The Design of Computer Networks for Optimum Reliability Subject to Criticality Constraints

NASSER HASSAN, MAGDY SAEB

Arab Academy for Science, Technology & Maritime Transport
Computer Engineering Dept.,
Alexandria, EGYPT

Abstract

The optimum design of computer communication network depends on capacity-related reliability CRR calculations. We show that some of the techniques found in the literature fail to provide the correct reliability expression in the case of packet-switched networks. Therefore, we present a modified approach for computing CRR especially in case of packet-switched networks.
The proposed algorithm is a two-stage approach: In the first stage, we find all success paths, simple and composite, at a given flow level. We also provide a method for computing the cut-set between source and terminal nodes.
In the second stage, we use the information obtained from stage one to get the CRR expression.
Finally, we develop the design optimization problem using the above-obtained expression as the objective function. The constraints are based on an overall hardening budget and the various link criticalities. A heuristic approach is then employed to get the least number of redundant links to be used to obtain the highest possible network reliability.

Key-words:
Computer networks,Reliability,Criticality,Optimization,packet-switched networks,Design. Proc.pp..1481-1488

1 Introduction

The design problem of computer communication networks for optimum reliability has forced itself back on the arena after the havoc caused by the failure of two essential communication satellites. However, the approaches, we found in the literature, have not taken into account the link criticality in their formulation of the optimization problem.
In this work, we try to shed some light on the importance of considering link criticality in the formulation of the capacity-related reliability (CRR) optimization problem. We start by reviewing the related recent literature on the subject matter. Noting that some techniques found in the literature fail to generate the correct reliability expression in case of packet-switched datagram networks, we discuss and provide a remedy for this irregularity. Moreover, we demonstrate our approach in computing the required reliability expression using two consecutive algorithms. Based on the resulting reliability expression, we find the link criticality, that we later use as a primary constraint in formulating the CRR optimization problem. Finally, we solve this problem using a heuristic approach.

2 Network Operational Environment (NOE)

For circuit-switched, packet-switched datagram and packet-switched virtual circuit networks, we define Network Operational Environment (NOE) characteristics as follows:

- The case of circuit switching and packet-switched using virtual circuit or datagram networks, where we need

to establish a node-to-node connection. As long as there is a connection between source and terminal nodes, one can send packets through the network although some of the links fail.

- The case of circuit switching and the case of packet switching using virtual circuit, all simple paths between source and terminal nodes are enumerated: Messages can be only sent through prescribed simple not composite paths in an ordered sequence.
- The case of packet switching datagram, where all simple and composite paths are enumerated, packets can flow through more than one path. They reach the terminal node in an out-of-order manner.
- The case of packet switching datagram, where packets are transmitted at each node in a different arrangement depending on the available path(s).

In all of the above cases, link capacity can not be ignored. The assumption that the link capacity is large enough to accommodate any packet bandwidth (size) is hardly practical. The link capacity depends on the available budget and is finite. Therefore, the mere existence of paths, simple or composite, is a necessary but not a sufficient condition for the analysis and design of computer communication networks.

3 Capacity-related reliability calculations

Several approaches have been established to deal with this problem. However, we will show some anomaly in one of these approaches. Then we turn to present our modifications, that we find necessary, to correct for these deviations. In order to achieve our objective, we need to present the following preliminaries:

The network is modeled as a probabilistic graph G (V, E), where nodes (V) identify communicating centers, and (E) identify connection links between the centers. In this graph, a link J has a finite capacity w_j that is known a priori. Let l be the total number of edges in G. A flow in the network is a function assigning a non-negative number f_I to each edge j so that $f_I \leq w_j$. We assume that the failure probabilities of links in the network are statistically independent. We also assume that the nodes do not allow any temporary storage of data flow. Thus, the network is defined as functioning correctly if and only if the specified amount of data flow can be transmitted from source node to destination node. A link j is said to be Up (Down) if it is functioning (not functioning) and is denoted by J (J'). An (s, t) cut is a disconnecting set. All communication between a prescribed (s, t) node pair is disrupted once an edge in (s, t) cut has failed. In an (s, t) cut, called cut i, C_i is minimal if no proper subset of it represents a 'cut'. The cut set $C_{s,\,t}$ is the set of all minimal cuts for graph G. Let the total number of cuts be n. The capacity of a cut set, W (C_i), for a minimal cut C_i is the sum of capacities of edges in C_j. From max-flow min-cut theorem [Ford 62], W_{max} is given by

$$W_{max} = \min_l \{W(C_i)\} \qquad (1)$$

In order to compute the reliability expression we need to enumerate both simple and composite paths.

4 Simple and composite-path enumeration

A simple path i, P_i, for an (s, t) node pair is formed by the set of UP edges such that no node traversed more than once. Any proper set of simple paths does not result in a path between these two node pairs. The path set $P_{s,\,t}$ is a set which elements are simple paths. The capacity of such a path is given by

$$W(P_i) = \min_j \{wj\} \quad \forall j \in \{P_i\} \qquad (2)$$

Depending on W_{min}, some or all-simple paths may fail to satisfy the capacity constraint. Thus, simple path (minimal cut) has to be carefully thought about while considering the CRR measure.

A k-composite path CP_i (k) is defined as the union of set of edges in any k simple

paths ($P_{i)}$ where i ∈ l, and l ≤ k ≤. We note that a k-composite path describes a subgraph of G (V, E). Moreover, all simple paths represent k-composite paths where k = one. One can show that for m-simple paths representing (s, t) connectives of the network, the total number of possible k-composite paths is $2^m - 1$. However, this number of composite paths can be minimized using the capacity constraint, and with the help of the absorption Boolean identity A ∪ AB = A, one can delete the redundancy.

5 An anomaly associated with some composite-path enumeration techniques

We noted a problem in composite-path enumeration resulting from the incorrect capacity computation for the CP_i (k). The capacity of a simple path is obtained easily from equation 2. However, we need to devise an efficient technique to get the capacity of a CP_i (k) for k > 1. Misra in [MISRA82] does not discuss any method. Aggarwal in [AGGA82] and [AGGA88] proposed a technique to help obtain the capacity of a CP_i (k) of the ARPA network, which is a packet-switched datagram network. *Nonetheless, this technique failed to generate the correct result, since it neglects one of the important Network Operational Environment (NOE) characteristics, that we discussed above, namely packet-switched datagram network.*

Aggarwal's technique states that: "The capacity of composite path is the sum of the capacities of the paths if there is no common link between them". For example, consider the ARPA network shown in Figure (1), [AGGA82]. The combination capacity of P_1: (6,7) and $P_{2\ (1,4,5,3)}$ can be determined as

$$W(CP_{1,2}(2)) = W(P_1) + W(P_2) = 1 + 3 = 4 \text{ units} \quad (3)$$

Now if we want to transmit five units (W_{min} = 5) from source node to terminal node T, the composite path $CP_{1,2}$ will fail to carry out these units according to Aggarwal's technique because $W(CP_{1,2}) < W_{min}$. However, these five units can be transmitted by sending two units through link 1 and link 4, and three units through link 6. Then rearrange these units at the common node between the two paths P_1 and P_2, which is allowable in this environment, to send out one unit through link 3 and four units through link 7. Therefore, in packet-switched datagram networks, we can consider the composite path $CP_{1,2}$ as a success path that can carry the five units.

We observe that for two or more disjoint paths, having at least one node common amongst themselves (other than the source and terminal nodes), Aggarwal's technique leads to incorrect results. Therefore, we conclude that, this method is not accurate enough to model packet-switched datagram networks operational environment.

6 Suggested remedy to the found anomaly

To solve the above problem, we provide a definition and a lemma to evaluate the capacity of a composite path CP_i (k), for k>1. To achieve this, we apply the concept of composite path cut (CPC) as follows:

Definition 1: A CPC_i (j) is a modified (s, t) cut C_j for the graph G (V, E) and is defined for a composite path CP_i (k). CPC_i (j) is

$$CPC_i(j) = CP_i(k), Cj \text{ for } j = 1 \ldots n \quad (4)$$

Since CP_i (k) describes a subgraph of G, then CPC_i (j) represents a cut for the CP_i (k) induced graph. The failure of the edges in CPC_i (j) leads to communication disruption between a prescribed (S, T) node pair. Further, note there are n number of CPC_i (j) for a composite path CP_i (k).

Lemma 1: The weight of a composite path, W (CP_i (k)) is given by:

$$W(CP_i(k)) = \min \{W(CPC_i(j))\}$$

Where W (CP_i (j)) represents the weight of a CPC_i (j) and is obtained by applying (2) to various CPC_i (j).

This lemma gives the same results, for k=1, as obtained by equation 2.

Definition 2: A composite path CP_i (k) is a success state of the network if it satisfies the capacity or flow constraint:

$$W(CP_i(k)) \geq Wmin$$

Otherwise, the CP_i (k) is a failure state. Moreover, a CP_i (k) is defined as

redundant state of the network if there is at least one success state CP_j (u) such that CP_j (u) $\subseteq CP_i$ (k).In [RAI91], Rai gives the following notion of a cross-link and its weight is used to detect a failure k-composite path a priori.

Definition 3: A cross-link (k), defined for the composite path, is the set of links common to the k simple paths forming the CP_i (k).

Definition 4: The weight of a cross-$link_i$ (k), referred to as: *W* (cross-$link_i$ (k)), is found by using the method of equation 2.

Theorem 1: A composite path CP_i (k) is a failure state if *W* (cross-$link_i$ (k)) < $W_{min,}$ for cross-$link_i$ (k) Φ, where Φ is ?. The proof is given in the Appendix of this article.

7 Data Representation for implementing path enumeration

In this work, we used bit-vector representation for the implementation of simple and composite path enumeration. A simple path in a network with l links is represented by l bits. A binary 1 denotes an "UP" link. However, a binary 0 denotes a "don't care" state rather than a "DOWN" state. This represents link absence. For example, consider the four simple paths;
P_1: (1,2), P_2: (3,4), P_3: (1,4,5),
P_4: (2,3,5). These are stored in memory for example as:
P_1: (1,2) = 0000000000000011
P_2: (3,4) = 0000000000001100
We used a 16-bit word (*w*) to represent network paths connected by a maximum of 16 links. With this type of data representation, the storage requirements for simple and composite path depends on the total number of links in the network and not on the size of the path. That is, irrespective of the number of links in a path, there will be always a 16-bit location reserved for it. However, the added cost of pre- and post-processing of the path l-bit testing is a one-time operation. Moreover, set theoretic operations like union, intersection, etc., can be used in detecting and eliminating redundant states. For example, assume a reference term X, and a test term XY (that is redundant subset of X):

Check one:

Reference X	11001	
Test XY	11101	
Using bit-wise OR	11101	
Test XY		11101
Using bit-wise XOR	00000	

A result "00000" shows that XY is redundant. A duplicate term is detected using the same approach.

Check two:

Reference X	11001	
Test XY		11101
Using bit-wise AND	11001	

A result equals to the reference term X shows that XY is redundant.
The set operations are implemented easily and the computation time is independent of the size of the network. Nevertheless, the number of bits representing a path increases the computation time by one unit every *w* additional links.

8 The Cut-set algorithm

We developed a simple algorithm, using bit vector representation, to find the cuts of a network. These cuts are necessary in computing the composite path capacity. We start by giving the required notation as follows:

P1 simple paths matrix, with rows denoting the paths and the columns corresponding to the links contained these paths.
$P1_{i, j}$ 1, if link j is present in path $P1_i$, 0 otherwise,
P2 non-success simple paths matrix,
P3 non-success simple and composite paths matrix,
P4 success simple and composite paths matrix,
P5 cut set matrix having all the minimal cuts of the network,
$P5_{i, j}$ = 1, if link is present in the cut $P5_i$, zero otherwise.
W_{min} system capacity,
LC link capacity column matrix $LC_j = w_j$, where w is the capacity of link j,
X cut capacity column matrix,
W_{max} maximum network flow,
$CP_{m, n}$ composite path consist of path m and n.
CPC composite path cuts matrix.

The algorithm can be summarized as follows:

Algorithm Cutset;
Begin
read input file containing **P1**and link number l;
Transfer each row in **P1**, using bit vector representation, to the equivalent decimal number and save the decimal numbers in a decimal column matrix **P1C**;
Let x = 2^l – 2;
Labe1: *If* x P1C $_i$ P1C $_i$ for any i,
that is x is not a redundant state
of any of the simple paths,
Then the position numbers of
the bits having zero value in
x-binary number represents
the link numbers in a cut.
Store these cuts in **P5**;
{The intersection here is an AND operation}
if x 0 then decrement the
decimal number x
and go to label1;
delete the redundant cuts in **P5**
using the following absorption law:
if cut $_x$ cut $_y$ = cut$_x$ or vice versa
then retain cut$_y$ (or cut$_x$)
else retain cut$_x$ and cut$_y$;
End.

As an example, consider a network with three nodes and three links. Here we have two simple paths in **P4.**The two matrices **P1** and **P1C** are given by:

$$P1 = \begin{bmatrix} 010 \\ 101 \end{bmatrix} \equiv P1C = \begin{bmatrix} 2 \\ 5 \end{bmatrix} \qquad (5)$$

9 Success paths Enumeration algorithm

In the previous section, we discussed the success and non-success states of a network from the aspect of simple paths. For a given W_{min} capacity, we test all simple paths and then partition them into success and non-success groups stored in matrix **P4** and matrix **P2** respectively. Let the number of paths in P2 be β. The non-success simple paths in P2 is used then to obtain the k-composite paths, 1<k β. This can be achieved by generating a new matrix P3 that is initially equal to P2, and then we combine each path in P2 with all paths that are successors to the same path in P3. Accordingly, we can prevent any duplication in composite paths. If the composite path is a success state then it is added to P4, otherwise it is appended to P3. To eliminate any redundancy in P4, the Boolean identity A ∪ AB = A is used.

Algorithm SuccesspathEnumeration;
Begin
Read input file to get **P1**, **P5**, **LC**, W_{min};
Get max network flow as follows:
W $(P5)_i = \Sigma_j LC_j \ \forall P5_{i,j} = 1$;
$W_{max} = \min \{W(P5)_i\}$;
If $W_{max} < W_{min}$ then stop;
Generate a column matrix W(p1) using the equation
W $(P1) = \min\{LC_j\} \ \forall P1_{i,j} = 1$;
Determine the success and failure paths from matrix **P1**. This is achieved by noting that
if the W(P1$_i$) W $_{min}$ for any i, then the set of links in row i is a success path and it is transferred to the binary matrix **P4**;
Formulate the second binary matrix **P2** with all entries of **P1** excluding the rows already transferred to **P4**. That is **P2** contains only the failure paths. The column matrix is also correspondingly modified and is renamed W (P2);
Initialize P3 by setting it equal to P2;
Initialize W (P3) by setting it equal to W(P3);
Let m = 1;
Label1:
Let n = m+1;
Label2:
Determine success and failure composite paths from **P2** as follows:
Combine row m of P2 with row n of P3 using logical OR operation;
Generate the CPC matrix using the identity;
$CPC_i = CP_{m,n}$ P $_i$5 , i = 1,2 …, k using logical OR operation;
Generate the column matrix
W (CPC_i)= LC $_j$, j=1,2,….,L;
Determine the capacity of the composite path CC = min $\{CPC_i\} \ \forall$ i;
If CC W $_{min}$ then it is listed in **P4**, otherwise it is placed in **P3**;
The capacity of the newly formed row is the capacity of the composite path and is placed in the appropriate list in W(P4);
Increment n and go to label2;
Continue until all rows are exhausted;
Increment m and go to label1;
Continue until all rows of P2 are exhausted;

Delete the redundant success paths in P4 using the absorption law:
If $P4_x$ P4 $_y = P4_y$, then retain $P4_y$
Else, retain both $P4_y$ and $P4_x$;
{The paths in P4 are the success paths}

End.

As an example of applying this algorithm, we apply it on the simplified ARPA net shown in Figure (1).

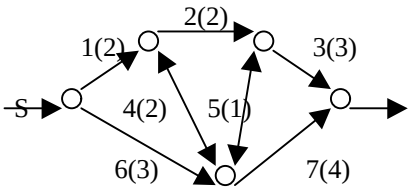

Figure 1: A simplified ARPA net where the link capacity is shown between brackets

The cut-set between S and T at a maximum flow of 20 units is given by
{(1,6), (2,4,6), (3,7), (2,5,7), (1,4,5,7)}
The LC matrix is:

LC = [2 2 3 2 1 3 4] (6)

The **P5** matrix is given by:

$$\mathbf{P5} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (7)$$

At a maximum flow of five units
W (P5) is given by:

$$\mathbf{W}\ (P5) = \begin{bmatrix} 5 \\ 7 \\ 7 \\ 7 \\ 9 \\ 8 \end{bmatrix} \quad (8)$$

For the network of Figure (2) shown below, the cut-set between S and T is: {(1,2), (4,5), (1,3,4), (2,3,4)} and the resulting matrices are given by:

LC = [15 5 5 10 15] (9)

$$\mathbf{P5} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (10)$$

$$\mathbf{W\ (P5)} = \begin{bmatrix} 20 \\ 25 \\ 20 \\ 40 \end{bmatrix} \quad (11)$$

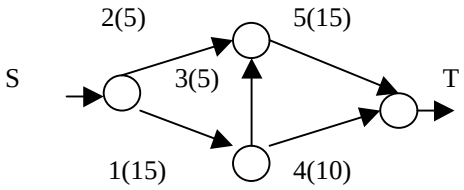

Figure 2: A bridge network test problem

10 The reliability expression evaluation

In this section, we apply a procedure outlined in [?] to evaluate the reliability expression, using our nine specially developed binary operators. We implemented these operators in a computer program to obtain the reliability expression.
As an example, the resulting expression for the network of Figure (3) is given by:

CRR (5) = p_1p_4 + q1p2p5 + p1p2q4p5 + p1q2q4p5 + q1p2p3p4q5
at a flow level of five units.

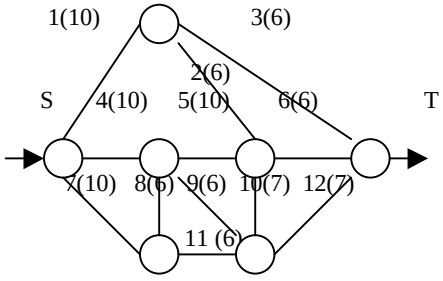

Figure 3: A seven-node twelve-link network.

For the network of Figure (3), shown above, the reliability expression matrix is given by:

1	1	1	1	0	1	0	0	1	0	0	1
1	1	1	−1	0	1	1	0	0	0	1	1
1	1	1	1	0	1	1	0	−1	0	1	1
1	−1	1	1	1	1	0	0	1	0	0	1
1	−1	1	1	1	1	0	0	−1	1	0	1
1	1	1	1	1	1	−1	0	−1	1	0	1
1	1	1	1	1	1	1	0	−1	1	-1	1
1	1	1	1	-1	1	−1	1	−1	0	1	1
1	1	1	1	1	1	−1	1	−1	−1	1	1
1	1	1	−1	0	1	1	1	1	0	-1	1
1	−1	1	1	1	1	1	0	−1	−1	1	1
1	−1	1	1	1	1	−1	1	−1	−1	1	1
1	−1	1	−1	1	1	1	1	0	0	1	1
1	1	1	1	1	1	1	0	0	0	1	1
1	1	1	−1	1	1	1	1	−1	1	-1	1
1	−1	1	1	-1	1	1	0	1	1	1	1
1	−1	1	1	-1	1	1	−1	1	1	1	1
1	−1	1	1	-1	1	−1	1	1	1	1	1
1	−1	1	−1	-1	1	1	1	1	1	1	1
−1	1	1	1	1	1	1	1	1	0	-1	1

Thus the resulting reliability expression given by:

CRR (15) =

$p_1 p_2 p_3 p_4 p_6 p_9 p_{12}$
+ $p_1 p_2 p_3 q_4 p_6 p_7 p_{11} p_{12}$
+ $p_1 p_2 p_3 p_4 p_6 p_7 q_9 p_{11} p_{12}$
+ $p_1 q_2 p_3 p_4 p_5 p_6 p_9 p_{12}$
+ and so on…
+…..
+$q_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 q_{11} p_{12}$
= 0.69221 (12)

11 The criticality measure of a link

The index of link criticality of a certain link (ICR) is defined as the normalized boundary value probability of a certain component in a network [SAEB87]. It is a measure of the degradability the network due a failure of a link. Using the obtained expression for the network reliability, one can deduce an expression for the link criticality of a certain link as shown in the above reference. The equation employed is of the form:

$$ICR_i = (\partial Q/\partial q_i).(qi/Q) \qquad (13)$$

Where ICR_i is the criticality of link i, and q_i is the failure probability of link i whereas Q is the system probability of failure. As shown in the same reference there is no need for using any numerical differentiation, since we can obtain the ICR as follows:

1. Obtain the partial derivative by substituting 1 and 0 and then 0, and 1 for p_i and q_i in the reliability expression and subtracting the two resulting values to obtain this derivative. This method is called the probability of boundary conditions and is given by:

$\partial Q/\partial q_i = (R \mid p_i = 1, q_i = 0) - (R \mid p_i = 0, q_i = 1)$

2. Multiply by the link probability of failure divided by the system failure probability.

This procedure is discussed in detail in the cited reference. We will use this measure as a constraint in the design optimization problem. The notion behind this is based upon *the hypothesis that a system with uniform link criticality is less vulnerable to natural or man-made induced failures.* The hardening of the system links will be subject to cost and now to criticality constraints as shown in the next section.

12 The design optimization problem formulation

The design optimization problem is formulated as follows:

Maximize CRR
Subject to:

$$\Sigma_i c_i x_i \le b, \quad \text{for } i=1,2,\ldots n$$

and

$$ICR_i \le k_i \text{ for } i=1,2\ldots n$$

We adopted a heuristic approach to find the optimum R. The results are as follows:

Example: For the above network, the following table demonstrates the cost of link hardening by adding similar and redundant links in parallel:

Link No.	Hardening cost (units)	Reliability
1	2	0.9
2	5	0.9
3	3	0.8
4	2	0.7
5	2	0.8
6	2	0.7
7	4	0.8
8	4	0.8
9	4	0.8
10	4	0.8
11	2	0.7
12	4	0.8

At flow level of 19 units, the CCR is found equal to 0.62502. We assume a total hardening budget of 40 units and a constraint on the IRC of the form

$$0 \le ICR_i \le 0.5,$$

Thus, we obtain an optimum reliability of 0.795903. Assuming redundancy upper limit of 2 links, the following table shows the optimum redundancy allocated to each link

Link	Optimum redundancy	ICR
1	2	0.430
2	1	0.130
3	1	0.130

4	1	0.102
5	1	0.099
6	2	0.096
7	1	0.07
8	1	0.068
9	1	0.040
10	1	0.030
11	1	0.030
12	2	0.020

Summary and Conclusions

In this work, we have presented an approach for designing computer communication networks based on the link criticality measure. The method is summarized as follows:

- We find all success paths at a given system flow level enumerating all simple and composite paths that can accommodate this flow level. We also develop a simple approach to obtain the cut-set between source and terminal nodes,
- The above-obtained information is then employed to generate the capacity related reliability CRR expression. We have shown that some of the techniques found in the literature fail to generate the correct results specifically in the case of packet-switched networks. Moreover, we have furnished a remedy to this anomaly.
- Finally, we compute the various links' criticality measure using the approach of boundary value probability. Based on this criticality measure and cost constraints, we formulate the design optimization problem. The design optimization problem is then solved using a heuristic approach and the numbers of redundant links required are computed.

The resulting system will have a practically uniform link criticality. Therefore, and using our hypothesis, it is less vulnerable to natural or man-made failures. Consequently, we believe that such an approach in designing computer communication networks can appreciably improve the quality of service provided by modern computer networks.

References

[FORD62] L. R. Ford, D. R. Fulkerson, Flows in networks, 1962, Princeton University Press.

[AHO74] A. V. Aho, J. E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms. Reading, MA: Addison-Wesley, 1974.

[ROSEN79] A. Rosenthal, "Approach to comparing cut-set enumeration algorithm," IEEE. Trans. Reliability, vol. R-28, April 1979, pp. 62-65.

[SATY81] A. Satyanarayana, Jane N. Hagstrom, " A new algorithm for the reliability analysis of multi-terminal network," IEEE. Trans. Reliability, vol. R-30, Oct. 1981, pp. 325-334.

[AGGA82] K.K. Aggarwal, et al., "Capacity consideration in reliability analysis of communication systems," IEEE. Trans. Reliability, vol. R-31, June 1982, pp. 117-181.

[MISR82] K. B. Misra, P. Prasad, "Comments on: Reliability evaluation of flow networks," IEEE. Trans. On Reliability, vol. R-31, June 1982, pp. 174-176.

[COLB87] C. J. Colbourn, The combinatorics of Network Reliability, Oxford University Press, 1987.

[HAR87] S. Harir, C. S. Raghavendra, "SYREL: A symbolic reliability algorithm based on path and cut-set method," IEEE. Trans. Computers, vol. C-36, Oct. 1987, pp. 1224-1232.

[LOCKS87] M. O. Locks, F. ASQC, " A minimizing algorithm for sum of disjoint products," IEEE. Trans. on Reliability, vol. 36, Oct. 1987, pp. 445-453.

[SAEB87] Magdy Saeb, Roland Schinzinger, "On measures of network reliability and critical components," IEEE. Asilomar Conference Proceedings on Circuit, Systems & Computers, California, VLSI session, 1987.

[AGGA88] K. K. Aggarwal, "A fast algorithm for the performance index of a telecommunication network," IEEE. Trans. on Reliability, vol. R-37, April 1988, pp.65-69.

[PAGE88] L. B. Page, J. E. Perry, "Reliability of directed networks using the factoring theorem," IEEE. Trans. Reliability, vol. R-38, Dec. 1988 , pp. 556-562.

[RAI90] S. Rai, D. P. Aggarwal, Distributed Computing Network Reliability, 1990, IEEE. Computer Society Press.

[WILSN90] J. M. Wilson, "An improvement minimizing algorithm for sum of disjoint product," IEE Trans. Reliability, vol. 39, April 1990, pp. 42-45.

[RAI91] S. Rai, S. Soh, " A computer approach for reliability evaluation of telecommunication networks with heterogeneous link-capacities," IEEE. Trans. on Reliability, vol. R-40, Oct. 1991, pp. 441-451.

[SOH91] S. Soh, S. Rai, "CAREL: Computer aided reliability evaluator for distributed computer networks," IEEE. Trans. On Parallel and Distributed Systems, vol. 2, April 1991, pp.122-213.