International Journal of Computer Science and Engineering (IJCSE)
Vol.1, Issue 1 Aug 2012 25-37

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

GLOBAL RELIABILITY EVALUATION OF DISTRIBUTED COMMUNICATION NETWORKS

MOHD ASHRAF & RAJESH MISHRA

School of Information and Communication Technology, Gautam Buddha University, India

ABSTRACT

This paper presents an efficient method to compute the global reliability of a distributed computer mesh network. In doing so, the paper proposes an algorithm which is based on edge deletion approach to enumerate all the spanning trees for a large complex computer communication network. These spanning trees are further used as an input to multi-variable inversion-based sum of disjoint product approach to obtain the reliability expression. The algorithm has been illustrated by suitable.

KEY WORDS: Network Reliability, DCN, Spanning Tree, Global Reliability MVI.

INTRODUCTION

Reliability evaluation and analysis have been the main contributors to the design, optimization, deployment, resources shearing and maintenance of traditional data distribution networks and non-network critical systems like space phased-mission systems. Researchers have proposed various approaches for evaluating reliability of these systems, in which reliability is generally defined as the probability that the system will perform its intended function under stated conditions for a specified period of time [1]. Advances in computer technology have placed us at the doorstep of a new era where a large number of communication devices will provide access to information anytime and this need to have the computers communication with each other has led to increased demand for a reliable distributed computing network (DCN). An important performance metric in the design of highly reliable DCN is provided by its global reliability parameter [2]. As there are fast and computationally simple methods available in the literature for terminal reliability evaluation but for the full utilization of facilities available in the system, it is important that each node is able to communicate with other nodes of the communication network.

Network reliability can be easily computed from the reliabilities of individual components if the network has strictly a series, parallel, and series–parallel or parallel series configuration. However, the computation becomes quite involve if one has to deal with a network that is complex, or non-series –parallel. Basically, network reliability problem are NP-hard [3], [4] in nature, and drawn the attention of many researchers to develop efficient algorithm for obtaining their solution. A close survey of the literature reveals that not much has been done to develop efficient as well as conceptually simple method for the global

reliability evaluation for a distributed mesh networks. Daniesel [5] and Fratta and Montanari [6] have provided techniques for path and cut enumeration by using the method of symbolic soloution for simultaneous equation. Aggarwal et al. [7] have presented a state removal algorithm to find all possible paths. The algorithm does not require matrix multiplication and the size of the matrix reduce in every step. A state removal algorithm is also used by Rai and Aggarwal [8] to determine all simple path of a graph. The method does not require matrix multiplication; however it does require rearrangement of the path in ascending order of cardinality.

For enumeration of global path and terminal paths in which the path are enumerated with the help of connection/adjacency matrix using graph theoretic concept suggested by Samad [9]. The algorithm has disadvantages of the path generation in ascending order of cardinality, further Samad [10] proposed an algorithm to enumerate all global paths simultaneously between all single terminal pair communication network. The proposed method is inefficient and computationally complex for large networks when the terminal nodes are dramatically increased.

Aziz et al. [11], presented an algorithm which enumerates path sets of reliability graphs using the method of indexing. In this algorithm, an index of connection matrix [C] is prepared at the outset which give the location of all the non-zero entries of the connection matrix. During the process of enumeration of path sets using this index only the non–zero entries of [C] are picked up and useful multiplication are performed, thereby eliminating the useless multiplication the number of multiplication is reduced significantly. Still these algorithms are inefficient for a large complex DCN. In the next part of this paper the author has been proposed an algorithm for enumeration of all spanning trees of a DCN that is the primary requirement of global reliability evaluation process taking connectivity constraint for any SDP approach.

Acronyms : NSP = Non series-parallel, SDP = sum of disjoint product, SP = Series parallel, DCN = distributed computing network, MVI = Multiple variable Inversion

METHODOLOGY AND ASSUMPTIONS

Figure 1 depicts the proposed methodology has been adopted for applying the SDP techniques for *all-terminal* reliability evaluation. Clearly, the key issues in applying the approach are network representation, enumeration of all possibilities (spanning trees) for all-nodes connectivity and by making these possibilities disjoint with each other to form the reliability expression.

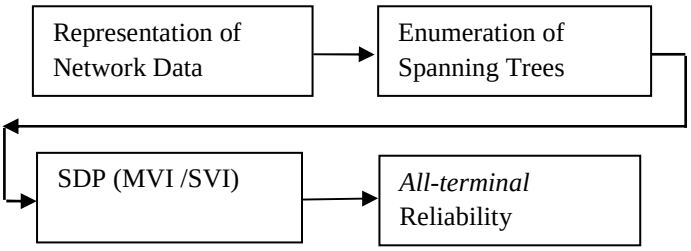

Figure 1: Block Diagram of Technique for Evaluating All-Terminal Reliability Using SDP Approach

Therefore, in the proposed approach, after the enumerating the *all spanning trees*, the next step in the reliability evaluation is to obtain the symbolic expression in terms of the probability of the various components being operational/non-operational. If the spanning trees are mutually exclusive, the probability of the union of '*n*' events can be written as:

$$\Pr(E_1 \cup E_2 \cup \ldots \cup E_n) = \Pr(E_1) + \Pr(E_2) + \ldots + \Pr(E_n) \quad (1)$$

As the generated spanning trees are not mutually exclusive, it is desired to obtain the disjoint of the probabilistic expression of the spanning trees. Various Boolean algebra methods have been reported in the literature for disjointing the probabilistic terms so that the simple additive expression can be used for reliability estimation. One such method available in the literature as suggested by Chaturvedi and Misra [12], based on the principle of SDP has been employed in the present study. The method decomposes the set of *all spanning trees* into another set of mutually exclusive spanning trees, which has a one-to-one correspondence with the reliability expression. And we make the following assumptions in our analysis.

1. A communication network is modeled by probabilistic connected graph.
2. The nodes of the network are perfectly reliable.
3. The network and its branch have only two states (a) working or (b) failed.
4. The failure probability for each link or node is given as a fixed probability for a given mission time or in terms of a lifetime distribution.

ALGORTHIM FOR ENUMERATING SPANNING TREES AND COMPUTING DCN RELIABILITY

Problem Statement

Given a DCN graph $G = (V, E)$ consists of an end user (sink node) s with a set of target source nodes $|V|$ or n and a set of edges (or links) $|E|$ or e. If an edge connects two vertices i and j; j is said to be adjacent to i. The n number of target sensor nodes in the graph is assigned number 1, 2, 3…n sequentially.

The e number of links of the network can be arbitrarily and sequentially assigned numbers. With

this graph model, depending on the state (working or failed) of vertices (or nodes) and / or edges (or links) with specified probability, the network can be considered either working or failed with estimated probability.

In this section we propose an algorithm for a DCN, where the network of *e* branches and *n* node will have simple path, touching all the nodes of the cardinality *(n-1)*. Thus the combination ${}^{m}C_{n-1}$ contains all simple paths, i.e. spanning trees of the DCN.

Therefore, a network graph *G= (V, E)* consists of a set of vertices (or nodes) |*V*| or *n* and *notation* are as follows:

n = Number of node in the network

e = Number of branch in the network

i = Index variable; i=1,2,3,4…….

$V_i = i^{th}$ node of any network $1 \leq i \leq n$

$B_i = i^{th}$ branch of any network $1 \leq i \leq e$

M_{dn} = Minimum degree of a node in network

A [ ] [ ] = Incidence matrix. Each entry A[i][j]

D [ ] = Degree vector corresponding to incidence matrix

FLOW CHART

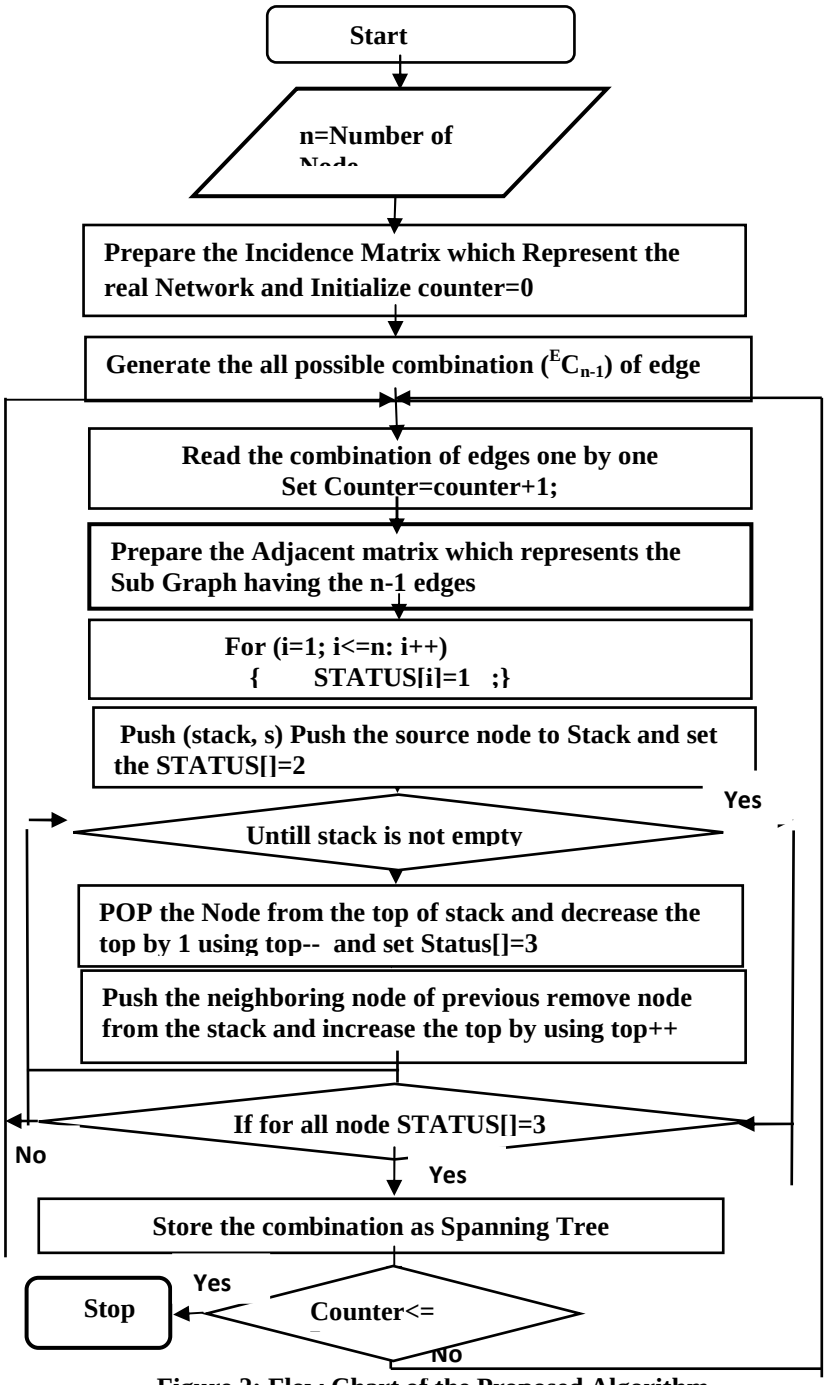

Figure 2: Flow Chart of the Proposed Algorithm

Steps of the algorithm

The algorithm has been coded and implemented in the JAVA environment. For reader's benefit, each step of the algorithm is illustrated with suitable example. Finally the Step # 8 provides all spanning tree for g-Reliability evaluation. The steps of the algorithm are as follows:

Step#1: [Initialize] Label all the node of logic diagram of DCNS from 1 to n and all the Branches from 1 to m in arbitrary manner where n is number of node and m is number of Branches.

Step#2: Develop the incidence matrix A [i][j] and its corresponding degree vector. This incidence matrix is being used to find the node with minimum degree (M_{dn}).

Step #3: Generate the combination of branches of n-1 branch and its corresponding Degree Vector D [i] where i =1 to n by adding all (n-1) degrees of the incidence matrix.

Step #4: For any value of i (i =1 to n), if D[i]=0 then remove this combination and proceed to next combination generation (Return to Step # 3)

Step # 5: Test D [i] ∀ i =1 to n

- **a)** D [i] =1 identify the branch which is incident to node i and it is present in the combination.
- **b)** Remove the branch from the combination and update the degree vector.
- **c)** If the removal of branch has caused removal of a node from the degree vector then go to Step #6 otherwise drop the combination and return to step #3

Step # 6: If number of branch present in the combination are more than 2 then return to **step # 4** otherwise go to next step # 7.

Step # 7: If the non-zero entries in the updated degree vector are three, save this combination otherwise drop it.

Step # 8: If all the combination in which branch incident on a minimum degree of node is included has been generated then stop otherwise return to **step # 5**.

ILLUSTRATION

Consider a 6-node, 9-link network with its adjacency matrix shown in Figure 3. The step by step enumeration of spanning trees and their corresponding degree vectors are explain in the following steps:

Step # 1 Label all the node and branches in any arbitrary manner

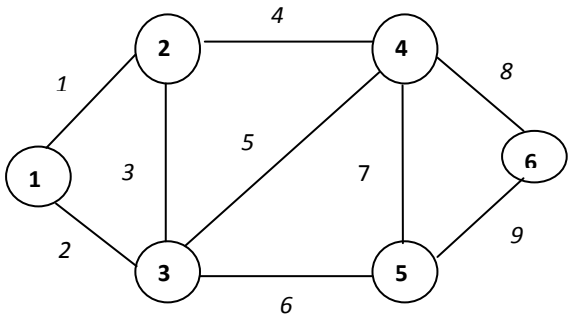

Figure 3: A Six Node Nine Link Mesh Network

Step # 2 Construct the incidence matrix and its corresponding degree vector for the graph as shown in Figure 2 and find the minimum degree node.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 4 \\ 3 \\ 2 \end{bmatrix}$$

In this network the minimum degree node (M_{dn}) are ‘1’ and ‘6’

Step # 3 Generate the combination of branches (combination having n-1 branches) with minimum degree ‘2’ and its corresponding degree vector by adding all (n-1) degrees of the incidence matrix.

Step # 4 For any value of i (where i = 1, 2, 3,……n) ; If D[i] = 0 then drop this combination and proceed to next combination as given in following stair:

	Combination					**Degree Vector**					
1.	1	2	3	6	7	2	2	2	1	2	0
2.	1	3	5	6	7	1	2	3	2	2	0
3.	1	2	3	4	7	2	3	2	2	1	0
4.	1	3	4	5	7	1	3	2	3	1	0
5.	3	6	7	8	9	0	1	2	2	3	2

Step # 5. Check D[i] for i= 1, 2, 3, 4…….n, if for any i, D [i] =1 then identify the branch which is incident to node i and present in the combination and update the degree vector

Example (a) Consider the combination 1, 2, 3, 6, 8 and its corresponding incidence matrix and degree vector is given below

$$A[\,][\,] = \begin{matrix} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 & 3 & 6 & 8 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{matrix} \quad D = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Node ‘4’ is of degree ‘1’ and branch which is incident on node ‘4’ is ‘8’. Branch ‘8’ is present in the combination and therefore it should be removed

$$A[\,][\,] = \begin{matrix} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 & 3 & 6 & 8 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix} \quad \text{A new Degree vector} = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

The sub network after the removal of this branch has a degree vector of [2, 2, 3, 0, 1, 0]. Here it is observed that the branch removal has caused removal of two nodes, therefore, this combination does not represent a spanning tree. Drop this combination and return to **step #3**

For Example (b) 1, 2, 6, 8, 9 $\rightarrow$ Degree vector $\begin{bmatrix} 2 & 1 & 2 & 1 & 2 & 2 \end{bmatrix}$

$$A = \begin{matrix} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 & 6 & 8 & 9 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{matrix} \quad \text{Degree Vector } D[\,] = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

Consider the combination (1,2,6,8,9) which has degree vector [2,1,2,1,2,2] . Node ‘2’ in the degree vector is of degree ‘1’ and the branch which is incident on node ‘2’ and present in degree vector is ‘1’. Branch ‘1’ is therefore, removed for the combination and the new updated degree vector is [1, 0, 2, 1, 2, 2] representing sub network with branches (2, 6, 8, 9)**.**

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 6 & 8 & 9 \end{array} \\ \begin{bmatrix} 0 & \boxed{1} & 0 & 0 & 0 \\ 0 & \boxed{0} & 0 & 0 & 0 \\ 0 & \boxed{1} & 1 & 0 & 0 \\ 0 & \boxed{0} & 0 & 1 & 0 \\ 0 & \boxed{0} & 1 & 0 & 1 \\ 0 & \boxed{0} & 0 & 1 & 1 \end{bmatrix} \end{array} \quad \text{Updated Degree Vector D [ ] =} \quad \begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

As the branch removal has caused removal of single node only and the four branches are left in the combination.

Therefore two more branches can be removed. Node ‘1’ is of degree ‘1’ in the new sub network and branches ‘1’ & ‘2’ is incident on node ‘1’. Branch ‘1’ is therefore, is removed from the combination and updated new degree vector for sub network is [0, 0, 1, 1, 2, 2] representing combination of branches 6, 8, 9. This branch removal also causes the removal of one node only.

Step # 6: It can be observe that number of branches is more than the two. Therefore one more branch can be removed **(Go to step 5)**

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 6 & 8 & 9 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & \boxed{0} & 0 \\ 0 & 0 & 0 & \boxed{0} & 0 \\ 0 & 0 & 1 & \boxed{0} & 0 \\ 0 & 0 & 0 & \boxed{1} & 0 \\ 0 & 0 & 1 & \boxed{0} & 1 \\ 0 & 0 & 0 & \boxed{1} & 1 \end{bmatrix} \end{array} \quad \text{Degree Vector D [ ] =} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

Here, node ‘3’ is of degree ‘1’ and ‘2’ in the new sub network and branches ‘2’ and ‘6’ are incident on node ‘3’ and shown in degree vector. Branch ‘6’ is therefore removed from the combination and updated new degree vector is [0, 0, 0, 1, 1, 2] representing the combination of branches (8, 9).

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 6 & 8 & 9 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \text{ New degree Vector } = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 2 \end{bmatrix}$$

Step # 7 Degree vector corresponding to sub network 8, 9 has three non-zero entries, therefore, the combination 1, 2,6,8,9 represent a spanning tree.

This branch removal also causes the removal of one node only. Further the degree vector corresponding to sub network (8, 9) has three non-zero entries, therefore the combination (1, 2, 6, 8, 9) represent a spanning tree, store this combination.

Step # 8 If all the combinations in which branch incident on a minimum degree of node are included have been generated then stop otherwise return to **step # 5**.

EXPERMENTAL RESULTS AND DISCUSSIONS

Authors have applied the proposed algorithm to several networks taken from the literature of varied complexities, and verified the reliability obtained by other researchers. Experimental results on several networks taken for this study from the published work. Among them, the results of comparison all spanning

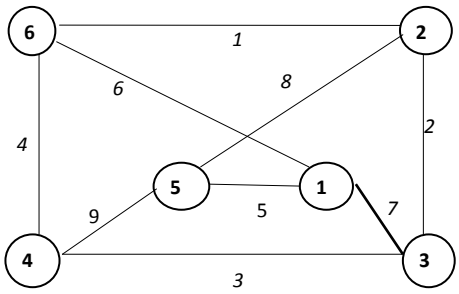

Figure 4: 6N9L (Fig. 4 of [13]).

trees for few networks (shown in Figure 4 - Figure 9) [13, 14] are provided in Table 1 to show the efficacy of the algorithm and proposed framework to evaluate global reliability using connectivity criterion. Besides, Table 1 also provides the enumeration time of each network with reliability.

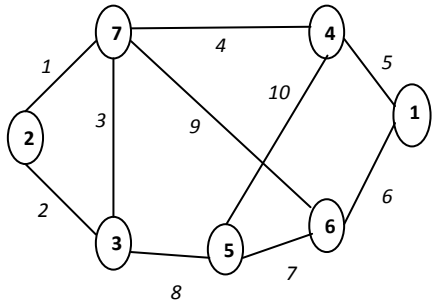

Figure 5: 7N10L (Fig. 5 of [13]).

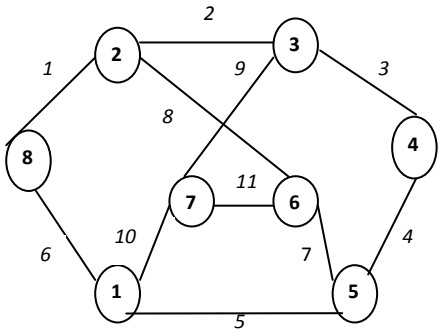

Figure 6: 8N11L (Fig. 6 of [13]).

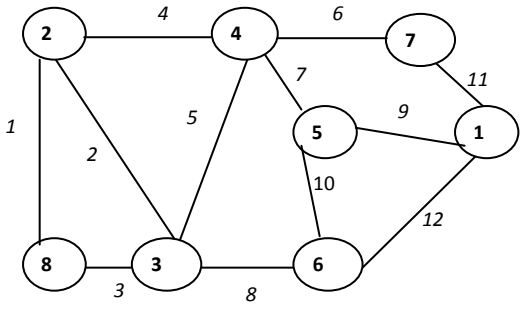

Figure 7: 8N12L (Fig 9 of [14].

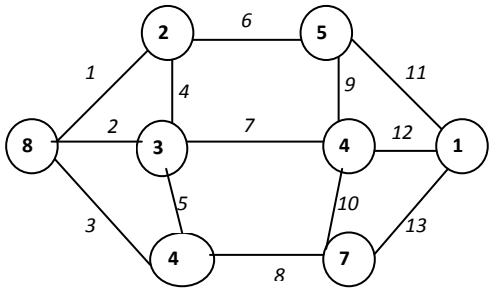

Figure 8: 8N13L (Fig. 8 of [14]).

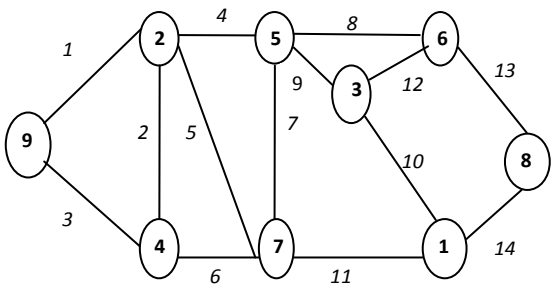

Figure 9: 9N14L (Fig. 10 of [14]

Table: 1 Spanning trees enumeration with enumeration time

Size of graph	Number of spanning tree	CPU time in micro second	Density of Graph 2*l/n(n-1)	Global reliability figure for p = 0.9
6N9L	81	$4.8x10^{-6}$	0.6	0.993263229
7N10L	96	$6.3 x10^{-5}$	0.476	0.972218165
8N11L	168	$9.5 x10^{-5}$	0.392	0.969699135
8N12L	247	$1.1 x10^{-4}$	0.428	0.971153158
8N13L	576	$1.89 x10^{-4}$	0.464	0.991942811
9N14L	647	$5.78 x10^{-4}$	0.388	0.970104348

CONCLUSIONS

This paper proposed an efficient approach to enumerate all possible spanning trees, which are used to compute global reliability of large DCN with modest memory and time requirement. The algorithm has been implemented in java and was run on Sun Solaris platform. The proposed algorithm is usually based on the data structure and the graph theoretic concepts. The performance of the algorithm is dependent on the computer specification on which the program run, and last but not least, the coding of program. A better implementation or faster machine would increase the performance of a program. Moreover, all methods of reliability computation are known to be computationally intractable or NP-hard, which make difficult to compare the technique from the aspect of time or memory complexity. The proposed approach was compared with some recent path set based algorithms in terms of their enumeration time to prove its simplicity and performance through its application on several example considered by researchers.

REFERENCES

1. Koushanfar F. and S. Slijepcevic, Error-tolerant Multi-model Sensor Fusion, IEEE CAS Workshop, Pasadena CA, Sept 2002.
2. Aggarwal A. and A. Satyanarayan, An (O/E) time algorithm for computing the reliability of a class of directed networks, Operation Research, Vol. 32, pp. 493-517, **1984**.
3. Wilkove R. S., Analysis and design of reliable computer network, IEEE Transactions on Communications. Vol. 20 (3), pp. 660-678, **1972**.
4. Provan J. S. and M. O. Ball, Computing network reliability in time polynomial in the number of Cuts, Operation Research, Vol. 32, pp 516-526, **1984**.
5. Denielson G., On Finding the Simple Path Sets and Circuit in a Graph, IEEE Transactions on Circuit Theory, Vol. CT-15, pp. 294-295, **1968**.

6. Fratta and U. G. Montanari, A Vertex Elimination Algorithm for Enumerating All Simple Path Sets in a Graph, Networks, Vol. 20, pp. 151-177, **1975**.
7. Aggarwal K. K., Gupta J.S. and K. B. Mishra, A simple method for reliability evaluation of a communication system, IEEE Trans. on Communications, pp. 563-566, **1975**.
8. Rai S. and K. K. Aggarwal, An efficient method for reliability evaluation of general network, IEEE Transactions on Reliability, Vol. R-273(3), pp. 206-356, **1978**.
9. Samad M. A., An Efficient Method for Terminal and Multiterminal Path Set Enumeration, Microelectronics and Reliability, Vol. 27, No. 3, pp. 443-446, **1987**.
10. Samad M. A., Methods for Global Reliability Evaluation of any Large Complex System, Reliability Engineering, Vol. 18, pp. 47-55, **1987.**
11. Aziz M. A., Sobhan M. A. and M. A. Samad, Empirical Formulas and Global reliability Evaluation od Reliability Graphs, Microelectronics & Reliability, Vol. 33, No. 9, pp. 1233-1236, **1993**.
12. Chaturvedi, S. K. and K. B. Misra, A hybrid method to evaluate reliability of complex networks, International Journal of Quality and Reliability Management, Vol. 19, No. 8, pp. 1098-1112, **2002**.
13. Rath D. and K. P. Soman, A Simple Method for Generating k-Trees of a Network, Microelectronics and Reliability, Vol. 33, No. 9, pp. 1241-1244, **1993**.
14. Chaturvedi S. K. and K. B. Misra, An Efficient Multi-Variable Inversion Algorithm for Reliability Evaluation of Complex System using Path Sets, International Journal of Reliability, Quality and Safety Engineering, Vol. 9, No. 3, pp. 237-259, **2002.**
15. Mishra R., Chaturvedi S. K., A Cutsets-based Unified Framework to Evaluate Network Reliability Measures, IEEE Trans. on Reliability, vol. 58-4, pp. 658-666,**2009**