as with each of the other conditioned reductions.

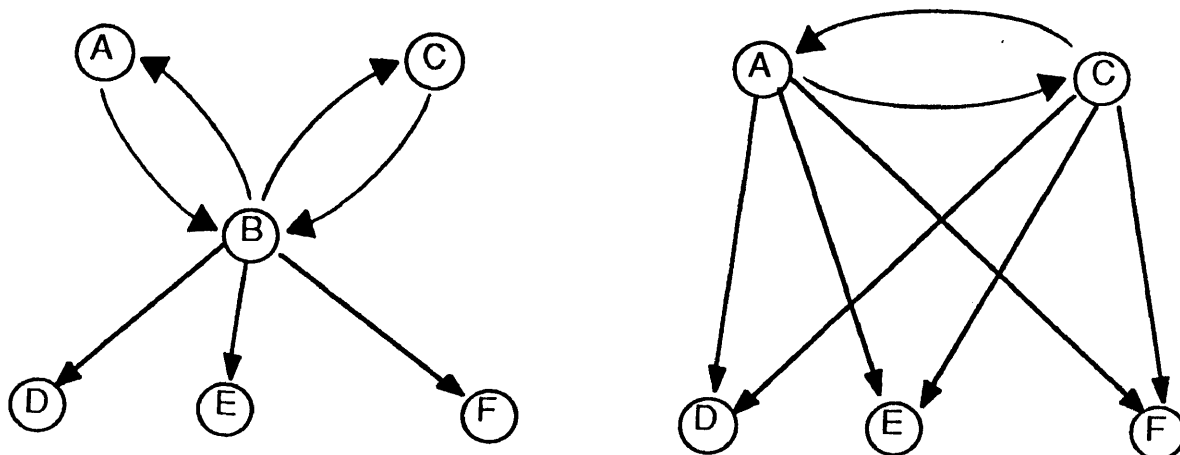


Figure 4.20. Reduced Butterfly Construct.

4.3 Conditioned Reductions and Shortest Path Calculations

The rules for simplifying a special construct present within a stochastic network via conditioned reductions follow a general scheme. Namely, once one of the constructs discussed in Section 4.2 has been identified, the following steps are taken to generate desired subnetworks:

- (1) Identify all factoring arcs for that construct.
- (2) List all possible combinations of lengths for the factoring arcs. Each combination will define a distinct subnetwork.
- (3) For each combination of lengths, compute the product of the probabilities associated with each length. Due to the independence assumption, this product is the probability of that subnetwork occurring.
- (4) Remove the central node from the construct and graphically complete all possible paths through the construct. (Note: any loop arc (i, i) formed can be eliminated from that subnetwork structure.)
- (5) Add the lengths of the component arcs to get the lengths of the new arcs in the subnetworks.
- (6) Define the probabilities for the new arcs. An arc with a constant length has probability 1. When an arc with a constant length is combined with an arc whose length is governed by a discrete distribution D , the length of the resulting arc is a random variable governed by D .

To demonstrate how to determine the distribution of the shortest path length in a network when conditioned reductions are used, consider the network G shown in Figure

4.21; notice that G is not a series-parallel network. Suppose each arc of G can take on lengths in the set of integers $\{1,3\}$ with equal probability.

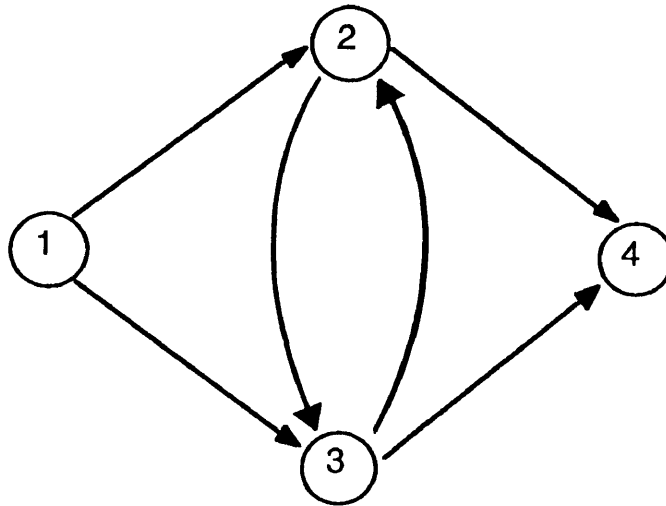


Figure 4.21. Network with Loop Construct.

In this network, there is a loop structure with central node 2 and a loop structure with central node 3. For this example, the choice of central node is not important, so node 2 is arbitrarily selected. (In a later section, criteria are given for choosing nodes for more efficient subsequent factoring.) According to Step (1) above, there will be four subnetworks generated from the four possible combinations of arc lengths from the factoring arcs (1,2) and (2,4). These combinations are $L(1,2)=1$ and $L(2,4)=1$ in S_1 ; $L(1,2)=1$ and $L(2,4)=3$ in S_2 ; $L(1,2)=3$ and $L(2,4)=1$ in S_3 ; and, $L(1,2)=3$ and $L(2,4)=3$ in S_4 . The four subnetworks are shown in Figure 4.22.

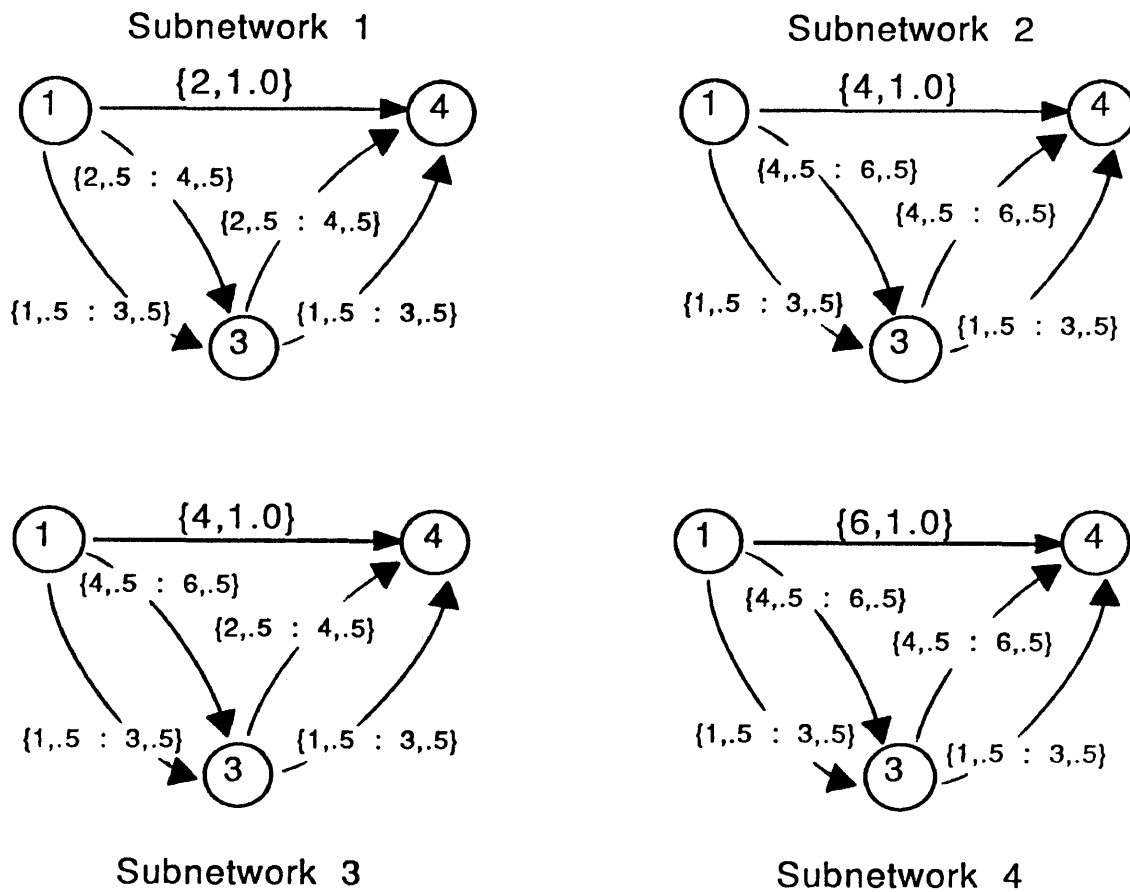


Figure 4.22. Reduced Subnetwork Structures.

From Step (3) above, the probability associated with each subnetwork is the product of the probabilities corresponding to the constant lengths used to define that subnetwork. In Figure 4.22

$$\Pr(S_1) = \Pr[L(1,2)=1] * \Pr[L(2,4)=1] = .25,$$

$$\Pr(S_2) = \Pr[L(1,2)=1] * \Pr[L(2,4)=3] = .25,$$

$$\Pr(S_3) = \Pr[L(1,2)=3] * \Pr[L(2,4)=1] = .25,$$

$$\Pr(S_4) = \Pr[L(1,2)=3] * \Pr[L(2,4)=3] = .25.$$

Notice that each subnetwork in Figure 4.22 can now be simplified to an equivalent two-node network via series and parallel reductions. These simplified networks are shown in Figure 4.23.

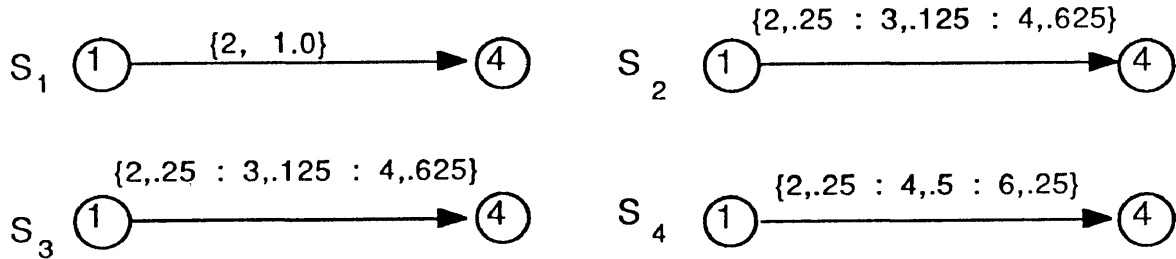


Figure 4.23. Simplified Subnetworks.

The distribution of the shortest path length through the original network in Figure 4.21 is then calculated as shown in Table 1. Namely, this final distribution is found by combining the shortest path distributions in each subnetwork $S_i, i=1,2,3,4$, using as weights the appropriate $\Pr(S_i)$.

Table 1

DISTRIBUTION OF SHORTEST PATH LENGTH IN THE NETWORK

Length of the Shortest Path	Distribution in Final Subnetworks				Distribution for Original Network
	S_1	S_2	S_3	S_4	
	.25	.25	.25	.25	
2	1.0	.25	.25	.25	.4375
3		.125	.125		.0625
4		.625	.625	.5	.4375
6				.25	.0625

4.4 Total Factoring

Although the special constructs required for the basic and conditioned reductions are found in many network configurations, realistic network configurations exist that do not, on initial inspection, contain any of these constructs. In these cases, the factoring approach and the steps for conditioned reductions can still be applied; however, all of the arcs that are incident with the node chosen for removal will now be factoring arcs. That is, all possible combinations of arc lengths around a central node will be completely enumerated. If total factoring were required for each node of the network, except for the source and terminal nodes, this approach would be equivalent to complete enumeration which, as discussed earlier, is computationally infeasible for all but the smallest networks. Fortunately, applying total factoring on a node in a complex network will often yield subnetworks that can be simplified via the basic and conditioned reductions.

For example, the network shown in Figure 4.24 does not contain any of the constructs discussed in Sections 4.1 and 4.2.

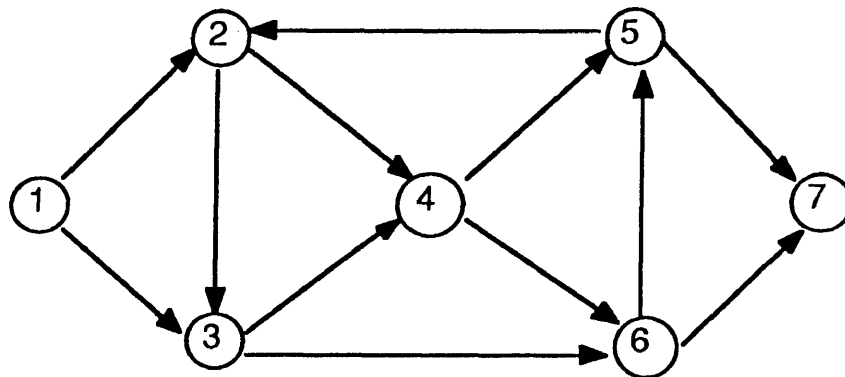


Figure 4.24. Network with No Special Construct.

To demonstrate total factoring, let node 2 be the central node to be removed from the network. The subnetworks generated by removing node 2 will each have the form shown in Figure 4.25.

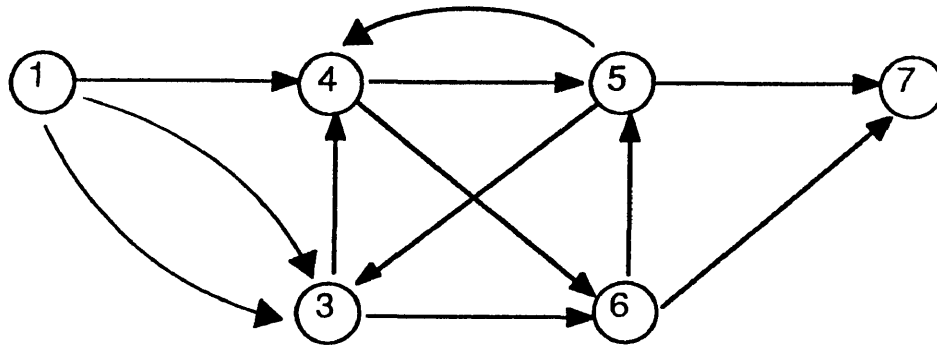


Figure 4.25. Subnetwork Structure After Factoring.

After performing a parallel reduction on the arcs connecting nodes 1 and 3, we can now identify an octopus construct with central node 4 in this subnetwork structure. When node 4 is removed from this subnetwork structure, the new subnetwork, shown in Figure 4.26, results. In this new subnetwork, there is a simple loop construct with central node 3, an octopus construct with central node 6, and a butterfly construct with central node 5. Since there are fewer factoring arcs associated with the loop construct than with the octopus or butterfly, node 3 is chosen for factoring. After factoring on node 3, the subnetworks, whose structure is shown in Figure 4.27, are generated. These subnetworks have the same graphical form as the example given in Figure 4.21 and, thus, can be solved in the same manner.

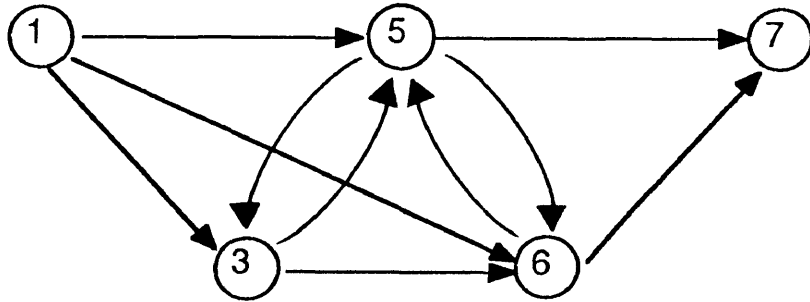


Figure 4.26. New Subnetwork Structure.

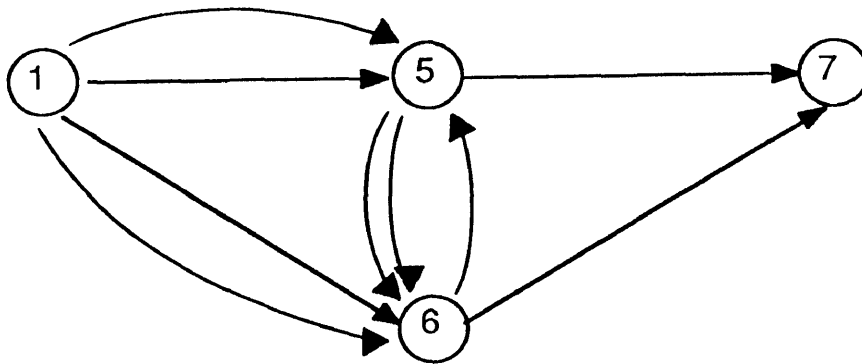


Figure 4.27. Final Subnetwork Structure.

4.5. Determining the Shortest Path Length Distribution

In previous sections, the distribution of the shortest path has been found for series-parallel networks and for networks where only one "level" of subnetworks needs to be generated. For many networks, such as the one given in Figure 4.24, reductions must be repeatedly applied before the original network is simplified to an equivalent two-node network. That is, each generated subnetwork can itself be reduced by further application of conditioned and basic reductions.

In general, the final subnetworks that simplify to two-node structures (weighted by their associated probabilities of occurrence) are the only contributors to the overall distribution of the shortest path length in the network. Hence, for each possible subnetwork, it is important to keep track of the probability of that subnetwork occurring. Suppose a conditioned reduction is applied to a network G , and several subnetworks, denoted S_k , are generated. Then, by the independence assumption, the probability of each subnetwork S_k is calculated according to Step (3) in the general reduction procedure. Now suppose that another conditioned reduction is used to simplify subnetwork S_k and an additional k' subnetworks, denoted $S_{k,j}$, are generated. The current status of the computations can be represented by the tree depicted in Figure 4.28. The shaded leaves of this subnetwork tree represent the currently unresolved subproblems. The probability of the j^{th} subnetwork of S_k occurring, $\Pr[S_{k,j}]$, is just $\Pr[S_{k,j}] = \Pr[S_{k,j} | S_k] \Pr[S_k]$, where $\Pr[S_{k,j} | S_k]$ is calculated according to Step (3) in the general reduction procedure. This process of generating new subnetworks from existing subnetworks continues until a generated subnetwork simplifies to a two-node structure.

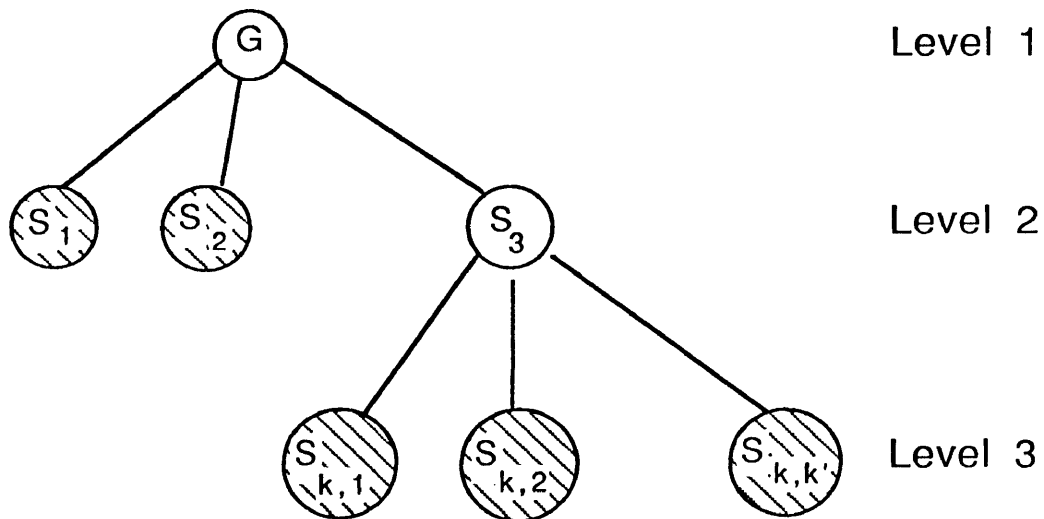


Figure 4.28. Subnetwork Tree.

Note that all subnetworks at level 2 of the subnetwork tree have the same graphical structure. The only differences among these subnetworks are the lengths and probabilities occurring on the arcs. More generally, all subnetworks on the same level of the subnetwork tree have the same graphical structure. So if a conditioned reduction is required to simplify S_k , the same conditioned reduction applies to S_1, S_2, \dots, S_{k-1} . Also, if some $S_{k,j}$ in Figure 4.28 simplifies to a two-node network without additional conditioned reductions, each subnetwork on level 3 will also simplify to a two-node network without generating further subnetworks.

To calculate the distribution of the shortest path in the original network, we only need to consider subproblems at the final level of the subnetwork tree; that is, those leaf subnetworks that can be simplified to two-node networks without further conditioned reductions. For each leaf subnetwork in the final level, we multiply the probability of each arc length appearing in the associated distribution by the probability of that subnetwork occurring. Then, for every possible length, we sum the contributing probabilities across all leaf subnetworks. This yields the distribution of the shortest path length in the original network.

SECTION 5.
IMPLEMENTATION OF THE REDUCTION ALGORITHM

The basic and conditioned reduction techniques presented in the previous section have been implemented in a computer program to automate the process of finding the distribution of the shortest path. The program is written in FORTRAN 77 and was implemented on a Microvax 3200 computer. *** Although the reduction techniques apply to arbitrarily large networks, practical limitations on computer resources place limits on the size of network that can be input to the program. Currently, the program will accept a network with a maximum of 20 nodes and 40 arcs.

An important aspect of the program is the compact storage of the node and arc data. The following information is kept for each node: (1) the first arc leaving that node (OUTSTRT), (2) the first arc entering that node (INSTRT), (3) the number of leaving arcs (OUTDEG), and (4) the number of incoming arcs (INDEG). The following information is tracked for each arc: (1) the origin of the arc (FROM), (2) the destination of the arc (TO), (3) the size of the distribution for that arc (DIS_SIZE), (4) the lengths assumed by each arc (LENGTH), and (5) the corresponding probability for each length (PROB). Additional information describing the sets of arcs entering and leaving each node is stored in linked lists called NEXTIN and NEXTOUT. For example, the node and arc data for the network in Figure 4.20 are stored as shown in Table 2. The various elements of this data structure are dynamically created as each network is read into the program.

*** Although other programming languages lend themselves better to implementing data structures and recursion, the author's primary programming language is FORTRAN.

Table 2

STORAGE OF ARC AND NODE DATA FOR A SAMPLE NETWORK

ARC DATA							NODE DATA			
FROM	TO	NEXTIN	NEXTOUT	DIS_SIZE	LENGTH	PROB	OUTSTRT	INSTRT	OUTDEG	INDEG
1	2	0	0	2	1	.5	2	0	2	0
					3	.5	4	5	2	2
1	3	0	1	2	1	.5	6	3	2	2
					3	.5	0	6	0	2
2	3	2	0	2	1	.5				
					3	.5				
2	4	0	3	2	1	.5				
					3	.5				
3	2	1	0	2	1	.5				
					3	.5				
3	4	4	5	2	1	.5				
					3	.5				

In addition to using this array representation with linked lists to efficiently store and access the node and arc data, an effort was made to minimize other storage requirements. A common approach used to minimize storage throughout the program was to use generalized bit vectors to store information. A bit vector was used, for example, to accumulate the final distribution data for the shortest path length. At the start of the program, this vector is initialized to zero. Then, as subnetworks are completely reduced, the appropriate distribution information is added to the bit vector. As an illustration, suppose a network problem is being solved where the first subnetwork to be simplified contributes lengths {2,4,7} with corresponding probabilities {.1,.06,.3} to the final distribution of the shortest path length. Suppose, further, that the next subnetwork to be completely reduced contributes lengths {2,5,6,7} with probabilities {.03,.01,.14,.002}. Figure 5.1 shows the state of the bit vector for the final distribution of the shortest path length after the second subnetwork has been solved. This bit representation is appropriate when the arc lengths assume nonnegative integer values, as is typically assumed. However, it could be modified to deal with real data using the concept of "buckets"¹⁶.

0	0	.13	0	.06	.01	.14	.302	0	...
---	---	-----	---	-----	-----	-----	------	---	-----

Figure 5.1. Bit Vector Used for Final Distribution Data.

Bit vectors are also used in the management of the arc distribution data when series and parallel reductions are performed. Notice that by using bit vectors, the distribution information can be easily maintained in "sorted order" by length. The combined use of linked lists and bit vectors to store and manipulate data enhances the overall efficiency of the program.

Utilizing these data structures to store the necessary arc, node, and distribution information, the following algorithm is applied to determine the distribution of the shortest path:

```

Input network data and store it
Use all applicable basic reductions to simplify the network
If the network has been completely reduced
  then
    Store the distribution information (note: this is the
      complete distribution of the shortest path length)
  else
    Identify a node to factor on
    Apply the appropriate reduction, generate the subnetworks
      and place them in a stack
    While there are subnetworks in the stack
      Remove the subnetwork on the top of the stack
      Use basic reductions to simplify the subnetwork
      If the subnetwork has been completely reduced
        then
          Store the distribution information for that
            subnetwork
        else
          Identify a node within the subnetwork to
            factor on
          Apply the appropriate reduction, generate new
            subnetworks and place them on top of the
            stack
        end if condition
      end while condition
    end if condition
  Print the distribution of the shortest path length

```

The algorithm used to find the longest path is identical to that used to find the shortest path except that maximum values are recorded in place of minimum values throughout the program.

By applying this algorithm, a tree of subnetworks, as demonstrated in Section 4.5, is grown. Level 1 of this tree represents the original network. When this network is decomposed into a set of equivalent subnetworks, another level is added to the tree; that is, the immediate "successor" subnetworks of the original network constitute level 2 of the tree. The simplification of these subnetworks adds yet another level to the tree. This process continues until the subnetworks on some level of the tree simplify to two-node networks, giving the necessary arc data for the distribution of the shortest path length.

In terms of efficient storage of the subproblems, the generated subnetworks are stored using a stack, as described in the above algorithm. When the original network is decomposed into several subnetworks, the storage space for the original network is used to store the information for one of these subnetworks. In general, when any subnetwork is simplified by a conditioned reduction, the storage space for the parent subnetwork is used for one of the successor subnetworks since the information about the parent subnetwork is no longer needed. This process in effect carries out a depth-first search of the subnetwork tree.

To illustrate, consider a network G that initially reduces to two subnetworks, G_1' and G_2' . According to the algorithm, we would first try to simplify G_2' . Suppose that G_2' itself generates three subnetworks, G_1'' , G_2'' , and G_3'' . Next, we would try to simplify G_3'' ; suppose that G_3'' produces two series-parallel networks, G_1''' and G_2''' . Figure 5.2 shows the partial subnetwork tree that has been grown to this point. Notice that at this point only five subnetworks are stored in the stack of networks: G_1' , G_1'' , G_2'' , G_1''' and G_2''' . In completely solving this hypothetical network problem, no more than five subnetworks will ever occur in the stack of networks even though a total of 21 networks will be generated and processed.

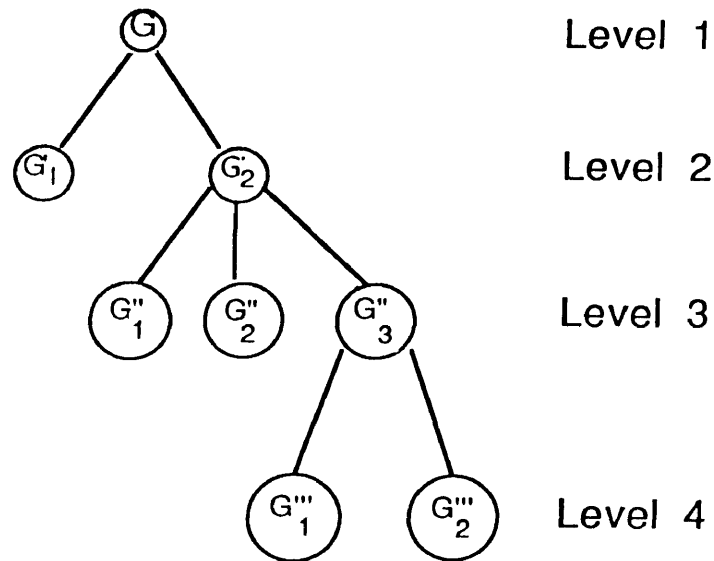


Figure 5.2. Partial Subnetwork Tree.

Additional effort was made to minimize the "width" of the subnetwork tree by judiciously choosing the node for factoring. Specifically, the factoring node i is chosen to fulfill the following criteria:

- (1) $\min[\text{indegree}(i), \text{outdegree}(i)] \leq \min[\text{indegree}(j), \text{outdegree}(j)]$ for all nodes j in the current subnetwork, and
- (2) among all nodes k that satisfy (1), $\omega(i) \leq \omega(k)$.

Our empirical findings have shown that in most cases, choosing the factoring node based on these criteria results in a smaller total number of subnetworks that need to be solved.

In general, the size of the subnetwork tree depends on the complexity, number of nodes, and the size of the distribution of lengths for each arc in the original network. If there are n nodes in the original network, there will be at most $n-1$ levels in the subnetwork tree. If there are k subnetworks generated during each simplification, then there would be a total of $1 + k + k^2 + \dots + k^{n-2} = (k^{n-1} - 1)/(k - 1)$ subnetworks to solve. However, the maximum number of subnetworks that ever need to be stored is $(k-1)(n-1) - (k-2)$. These are worst-case estimates. In practice, the number of subnetworks solved and the

maximum number stored have been far fewer than those predicted by the worst-case estimates. The results presented in the next section demonstrate that a fairly modest computational effort is usually required.

SECTION 6.

EXAMPLES

In this section, the structural factoring approach will be applied to several network examples from the literature. The first two examples are acyclic networks where only basic reductions are necessary to find the distribution of the shortest path or longest path in the network. The third example is a complex cyclic network that requires both octopus and butterfly reductions to simplify the network. In the last example, the exact two-terminal reliability of a large network is determined.

6.1 Kleindorfer Crossing Network

The first problem, shown in Figure 6.1, is the "crossing network"**** analyzed by Kleindorfer.⁸ In this paper, Kleindorfer gives bounds on the cumulative distribution for this network where each arc assumes a length in the set $\{1,2,3,4,5\}$ with equal probability.

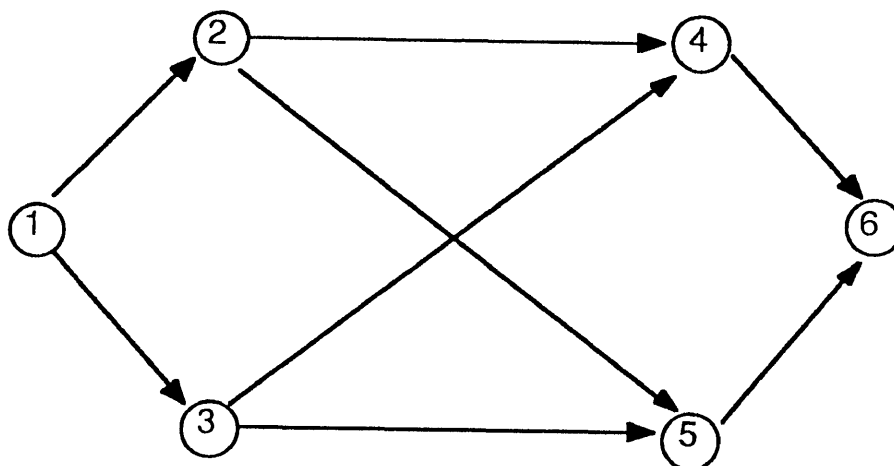


Figure 6.1. Crossing Network.

By applying structural factoring to the fan structures in this network, the exact distribution of the shortest path length from node 1 to node 6 can be determined. Table 3 gives the the exact distribution of the shortest path for this network; Table 4 shows the exact cumulative distribution of the critical (longest) path through the same network as well as the Kleindorfer bounds.

**** Also referred to as the wheatstone bridge network

Table 3

EXACT DISTRIBUTION FOR THE SHORTEST PATH THROUGH THE CROSSING NETWORK

Length	Exact Probability
3	0.03064064
4	0.08365312
5	0.14335488
6	0.18986496
7	0.20426496
8	0.16326144
9	0.10479360
10	0.05362176
11	0.02052864
12	0.00505344
13	0.00087552
14	0.00008448
15	0.00000256

Table 4

EXACT DISTRIBUTION FOR THE CRITICAL PATH THROUGH THE CROSSING NETWORK
COMPARED WITH THE KLEINDORFER CUMULATIVE BOUNDS

Length	Kleindorfer Lower Bound	Exact Cumulative Probability	Kleindorfer Upper Bound
3	0.000	0.00000256	0.008
4	0.000	0.00008704	0.032
5	0.000	0.00096256	0.080
6	0.002	0.00801600	0.160
7	0.014	0.02654464	0.280
8	0.055	0.08016640	0.424
9	0.149	0.18496000	0.576
10	0.312	0.34822144	0.720
11	0.528	0.55248640	0.840
12	0.731	0.74235136	0.920
13	0.882	0.88570624	0.968
14	0.969	0.96935936	0.992
15	1.000	1.00000000	1.000

Notice in Table 4 that Kleindorfer's lower bound is much closer to the exact cumulative distribution than the upper bound. Recall from Section 3 that bounding approaches have largely concentrated on providing only a lower bound on the distribution. For critical path problems, the expected project duration time, which is the expected length of the critical path, is often of interest. The exact expected project duration can be calculated using the exact distribution of the critical path length, and this value is 11.203136. Kleindorfer gives a lower bound, $LB=9.000$, and an upper bound, $UB=11.358$, for this problem. Shogan¹³ also gives bounds on the expected project duration for this problem: $LB=10.6$ and $UB=11.358$. Only 3.7 seconds of central processing unit (cpu) time were required to solve this problem exactly.

Since each arc in this network has five possible lengths, the complete enumeration approach would consider $5^8=390,625$ possible states of the network to determine exactly the distribution of the shortest path length. For each possible state of the network, the shortest path length must be identified. A computer program, written by the author, that implements this complete enumeration approach took 57.21 seconds of cpu time to execute, and the resulting distribution agreed with the one given in Table 3. The conditional factoring approach clearly provides a drastic improvement relative to complete enumeration for determining an exact solution.

6.2 Fulkerson Longest Path Example

In this section, we examine a network problem presented by Fulkerson.⁵ The network has a simple four-node architecture, shown in Figure 6.2, where the length of each arc is uniformly distributed on the interval $[0,2]$.

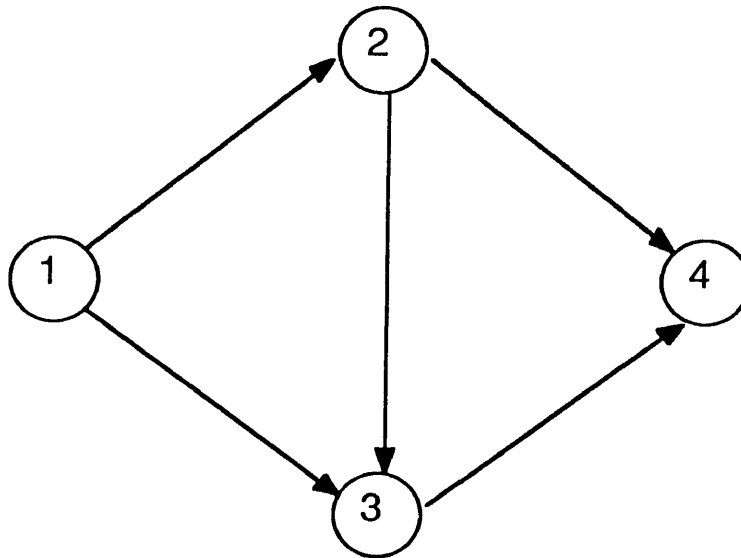


Figure 6.2. Fulkerson Example.

In this network, two fan constructs can be identified, one centered at node 2 and the other centered at node 3. The exact distribution of the longest path through the network has been determined using our algorithm and is given in Table 5.

Table 5

EXACT DISTRIBUTION OF THE LONGEST PATH FOR THE FULKERSON EXAMPLE

Length	Exact Probability
0	0.004115226
1	0.045267490
2	0.201646091
3	0.304526749
4	0.296296296
5	0.111111111
6	0.037037037

From the distribution in Table 5, the exact expected length of the critical path is calculated to be 3.32510288. In the Fulkerson paper, the lower bound calculated by Fulkerson's method is given as 3.22. This problem required 3.5 seconds of cpu time to solve for the exact distribution and four subproblems were generated.

6.3 Shogan Example

The next problem is a cyclic network, shown in Figure 6.3, presented by Shogan¹⁴. This is a complex network that requires both octopus and butterfly reductions to obtain the distribution of the shortest path length from node 1 to node 7.

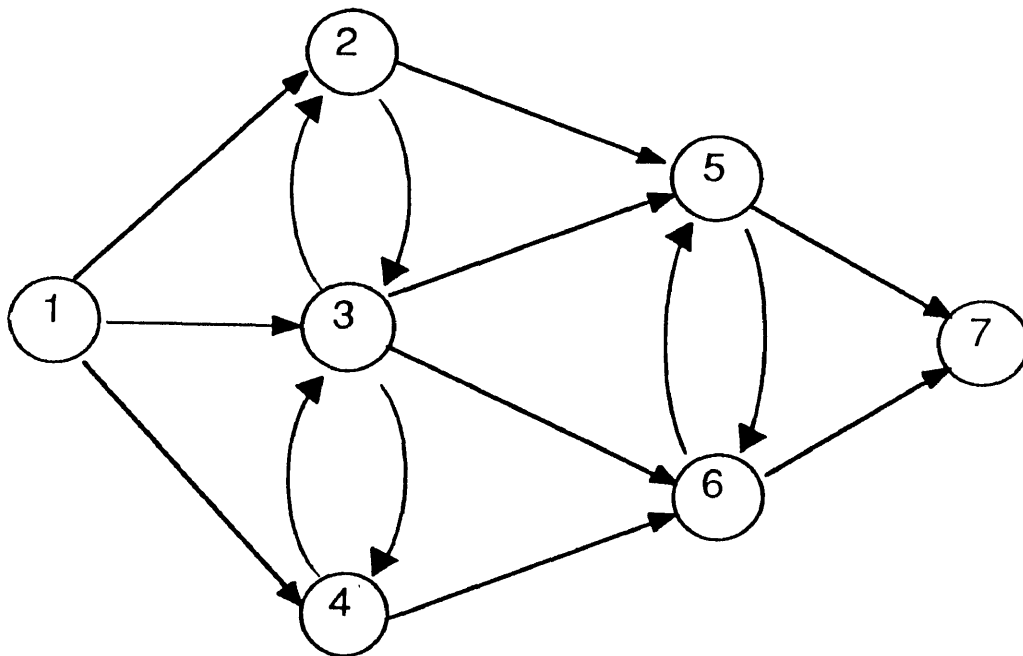


Figure 6.3. Shogan Example.

For this example, the set of lengths for each arc is $\{0,2,4\}$ with corresponding probabilities $\{.4,.2,.4\}$. The exact distribution of the shortest path length through this network is given in Table 6.

Table 6

EXACT DISTRIBUTION OF THE SHORTEST PATH LENGTH FOR THE SHOGAN NETWORK

Length	Exact Probability
0	0.3047540654
2	0.2810753843
4	0.2784418939
6	0.0986955899
8	0.0339240387
10	0.0028468838
12	0.0002621440

To solve this network for the distribution of the shortest path length, 1528 subnetworks were generated, but the maximum number of networks stored was 27. Although a large number of subproblems were generated, this problem took only 8.19 seconds of cpu time to solve. By contrast, it took 6.787 hours of cpu time to solve this same problem by completely enumerating all possible combinations of path lengths.

This problem can also be considered as a two-terminal reliability problem where the probability that each arc is operational is 0.4. The exact two-terminal reliability of the network, as shown in Table 6, is 0.3047540654. Shogan gives the following bounds on the two-terminal reliability: LB=0.1971 and UB=0.3527.

6.4 Large Kleindorfer Network

The last example is a large network with 20 nodes and 38 arcs that was also presented by Kleindorfer⁸. This network is shown in Figure 6.4.

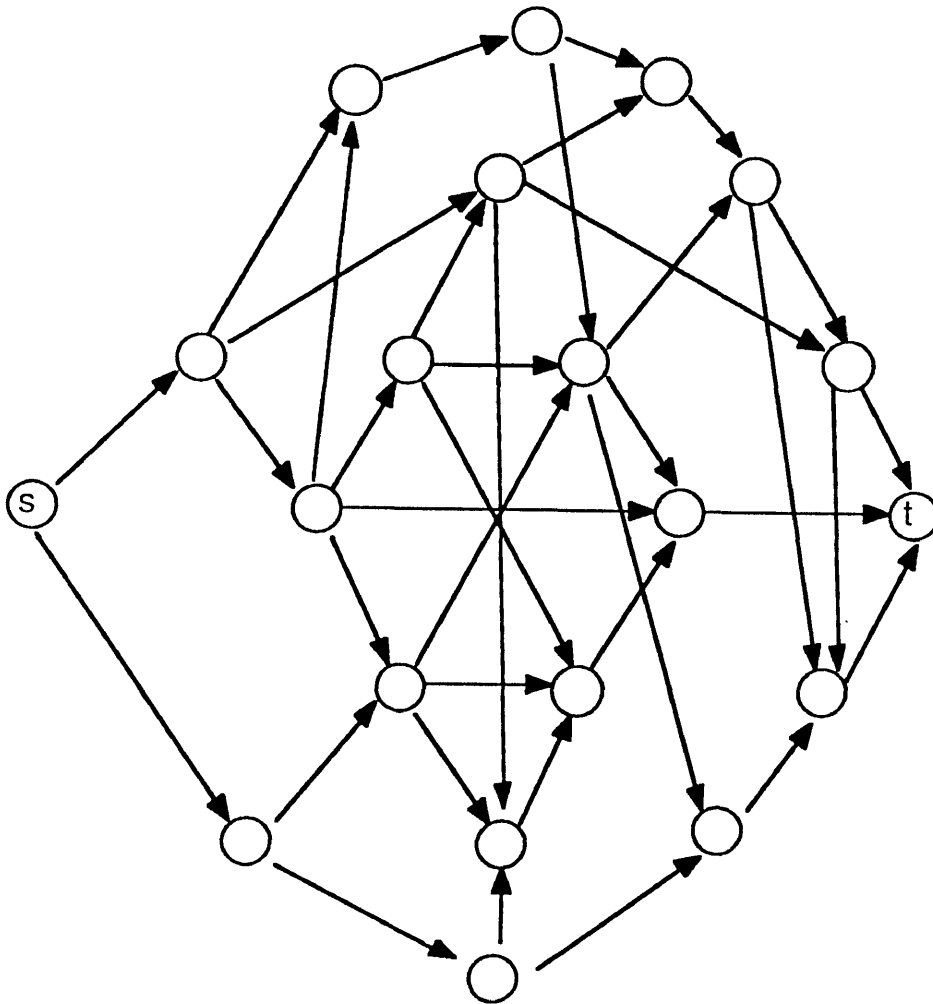


Figure 6.4. Large Kleindorfer Network.

The two-terminal reliability of this network, in which each arc is operational with probability 0.9, can be determined by letting each arc take on lengths {0,1} with corresponding probabilities {0.9,0.1}. Using the program with these parameters, the exact s-t reliability of this network is found to be 0.98612801. The subnetwork tree generated for this problem had 15 levels. A maximum of 15 subproblems were stored at any given time, and 32,767 total subproblems were generated. The program required 2.4 minutes of cpu time to solve this problem. For this problem, complete enumeration would not be feasible since 2^{38} or approximately 2.75×10^{11} states of the network would be individually examined to find the shortest path length through each.

SECTION 7. CONCLUSIONS

The structural factoring approach allows the exact distribution of the shortest path length through a stochastic network to be determined. The application of structural factoring to the fan, loop, octopus, and butterfly constructs significantly increases the tractability and applicability of the approach, since complete enumeration of all states is not necessary to simplify these constructs. Although there are network architectures that do not initially contain these special constructs, factoring can often be locally applied to produce subnetworks that do contain such constructs. Hence, this approach extends the computational range, especially in the case of cyclic networks, to larger and more complex networks than have previously been solved as compared with other exact methods. Structural factoring can also be applied to find the distribution of the critical path length in acyclic networks and the two-terminal reliability for a network.

Although the technique of structural factoring is theoretically unconstrained, the size of the network that can be solved using the algorithm is restricted by computational resources. In the worst case, the computational effort can grow exponentially with the size of the problem, which is indicative of the NP-complete status of the stochastic path problem. Computational experience with the structural factoring algorithm has nonetheless been encouraging. Moreover, the ability to solve certain nontrivial problems exactly can serve as a baseline for assessing the accuracy of various approximation schemes proposed to solve larger problems.

BIBLIOGRAPHY

1. Ball, Michael O., "Computational Complexity of Network Reliability Analysis: An Overview", *IEEE Trans. on Rel.*, vol. R-35, 1986, pp. 230-239.
2. Dodin, Bajis, "Approximating the Distribution Functions in Stochastic Networks", *Computers & Operations Res.*, vol. 12, 1985, pp. 251-264.
3. Elmaghraby, Salah E., Activity Networks: Project Planning and Control by Network Models. John Wiley & Sons, 1977.
4. Frank, H., "Shortest Paths in Probabilistic Graphs", *Operations Research*, vol. 17, 1969, pp. 583-599.
5. Fulkerson, D. R., "Expected Critical Path Lengths in PERT Networks", *Operations Research*, vol. 10, 1962, pp. 808-817.
6. Hansler, E., G. K. McAuliffe, and R. S. Wilkov, "Exact Calculation of Computer Network Reliability", *Networks*, vol. 4, 1974, pp. 95-112.
7. Hillier, Frederick S., and Gerald J. Lieberman, Introduction to Operations Research, 3rd ed. Holden-Day, Inc., 1980, pp. 246-259.
8. Kleindorfer, George B., "Bounding Distributions for a Stochastic Acyclic Network", *Operations Research*, vol. 19, 1971, pp. 1586-1601.
9. Larson, Richard C., and Amedeo R. Odoni, Urban Operations Research. Prentice-Hall, Inc., 1981, p. 455.
10. Malcolm, D. G., J. H. Roseboom, C. E. Clark, and W. Fazar, "Application of a Technique for Research and Development Program Evaluation", *Operations Research*, vol. 7, 1959, pp. 646-669.
11. Martin, J. J., "Distribution of the Time Through a Directed, Acyclic Network", *J. of the Operations Res. Society of America*, vol. 13, 1965, pp. 46-66.
12. Mirchandani, Pitu B., "Shortest Distance and Reliability of Probabilistic Networks", *Computers & Ops. Res.*, vol. 3, 1976, pp. 347-355.
13. Shogan, A. W., "Bounding Distributions for a Stochastic PERT Network", *Networks*, vol. 7, 1977, pp. 359-381.
14. Shogan, Andrew W., "A Recursive Algorithm for Bounding Network Reliability", *IEEE Trans. on Reliability*, vol. r-26, 1977, pp. 322-327.
15. Sigal, C. Elliott, A. Alan B. Pritsker, and James J. Solberg, "The Stochastic Shortest Route Problem", *Operations Research*, vol. 28, 1980, pp. 1122-1129.
16. Tenenbaum, Aaron M., Moshe J. Augenstein, Data Structures Using Pascal, Prentice-Hall, Inc., 1981, p. 473.

VITA

Kelly Jeanne Hayhurst

The author was born in Arlington, Virginia, August 19, 1962. She graduated from South Charleston Senior High School in South Charleston, West Virginia, in June 1980, and received her Bachelor of Science degree in Mathematics from Virginia Polytechnic Institute and State University in June 1985. The author has been employed by NASA Langley Research Center in Hampton, Virginia as a cooperative education student from January 1982 to June 1985 and as a full-time aerospace technologist from July 1985 to the present. In August 1986, the author entered the College of William and Mary as a graduate student in the Department of Mathematics.