A Method for Computing the Network Reliability of a Computer Communication Network

○
Kyung-Jae Ha, Ssang-Hee Seo
Dept. of Computer Engineering, Kyungnam University

<ABSTRACT>

The network reliability is to be computed in terms of the terminal reliability. The computation of a termial reliability is started with a Boolean sum of products expression corresponding to simple paths of the pair of nodes. This expression is then transformed into another equivalent expression to be a Disjoint Sum of Products form. But this computaion of the terminal reliability obviously does not consider the communication between any other nodes but for the source and the sink. In this paper, we derive the overall network reliability which is the probability of communication that each node in the network communicates with all other remaining nodes. For this, we propose a method to make the SOP disjoint for deriving the netwok reliability expression from the system success expression using the modified Sheinman's method. Our method includes the concept of spanning trees to find the system success function by the Cartesian products of vertex cutsets.

1. Introduction

System reliability evaluation is basic step in a reliability studies. Therefore, derivation of the symbolic reliability expression in a simplified and compact form for a general system is very desirable. In system relaibility analysis, it is customary to represent the system by a probabilistic graph in which each node and each branch has a probability of being operative. A very common example of a general system is the computer communication networks such as the internet. The most common problem which arises in the analysis of such a network is to compute in a efficient and systematic manner the network reliability between each of the all node pairs.

In the graphical representation of a computer communication network(CCN), specific communication links and computers are represented as branches and nodes of the graph. The terminal reliability, a commonly used measure of connectivity, is the probability of obtaining service between a pair of operative centers, called souce and sink, in terms of reliability for each communication link/node in the network. This calculation obviously does not consider the communication between any other nodes but for the source and sink. Here, we find the probability of obtaining a situation in which each node in the network communicates with all other remaining communication nodes. In case that this probability

called Network Reliability of a CCN is to be calculated using the concepts of terminal reliability only, one can proceed by finding all possible paths between each of the n(n-1)/2 node pairs. Since this is impractical for graphs with a large number of nodes, alternative methods have been proposed in the literature.[1][2]

The Boolean algebra method is a power tool for reliability analysis because it has advantages over other techniques as summarized in [3]. Several algorithms for the generation of disjoint products are known but they do not obtain a minimal formula.

In the Boolean algebra method, following two factors determine the quality of algorithms for the generation of disjoint products[4];

- computational efficiency,
- resulting number of disjoint products.

In this paper, we propose an alternative method to obtain the SDP using the Sheinman's method which is used to simplify the Boolean function[5]. We modify the original algorithm in order to transform a canonical function to a SDP expression.

Our method includes the concept of spanning trees to find all possible paths between each of the all node pairs.[6]

2. Derivation of a system success function from a CCN

The first step in most reliability evaluation techniques is to enumerate all minimal paths between each of the all of node pairs in the probabilistic graph.

A following assumptions are made with respect to the probabilistic graph.

a. All nodes of graph are perfectly reliable.

b. Links are half- or full-duplex. The network is modeled by a undirected graph.

c. Link states are s-independent. Failure of one link does not affect the probability of failure of other links.

d. Each link and the network have only 2 states good, the operating state and bad, the failed state.

e. The network is free from self loops and cycles.

In any general network, the network reliability expression is usually derived from the graph of the network into two steps.

In the first step, the success paths or cutsets are enumerated and success expression is formed by taking union of them. In the next step, this success expression is modified to give an equivalent canonical form or sum of products expression in which all product terms are mutually disjoint.

In the first step, one can proceed by finding all possible paths between each of the n(n-1)/2 node pairs for computing the network reliability.[7] Since this is impractical for graphs with a large number of nodes, we use the concept of spanning trees in [6].

From the definition of a spanning tree, any T_i will link all n nodes of G with (n-1) branches and represents the minimum interconnections required for providing a communication between all nodes. For this, the backtrack programming method[8] or the elementary tree transformation[13] are used to find all the spanning trees of a graph. This approach is difficult to computerize, so we borrow another method in Ref.[6] which uses Cartesian products of (n-1) vertex cutsets C_i whose elements are the branches connected to any of the(n-1) nodes of G. Thus,

$$C = C_1 \times C_2 \times \cdots \times C_{n-1} = \overset{n-1}{\underset{i=1}{\times}} C_i \qquad \cdots\cdots(1)$$

Where C is a set of subgraph of G with (n-1) branches. It has been proved[9] that any circuit of G with (n-1) branches will have an even number of identical appearances in C. If these terms are recognized, then deleted from C, the normalized Cartesian product C^* contains only those subgraphs which do not repeat an even number of times and are of cardinality (n-1). From the concept of spanning tree, C^* is thus, the set of all T_i's of a connected graph G.

Finally, We can get a Boolean expression of the system success function by union operations on the elements in C^*.

<Example>

In order to enumerate the spanning trees for

a bridge network in Fig. 1, the three vertex cutsets can be obtaind. They are:

$C_1 = (x_1, x_2)$, $C_2 = (x_4, x_5)$, $C_3 = (x_1, x_3, x_4)$

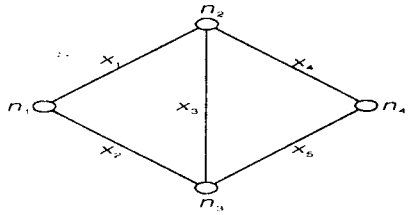

Fig.1 a network example

By using (1), C becomes as follow.

$$C = (x_1, x_2) \times (x_4, x_5) \times (x_1, x_3, x_4)$$
$$= (x_1x_4 , x_1x_5 , x_2x_4 , x_2x_5) \times (x_1, x_3, x_4)$$
$$= (x_1x_3x_4 , x_1x_3x_5 , x_1x_4x_5 , x_1x_2x_4 , x_2x_3x_4 , x_1x_2x_5 , x_2x_3x_5, x_2x_4x_5) \quad \text{--------(2)}$$

Since no term in (2) has an even number of identical appearances, C^* is the same as C. The 8 elements of set C^* thus represent 8 different spanning trees.

We can obtain the system success function:

$$S = x_1x_3x_4 \cup x_1x_3x_5 \cup x_1x_4x_5 \cup x_1x_2x_4 \cup x_2x_3x_4 \cup x_1x_2x_5 \cup x_2x_3x_5 \cup x_2x_4x_5 \quad \text{--------(3)}$$

3. Computation of the network reliability

3-1. The Boolean Algebra method

In Sec. 2, we first derived the Boolean sum of products expression for the system success function as a pure Boolean algebraic statement. If it is to be interpreted as a probability expression, certain modifications are necessary. The modifications are necessary because the following relation for expressing the probability of the union of n events is true only if the events are mutually exclusive

$$P_r(E_1 \cup E_2 \cup \dots \cup E_n) = P_r(E_1) + P_r(E_2) + \dots + P_r(E_n)$$

For example, consider a simple Boolean expression,

$$S = x_1x_2 + x_1x_3x_4$$

This function is represented on a Karnaugh map in Fig. 2. This map can be reinterpreted as a probability map where x_1, x_2, x_3, x_4 represent four primary events with individual probabilities of occurrence p_1, p_2, etc. and individual probabilities as q_1, q_2 etc.

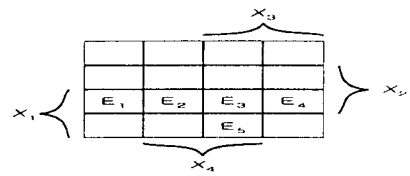

Fig.2 Karnaugh map for $S = x_1x_2 + x_1x_3x_4$

On this basis, therefore, the probability of event S is the algebraic sum of five elements defined by the five locations containing a 1 in the Karnaugh map, i.e.,

$$P_r(S) = P_r(E_1) + P_r(E_2) + P_r(E_3) + P_r(E_4) + P_r(E_5) \quad \text{--------(4)}$$

where,

$P_r(E_1) = p_1p_2q_3p_4$, $P_r(E_2) = p_1p_2q_3q_4$, $P_r(E_3) = p_1p_2p_3p_4$, $P_r(E_4) = p_1p_2p_3q_4$, $P_r(E_5) = p_1q_2p_3p_4$

This equation could have been obtained directly from the original Boolean expression by converting the same into its canonical form as:

$$S^* = x_1x_2 + x_1x_3x_4 = x_1x_2(x_3+x_3')(x_4+x_4') + x_1x_3x_4(x_2+x_2') \text{ , or}$$
$$S^* = x_1x_2x_3x_4 + x_1x_2x_3x_4' + x_1x_2x_3'x_4 + x_1x_2'x_3x_4 \quad \text{--------(5)}$$

There is now a one-to-one correspondence between the terms of the two equations. However, it must be realized that one can not use the original Boolean expression to derive Pr(S) directly, as

$$P_r(S) \neq p_1p_2 + p_1p_3p_4$$

In terms of the probability map interpretation, the modification is necessary to compensate for the fact that the groupings of the x_1x_2 and $x_1x_3x_4$ terms are not disjoint. In this example, $x_1x_2x_3x_4$ is common to both. An alternative solution therefore would be to modify the Boolean terms until they represent a disjoint grouping and one possibility in this case is:

$$S_d = x_1x_2 + x_1x_2'x_3x_4$$

Which leads directly to

$$P_r(S) = P_r(S_d) = p_1p_2 + p_1q_2p_3p_4 \quad \text{--------(6)}$$

For computing a network reliability in this example, the path reliabilities of x_1, x_2, x_3, x_4 are given that $p_1=0.9$, $p_2=0.8$, $p_3=0.7$, $p_4=0.6$,

respectively. In this case, The network reliability, R_s, calculated by (5) or (6) is R_s= 0.7956.

The above Boolean expression (5) thus represent a valid to full canonical form and can still be interpreted as a probability expression. The key problem of this Boolean algebra method thus is to transform the Boolean expression of the system success function to a form such that all terms are mutually disoint. Several methods for the generation of disjoint products are known.[10,11,12] Most of these methods are based on the concept that two conjunctive terms T_1 and T_2 will represent disjoint grouping if there exists at least on literal in a T_1 such that the same literal occurs in its complemented form in T_2. Therefore these techniques of generating disjoint terms require step by step testing for disjointness.

3-2. Our method to obtain the SDP expression

In the original Schinman's method, the canonical form is prerequsite for the simplification process and its minterms are designated by decimal numbers. This method is principally based on the Shannon's expansion theorem that a canonical Boolean function is recursively expanded into the form around variable x_i:

$$f = x_i \cdot f_i \mid_{xi=1} + \bar{x_1} \cdot f_i \mid_{xi=0}$$

The algorithm is as follows[5]:

Step 1) Arrange the decimal numbers representing the given function in a column.

Step 2) Divide the decimal numbers into two columnar groups, one headed with $\bar{x_1}$ and one headed x_1. The $\bar{x_1}$ column contains the numbers of the original function which are smaller than 2^{n-1} - the binary weight of x_1 - and the x_1 column contains the numbers which are equal to or greater than 2^{n-1}, first subtracting 2^{n-1} from each.

Step 3) Include a third column, headed by a dash to indicate the redundancy of x_1 and $\bar{x_1}$, consisting of the numbers which are common to columns x_1 and $\bar{x_1}$. Check the corresponding numbers in columns x_1 and $\bar{x_1}$ to record the fact that they are redundant. If any of the numbers in the dashed column have been previousely checked in both the x_1 and $\bar{x_1}$ columns, they should also be checked in the dashed column.

Step 4) Examine each column. If any column consists of only checked numbers, eliminate the column entirely. Each of the columns must now be expanded about x_2, x_3 , .. , x_n respectively by repeating the above steps.

Step 5) When the function is expanded about the final variable x_n, the residues must be 0. At this step, the prime implicants may be determined by simply tracing a path back to the start and reading the appropriate columnar headings.

Step 6) Draw the prime implicant chart to reduce the number of implicants so as to obtain a simplified Boolean function.

Step 7) Stop.

The strategy used in our algorithm is to modify the above Scheinman's algorithm in which the checked terms are not taken over subsequently in the process in order to obtain the disjoint expression.

Our algorithm is as follows:

Step 1) and Step 2) are same as above.

Step 3) Same as above but the checked terms are not taken over subsequently in the process for variables x_2, x_3 , .. , x_n.

Step 4) and Step 5) Same as above.

Step 6) Stop.

In our modified algorithm, the prime implicant chart is not needed because all the implicants obtained are mutually exclusive.

3-3. Application of the proposed algorithm

To compare our method with another method in Ref.[6], we consider the same bridge network in the example of Fig.1. The system success function which is obtained using the concept of the spanning trees is the same as (2) in Sec. 2. We apply our algorithm on the system success function to obtain the simplified network reliability expression in a sum of disjoint form. Fig. 3 shows the application of the proposed algorithm.

In this example, the full canonical form of the system success function is:

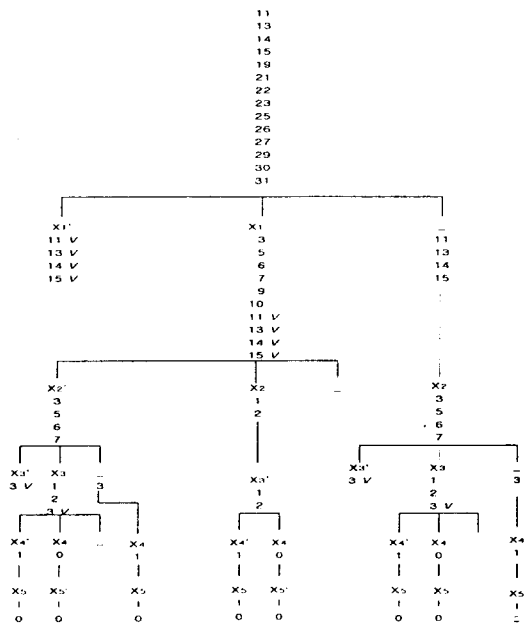

Fig. 3. Application of the proposed algorithm

$S^*(x_1, x_2, x_3, x_4, x_5) =$

$\sum(11, 13, 14, 15, 19, 21, 22, 23, 25, 26, 27, 29, 30, 31)$

The disjoint expression or the network reliability expression resulted from the application using the proposed algorithm is :

$$S_d = x_2x_4x_5 \cup x_1x_2\bar{x}_4x_5 \cup x_2x_3\bar{x}_4x_5 \cup x_2\bar{x}_3x_4x_5 \cup x_1x_2\bar{x}_3\bar{x}_4x_5 \cup x_1x_2x_3\bar{x}_4\bar{x}_5 \cup x_1x_2\bar{x}_3x_4\bar{x}_5 \quad \text{--------(7)}$$

Here, it is given that $p_1=0.9$, $p_2=0.8$, $p_3=0.7$, $p_4=0.6$ and $p_5=0.5$. These represent the path reliabilities of x_1, x_2, x_3, x_4, x_5 respectively.

From (7), the network reliability expression is:

$$R_s = p_2p_4p_5 + p_1q_2p_4p_5 + p_2p_3p_4q_5 + p_2p_3q_4p_5 + p_1q_2p_3q_4p_5 + p_1q_2p_3p_4q_5 + p_1p_3q_3q_4p_5 + p_2p_2q_3p_2q_5$$

The network reliability becomes then, $R_s = 0.7450$ which is the same as in Ref.[6] although the number of terms in the reliability expression is the same as that in [6].

For the CCN having equal probabilitys of survival p for each communication link, (7) simplifies to:

$$R_s = p^3 + p^3(1-p) + p^3(1-p) + p^3(1-p) + p^3(1-p)^2 + p^3(1-p)^2 + p^3(1-p)^2 + p^3(1-p)^2$$
$$= 8p^3 - 11p^4 + 4p^5 \quad \text{--------(8)}$$

In deriving (8), we have assumed perfect nodes. As computer outage account for as much as 90% of failure in most CCN's, we have to consider the reliability of nodes as less than 1 in in such situations. In such a case, (8) is to be multiplied by a factor ($p_{n1}p_{n2}p_{n3}p_{n4}$), where p_{ni} represents the reliability of node n_i [15].

4. Conclusion

In this paper, we derive the overall network reliability which is the probability of communication that each node in the network communicates with all other remaining nodes. For this, we propose a method to make the SOP disjoint for deriving the netwok reliability expression from the system success function using the modified Sheinman's method. A simple example is worked out to illustrate the our concept.

The proposed algorithm is applicable for large number of nodes and is easy to be implemented on the computer because of the generality of the Schinman's method. Our algorithm for obtaining the sum of disjoint expression does not require step by step testing for disjointness and only require operations on the decimal numbers iteratively, although the number of terms in the reliability expression is the same as that in [6].

We are implementing the algorithm on the computer and trying to apply it to the various benchmark networks such as in [2] for giving the superiority of our method.

<REFERENCES>

[1] K.K. Aggarwal, *Reliability Engineering*, Kluwer Acaademic Pub., 1993.

[2] S. Soh, S. Rai, "Experimental Results on Preprocessing of Path/Cut Terms in Sum of Disjoint products Technique", IEEE Trans. on Reliability, Vol. R-42, no. 1, pp. 24-33, Mar. 1993.

[3] M.O.Locks, "Recursive Disjoint Products: a Review of Three Algorithms", IEEE Trans. on Reliability, Vol. R-31, No. 1, pp.33-35, Apr. 1982.

[4] K. D. Heidtmann, " Smaller Sums of Disjoint

Products by Subproduct Inversion", IEEE Trans. on Reliability, Vol. R-38, No.3, pp. 305-311, Aug. 1989.

[5] A. H. Schinman, "A Method for Simplifying Boolean Functions", The Bell Systems Technical Journal, pp. 1337-1346, Jul. 1962.

[6] K. K. Aggarwal, S. Rai, "Reliability Evaluation in Computer-Communication Networks", IEEE Trans. on Reliability, Vol. R-30, No.1, pp. 32-36, Apr. 1981.

[7] R. S. Wilkov, "Analysis and Design of Reliable Computer Networks", IEEE Trans. on Communications, Vol. COM-20, pp. 660-678, Jun. 1972.

[8] B. Carre, *Graphs and Networks*, Clarendon Press, Oxford, 1979.

[9] M. Piekarski, "Listing of All Possible Trees of a Linear Graph", IEEE Trans. on Circuit Theory, Vol. CT-12, pp. 124-125, Mar 1965.

[10] J. A. Abraham, "An Improved Algorithm for Network Reliability", IEEE Trans. on Reliability, Vol. R-28, pp.58-61, Apr. 1979.

[11] M. O. Locks, "A Minimizing Algorithm for Sum of Disjoint Products", IEEE Trans. on Reliability, Vol. R-36, pp.445-453, Oct. 1987.

[12] F. Beichelt, L. Spross, "An improved Abraham-Method for Generating Disjoint Sums", IEEE Trans. on Reliability, Vol. R-36, pp.70-74, Apr. 1987.

[13] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall, 1974.

[14] S. Rai, K. K. Aggarwal, "An Efficient Methods for Reliability Evaluation of a General Network", IEEE Trans. on Reliability, Vol. R-27, No.3, pp.206-211, Aug. 1978.

[15] K. K. Aggarwal, J. S. Gupta, K. B. Misra, "A Simple Method for Reliability Evaluation of a Communication System", IEEE Trans. on Communication, Vol. COM-23, pp.563-565, May 1975.