#157

A NEW ALGORITHM FOR NETWORK RELIABILITY COMPUTATION*

A. Grnarov[†], L. Kleinrock[††] and M. Gerla[††]

[†]Electrical Engineering Department
University of Skopje, Yugoslavia

[††]UCLA Computer Science Department
Los Angeles, California 90024

I. Introduction

The analysis of reliability networks has been the subject of considerable research. The terminal reliability algorithms available in the literature are based on state enumeration,[1–4] factoring,[5] reduction to series-parallel network,[6,7] path enumeration,[8–20] cut-set enumeration,[21,22] and case analysis[23]. Recently Satyanarayana and Prabhakar[24] proposed an efficient algorithm based on the acyclic subgraphs of the given probabilistic graph.

The most efficent path enumeration algorithms use reduction to mutually exclusive events by Boolean algebra. The drawbacks of these algorithms, however, are the fact that they generate a large number of terms,[10,20] they can efficiently handle systems of only moderate size (say, fewer than 20 paths or cut-sets between the input-output node pair),[11] and they do not permit an efficient determination of the resulting function when the number of elements in the network is large.[15]

In this paper a new algorithm for terminal reliability computation of a general network is presented. The algorithm belongs to the path enumeration algorithms (which use Boolean algebra) and is based on performing the here defined \$-operation (modified #-operation)[25] on the set of all simple paths. A simple path is represented by a particular binary string (path identifier); thus,only logical operations need to be performed. The proposed algorithm is more efficient than existing algorithms since it does not suffer from the previously mentioned drawbacks. It can be applied to both oriented and non-oriented networks and easily modified to produce symbolic expression for terminal reliability.

The algorithm was coded in FORTRAN IV and run on a DEC-10 timesharing computer system. Execution times confirm the advantages of the proposed algorithm over existing algorithms.

II. Derivation of the Algorithm

For a network consisting of N nodes and E links, we consider the set of paths between a given source s and a given destination t. Each path in the set is identified by a path identifier defined as follows:

Definition 1: The path identifier IP_k for the path π_k is defined as a string of n binary variables

$$IP_k = x_1 x_2 \cdots x_i \cdots x_n$$

where

	$x_i = 1$	if the i^{th} element of the network (node or link) is included in the path π_k
	$x_i = x$	otherwise
and n = number of network elements that can fail, i.e.:		
	$n = N$	in the case of perfect links and imperfect nodes
	$n = E$	in the case of perfect nodes and imperfect links
	$n = N + E$	in the case of imperfect links and imperfect nodes

According to *Definition 1*, it can be seen that a path identifier has the form of the cube in Boolean algebra. Hence, by applying the #-operation ("sharp" operation) between two path identifiers, denoted by $IP_k \# IP_j$, we obtain the set of all subcubes (events) of IP_k not included in IP_j. Unfortunately, the generated subcubes are not mutually disjoint. The #-operation should be repeated on the set of subcubes again and again until a set of disjoint subcubes is obtained.

In the case when the paths π_j, $j = 1, 2, \cdots, m-1$, have been examined, the determination of the set of subcubes of the path π_m, which are not included in the previous paths, can be performed as

$$S_m = (\ldots((IP_m \# IP_1) \# IP_2) \# \ldots \# IP_j) \# \ldots) \# IP_{m-1} \quad (1)$$

Since the #-operation is performed using the largest possible cubes (path identifiers), the generation of S_m is faster than with Fratta's[10] and Rai's[15] methods, and also there is no need for the storing of the new terms. The drawbacks of performing (1) are the generation of the repeated subcubes and the need for the repeated application of the #-operation between the subcubes in S_m. To avoid these drawbacks a new \$-operation, called "exclusive sharp" operation, will be introduced later in *Definition 4*.

Before the presentation of *Definition 4* it is useful to make the following consideration. Since the path identifiers contain only symbols 1 and *x*, the #-operation by definition (see Table 1 below) can produce only 0 as a new symbol.

*This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract MDA 903-77-0272.

The symbols 0 generated in step j are designated as 0^j. If in the cube generated by #-operation there is only one 0^j, we call this a unique 0. Now we quote the definition of the coordinate #-operation between two cubes as given in Miller[25] (page 173), and we also present the definitions of the $@_j$-operation and the \$-operation.

Definition 2. The coordinate #-operation is defined as given in Table 1.

Table 1

$a_i \backslash b_i$	0	1	x
0	z	y	z
1	y	z	z
x	1	0	z

Definition 3. The $@_j$-operation between two cubes, say $C^r = a_1a_2 \cdots a_i \cdots a_n$ and $C^s = b_1b_2 \cdots b_i \cdots b_n$, is defined as

$$C^r \,@_j\, C^s = \begin{cases} C^r & \text{if } a_i\#b_i = y \text{ for any unique 0 or} \\ & \text{if } a_i\#b_i = y \text{ for all } a_i = 0^v \text{ for any v} \\ C_1^r \,\#\, C^s \cup C_0^r & \text{otherwise} \end{cases}$$

where

C_1^r is a cube obtained from C^r by setting to 1 all $a_i = 0^j$, for which $a_i\#b_i = y$,

and

C_0^r is a cube obtained from C^r by setting to x all $a_i = 0^j$, for which $a_i\#b_i \neq y$.

Definition 4. The \$-operation between two cubes C^r and C^s is defined as

$$C^r \,\$\, C^s = \begin{cases} \Phi & \text{if } a_i\#b_i = z \text{ for all i} \\ S_k & \text{if } a_i\#b_i = y \text{ for any i} \\ C' & \text{otherwise} \end{cases}$$

where Φ is the empty set, $S_j = S_{j-1} \,@_j\, C^s$, $S_1 = C' \,@_1\, C^s$ and $j = 2, 3, \cdots k$ (k is the number of steps in which any symbol(s) 0 is generated) while $C' = c_1c_2 \cdots c_i \cdots c_n$ and $c_i = a_i\#b_i$.

According to *Definition 4* it follows that if we substitute the #-operation in (1) by the \$-operation, the set S_m, consisting of disjoint cubes only, will be generated. The probability of the event, corresponding to a cube C in S_m, can be calculated as

$$P(C) = P \cdot Q \prod_{j=2}^{k} (1 - P_j) \tag{2}$$

where

$P = \Pi\, p_i$	for all i satisfying $c_i = 1$
$Q = \Pi\, q_i$	for all i satisfying $c_i =$ unique 0
$P_j = \Pi\, p_i$	for all i satisfying $c_i = 0^j$
$q_i = 1 - p_i$	

and p_i is the probability that the ith element is up.

Now we can propose the algorithm PROB for terminal reliability computation:

ALGORITHM PROB:

```
step 1.          Find path identifiers for all simple paths
                 between node s and node t

step 2.          Sort them by increasing weight (where
                 weight = number of 1's in the path
                 identifier string)

step 3.          Set $P_{st} = P(IP_1)$

step 4. (loop)   For $m = 2, 3, \cdots, k$ determine
                 $S_m = (\cdots((IP_m \,\$\, IP_1)\$IP_2 \cdots)\$IP_{m-1}$
                 $P_{st} = P_{st} + P(S_m)$

step 5.          END
```

In the algorithm, $P(S_m)$ is the sum of the probabilities of all subcubes in S_m.

As an example, the PROB algorithm is applied to the simple bridge network given in Figure 1. (Figure 2 in Lin[11] and Figure 1 in Fratta[10].)

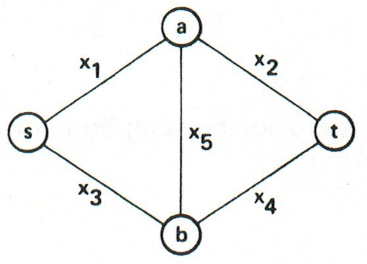

Figure 1. A Bridge Network

steps 1,2. Table 2 presents the set of all simple paths and their path identifiers (for simplicity we consider the case with perfect nodes and imperfect links).

Table 2

m	π_m	IP_m
1	x_1x_2	11xxx
2	x_3x_4	xx11x
3	$x_1x_5x_4$	1xx11
4	$x_3x_5x_2$	x11x1

step 3. $P_{st} = [p_1p_2]$

step 4. $m = 2$:

$$\begin{array}{l} \text{xx11x} \\ \underline{\text{11xxx}} \\ S_2 = \text{0011x} \end{array} \qquad j = 1$$

$P_{st} = P_{st} + [p_3p_4(1 - p_1p_2)]$

$$\begin{array}{lll} m = 3: & & \\ & 1xx11 & \\ & \underline{11xxx} & \\ & 10x11 & \text{unique } 0,\ j = 1 \\ & \underline{xx11x} & \\ S_3 = & 10011 & \text{unique } 0,\ j = 2 \end{array}$$

$$P_{st} = P_{st} + [p_1 p_4 p_5 q_2 q_3]$$

$$\begin{array}{lll} m = 4: & & \\ & x11x1 & \\ & \underline{11xxx} & \\ & 011x1 & \text{unique } 0,\ j = 1 \\ & \underline{xx11x} & \\ & 01101 & \text{unique } 0,\ j = 2 \\ & \underline{1xx11} & \\ S_4 = & 01101 & \end{array}$$

$$P_{st} = P_{st} + [p_2 p_3 p_5 q_1 q_4]$$

step 5. END

In the example [x] is used to denote the numerical value of x.

Since the proposed algorithm does not produce cancelling terms, the obtained result for P_{st} requires computation of 4 terms in comparison with 5 terms in Lin[11] and 6 terms in Fratta[10].

As a second example, for the network shown in Figure 2 (Figure 1 in Rai[15], Figure 10 in Lin[11] and Figure 3 in Satyanarayana[24]) the reliability P_{st} is obtained by computing 16 terms. The results in Rai[15], Lin[11] and Satyanarayana[24] require computation of 22, 61 and 123 terms respectively.

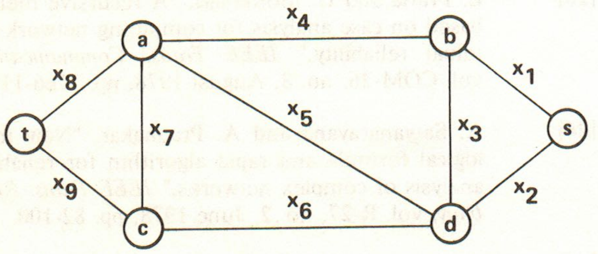

Figure 2. Modified Graph of ARPA Network

The third example, given in Figure 3, was also used in Fratta[10], Abraham[20] and Aggarwal[19].

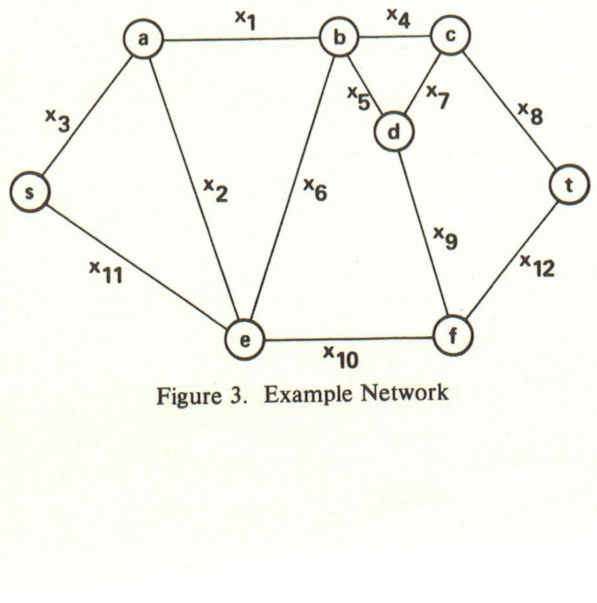

Figure 3. Example Network

The reliability evaluation requires the computation of 38 intermediate terms, while there are 72 terms in Abraham[20]. Furthermore, the algorithm PROB requires 34 comparisons for the determination of S_{23} while the algorithms reported in Abraham[20] and Aggarwal[19] require 54 and 173 comparisons respectively. For finding S_{24}, these numbers are 26, 45 and 159 respectively.

The algorithm was coded in FORTRAN IV and run on a DEC-10 computer system. The execution of algorithm PROB for a 9 node, 13 link example (given in Figure 7 in Lin[11]) took 0.76 seconds. The same example required 71 seconds execution time on a faster machine (namely, a CDC 6500) using the algorithm reported in Lin[11]. For the examples given in Figures 10 and 11 in Fratta[23] the computing times with PROB were 0.88 and 1.41 seconds. The execution of the algorithm in Fratta[23], which was coded in ALGOL W on an IBM 360/67 computer, took 4 and 15 seconds respectively.

III. Concluding Remarks

The paper presents a new algorithm for network terminal reliability. The proposed algorithm is based on the implementation of the introduced \$-operation on the set of the path identifiers only. By application of the \$-operation, only the disjoint subcubes are generated. The probability of the corresponding event can be calculated in a straightforward manner. The algorithm is efficient since it does not generate a large number of terms, and there is no need for the generation and storage of new terms other than the path identifiers. Since determination of the set S_m is easy, the algorithm can efficiently handle medium size networks. It can be applied to both oriented and non-oriented networks and can be easily modified to produce symbolic expressions for terminal reliability.

The comparison with existing algorithms on the basis of the number of terms, the number of operations and program execution time confirms the superiority of the proposed algorithm over path enumeration algorithms as well as algorithms employing other techniques.

REFERENCES

[1] M. L. Shooman, *Probabilistic Reliability, An Engineering Approach,* New York: McGraw-Hill, 1968.

[2] F. Henley, R. Williams, *Graph Theory in Modern Engineering,* New York: Academic Press, 1973.

[3] O. Wing, P. Demetriou, "Analysis of probabilistic networks," *IEEE Transactions on Communication Technology,* vol. COM-12, September 1970, p. 38-40.

[4] S. Arnborg, "Reduced state enumeration - Another algorithm for reliability evaluation," *IEEE Transactions on Communications,* vol 27, no. 2, June 1978, pp. 101-105.

[5] F. Moshowitz, "The analysis of redundancy networks," *AIEE Trans. Commun. Electr.*, vol. 77, November 1958, pp. 627-632.

[6] Y. Kim, K. Case and P. Ghare, "A method for computing complex system reliability," *IEEE Trans. Reliability*, vol. R-21, no. 4, November 1972, pp. 215-219.

[7] K. Misra, "An algorithm for the reliability evaluation of redundant networks," *IEEE Trans. Reliability*, vol. R-19, November 1970, pp. 146-151.

[8] C. Lee, "Analysis of switching networks," *Bell System Tech. J.*, November 1955, pp. 1287-1315.

[9] D. Brown, "A computerized algorithm for determining the reliability of redundant configurations," *IEEE Trans. Reliability*, vol. R-20, March 1971, pp. 121-124.

[10] L. Fratta, U. Montanari, "A Boolean algebra method for computing the terminal reliability in a communication network," *IEEE Trans. Circuit Theory*, vol. CT-20, May 1973, pp. 203-211.

[11] P. Lin, B. Leon, and T. Huang, "A new algorithm for symbolic system reliability analysis," *IEEE Trans. Reliability*, vol. R-25, no. 1, April 1976, pp. 2-14.

[12] L. Fratta, U. Montanari, "Synthesis of available networks," *IEEE Trans. Reliability*, vol. R-25, no. 2, June 1976, pp. 81-87.

[13] J. de Mercado, N. Spyratos and B. Bowen, "A method for calculation of network reliability," *IEEE Trans. Reliability*, vol. R-25, no. 2, June 1976, pp. 71-76.

[14] P. Canarda, F. Corsi and A. Trentadue, "An efficient simple algorithm for fault free automatic synthesis from the reliability graph," *IEEE Trans. Reliability*, vol. R-27, no. 3, August 1978, pp. 215-221.

[15] S. Rai and K. Aggarwal, "An efficient method for reliability evaluation of a general network," *IEEE Trans. Reliability*, vol. R-27, no. 3, August 1978, pp. 206-211.

[16] T. Case, "A reduction technique for obtaining a simplified reliability expression," *IEEE Trans. Reliability*, vol. R-26, no. 4, October 1977, pp. 248-249.

[17] H. Nakazawa, "A decomposition method for computing system reliability by a Boolean expression," *IEEE Trans. Reliability*, vol. R-26, no. 4, October 1977, pp. 250-252.

[18] K. Aggarwal, J. Gupta and K. Misra, "A simple method for reliability evaluation of a communication system," *IEEE Trans. Communications*, vol. COM-23, No. 5, May 1975, pp. 563-566.

[19] K. Aggarwal, K. Misra and J. Gupta, "A fast algorithm for reliability evaluation," *IEEE Trans. Reliability*, vol. R-24, April 19675, pp. 83-85.

[20] J. Abraham, "An improved algorithm for network reliability," *IEEE Trans. Reliability*, vol. R-28, no. 1, April 1979, pp. 58-61.

[21] P. Jensen and M. Bellmore, "An algorithm to determine the reliability of a complex system," *IEEE Trans. Reliability*, vol. R-18, November 1969, pp. 169-174.

[22] E. Hansler, G. McAulife and R. Wilkov, "Exact calculation of computer network terminal reliability," *Networks*, vol. 4, 1974, pp. 95-112.

[23] L. Fratta and U. Montanari, "A recursive method based on case analysis for computing network terminal reliability," *IEEE Trans. Communications*, vol. COM-26, no. 8, August 1978, pp. 1166-1177.

[24] A. Satyanarayana and A. Prabhakar, "New topological formula and rapid algorithm for reliability analysis of complex networks," *IEEE Trans. Reliability*, vol. R-27, no. 2, June 1978, pp. 82-100.

[25] R. Miller, *Switching Theory, Volume 1: Combinational Circuits*, New York, Wiley, 1965.