An Efficient Connectivity Measuring Approach for the Communication Network

Maneesha Dabas [1], P.C. Saxena [2], Sangeeta Sabharwal [3]

**[1] Division of Computer Engineering, Netaji Subhash Institute of Technology
New Delhi, Delhi – 110075, India**

**[2] Department of Computer Science, Jawaharlal Nehru University
New Delhi, Delhi - 110067, India**

**[3] Division of Computer Engineering, Netaji Subhash Institute of Technology
New Delhi, Delhi – 110075, India**

Abstract

The purpose of designing an optimal and fault tolerant communication network is to achieve a precise performance without any disruption at a minimal cost. A fault tolerant network is able to maintain connectivity under the failure conditions only if there are multiple links disjoint paths for each node pair (nodes are assumed reliable). Main objective in designing a network topology is to minimize the cost of the network while satisfying the pre specified connectivity. It has also been proved that designing a minimum cost topology under the connectivity constraint is a NP hard problem. Several Meta- heuristic iterative techniques like Genetic Algorithm, Simulated Annealing, Tabu Search etc have been proposed by many researchers for designing the network topologies. There is a need to measure the connectivity of each intermediate solution in these iterative techniques. Measuring connectivity by finding all disjoint paths for each node pair is a very time consuming task. Researchers also proposed to use node degree as a parameter for measuring the connectivity rather than exploring all disjoint paths for each node pair. But it is also true that node degree is not a sufficient condition for measuring the connectivity of the network. In this paper, an approach for measuring the connectivity of a given network is proposed which is as efficient as finding all disjoint paths in determining the connectivity at less computational effort. Performance of the proposed approach is evaluated using several examples.

Keywords: Connectivity, Disjoint paths, Degree, K-connected network.

1 Introduction

Communication network link connectivity is defined as the minimum number of links that need to be removed to disconnect the network [3]. It can be measured by finding all disjoint paths for each node pair. Connectivity of a network is equal to the number of disjoint paths of a node pair who has minimum number of disjoint paths among all node pairs. Connectivity is greatly measured in designing a k- connected communication network topology.

As we know that, most common network topology design deals with finding the best layout of the elements subject to fault tolerance [2]. This is a well-known optimization problem and difficult to solve. For N number of nodes, maximum number of links are n*(n-1)/2, and the maximum number of topological configurations of n nodes are $2^{n*(n-1)/2}$. The risk of combinatorial explosion is obvious [11][12][13]. As a result, various constructive heuristic methods[3][6][14][16] and meta heuristic methods[7][8][10][15] are often proposed for design of k-connected networks to reduce the search of candidate topologies and provide suboptimal solutions.

Meta heuristic iterative techniques based on the genetic algorithms [7] [8] [10], simulated annealing [15] etc. are proposed by researchers to design a k connected network. These are the iterative techniques and desired connectivity is required to check for each intermediate solution. Finding all disjoint paths for each node pair to verify the connectivity is a very time consuming task which has time complexity of $O(VE^2)$[16]. To make it simpler, some researchers measure the connectivity using the node degree only. For example, Ewa Szlachcic [7] used evolutionary algorithm to design a minimum cost network subject to connectivity and average message delay constraint. Connectivity in this paper is described using the nodes degree and it is considered that a network is k-connected if it satisfies the condition in the following equation:

Degree (v_i) ≥ k for all v (1)

Berna Disgz et. al.[11] proposed an local search genetic algorithm for optimal reliable design of a network. Nodes degree is used to explore boundaries between the feasible and infeasible solutions to avoid unnecessary computation of fitness function to check all terminal reliability. H. sayoud et.al [10] also assumed that k connectivity can be checked using the condition given in the equation 1. In [1], Tomas Fencl provided example which proves that condition in equation 1 does not result into k connected network and suggested that if node degree is used as a connectivity measure then it is necessary to use another condition given in equation 2 instead of condition given in equation 1.

Degree (v_i) = h where h € {k, k+1,...} for all v (2)

But we found that the condition suggested by Tomas Fencl [1] also does not result into a k-connected network in certain cases as shown in figure 1. Degree of all nodes in figure 2 is 3 but the network is not a 3-conencted network. In other words, failures in the links 3-5 and 4-6 result into a disconnected network.

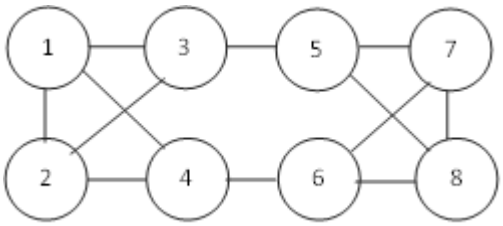


Figure 1.

So considering only the node degree in all way is not a sufficient condition to verify the connectivity of a network. And measuring exact connectivity using the algorithms [16] is a very time consuming task especially when used in the topology design using genetic algorithm. Hence, there is a need of computational effective approach which would be able to measure the exact connectivity of a network. In this paper, an approach has been proposed for measuring the connectivity of a given network which is computationally efficient. The paper is organised as follows. Section 2 presents an approach for measuring the connectivity of a network. Illustrative examples are discussed in section 3 and some concluding remarks are given in section 4.

2. Proposed Connectivity Measuring Approach

A communication network consists of a number of nodes connected via communication lines. In graph theoretic terms, a communication network having N nodes and M communication lines can be regarded as a Graph G=(V, E) consisting of a finite non- empty set V = {v_1, v_2, …. , v_N} of N vertices and a set E = { e_1, e_2,……., e_M}. The vertices of the graph correspond to the nodes (computers) of the network and its edges correspond to the communication links. Throughout out discussion, we shall use Network and Graph interchangeably.

In this paper, an approach has been proposed an approach which measures the connectivity of a given network by finding the 2-connected networks. And at any point, if it is not possible to find a 2-connected network, we try to find out a 1-connected network. Connectivity of a network is always equal to two times the number of 2-connected network plus one if a 1- connected network is also found. A given network may be 1-connected network, 2-connected network or k-connected network. The necessary and sufficient conditions for all three types of the network representations that need to be checked are discussed in section 2.1. Necessary conditions are checked first and if fulfilled then only the sufficient conditions are required to be checked. Using this way, we can save computation time by avoiding examining the infeasible networks.

2.1 Classification of a Network and their Necessary and Sufficient Conditions

A network can be classified as a 1-connected network, 2-connected network and k-connected network. Here, we describe all the necessary and sufficient conditions that need to be checked for all types of networks.

2.1.1 1-Connected Network

A network is said to be 1-connected if all nodes are reachable to each other. Connectivity of network is one only if there exists a spanning tree traversing all nodes in the network. Necessary and sufficient conditions for a 1-connected network are given below:

Necessary condition:
Number of links in the network should be at least N-1.

Sufficient condition:
There should be at least one spanning tree which traverses all nodes in the network.

2.1.2 2-Connected Network:

A network is said to be a 2-connected network if there are at least two disjoint paths for each node pairs. The necessary and sufficient conditions for a 2-connected network are given below:

Necessary conditions:

Minimum number of links should be N.
Degree (v_i) ≥ 2 for all v_i.

Sufficient conditions:

There are various possible representations for a 2-connected network. If a traversed network satisfies any of the following representations described below, the network is said to be a 2-connected network.

A network is represented by a cycle containing all nodes. This representation can be proved by a theorem given below:

Theorem 1. Let G be a graph represented by a cycle containing all nodes. Then, G is a 2-connected network.

Proof: It is trivial that there are two disjoint paths between each node pair in a cycle i.e. one in the forward direction and second in backward direction from one node to another node.

A network is represented by two disjoint sub networks which are connected via two links. Theorem which proves this representation is given below:

Theorem 2. Let G_1 and G_2 are two graphs representing two disjoint 2-connected sub graph. If G_1 and G_2 are connected via two links $l_1$and l_2, then $G_1 \cup G_2 \cup l_1 \cup l_2$ is also a 2-connected network.

Proof: It is also trivial that if two 2-connected networks are connected via two links, then resultant network would also be a 2-connected network. As every node in G_1 have two disjoint paths to reach every other node in G2.

A network is represented by a 2-connected sub network N_1 and a spanning tree N_2. And all nodes of the spanning tree N_1 having degree one are connected to at least two nodes of sub network N_1.

Theorem 3. Let G_1 be a graph representing a 2-connected network. G_2 be a graph representing a network in which all nodes are connected. Let v_1, v_2,…., v_k are the k nodes in G_2 who have degree one. If these k nodes are connected to at least two nodes of G_1 via links $l_1, l_2, \ldots, l_k$, then, G_1 u G_2 u l_1 u l_2 u l_3……..u l_k is also a 2-connected network.

Proof: This is similar to theorem 2 except G_2 is not a 2-connected network. Once all nodes having degree one are connected to at least two nodes of G_1, it means that there are two disjoint paths to reach G_1 from the nodes of G_2 because all nodes in G_2 are reachable to those nodes whose degree are one and vice-versa.

More than one disjoint sub networks representing the cycles (containing all nodes of the network) and if these cycles are connected to other cycles via at least two links. Theorem 2 can be used as a basis for proving this representation.

More than one of disjoint sub networks representing cycles (containing subset of the nodes of the network) and nodes which are not the part of the cycles either represent a single sub network or multiple sub network. Theorem 2 and Theorem 3 can be used for proving this representation.

The representation described above can be found by traversing the given network from any randomly selected node. If the traversed network results into any of the representations discussed above, network is said to be a 2-connected network.

2.1.3 k-connected Network

A network is said to be a k-connected network if there are at least k disjoint paths for each node pair. k may be either even or odd. The necessary and sufficient conditions that need to be checked for k-connected networks are given below:

Necessary Conditions:

Minimum number of links should be (N*k)/2.
Degree (v_i) >= k for all v_i.

Sufficient Conditions:

If any network satisfies the necessary conditions described above, then the given network would be a k-connected network if it also satisfies the conditions listed in step1, step 2 and step 3 as given below:

Step 1. Traverse the given network and check whether the traversed network satisfies any of sufficient condition described for a 2-connected network (using marked links only). If yes, then increase the connectivity k by 2 and repeat step1 otherwise go to step 2.

Step 2. If the traversed network at any point does not satisfy the condition in step 1, then start joining the traversed disjoint sub network until a single sub network is formed. While joining the two disjoint sub networks, it is also checked whether each disjoint sub network is joined to other via at least k+2 unmarked links. If at any point, number of links joining two disjoint sub networks are less than k+2 then go to step 3, otherwise Set k= k+2

and go to step 1.

Step 3. This is the case for a network having odd connectivity. If every traversed node is incident on some unmarked links, then check whether it is possible to form a spanning tree using these unmarked links. If yes, set k = k+1. Otherwise, check whether it is possible to form two disjoint sub networks using both marked links and unmarked links. If yes, check how many links (marked or unmarked) are there which connect these two sub networks. If there exist either k+1 or more links, then set k =k+1. Resultant network is a k-connected network.

2.2 Proposed Algorithm for Measuring Connectivity of a Network.

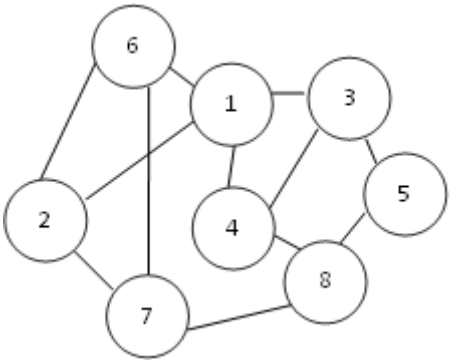
Input: Network topology represented by Graph G (V, E)
Output: link connectivity of the network
Initialize G` =0, k =0
Begin
Calculate the degree of all nodes.
Set Network_degree = degree of node whose has lowest degree among all nodes.
If (Network_ degree = 1)
Apply either breadth first search or depth first search algorithm to find out whether all nodes are connected. And if all node are connected, then
Set k=1;
Go to step 5.
While (Network_degree >= 2)
Start traversing the network until all nodes are traversed.
Mark all traversed links.
If the sub network represented by the marked link satisfies any of the sufficient conditions for a 2-connected network described in section 2.1.2.
Then,
Set k = k+2;
Set Network_degree= Network_degree-2;
Else
If traversing results into more than one disjoint sub networks, then start joining these disjoint sub networks using the marked/unmarked links until a single sub network is formed. Number of links which join two disjoint sub networks should be more than or equal to k+2. If at any point, number of links joining two disjoint sub networks are less than k+2 then go to step 4, otherwise
Set k= k+2;
Network_degree= Network_degree – 2;

If every node is incident on some uunmarked links, then check whether it is possible to form a spanning tree using these unmarked links. If yes, set k = k+1 and go to step 5.
Else
Check whether it is possible to form two disjoint sub networks using both marked links and unmarked links. If yes, check how many links (marked or unmarked) which connect these two sub network. If there exists at least k+ 2 links, then set k =k+1.
The network is a k-connected network.
End

3. Illustrative Examples

In this section, we will illustrate the proposed approach using various examples. We have illustrated all representations of a 2-connected network in example 1. The network in example 2 describes the case when a given network's nodes have degree 3, still network is not a 3-connected network. Details description for checking the sufficient conditions for a 4- connected network is given in example 3 as the network satisfies the necessary conditions for a 4-connected network.

Example 1. Consider a network having 8 nodes, 12 links with its adjacency matrix shown in figure 2 given below.



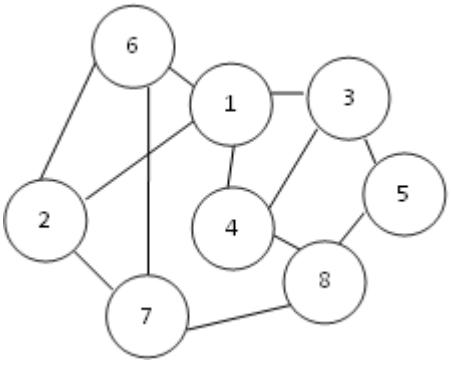
0	1	1	1	0	1	0	0
1	0	0	0	0	1	1	0
1	0	0	1	1	0	0	0
1	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1
1	1	0	0	0	0	1	0
0	1	0	0	0	1	0	1
0	0	0	1	1	0	1	0

Figure 2. Given network and its adjacency matrix

Traversed network representation depends on the starting node and the heuristic used for traversing. Hence, if we traverse the network in figure 2, it

may results into any of the four grouping of sub networks as shown in figure 3, figure 4, figure 5 and figure 6 which are explained below:

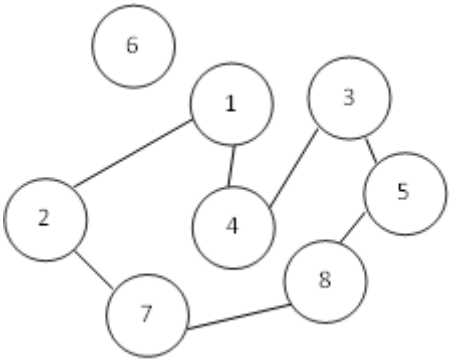
The sub network in figure 3 results when we start the traversing from node 7 and always choose highest numbered node among the adjacent nodes. Step by step description for getting the sub network in figure 4 is as follows: Let's choose node 7 as a starting node. Mark node 7 as traversed node. Node 8 and node 6 are adjacent to node 7. We chose highest numbered node first among the adjacent nodes. So, node 8 and link 7-8 are marked as traversed in the network. Links are symmetric as network is represented as an undirected graph. In the next step, find out the adjacent nodes to the current traversed node i.e. node 8. Node 5 and node 4 are the adjacent node to node 8. So mark node 5 and link 8-5 as traversed in the network. Node 5 is adjacent to only one node which is not traversed yet i.e. node 3. Mark node 3 and link 5- 3 as traversed. Node 3 is adjacent to node 1 and node 4. Mark node 4 and link 3-4 as traversed. Next, we mark node1 and link 4-1. Then, node 6 and link 1-6 are marked. After that, node 2 and link 6-2 are marked. Now all nodes are traversed at this point. Check whether there is a link between the starting node and current traversed node. If yes, mark that link also. So link 2-7 is marked as traversed. Resultant traversed network and adjacency matrix containing marked (shown bold, italic and underline) and unmarked links are shown in figure 3 and it forms a cycle, which itself a 2-connected network. Hence the given network is a 2-connected network.



0	1	1	***1***	0	***1***	0	0
1	0	0	0	0	***1***	***1***	0
1	0	0	***1***	***1***	0	0	0
1	0	***1***	0	0	0	0	1
0	0	***1***	0	0	0	0	***1***
1	***1***	0	0	0	0	1	0
0	***1***	0	0	0	1	0	***1***
0	0	0	1	***1***	0	***1***	0

Figure 3. Network and Adjacency Matrix after traversing

It is not necessary that the given network always results into a cycle as we have already said that it depends on the starting node and heuristics used for traversing. For example, if we traverse the same network choosing node 1 as a starting node, any adjacent node to current node becomes the new current node. Step by step description for constructing the sub networks in figure 4 is as follows: choose node 1 as a starting node. Now, node 6 and node 2 are adjacent to node 1. Suppose node 2 is chosen. Mark node 2 and link 1-2 as traversed. Suppose node 7 is chosen next. Mark node 7 and link 2-7. Following this manner, we come across the node 4. At this point, node 6 is the only node which is not traversed yet. But it is not adjacent to current traversed node 4. As there is a link between starting node 1and more recently traversed node 4. So, mark link 3-1 as traversed. Now start traversing from node 6 and mark node 6 as traversed. As there is no node which is left for traversing. Hence, traversed network results into two sub networks i.e. one cycle containing nodes 1,2,7,8,5,4,3 and one single node 6. Adjacency matrix after traversing is shown in figure 4.



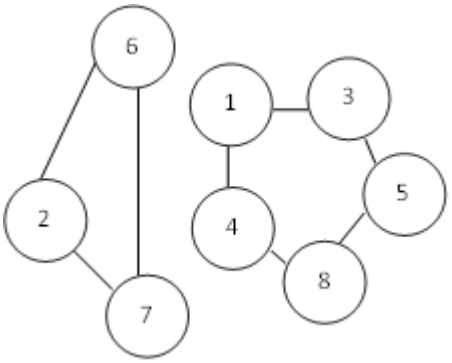
0	1	1	***1***	0	***1***	0	0
1	0	0	0	0	1	***1***	0
1	0	0	***1***	***1***	0	0	0
1	0	***1***	0	0	0	0	1
0	0	***1***	0	0	0	0	***1***
1	1	0	0	0	0	1	0
0	***1***	0	0	0	1	0	***1***
0	0	0	1	***1***	0	***1***	0

Figure 4 Traversed Network and Adjacency Matrix after Traversing

To satisfy the 2-connectivity, there should exist two unmarked links which connect node 6 and the cycle. If we see the adjacency matrix, there are three unmarked links i.e. 6-1, 6-2, 6-7 which

connect node 6 to the cycle. Hence the sufficient condition is also satisfied and network in figure 4 is a 2-connected network once we add the any of two links among links 6-1, 6-2, 6-7.

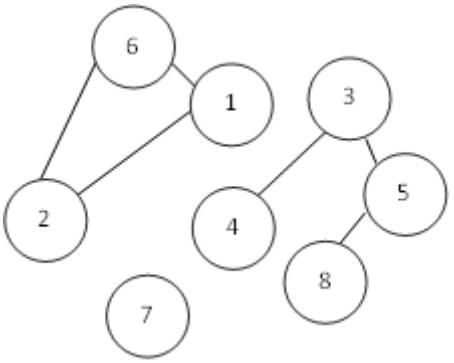
If traversing the same network results into two disjoint sub networks as shown in figure 5, we check whether there exists at least two unmarked links which connect these two sub networks. In this example three links 1-6, 1-2, 7-8 which connect the two disjoint sub networks. Hence the sufficient condition for this is also satisfied. So Network is a 2-connected network once we add the any of the two links among links 1-6, 1-2, 7-8.



0	1	***1***	***1***	0	1	0	0
1	0	0	0	0	***1***	***1***	0
1	0	0	1	***1***	0	0	0
1	0	1	0	0	0	0	***1***
0	0	***1***	0	0	0	0	***1***
1	***1***	0	0	0	0	***1***	0
0	***1***	0	0	0	***1***	0	1
0	0	0	***1***	***1***	0	1	0

Figure 5. Traversed Network and its Adjacency Matrix after Traversing.

Another possibility in which traversed network may result into sub networks is shown in figure 6. There are two disjoint sub network S1 (1, 2, 6), S2 (3, 4, 5, 8) which form cycle and one sub network S3 (7) which do not form a cycle. In this case, we check whether there exist at least two links which connect sub network S1 and sub network S2. Looking at the adjacency matrix in figure 6, there are two links 1-3 and 1-4 which connect sub network S1 and sub network S2. Next we check whether there exist two links which connect node 7 to S1 or S2 or both. And if we look at the adjacency matrix, there are three links 7-8, 7-6, 7-2 which connect node 7 to S1 and S2. Hence network is a 2-connected network.

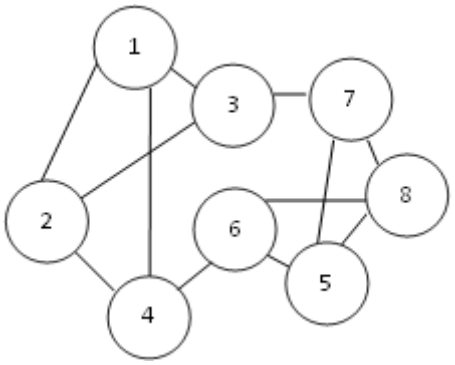


0	***1***	1	1	0	***1***	0	0
1	0	0	0	0	***1***	1	0
1	0	0	***1***	***1***	0	0	0
1	0	***1***	0	0	0	0	***1***
0	0	***1***	0	0	0	0	***1***
1	***1***	0	0	0	0	1	0
0	1	0	0	0	1	0	1
0	0	0	***1***	***1***	0	1	0

Figure 6. Traversed Network and its Adjacency Matrix after Traversing.

Hence, it is proved that using the sufficient conditions described in section 2.12. for a 2-connected network, we can measure the 2-connectivity of a given network.

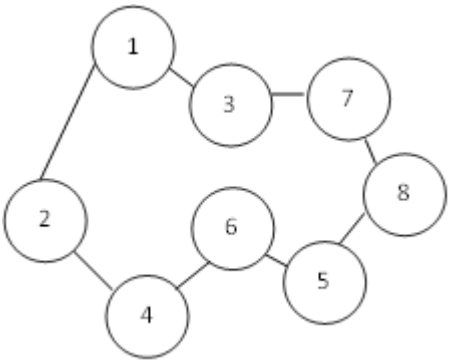
Example 2. Consider a network having 8 nodes and 12 links network with its adjacency matrix shown in figure 7.



0	1	1	1	0	0	0	0
1	0	1	1	0	0	0	0
1	1	0	0	0	0	1	0
1	1	0	0	0	1	0	0
0	0	0	0	0	1	1	1
0	0	0	1	1	0	0	1
0	0	1	0	1	0	0	1
0	0	0	0	1	1	1	0

Figure 7. Given Network and Its Adjacency Matrix.

The given network satisfies the necessary conditions of a 3-connected network and also satisfies the equation 2 given in section 1 but it is not a 3-connected network. As k is odd, so it is first checked for k-1 connectivity i.e. 2. We start traversing the network considering node 1 as a starting node and select the adjacent nodes randomly. Traversed network may result into various grouping of sub networks. For example, if traversing results in to the sub network shown in figure 8 along with its adjacency matrix.

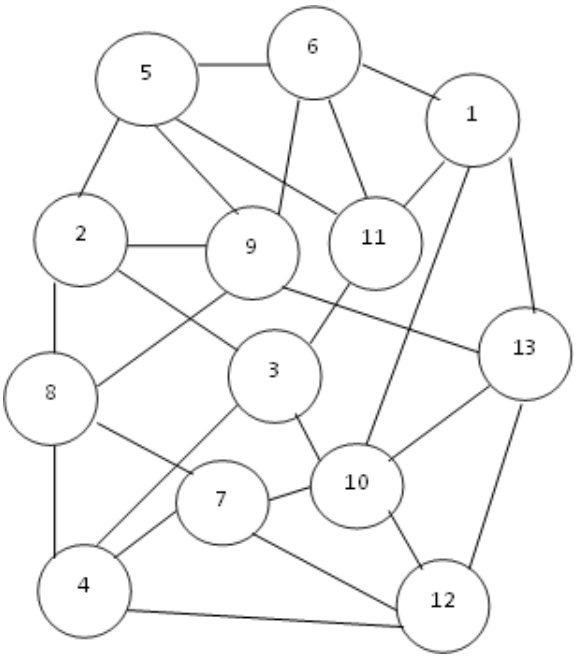


0	***1***	***1***	1	0	0	0	0
1	0	1	***1***	0	0	0	0
1	1	0	0	0	0	***1***	0
1	***1***	0	0	0	***1***	0	0
0	0	0	0	0	***1***	1	***1***
0	0	0	***1***	***1***	0	0	1
0	0	***1***	0	1	0	0	***1***
0	0	0	0	***1***	***1***	1	0

Figure 8. Traversed network and Adjacency Matrix after traversing.

The sub network in figure 8 forms a cycle which is itself a 2-conencted network. Hence condition in step 1 of section 2.1.3 is satisfied. Next we check whether unmarked links are able to form a spanning tree. There are only four remaining links and spanning tree requires at least 7 links. It is not possible to form a spanning tree using four unmarked links for a network of 8 nodes. So, we check whether it is possible to form disjoint sub network using both marked and unmarked links. In this example, it is possible to form two disjoint sub network or cycles i.e. C1 (1-4-2-3-1) and C2 (8-6-5-7-8) and if we see in the adjacency matrix, there are only two marked links 3-7 and 4-6 which connect these two sub networks. Hence the condition of k links for this situation is not satisfied as there are only k-1 links and failure in these two links results into a disconnected network. Network in figure 7 is not a 3- connected network even though it satisfies the equation 2 discussed in section 1.

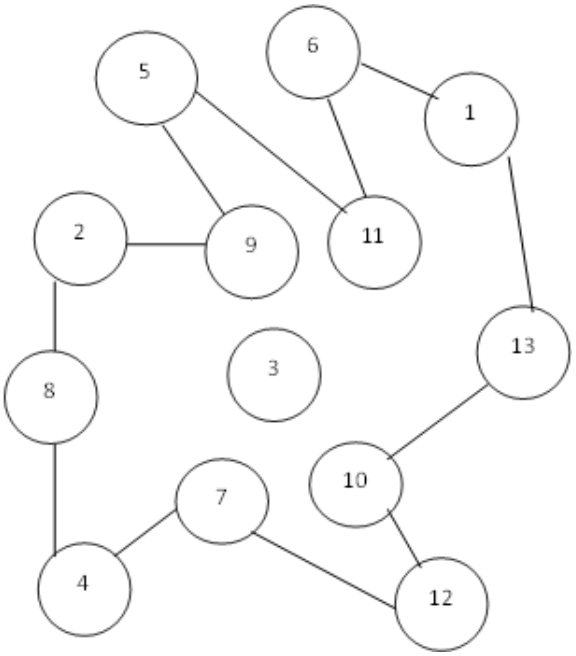
Example 3. Consider a network having 13 nodes and 27 links with its adjacency matrix as shown in figure 9.



0	0	0	0	0	1	0	0	0	1	1	0	1
0	0	1	0	1	0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	1	1	0	0	0	1	0
0	1	0	0	0	1	0	0	1	0	1	0	0
1	0	0	0	1	0	0	0	1	0	1	0	0
0	0	0	1	0	0	0	1	0	1	0	1	0
0	1	0	1	0	0	1	0	1	0	0	0	0
0	1	0	0	1	1	0	1	0	0	0	0	1
1	0	1	0	0	0	1	0	0	0	0	1	1
1	0	1	0	1	1	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	0	0	0	0	0	1	1	0	1	0

Figure 9 Given Network and its Adjacency Network.

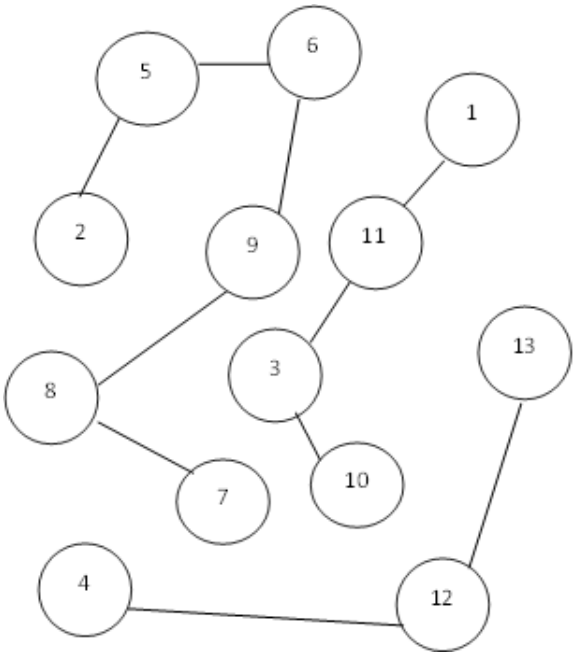
The given network satisfies the necessary conditions for a 4-connected network. The network when traversed using heuristic in which adjacent nodes are selected randomly, results into the representation shown in figure 10.



0	0	0	0	0	***1***	0	0	0	1	1	0	***1***
0	0	1	0	1	0	0	***1***	***1***	0	0	0	0
0	1	0	1	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	***1***	***1***	0	0	0	1	0
0	1	0	0	0	1	0	0	***1***	0	***1***	0	0
1	0	0	0	1	0	0	0	1	0	***1***	0	0
0	0	0	***1***	0	0	0	1	0	1	0	***1***	0
0	***1***	0	***1***	0	0	1	0	1	0	0	0	0
0	***1***	0	0	***1***	1	0	1	0	0	0	0	1
1	0	1	0	0	0	1	0	0	0	0	***1***	***1***
1	0	1	0	***1***	***1***	0	0	0	0	0	0	0
0	0	0	1	0	0	***1***	0	0	***1***	0	0	1
1	0	0	0	0	0	0	0	1	***1***	0	1	0

Figure 10 Traversed Network and its Adjacency Matrix after Traversing.

Traversed network in figure 10 contains a cycle C1 containing the subset of nodes in the network and a single node 3 which is not connected. To make it 2-connected, connect node 3 to any two nodes of the cycle C1. If we m,lark link 3-4 and link 3-2. Resultant sub network is a 2-connected network. Set connectivity = 2. The network is traversed again using unmarked links which is shown in figure 11.



0	0	0	0	0	***1***	0	0	0	***1***	***1***	0	***1***
0	0	1	0	***1***	0	0	***1***	***1***	0	0	0	0
0	1	0	1	0	0	0	0	0	***1***	***1***	0	0
0	0	1	0	0	0	***1***	***1***	0	0	0	***1***	0
0	***1***	0	0	0	***1***	0	0	***1***	0	***1***	0	0
1	0	0	0	***1***	0	0	0	***1***	0	***1***	0	0
0	0	0	***1***	0	0	0	***1***	0	1	0	***1***	0
0	***1***	0	***1***	0	0	***1***	0	***1***	0	0	0	0
0	***1***	0	0	***1***	***1***	0	***1***	0	0	0	0	1
1	0	***1***	0	0	0	1	0	0	0	0	***1***	***1***
1	0	***1***	0	***1***	***1***	0	0	0	0	0	0	0
0	0	0	***1***	0	0	***1***	0	0	***1***	0	0	***1***
1	0	0	0	0	0	0	0	1	***1***	0	***1***	0

Figure 11 Traversed Network and Adjacency Matrix after Traversing.

Traversed network in figure 11 is a grouping of three sub networks i.e. C1 (1-11-3-10) which form a cycle, C2 (4-12-13) and C3 (7-8-9-6-5-2). To verify the 2-conenctivity, check whether there exists at least two unmarked or marked links which can connect node 2 and node 7 of C3 to the nodes of C1. Links 2-3 and link 7-10 are marked and C1 and C3 result into a single 2-connected sub network. Now check whether there exists at least two unmarked/marked links which connect node 4 and node 13 to newly formed 2-connected sub network. Link 4-3 and 9-13 are two

unmarked links which connect C2 to newly formed 2-conencted network and satisfy the 2-conencvity. Hence, it proved that network given in figure 9 is a 4-connected network.

4. Performance Evaluation

Running time of the algorithm proposed in the section 2.2 for measuring the connectivity depends on the running time required to traverse the network i.e. O (V+E). Once the network is traversed, next task is to join the entire connected disjoint component found in traversing the network. If during traversing m connected components are formed then join operation will be called m-1 times. Running time of joining all components are O (m*E). For a network whose network degree is k, this whole process is called k/2 times. So total running time of the algorithm measuring the connectivity would be O (k/2(V+E) + (m*E)) which is smaller than the Karp Edmond algorithm whose time complexity is $O(VE^2)$.

5. Conclusion and Future Work

In this paper, we have proposed an approach for measuring the connectivity of a given network. The proposed approach is robust enough that it gives exact connectivity of a network in all situations without finding out all disjoint paths between all node pairs.

References:

[1] Tomas Fencl, Pavel Burget, Jan Bilek, "Network topology design" control engineering practice, 2011 Elsevier Ltd.
[2] Ewa Szlachcic and Jacek Mlynek " Efficiency Analysis in Communication Network Topology Design" Fourth IEEE International Conference on Dependability of Computer Systems, 2009.
[3] K.N. Kamlesh, Srivatsa S.K , "Topological Design of Minimum Cost Survivable Computer Communication Networks: Bipartite Graph Method" (IJCSIS) International Journal of Computer Science and Information Security, Vol. 3, No. 1, 2009.
[4] Mostafa Abd-El-Barr'," Topological Network Design: A Survey" Journal of Network and Computer Applications, 2008 Elsevier Ltd.
[5] T.Fencl, P.Burget and J.Bilek "Network Topology Design" Proceeding of the 17th World Congress, The International Federation of Automatic Control Seoul Korea, July, 2008.
[6] Deng Zili, Yu Nenghai, Li Zheng" Designing Fault Tolerant Networks Topologies based on Greedy Algorithm " IEEE Third International Conference on Dependability of Computer Systems DepCoS-RELCOMEX 2008.
[7] Ewa Szlachcic "Fault Tolerant Topological Design for Computer Networks" Proceeding of the IEEE International Conference on Dependability of Computer Systems, 2006.
[8] Mostafa Abd-El-Barr', Ahmer Zakir, Sadiq M. Sait, and Abdulaziz Almulhern" Reliability and Fault tolerance based Topological Optimization of Computer Networks - Part I: Enumerative Techniques" IEEE 2003.
[9] Mostafa Abd-El-Barr', Ahmer Zakir, Sadiq M. Sait, and Abdulaziz Almulhern" Reliability and Fault tolerance based Topological Optimization of Computer Networks - Part II :Iterative Techniques"IEEE 2003.
[10] H. Sayoud, K. Takahashi, And B. Vaillant "Designing Communication Networks Topologies Using Steady-State Genetic Algorithms" Ieee Communications Letters, Vol. 5, No. 3, March 2001
[11] Berna Dengiz, Fulya Altiparmak, and Alice E. Smith, "Local Search Genetic Algorithm for Optimal Design of Reliable Networks", Ieee Transactions On Evolutionary Computation, Vol. 1, No. 3, September 1997.
[12] L.W. Clarke and G. Anandalingam, "An integrated system for designing minimum cost survivable telecommunication networks" IEEE Trans. System, Man and Cybernetics. Part A, Vol 26, No 6, pp 856-862, NOV 1996.
[13] Mario Geria and Leonard Kleinrock " On the Topological Design Of Distributed Networks" IEEE Trans. On communication, Vol. COM-25, No 1, January 1977.
[14] K. Steiglitz, P. Weiner and D.J. Kleitman "The Design of Minimum-Cost Survivable Networks" IEEE Transaction on Circuit Theory, Vol. Cr-16, November 1969.
[15] Samuel Pierre et.al "Topological Design of Computer Communication Networks using Simulated Annealing" Engg. Application. Artif. Intell. Vol. 8 No 1 pp 61-69, 1995.
[16] Edmands, Jack karp, Richard M. (1972) "Theoretical Improvement in the algorithmic efficiency for network flow problem" Journal of the ACM (Association for computing Machinery)" 19(2): 248-264.
[17] P.C. Saxena, Sangeeta Sabharwal, Maneesha "P.C. Saxena, Sangeeta Sabharwal, Maneesha, "An Efficient Constructive Approximation Approach to Design a K-Connected Network Topology" Second International Conference on Computers, Communication, Control and Information Technology (C3IT-2012)" Volume 4, 2012 pages 140-144, Procedia Technology, Elsevier.