ALGEBRAIC METHODS APPLIED TO NETWORK RELIABILITY PROBLEMS

D. R. Shier
D. E. Whited

Department of Mathematical Sciences
Clemson University

Technical Report #486

August, 1985
September, 1986 (Revised)

This research was supported by the U.S. Air Force Office of Scientific Research (AFSC) under Grant AFOSR-84-0154
An algebraic structure underlying network reliability problems is presented for determining the 2-terminal reliability of directed networks. An iterative algorithm is derived from this algebraic perspective to solve the (s,j)-terminal reliability problem simultaneously for all nodes j. In addition to providing an exact answer (in the form of a reliability polynomial), the algorithm also yields a nondecreasing sequence of approximate solutions guaranteed to be lower bounds on the exact solution. Empirical results, presented for two different implementations of the algorithm, show that useful approximate solutions can be obtained in a reasonable amount of computation time.
Abstract. An algebraic structure underlying network reliability problems is presented for determining the 2-terminal reliability of directed networks. An iterative algorithm is derived from this algebraic perspective to solve the (s, j)-terminal reliability problem simultaneously for all nodes j. In addition to providing an exact answer (in the form of a reliability polynomial), the algorithm also yields a nondecreasing sequence of approximate solutions guaranteed to be lower bounds on the exact solution. Empirical results, presented for two different implementations of the algorithm, show that useful approximate solutions can be obtained in a reasonable amount of computation time.

Key words. algorithm, directed graphs, reliability

AMS (MOS) subject classifications: 05C20, 62N05, 94C15
1. Introduction

The problem of determining the reliability of an existing or proposed communication system has received considerable attention in the engineering, statistical, and operations research literature \[1, 5, 9\]. For example, it is important to assess the probability that a message sent from a given source arrives at its destination, when the components comprising the system are subject to failure. Unfortunately, most reliability problems of any substance are now known to be NP-hard or \#P-complete \[4, 17, 18, 27\]. As a result, researchers have focused on special network structures (where polynomial-time algorithms are possible), or have resorted to simulation.

A number of special classes of undirected networks have recently been analyzed with success. Polynomial-time algorithms are now available for calculating certain reliability measures in series-parallel \[21\], inner-cycle-free \[14\], inner-four-cycle-free \[14\], and cube-free \[15\] planar graphs. Provan \[16\] has shown, however, that the problem of determining source-to-terminal reliability remains \#P-complete for the general class of planar graphs. In order to analyze more complex network topologies, the idea of pivotal decomposition \[5\] together with polygon-to-chain reductions \[28\] can be used to decompose the original problem into smaller subproblems.

Similar results and tools are not as available in the case of directed networks. The only significant types of directed networks that are known to admit a polynomial-time algorithm are "basically-series-parallel" networks \[2, 3\]. Also, unlike the case for undirected networks, certain simplifications are not available when carrying out pivotal decomposition in the directed case \[1\]. Nor does there exist an "optimal" factoring algorithm, such as that demonstrated for undirected networks \[19\].

This paper exploits the underlying algebraic structure of network reliability problems to produce a general iterative algorithm, applicable to arbitrary directed networks. While not polynomially-bounded, it is able to generate reasonable
approximations to exact network reliability with a modest amount of computation.

2. Algebraic Structure

Suppose that \( G = (N, E) \) is a directed network with node set \( N \) and edge set \( E \), in which nodes do not fail but edges fail independently of one another. The reliability of edge \( e \) (the probability that edge \( e \) functions) is denoted by \( p_e \). Nodes \( s \) and \( t \) designate the specified source and terminal of \( G \), and we are interested in calculating the 2-terminal reliability \( R_{st}(G) \):

\[
R_{st}(G) = \Pr \{\text{there exists a functioning path from } s \text{ to } t \text{ in } G\}.
\]

Associate with each edge \( k \in E \) a variable \( x_k \). Then the reliability polynomial \( F_{st}(x) = F_{st}(x_1, \ldots, x_r) \) associated with \( s \) and \( t \) is a polynomial in \( x_1, \ldots, x_r \) such that if the numerical values \( p_1, \ldots, p_r \) are substituted for the corresponding variables \( x_1, \ldots, x_r \) then the resulting value is the probability that a functioning path exists from node \( s \) to node \( t \). (If the \( x_i \)'s were simply Boolean variables, this polynomial would be identical with the structure function of the system [5].) The reliability polynomial can be concisely expressed using two operations \( \oplus \) and \( \otimes \) defined on polynomials.

To begin, let

\[
T^a = x_1^{a_1} x_2^{a_2} \ldots x_r^{a_r}
\]

denote a monomial term, where each \( a_i \in \{0,1\} \). The operation \( \otimes \) when applied to terms \( T^a \) and \( T^b \) yields the term \( T^c \) where \( c_i = \max \{a_i, b_i\} \). This operation is extended to arbitrary polynomials by distributivity. The operation \( \oplus \) is defined on polynomials \( f(x) \) and \( g(x) \) using

\[
f(x) \oplus g(x) = f(x) + g(x) - f(x) \otimes g(x).
\]

Operations related to \( \oplus \) and \( \otimes \) were apparently first suggested by Mine [13] and by
Kim et al. [10]. More recently Gondran and Minoux [8], and Shier [24], have formulated
network reliability using the operations $\oplus$ and $\ominus$ defined above.

Let $S$ denote the set of all polynomials that can be formed from monomial terms $T^a, T^b, \ldots$ by finite applications of the operations $\oplus$ and $\ominus$. Then it can be demonstrated [24] that $(S, \oplus, \ominus)$ forms a distributive lattice with smallest element 0 (the zero polynomial) and largest element 1 (the unit polynomial). Suppose that $P_{st}$ is the set of simple paths from node $s$ to node $t$ in $G$. Define the value $v(P)$ of path $P$ to be the product, with respect to $\ominus$, of the edge variables along the path:

$$v(P) = \ominus \prod \{x_k : k \in P\}.$$  

Then the reliability polynomial $F_{st}(x)$ can be expressed as

$$F_{st}(x) = \oplus \sum \{v(P) : P \in P_{st}\}.$$  

As an illustration, consider the standard bridge network in Fig. 1 having $s = 1$ and $t = 4$. Since there are four simple paths extending from $s$ to $t$, equation (1) becomes

$$F_{st}(x) = x_1 x_4 \oplus x_1 x_3 x_5 \oplus x_2 x_4 x_6 \oplus x_2 x_5.$$  

Equation (1) is just the standard expression for the inclusion-exclusion formula, applied to paths in the network [1]. Expanding such an expression, using the definitions of $\oplus$ and $\ominus$, and then substituting numerical values $p_k$ for the corresponding variables $x_k$ yields $R_{st}(G)$.

A number of techniques have been developed to calculate the quantity (1) for a general algebraic structure $(S, \oplus, \ominus)$ satisfying appropriate properties [6, 8]. These techniques can all be viewed as different methods of solving the system of equations

$$z = z \ M \oplus e_s,$$  

which is linear in the operations $\oplus$ and $\ominus$. Here $M = (m_{ij})$ is the weighted adjacency matrix for $G$, with $m_{ij} = x_k$ for $k = (i, j) \in E$ and $m_{ij} = 0$ otherwise. Also, $e_s$ denotes the $s$-th unit row vector. An (extremal) solution $z$ to these equations is known [6] to satisfy
Thus by solving such equations to find $F_{st}(x)$, and hence $R_{st}(G)$, we also obtain the $(s, j)$-terminal reliabilities for all $j \in N$. Moreover, unlike existing methods for calculating 2-terminal network reliability based on paths [1, 11], such algebraic methods do not need to first enumerate all simple paths joining the two terminals [6, 8]. These paths are automatically generated in the course of solving the set of equations (2).

A natural way of solving (2) is by means of an iterative procedure, whereby the current estimate for $z$ is substituted into the right-hand-side of (2), producing a new estimate for the solution vector $z$. In the next section, we discuss a specific iterative scheme for solving (2) that incorporates special data structures to streamline such computations.

3. An Iterative Scheme

The basic idea of the iterative scheme presented here is that of passing on, at each step, the information available at node $i$ to each of its neighbors $j$, where $(i, j) \in E$. Before stating the general iterative scheme, the ideas will first be illustrated using the network in Fig. 1. We will find all reliabilities $z_j = F_{sj}(x)$ relative to the source node $s = 1$.

In the algorithm, a polynomial label is associated with each node $j$. At any stage, $\text{LABEL}(j)$ will be a reliability polynomial based on a certain subset of paths from node $s$ to node $j$. In this sense, $\text{LABEL}(j)$ corresponds to a current estimate of the solution $z_j$ to equation (2). Initially, if there is an edge $k = (s, j) \in E$ then $\text{LABEL}(j) = x_k$. If there is no such edge then $\text{LABEL}(j) = 0$; in the case of the source node, $\text{LABEL}(s) = 1$. Those nodes, apart from $s$, receiving a nonzero initial label are placed on a list $L$. In this example, we have
Now we remove the "top" node $i$ from $L$ and update its neighbors $j$ using

$$\text{LABEL}(j) := \text{LABEL}(j) \oplus [\text{LABEL}(i) \otimes x_k],$$

where $k = (i, j)$. The above $(i, j)$ update simply incorporates into $\text{LABEL}(j)$ new paths from $s$ to $j$ that use the edge $(i, j)$. Any node $j$ whose label is changed by (3) is placed on $L$ if it does not already appear. This step removes, in our example, $i = 3$ and updates

$\text{LABEL}(2) = x_1 + x_2 x_6 - x_1 x_2 x_6$

$\text{LABEL}(4) = x_2 x_5$

$L = [2, 4]$

The corresponding network, with node labels attached, is shown in Fig. 2.

At the next step, node $i = 2$ is removed from the top of $L$. Nodes 3 and 4 are then updated, and node 3 is added to $L$:

$\text{LABEL}(4) = x_2 x_5 + x_1 x_4 - x_1 x_2 x_4 x_5 + x_2 x_4 x_6 - x_2 x_4 x_5 x_6$

$\quad - x_1 x_2 x_4 x_6 + x_1 x_2 x_4 x_5 x_6$

$\text{LABEL}(3) = x_2 + x_1 x_3 - x_1 x_2 x_3 + x_2 x_3 x_6 - x_2 x_3 x_6$

$\quad - x_1 x_2 x_3 x_6 + x_1 x_2 x_3 x_6$

$\quad = x_2 + x_1 x_3 - x_1 x_2 x_3$

$L = [4, 3]$

This process is continued until $L$ becomes empty. At this point, the polynomial label on any node $j$ represents $z_j = F_{s_j}(x)$. Table 1 shows the final labels for our example, together with the value obtained by substituting the common edge reliability $p$ for all $x_k$. 

Table 1

<table>
<thead>
<tr>
<th>$F_{sj}(x)$</th>
<th>$R_{sj}(G), \ p_k = p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$j = 1$</td>
<td>1</td>
</tr>
<tr>
<td>$j = 2$</td>
<td>$x_1 + x_2x_6 - x_1x_2x_6$</td>
</tr>
<tr>
<td>$j = 3$</td>
<td>$x_2 + x_1x_3 - x_1x_2x_3$</td>
</tr>
<tr>
<td>$j = 4$</td>
<td>$x_2x_5 - x_1x_2x_4x_6 + x_1x_2x_4x_5x_6$</td>
</tr>
</tbody>
</table>

The general form of the iterative procedure is specified by the following algorithm, where $L$ again represents the list of nodes whose labels have been changed.

1. [Initialization]
   
   for $j \neq s$ do
     
     if $k = (s, j) \in E$ then $\text{LABEL}(j) := x_k$
     else $\text{LABEL}(j) := 0$;
   
   $\text{LABEL}(s) := 1$;
   
   $L := [j: (s, j) \in E]$;
2. [Iterative Step]

while L ≠ [ ] do
    remove i from L;
    for k = (i, j) ∈ E do
        T := LABEL(j) ⊕ [LABEL(i) ⊗ x_k];
        if T ≠ LABEL(j) then
            LABEL(j) := T;
            if j ∈ L then enter j into L.

Upon termination of the algorithm, LABEL(j) will be the required reliability polynomial $F_{S_j}(x)$. Notice that there are several ways of managing the list L. In our example, we treated L as a queue, whereby nodes are processed in a FIFO (first-in-first-out) manner. It is also possible to treat L as a stack, whereby nodes are processed in a LIFO (last-in-first-out) manner. The effect of these two ways of managing L will be examined in Section 5. First, we discuss a number of useful properties of this iterative algorithm.

4. Properties

In this section we make use of the algebraic properties of $(S, ⊕, ⊗)$ to establish certain properties of the iterative algorithm presented in Section 3. It will be convenient to denote the variable attached to edge (i, j) by x. Also, the label on node j at the start of step m will be denoted by $L_m(j)$. Then the (i, j) update (3) of node j after step m is expressed as

$$L_{m+1}(j) = L_m(j) ⊕ x L_m(i).$$

Because the label on node j represents the sum with respect to $⊕$ of a set of simple
s-j paths and because this set of paths can expand through subsequent updates (4), we have

**Property 1.** If \( k \leq m \) then \( L_k(j) \oplus L_m(j) = L_m(j) \).

One important simplification derives from the following property. It states that only the "new" information \( N(i) \) added to the label of \( i \) since \( i \) was last on \( L \) needs to be propagated to its neighbors \( j \).

**Property 2.** Suppose that at step \( m \) an \((i, j)\) update is to be performed, where the labels on \( i \) and \( j \) are \( L_m(i) = L_k(i) \oplus N(i) \) and \( L_m(j) \) with \( k < m \). Step \( k \) represents the step at which an \((i, j)\) update previously occurred. Then at step \( m+1 \) the new label assigned to \( j \) will be \( L_{m+1}(j) = L_m(j) \oplus x N(i) \).

Proof. At step \( k \), node \( j \) receives the label \( L_{k+1}(j) = L_k(j) \oplus x L_k(i) \). Also, since \( k < m \) we have \( L_{k+1}(i) \oplus L_m(i) = L_m(j) \), by Property 1. Then

\[
L_{m+1}(i) = L_m(i) \oplus x L_m(i)
= [L_m(i) \oplus L_{k+1}(i)] \oplus x [L_k(i) \oplus N(i)]
= [L_m(i) \oplus L_k(i) \oplus x L_k(i)] \oplus x L_k(i) \oplus x N(i)
= [L_m(i) \oplus L_k(i) \oplus x L_k(i)] \oplus x N(i)
= [L_m(i) \oplus L_{k+1}(i)] \oplus x N(i)
= L_m(j) \oplus x N(i). \quad \star
\]

Because the labels on each node will be maintained as fully expanded polynomials (expressed using ordinary + and \( \times \)), it is desirable to know when certain of the terms in \( L_m(i) \) do not affect the label \( L_m(j) \). The following property provides one such condition.

**Property 3.** Suppose that at step \( m \) an \((i, j)\) update is to be performed with \( L_m(i) = A_1 + A_2 + \ldots + A_v \), \( L_m(j) = B_1 \oplus B_2 \oplus \ldots \oplus B_w \) and \( B_1 \subseteq xA_1 \). Then the updated label \( L_{m+1}(j) = L_m(j) \oplus x [A_2 + \ldots + A_v] \).

Proof. Let \( A = A_2 + \ldots + A_v \) and \( B = B_2 \oplus \ldots \oplus B_w \). Then
\[ L_{m+1}(j) = L_m(j) \oplus L_m(i) \]

\[ = [B_1 \oplus B] \oplus [A_1 + A] \]

\[ = [B_1 \oplus B] + [xA_1 + xA] - [xA_1 + xA] [B_1 \oplus B] \]

\[ = [B_1 \oplus B] + xA_1 + xA - xA_1 [B_1 + B + B_1 B] - xA [B_1 \oplus B] \]

\[ = [B_1 \oplus B] + xA_1 + xA - xA_1 - xA_1 B + xA_1 B - xA [B_1 \oplus B] \]

\[ = L_m(j) + xA - xA L_m(j) \]

\[ = L_m(j) \oplus x [A_2 + \ldots + A_v] . \]

Together, Properties 2 and 3 show that certain "cancellations" in the update step (4) of the iterative scheme can be predicted in advance, and thus unnecessary computation can be avoided. The next property demonstrates that the approximations to \( R_{Sj}(G) \), derived from successive labels at node \( j \), are monotone nondecreasing. The notation \( R_m(j) \) indicates the value obtained by substituting numerical values \( p_r \) for \( x_r \) into the polynomial \( L_m(j) \).

**Property 4.** If \( k < m \) then \( R_k(j) \leq R_m(j) \).

Proof. Since \( k < m \) we can express \( L_k(j) = T_1 \oplus T_2 \oplus \ldots \oplus T_v \) and \( L_m(j) = T_1 \oplus T_2 \oplus \ldots \oplus T_w \), where \( T_i \) is a monomial term representing some path \( P_i \) from \( s \) to \( j \) and \( v \leq w \). Then \( R_m(j) \) represents the probability that at least one path of \( \{P_1, P_2, \ldots, P_v\} \) is functioning and so is at least as large as the probability \( R_k(j) \) that at least one path of \( \{P_1, P_2, \ldots, P_v\} \subseteq \{P_1, P_2, \ldots, P_w\} \) is functioning. ♦

5. Computational Results

Several examples will be given in this section to illustrate the efficacy of a version of the iterative algorithm that makes use of Properties 2 and 3. The quality of the nondecreasing sequence of approximations to \( R_{Sj}(G) \) will also be examined, in particular as this relates to the discipline (FIFO, LIFO) used for managing the list \( L \). The iterative
algorithm was coded in FORTRAN 77 and all computations were performed using the IBM 3081 computer at Clemson University.

**Example 1.** This network, having 9 nodes and 19 edges, is taken from [20] and is shown in Fig. 3. There are 35 s-t paths and 5287 noncancelling terms in \( F_{st}(x) \). As discussed in Satyanarayana and Prabhakar [20], each noncancelling term corresponds to an "acyclic subgraph" of \( G \). Despite its small size, this example represents one of the most complex directed networks whose exact reliability has been reported in the literature.

The reliabilities \( R_{sj}(G) \) have been calculated using our iterative procedure and the FIFO/LIFO disciplines. For ease of presentation, the reliability polynomial \( F_{st}(x) \) has been evaluated with all \( p_k = p \) for the particular \((s, t)\) pair indicated in Fig. 3; all edge failures are assumed to be independent. Fig. 4 shows \( F(p) = F_{st}(p, \ldots, p) \) plotted versus \( p \) using the FIFO discipline. As expected, the various iterations produce an increasing sequence \( F_1, \ldots, F_9 \) of reliability curves that converge to the exact answer in 9 iterations. Each iteration produces a lower bound on \( R_{st}(G) \) and thus provides a conservative estimate for the true network reliability. Namely, the exact \((s, t)\)-reliability of the network is guaranteed to be at least as large as the value specified by the approximation. Notice that the curves for the fifth through ninth iterations overlap in the figure, thus providing excellent approximations to \( R_{st}(G) \). Also indicated on Fig. 4 are the cumulative CPU times (in seconds) required to complete the work through the end of the specified iteration. Thus, a total of 0.638 seconds were needed to obtain \( R_{st}(G) \), whereas only 0.061 seconds were needed to obtain an approximation that is virtually indistinguishable over the entire range \( 0 \leq p \leq 1 \).

Fig. 5 shows analogous information relative to the LIFO discipline. In this case, twelve iterations were required before convergence was obtained. (Several of the curves overlap so only 10 approximations are apparent in the figure.) Although the exact answer was obtained in 0.454 seconds (less than the comparable time for FIFO), the
LIFO discipline did not give as useful a set of approximations compared to the FIFO approach.

Example 2. This network, with 13 nodes and 27 edges, is derived from an example given by Martelli [12]; see Fig. 6. It is considerably more complex than Example 1, having 70 s-t paths and 34,983 noncancelling terms. Plots of $F(p)$ versus $p$ are shown in Figs. 7 and 8 for the FIFO and LIFO disciplines, respectively. Again it is observed that the LIFO method obtains the exact answer faster than the FIFO method. However, the quality of approximations produced by FIFO is superior to those produced by LIFO. Indeed, a very close approximation to the exact reliability polynomial is obtained by FIFO in 1.18 seconds, one-eighth of the time required to find the exact answer using FIFO and one-sixth of that required using LIFO.

Finally, five random networks on 12 nodes and 30 edges were generated for test purposes. The characteristics of these networks, together with the number of iterations required for convergence, are shown in Table 2. In order to compare the quality of the approximations generated for these examples, we have tabulated the CPU time (in seconds) required to achieve a relative error of $\alpha\%$ or less (at $p = 0.5$) in Table 3. The results for Examples 1 and 2 are also included.

In these random examples the FIFO and LIFO disciplines appear to be comparable in terms of the time required to obtain the exact answer. Again, however, the FIFO variant gives a fairly close approximation rather quickly and it completely dominates the LIFO variant in this respect.
Table 2

<table>
<thead>
<tr>
<th>Network</th>
<th>#s-t Paths</th>
<th>#Noncancelling Terms</th>
<th>#Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>14</td>
<td>1,263</td>
<td>9 7</td>
</tr>
<tr>
<td>R2</td>
<td>28</td>
<td>3,383</td>
<td>11 8</td>
</tr>
<tr>
<td>R3</td>
<td>41</td>
<td>7,583</td>
<td>8 10</td>
</tr>
<tr>
<td>R4</td>
<td>44</td>
<td>17,919</td>
<td>5 5</td>
</tr>
<tr>
<td>R5</td>
<td>34</td>
<td>42,687</td>
<td>10 8</td>
</tr>
<tr>
<td>Example</td>
<td>Discipline</td>
<td>CPU (secs) for accuracy within α%</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0%</td>
<td>1%</td>
</tr>
<tr>
<td>1</td>
<td>FIFO</td>
<td>.638</td>
<td>.248</td>
</tr>
<tr>
<td></td>
<td>LIFO</td>
<td>.454</td>
<td>.454</td>
</tr>
<tr>
<td>2</td>
<td>FIFO</td>
<td>9.23</td>
<td>4.26</td>
</tr>
<tr>
<td></td>
<td>LIFO</td>
<td>6.92</td>
<td>6.92</td>
</tr>
<tr>
<td>R1</td>
<td>FIFO</td>
<td>.049</td>
<td>.001</td>
</tr>
<tr>
<td></td>
<td>LIFO</td>
<td>.054</td>
<td>.054</td>
</tr>
<tr>
<td>R2</td>
<td>FIFO</td>
<td>.380</td>
<td>.076</td>
</tr>
<tr>
<td></td>
<td>LIFO</td>
<td>.362</td>
<td>.362</td>
</tr>
<tr>
<td>R3</td>
<td>FIFO</td>
<td>1.21</td>
<td>.172</td>
</tr>
<tr>
<td></td>
<td>LIFO</td>
<td>1.11</td>
<td>.630</td>
</tr>
<tr>
<td>R4</td>
<td>FIFO</td>
<td>3.06</td>
<td>.195</td>
</tr>
<tr>
<td></td>
<td>LIFO</td>
<td>2.85</td>
<td>2.85</td>
</tr>
<tr>
<td>R5</td>
<td>FIFO</td>
<td>5.07</td>
<td>.062</td>
</tr>
<tr>
<td></td>
<td>LIFO</td>
<td>5.11</td>
<td>1.58</td>
</tr>
</tbody>
</table>
6. Conclusions

This paper has explored an algebraic structure underlying certain network reliability problems. A promising iterative algorithm has been developed that allows both exact and approximate answers to be obtained. Rather than giving simply a single number, this algorithm produces a reliability polynomial that can then be easily evaluated at any particular input values $p_1, ..., p_r$. Also, in the process of determining $R_{st}(G)$ we also generate $R_{sj}(G)$ for all $j \in N$.

Empirical results have shown that the choice of data structure (FIFO, LIFO) can have a significant effect on the relative efficiency of the procedures as well as on the quality of the approximations. Whereas the LIFO approach frequently obtains the exact reliability polynomial faster than the FIFO approach, the latter produces better approximations -- ones that are quite close to the exact answer but are obtained in a fraction of the time. This desirable feature of the FIFO approach can be explained as follows, assuming that the $p_k$ are comparable in value. Under a FIFO discipline, nodes are processed in order of increasing distance from $s$. Thus, the first time node $j$ is labelled, it is done so relative to a path with the minimum number of edges. More generally, the FIFO approach ensures that the "more probable" (fewer edge) paths are incorporated as soon as possible. Subsequent (longer and less probable) paths contribute, but not as much, to the final label on node $j$. On the other hand, a LIFO discipline creates a depth-first rather than a breadth-first search of the network, and thus "early" approximations can be substantially improved by the incorporation of later (shorter) paths.

The approximate solutions generated by the iterative algorithm will always produce (conservative) lower bounds on the exact solution. If greater accuracy is required, such lower bounds can be used together with a simulation approach, such as Fishman's sampling procedure [7], that makes explicit use of lower bounds to obtain improved estimates. Alternatively, these lower bounds can be used in conjunction with existing
techniques that produce upper bounds on network reliability [22, 25] to obtain an interval that must enclose $R_{st}(G)$.

Finally, it should be emphasized that regardless of the list discipline used, some relatively challenging directed networks from the literature can be solved by our algorithm with a modest amount of computation. In particular, one example studied had 70 paths. We are not aware of any existing algorithm that has solved exactly a problem of this complexity. While the proposed approach appears to have potential, further experimentation will be necessary before any firm conclusions can be drawn concerning its general applicability. In order to solve larger, more realistic problems it may be possible to combine this approach with methods for decomposing the network into more manageable portions [23,26].

Acknowledgments

This paper benefited greatly from the careful scrutiny of the referees. This research was supported by the U. S. Air Force Office of Scientific Research (AFSC) under Grant AFOSR-84-0154. Reproduction in whole or in part is permitted for any purpose of the United States Government.

REFERENCES


END

FILMED

MARCH, 1988

DTIC