Generating sets for finding faults in the networks

Dr. H.B. Walikar[1] Dr. Ravikumar H. Roogi[2] Marriswamy Ramegowdgari[3], Dr. S. V. Shindhe[4]

Abstract

This paper describes generating sets the application intended for to find error and faults in the network. Now a day's all networks are having more congestion than older generation networks. So to avoid congestion in the network we are implementing new approach to find errors and faults in the network.

Keywords

Graph, Set, Tree, Networks, Divide and conquer algorithm.

1. Introduction

While forming reliable communication networks, we must guarantee that, after failure of a node or links, the surviving network still allows communication between all other nodes by choosing alternate path which gives strict requirement on the connectivity of the corresponding graph. A general network design problem which requires the underlying network to be resilient to link failures is known as the edge-connectivity survivable network design problem.

In this paper we are implementing finding the faults in network using divide and conquer technique, the given graph collects set of vertices and generates even number set of vertices and odd number set of vertices for some pre defined properties.

Divide and conquer method is a top-down technique for designing algorithms which consists of dividing the problem into smaller sub problems hoping that the solutions of the sub problems are easier to find. The solutions of all smaller problems are then combined to get solution for the original problem.

2. Basic Definitions

2.1 Algorithm

An algorithm is a sequence of unambiguous instruction for solving problem, for obtaining a required output for any legitimate input in a finite amount of time. [1]

2.2 Graph theory

A graph G consists of two sets V and E. The set V is finite, nonempty set of vertices; these fairs are called edges. The notations V (G) and E (G) represent the sets of vertices and edges, respectively, of graph G. we also write G= (V, E) to represent a graph. [2]

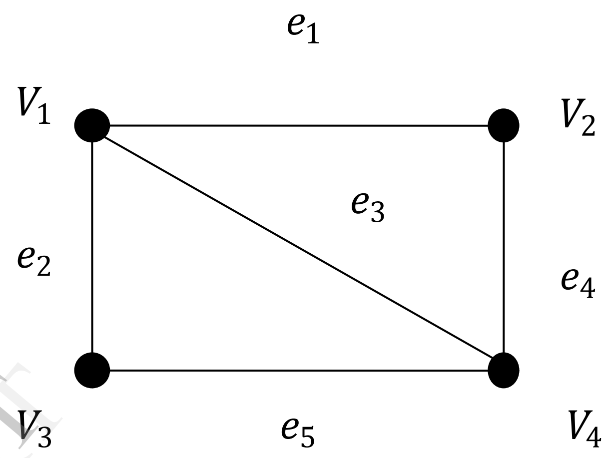

Fig.1

2.3 Set theory

A set is a collection of distinguishable objects, called its members or elements. If an object X is a member of a set S, we write $X \in S$, (read "X is a member of S" or, more briefly, "X is in S"). If X is not member of S, We write $X \notin S$. We can describe a se by explicitly listing its members as a list inside braces. For example, we can define a set S t contain precisely the numbers 1, 2, and 3 by writing $S = \{1, 2, 3\}$. Since 2 is the member of the set S, we can write $2 \in S$, and since 4 is not a member, we have $4 \notin S$.[3]

Ex: - $N = \{0, 1, 2, 3 \ldots\}$, the set of **natural numbers.**

2.4 Subset

If all the elements of set A are contained in set B, that is, if $X \epsilon A$ implies $\chi \in B$ then we write $A \subseteq B$ and say the A is subset of B.[3]

2.5 Networks

A network consists of two or more computers that are linked in order to share resources (Such as printers and CDs), exchanging files , or allow electronic communications .The Commuters on network may be linked through cables, telephone lines, radio waves, satellites or infrared light beams. [4]

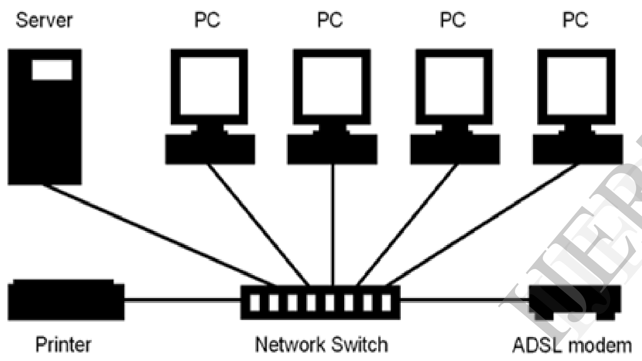

Fig.2

Example: A wired client-server network

2.6 Trees

A tree is finite set of one or more nodes such that there is a specially designated node called the root and reaming nodes are partitioned into n≥0disjoint sets $T_1 .....T_n$, where each of these sets is tree. The sets $T_1 ,.... T_n$ are called the sub trees of the root.[5]

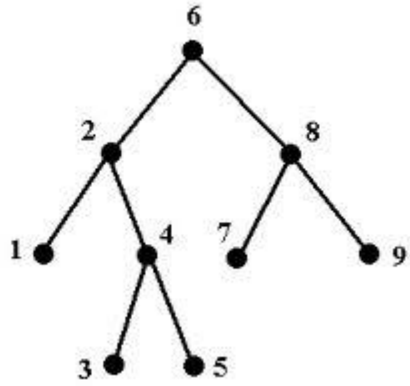

Fig.3

3. Proposed Algorithm

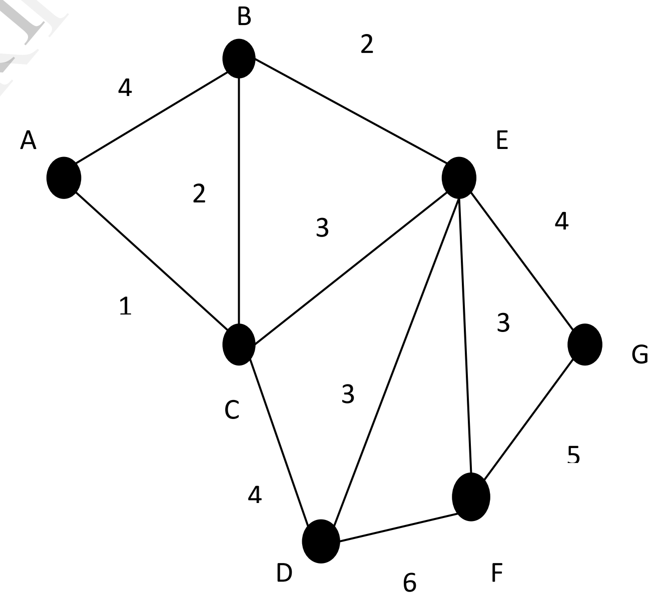

Fig.4

$$\begin{bmatrix} 0 & 4 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 2 & 0 & 2 & 0 & 0 \\ 1 & 2 & 0 & 4 & 3 & 0 & 0 \\ 0 & 0 & 4 & 0 & 3 & 6 & 0 \\ 0 & 2 & 3 & 3 & 0 & 3 & 4 \\ 0 & 0 & 0 & 6 & 3 & 0 & 5 \\ 0 & 0 & 0 & 0 & 4 & 5 & 0 \end{bmatrix}$$

Fig.5

Graph with weighted vertices
Collecting all the weights of the given graph consider as a universal set.
U= {0, 1, 2, 3, 4, 5, 6}

Property:-

Select even weight and odd weight edges from universal set then prepare subsets as below.
Even Set= {2, 4, 6}
Odd Set= {1, 3, 5}

3.1 Generating sets for finding faults in the networks Algorithm as follows

```
Step 1. Start
Step  2. Input Graph
Step 3. Collect set of all the weighted edges
Step 4. Count the number of edges and also count the
number of Odd and even edges.
Step 5. if((n%2==0)&&(n==working))
                    even=even+1;
else
                    odd=odd+1
Step 6. display two subsets of odd and even
Step 7. if(even==count)
                {
                    no fault in even edge
                    {
                         if(odd==count)
                         {
                              no fault in odd edge
                         }
                         else
                         {
                              fault is odd edge so go to
                              correct odd edge lines.
                              }
                              Correct the algorithm
Step 10. Stop
```

4. Conclusion

We are representing the Generating sets for finding faults in the networks. This is very simple to understand and implement. This approach executes sequentially and here we are using sets to find faults also it is time saving, because we are implementing using divide and conquer technique so this algorithm takes half of the time comparing with existing algorithm.

5. References

[1] A.M. Padma reddy Desgin and analysis of algorithms 6th Edition, 2012.
[2] Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran: Fundamental of computer algorithms, 2nd Edition, University press, 2007.
[3] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest "Introduction to Algorithms", PHI, Fourth Printing 2001.
[4] F Harary, "Graph Theory", Addison – Wesley, Reading, Mass, 1969
[5] http://fcit.usf.edu/network/chap1/chap1.html.
[6] Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran: Fundamental of computer algorithms, 2nd Edition.graph augmentation", J. of Algorithms, 14, pp. 214-225,1993.
[7] Jaewon Oh, Iksoo Pyo, Massord Pedram,"Constructing Minmal Spanning / Steiner Trees With Bounded Path Length", pp. 244-249,1996.
[8] R.Jyothi, B.Raghavachari, S.Varadarajan,"A 5/4- approximation algorithm for minimum 2-edge-connectivity", In SODA, pp. 725-734, 2003.
[9] H.Nagamochi,"An approximation for finding a smallest 2-Edge connected subgraph containing a specified spanning tree", Discrete Applied Mathematics, 126, pp.83-113,2003.
[10] Dominik Alban Scheder,"Approaches to Approximating the minimum weight k-Edge