Research Paper on Finding All Paths and Critical Nodes in a Network

Rashi Tanwar[1], Shikha Verma[2]

[1]Assistant Professor, Department of Computer Science & Applications, Sanatan Dharma College, Ambala Cantt
[2]Assistant Professor, Department of Computer Science & Applications, Sanatan Dharma College, Ambala Cantt

***

Abstract: Multipath Routing and critical node are two main factors that play very important role in network performance. In this paper multipath and criticalnodes in a network are discovered. Multipath routing is the technique of choosing multiple alternate paths within a network, which gives a variety of benefits such as increased bandwidth, improved security or fault tolerance. The multiple paths computation may be overlapped, node-disjointed or edge-disjointed with each other.

A Critical node is node whose in degree and out degree is greater than other nodes. That node is treated as a critical node because if all nodes connected to that critical node try to send data at the same time it degrades the performance of network, so to overcome this problem multipath concept is introduced. In this dissertation there is a discussion of such critical nodes.

To discover allpath and critical nodes a program is implemented in MatLab. Algorithm of program and flow chart of algorithm are also discussed.An algorithm has been proposed to find all paths and critical nodes in network.

Keywords: Routing ,shortest path,hoffman

Introduction

Routing is the way of selecting best paths in every network. In past, the routing was also mean to forward network traffic among different networks. But this latter function is much better described as simply forwarding. Routing is used for different kinds of networks, including the electronic data networks (such as the Internet), transportation networks and telephone network (circuit switching). This paper is concerned primarily with routing in electronic data networks using packet switching technology.

[8] The critical node is that node whose disruption, is immediately degrades the performance of a network. If a node is critical, then attention must be paid to that node to avoid its failure or removal of a network. So how to confirm critical nodes in a network is to predict the network partition. A critical node is the most precious node in a network. The parameters to evaluate a node is packet delivery ratio, end-to-end delay and throughput.

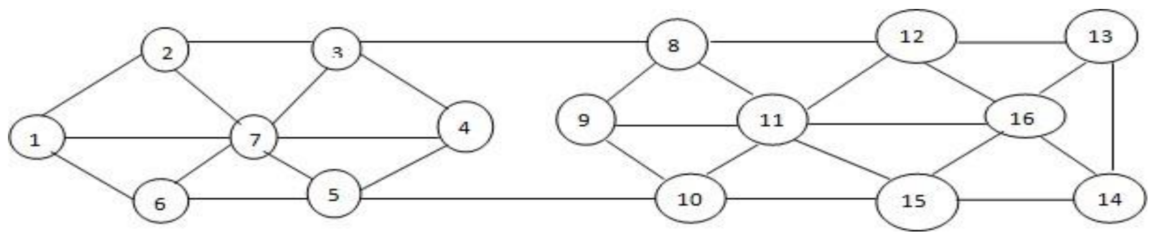

Figure 1.13 Node 8 & 10 are sufficient to destroy the whole network so, node 8 & 10 are critical nodes

Another definition of critical node is node whose in degree is greater than other nodes that node treated as a critical node because if all nodes connected to that critical node tries to send data at same time it degrades the performance of network, so to overcome this problem multipath concept is introduced. In this dissertation there is a discussion of such critical nodes or we can say node whose in degree and out degree is greaterthan other nodes is treated as critical node.

In figure 1.13 node 7 & 11 are critical nodes because 6 nodes are connected to these nodes

LITERATURE SURVEY

2.2 Different KSP algorithms

2.2.1 YEN K-SHORTEST PATH ALGORITHM

The shortest path (containing p<=n nodes) is found using a standard shortest path algorithm (e.g. Dijkstra's[36]) and placed in the result list(YEN'S list A). Yen [37] takes every node in the shortest path except terminating node and calculates another shortest path(spur) from eachselected node to the terminating node . For each such node, the path from the start node to the current node is the root path. Two restrictions are placed on the spur path:

1) It must not pass through any node on the root path(i.e. loop less)

2) It must not branch from the current node on any edge used by a previously found [k-] shortest path with the same root.

Node and edge marking is used to prevent the spur paths from looping or simply following the route of a previous [k-] shortest path. If a newspur path is found it is appended to the root path for that node, to form a complete path from start to end node, which is then a candidate for the next KSP. All such paths are stored (YEN'S list B) and the shortest remaining unselected path is selected as the next KSP, and transferred to the result list (YEN'S list A). The same process is repeated, calculating a spur path from each node in each new KSP, until the required numbers of KSPs have been found.

Various improvements to yen's algorithm have been commented in papers over the years. The most significant improvement can be made withthe use of heap to store YEN'S list B, giving an improvement performance although the overall computational complexity does not change. A further improvement is checking for non-existent spur paths (i.e. where the root path exists, but all spur paths from the root have been used previously).

If list B contains enough shortest paths all of the same minimum length, to produce all the required paths, it is not necessary to extract each path in turn and perform the above calculations; only to extract the required number of paths and place them directly in list A, as no other shorter paths will be found.

2.2.2 LAWLER

Lawler [38] presents a modification to Yen's algorithm, such that rather than calculating and then discarding any duplicate paths, they are simply never calculated.

The extra paths occur due to the calculation of spur paths from nodes in the root of the KSPs. Lawler points out that it is only necessary to calculate new paths from nodes that were on the spur of previous KSP. Consider a node on the root of a path; Yen calculates a spur path fromthis node every time a new KSP is required. The path calculated each time will be the same provided that no extra edge marking has taken place at this node. Extra edge marking will only take place if a KSP has been chosen that branches from this node i.e. this node is on the spur.When one of these paths becomes a KSP, then all paths from the root of that KSP will have already been calculated and stored in the heap of prospective KSPs (YEN'S list B).

It is therefore necessary to keep a record of the node where each path branched from its parent. This node marks the point from where the calculation of spur paths starts. The increased efficiency of Lawler can be explained therefore as, when finding more than two KSPs, the nextKSP will on average branch from the middle of the path and so approximately a 50% improvement in speed achieved over Yen.

2.2.3 KATOH

This algorithm [39] is claimed to be faster than Yen due to the way that the previous [k-] shortest path is broken down for the calculation of the next KSP. This path is broken into three sections rather than the (p-1<=n-1) sections for Yen. Two shortest-path trees are calculated for each section, one from the start node and another to the end node.

The three sections used in the algorithm are derived from the previous KSP by recording (exactly as in Lawler's algorithm) the point that eachpath branches from its parent. The sections are:

1) All nodes after the branch (P_a)

2) The branch node (P_b)

3) All nodes before the branch (P_c)

The algorithm then calculates one path from each of these sections. Restrictions are placed on this path such that it must deviate from its parentat some point and must not follow other previous KSPs. Each path is calculated start node in each section and from the given end node. The possible shortest paths are represented by paths across these two trees, where each path goes between the trees via a common node or a singleedge.

2.2.4 HOFFMAN

Hoffman's algorithm [40] requires that the shortest path between the two nodes has been found and that the shortest-path tree from all nodes to the terminal node is known.

In Yen and Lawler, the spur paths are calculated from each node O(n) on the previous[k-] shortest path in turn, using a shortest-path algorithm$O(n^2)$ with node marking, making a total time of $O(n^3)$. In contrast, Hoffman calculates the shortest-path tree from all nodes to the end node $O(n^2)$ at the start of execution. Then, to find the next shortest path, it is only necessary to search from each node O(n) on the previous KSP spur to every other node O(n), making the total time of $O(n^2)$. Each path is made up of three sections:

1) The start node to the selected(i.e. branching) node

2) The edge from the selected node to the new node

3) The branch of the shortest-path tree from the new node to the terminating node

Edge marking does not need to be used, as edge can be ignored when searching through those from the selected node. The shortest path from each selected node is placed in the heap. The next KSP is then the path with the shortest total length from the start node along the tree to the end node. As the shortest-path tree is known beforehand, it is only a matter of searching at most n*n edges to determine each KSP.

An important point to note in this path generation is that the paths evolve by the addition of exactly one edge to the existing root and spur treepaths. This gives rise to a complication that is automatically excluded by Yen's and Lawler's algorithms such that it is possible (and quite likely) that looping paths will be generated. These paths are essential to the operation of the algorithm and must be kept like any other previousor prospective KSP. The looping path obtained will follow a previous KSP, but with the addition of a loop making the path slightly longer. The importance is that elementary paths may evolve from the looping section, again by the addition of one edge. The way looping paths are generated is determined by the geometry of the network and whether it obeys the rules of Euclidean geometry. The generation of looping paths adds an extra overhead to this algorithm, which is not easily quantifiable, but will depend on the network features.

A further speed improvement is available with this algorithm, as it is not necessary to calculate the full path details (i.e. creating the route) until the path is finally selected as a KSP and removed from the heap. Additionally, the same shortest-path tree to the terminal node can be used for all-1 KSP calculations.

Proposed work

In this chapter we describe how objective function is achieved. Three functions are used to find critical nodes and all paths. These functions areRead_File, Get_all_paths and graph. Algorithm of these three functions explained below.

3.1 Algorithm of Read_File

```
START

Step 1: Initialize the required variables.Step 2: Open lable.txt file.

Step 3: Get the 1st line in lable.txt file from line variableStep 4: If line is character

        i)      C=C+1
        ii)     Get node from line
        iii)    Store in label array
        iv)     Get x-coordinate from line store on xy(c,1) in matrix xy
        v)      Get y-coordinate from line store on xy(c,2) in matrix xy
        vi)     Move on next line
        vii)    Go to step 4Step 5: Open edge.txt file

Step 6: Get the 1st line in edge.txt file from line variableStep 7: If line is character

      i)      C=C+1
      ii)     Get 1st node from line
      iii)    Search in label array and store in n1
      iv)     Get 2nd node from line
      v)      Search in label array and store in n2
      vi)     Change the adjacent value in matrixadj(n1,n2)=1 adj(n2,n1)=1
      vii)    Move on next line
      viii)   Go to step 7
```

STOP

3.2 Algorithm of Get_all_paths

START

Step 1: Initialize the required variables

Step 2: Enter the graph in matrix form and initial and final node.

Step 3: Calculate the value of count, where count is the total number of adjacent nodes of initial node.

Step 4: Calculate 1st path from initial node to final node and change the value in visited array as 1 and critical value as 1.

Step 5: Start from second last from calculated path and mark that node as K

(i) Find all adjacent nodes of the Kth node whose visited value is 0.

(ii) Push those nodes into stack.

(iii) Pop one node from stack and find the path along that node and increment the critical value by 1.

(iv) Repeat step 3 until stack will not be empty.

Step 6: Decrement value of K by 1 and again repeat step 5. If value of K and K+1 node is 0 change it to 1 and value of K and K-1 node as 0.Step 7: When all nodes in calculated path is completed change the value between 1st and 2nd node as zero.

Step 8: Decrement count by 1.

Step 9: Repeat step 4 to 8 until count is equal to 0.STOP

3.3 Algorithm of Graph

START

Step 1: Initialize the required variables

Step 2: Load files Labels.mat, adj.mat and xy.mat

Step 3: Plot the graph using xy1 and xy2 coordinates from read filesStep 4: Call Get_all_path function

Step 5: Count total no of paths findout in Get_all_path function

Step 6: Find total number of existence of a node in Get_all_paths function and store that value in Critical (n1)Step 7: Plot the graph of critical nodes with cyan color and other nodes with yellow color.

STOP

3.4 Explanation of algorithm with example

First of all label.txt and edge.txt file are read by using Read_File function.

Then a label array and xy matrix is created. In label array all nodes of networks are stored and in xy matrix coordinates related to nodes are stored.After this all paths in a network are calculated. Graph given below is used to explain Get_all_path algorithm.

1 Let us suppose the source and destination are A, E respectively.1 Find the total no of adjacent nodes of A i.e. Count=3

2 Find the first path from A to E i.e. A->B->D->E

3 In visit array change the value as 1 with respect to above nodes but leave the last node.

4 Now find the adjacent node of D whose visit value is not 1 and adjacent node is not equal to destination node.

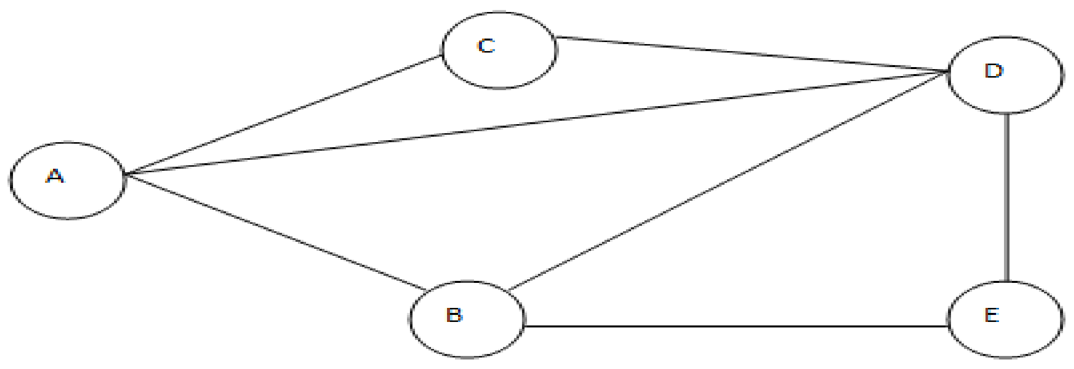

Figure 3.1 Graph to exemplify the algorithm5 In stack we have C

6 Now all adjacent are calculated, find the path with stack element i.e. C.

7 But there is no node adjacent to C whose visit is 0 so there is no path from c

8 Now adjacent of D are completed, change the value in adjacent matrix as 0 with respect to nodes B and D.9 Now find adjacent of B and store in stack.

10 A new path is generated i.e. A->B->E

11 Now adjacent of B are completed, change the value in adjacent matrix as 0 with respect to nodes A and B & adjacent value of B and D as 1.12 Again find the new path i.e. A->C->D->E and change the visit value as 1.

13 Now find the adjacent node of D whose visit value is not 1 and adjacent node is not equal to destination node.

14 Now in stack we have a new node B and we calculate a new path with the help of this node i.e. A->C->D->B->E.

15 15 Now adjacent of D are completed, change the value in adjacent matrix as 0 with respect to nodes B and D.

16 Find the adjacent of C, there are no such nodes, those are adjacent to C and visit value is 0.

17 Change the value in adjacent matrix as 0 with respect to nodes A and C & adjacent value of B and D as 1.18 Again find a new path i.e. A->D->E.

19 Find adjacent of D i.e. B, C.

20 But there is no such node, that is adjacent to C and visit value is 0.

21 21 So we calculate our new path with respect to B i.e. A->D->B->E.

22 Now all adjacent nodes are completed, we have all paths from A-E

1) A->B->D->E
2) A->B->E
3) A->C->D->E
4) A->C->D->B->E
5) A->D->E
6) A>D>B>E

After this we calculate the total number of paths and critical nodes in Graph function.

Experiments and Results

As we discussed in previous chapter the basic objective of this dissertation work is to find all paths in a network from a given source to destinationand also find critical nodes in the given network.

4.1 Results

Program explained in chapter 3 was implemented on different Graphs. The output of program is all paths from a given source to destination andcritical nodes in the network.

4.1.1 First Network

MatLab program was implemented on below network and we have to find out all paths from node A to node E and critical nodes in this network.Output of program is all paths from node A to node E and critical nodes i.e. B D and total no of paths from node A to node E i.e. 6.

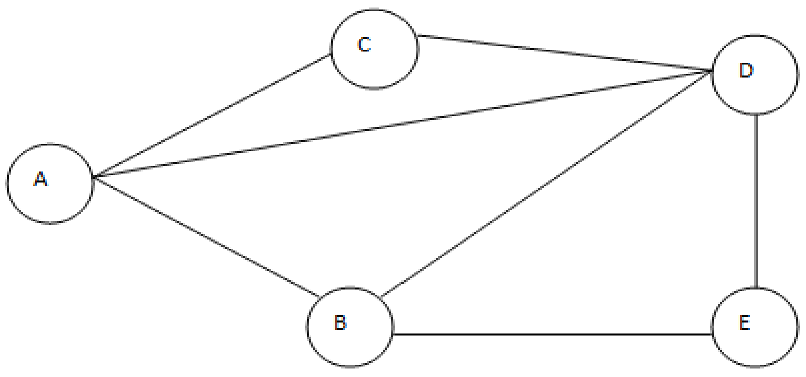

Figure 4.1 First Network on which program is implementedA->B->E hop: 2

A->D->E hop: 2

A->D->B->E hop: 3

A->B->D->E hop: 3

A->C->D->E hop: 3

A->C->D->B->E hop:4C=B D

>> numpaths6

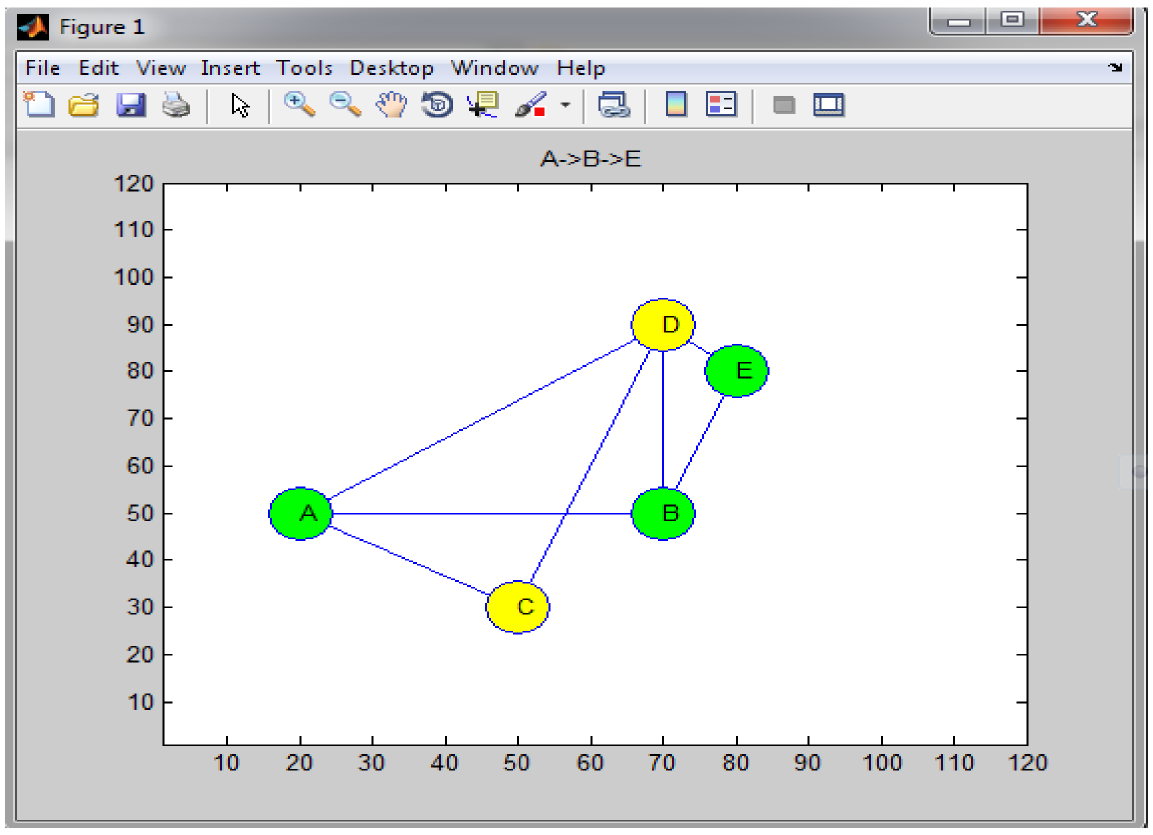

Figure 4.1 Path 1

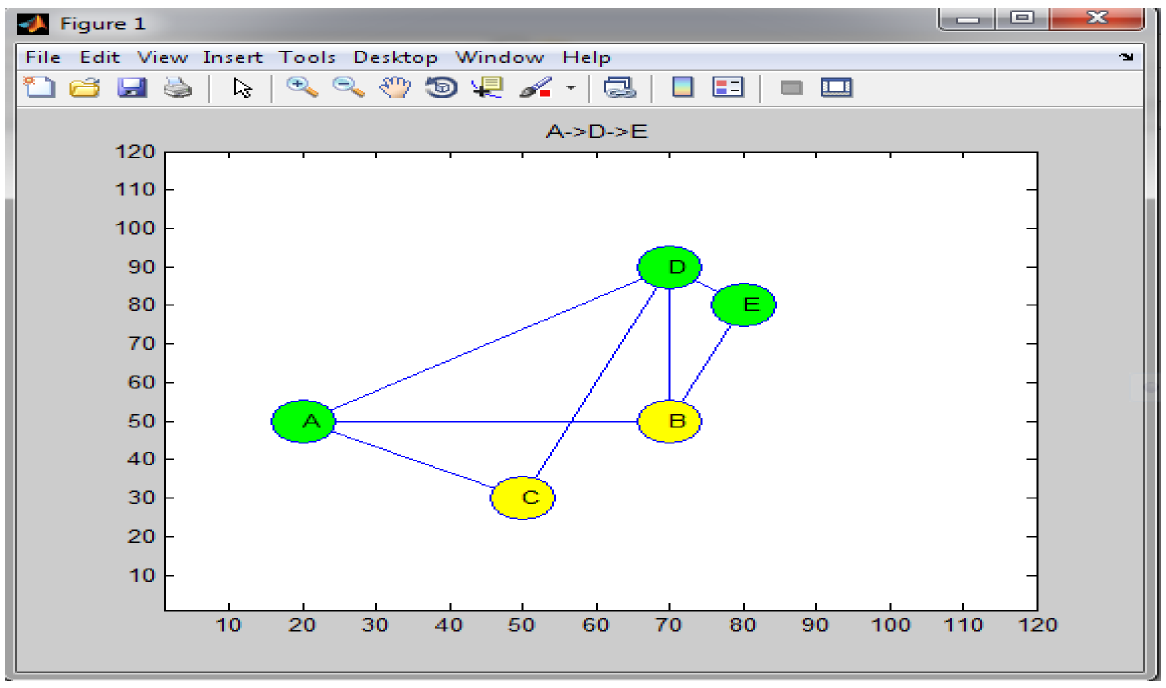

Figure 4.2 Path 2

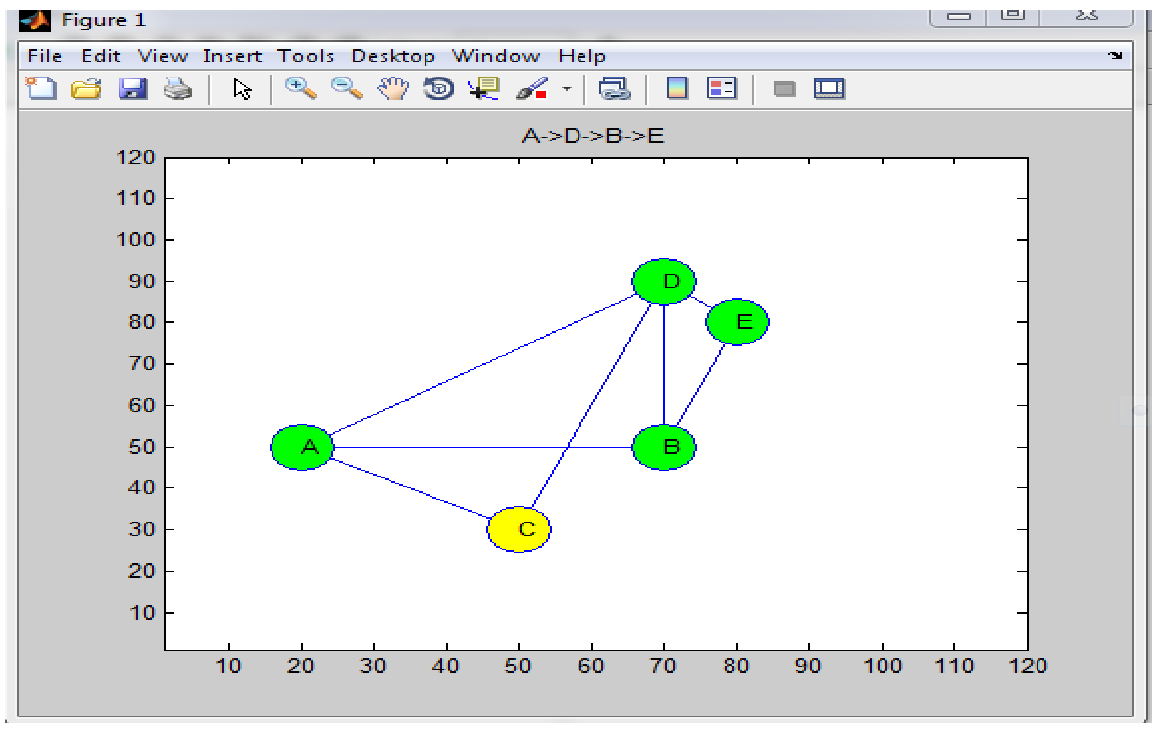

Figure 4.3 Path 3

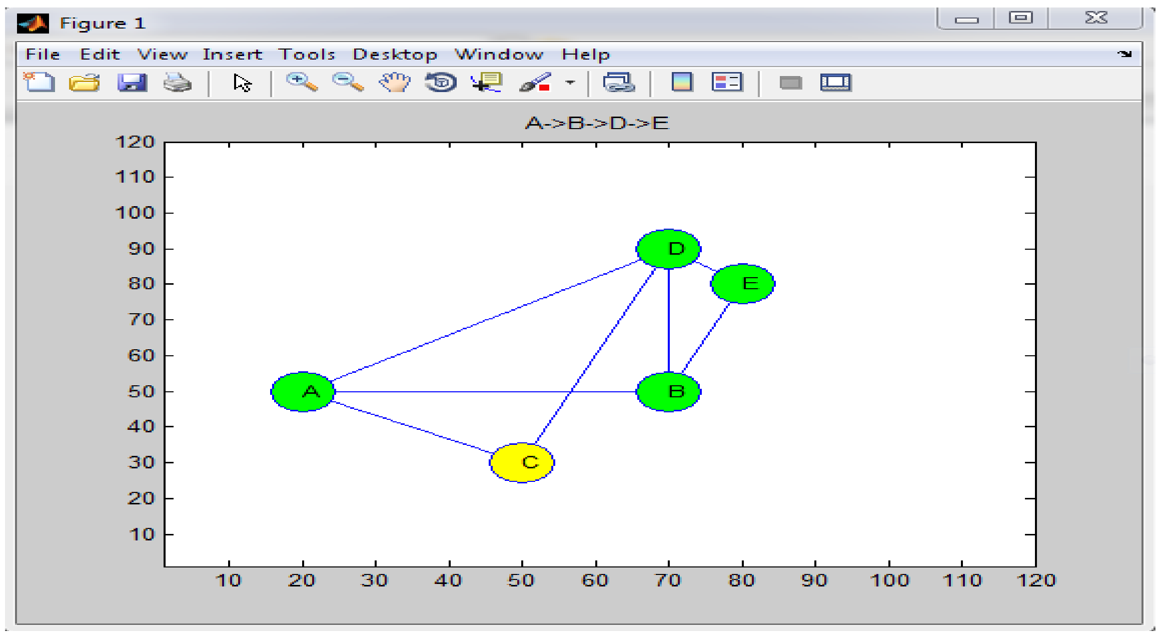

Figure 4.4 Path 4

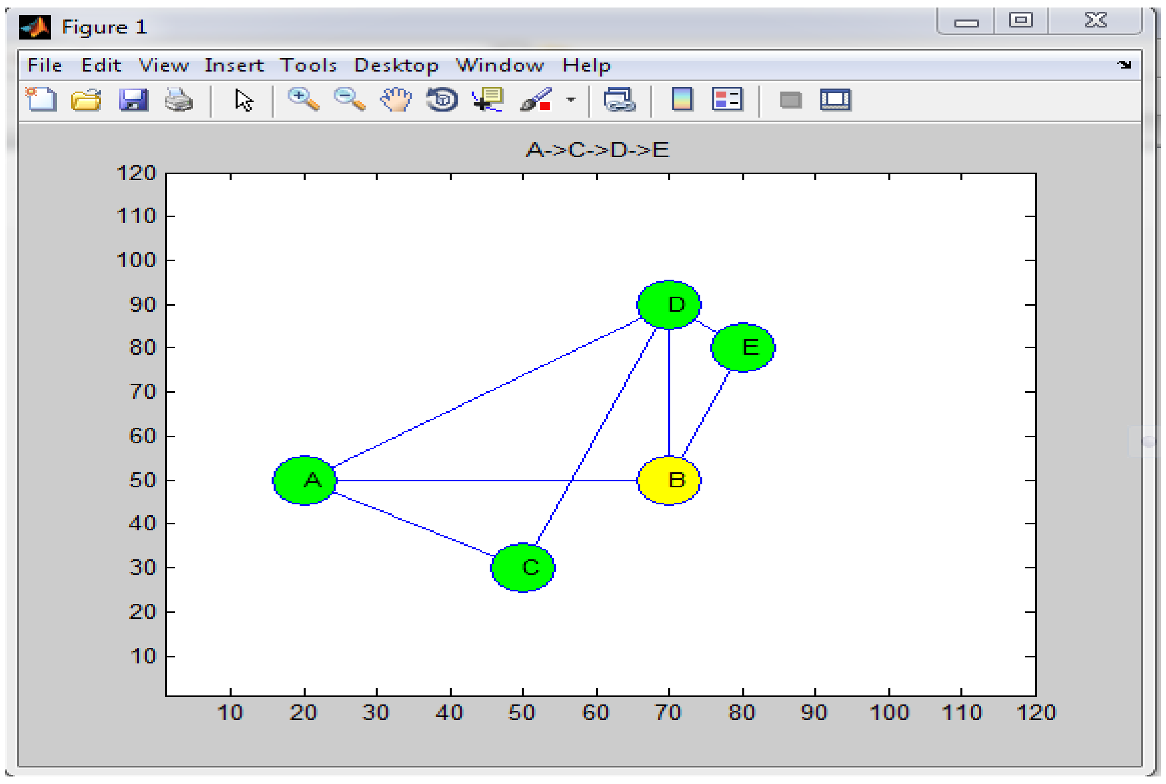

Figure 4.5 Path 5

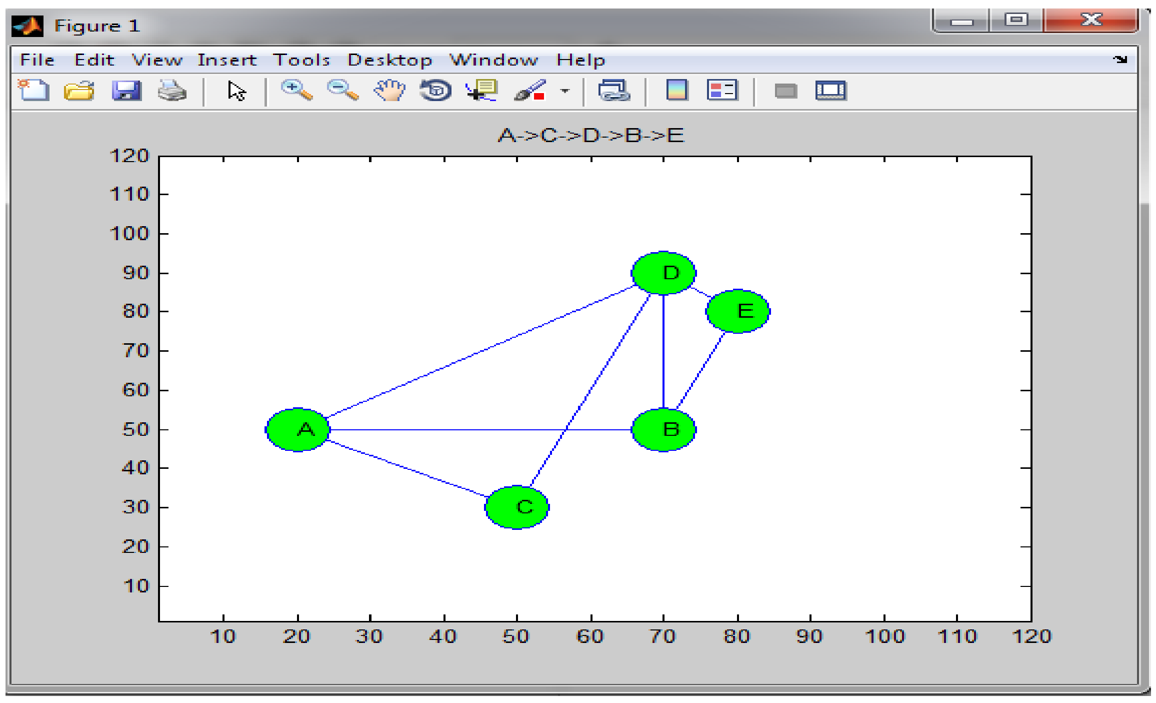

Figure 4.6 Path 6

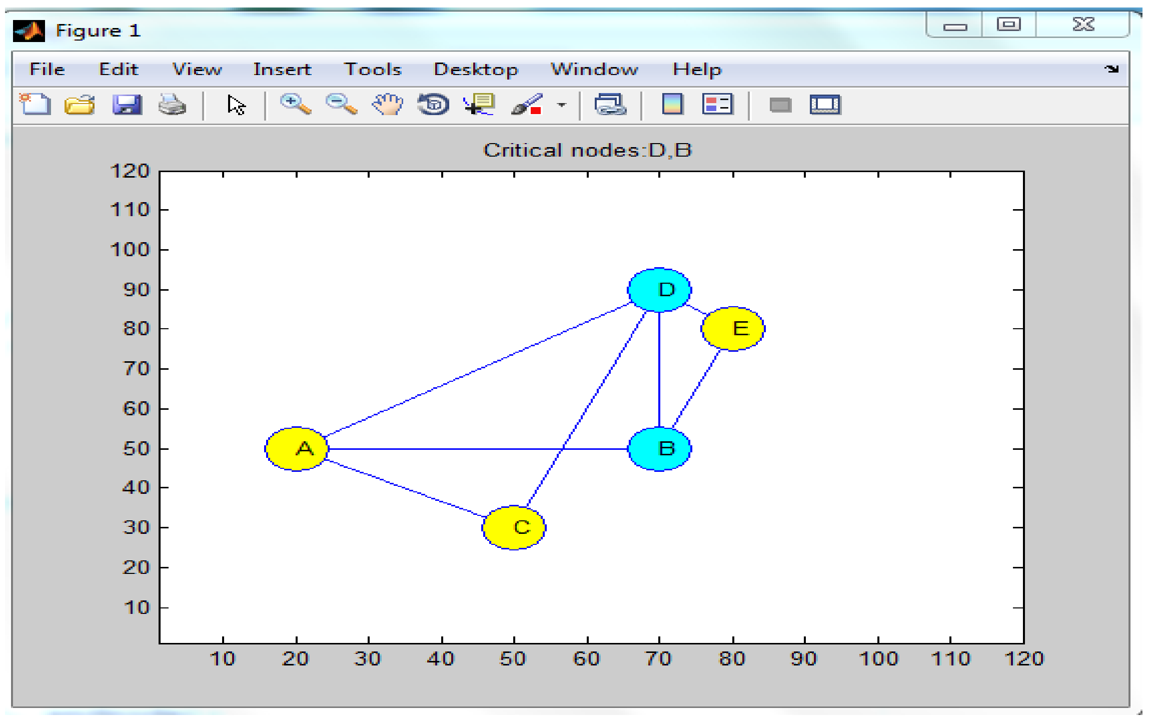

Figure 4.7 Critical Nodes in First Network

5.1 Conclusion

In this dissertation, all path and critical nodes are discovered, which can yield a variety of benefits such as fault tolerance, increased bandwidthand improved security.

To calculate all path and critical node a program is introduce in Mat Lab.

5.2 Future Scope

In future, this program may be changed into better one by reducing its time complexity.

References

[1] HOFFMAN, W. AND PAVLEY, R., "A Method for the Solution of the Nth Best Problem," J. of ACM, Vol. 6, No. 4 (October 1959), pp. 506-514.

[2] BELLMAN, R. AND ;KALABA, R. "On kth Best Policies," J. of SIAM, Vol. 8, No. 4 (December 1960), pp. 582-588.

[3] SAKAROVITCH, M., The k Shortest Routes and the k Shortest Chains in a Graph, Opns. Res. Center, University of California, Berkeley,Report ORC-32, October 1966.

[4] BOCK, F., KANTNER,H . ANDH AYNES, J., An Algorithm (The rh Best Path Algorithm) for Find-inq and Ranking Paths Through a Network, Research Report, Armour Research Foundation, Chicago ,Illinois, November 15, 1957.

[5] POLLACK, M., "The kth Best Route Through a Network," Opns. Res., Vol. 9, No. 4 (1961), pp. 578.

[6] CLARKE, S., KRIKORIAN, A. AND RAUSAN, J., "Computing the N Best Loopless Paths in a Net-work," J. of SIAM, Vol. 11, No. 4(December 1963), pp. 1096-1102

[7] Bock, F., Kantner, H. and Haynes, J. 1957 An Algorithm (The rh Best Path Algorithm) for Finding and Ranking Paths Through a Network,Research Report, Armour Research Foundation, Chicago, Illinois, November 15.

[8] Pollack, M. 1961 The k-th Best Route Through a Network, Operations Research, Vol. 9, No. 4, pp. 578-580.

[9] Sakarovitch, M. 1966 The k Shortest Routes and the k Shortest Chains in a Graph, Operations Research, Center, University of California,Berkeley, Report ORC-32.

[10] Martins,E. Q. V., Pascoal, M. M. B., and Santos, J. L. E. 1998 The K shortest paths problem. Research Report, CISUC.

[11] Shier, D. 1974 Computational experience with an algorithm for finding the k shortest paths in a network. Journal of Research of the NBS,78:139-164.

[12] Shier, D. 1976 Interactive methods for determining the k shortest paths in a network. Networks, 6:151-159.

[13] Shier, D. 1979 On algorithms for finding the k shortest paths in a network. Networks, 9:195-214.

[14] Bellman, R.E. 1958 On a routing problem. Quarterly Applied Mathematics, 1:425-447.

[15] Cherkassky, B.V., Goldberg, A.V., and Radzik, T. 1996 Shortest paths algorithms: Theory and experimental evaluation. MathematicalProgramming, 73:129-196.

[16] Moore, 1959 E.F. The shortest path through a maze. Proceedings of the International Symposium on the Theory of Switching, HarvardUniversity Press, 285-292.

[17] Dijkstra, E. 1959 A note on two problems in connection with graphs. Numerical Mathematics, 1:395-412.

[18] Dreyfus, S.E. 1969 An appraisal of some shortest-path algorithms. Operations Research, 17:395-412.

[19] Martins, E.Q.V. and Santos, J.L.E. 1996 A new shortest paths ranking algorithm. E. Martins and J. Santos. A new shortest paths ranking algorithm. Technical report, Departmentof Mathematics, Universityof Coimbra, (http://www.mat.uc.pt/~eqvm).

[20] Azevedo, J.A., Costa, M.E.O.S., Madeira, J.J.E.R.S., and Martins, E.Q.V. 1993 An algorithm for the ranking of shortest paths. EuropeanJournal of Operational Research, 69:97-106.

[21] Azevedo, J.A., Madeira, J.J.E.R.S., Martins, E.Q.V., and Pires, F.M.A. 1990 A shortest paths ranking algorithm, Proceedings of the Annual Conference AIRO'90, Models and Methods for Decision Support, Operational Research Society of Italy, 1001-1011.

[22] Azevedo, J.A., Madeira, J.J.E.R.S., Martins, E.Q.V., and Pires, F.M.A. 994 A computational improvement for a shortest paths ranking algorithm. European Journal of Operational Research, 73:188-191.

[23] Martins, E.Q.V. 1984 An algorithm for ranking paths that may contain cycles. European Journal of Operational Research, 18:123-130.

[24] Yen, J.Y. 1971 Finding the k shortest loopless paths in a network. Management Science, 17:712-716.

[25] Martins, E. Q. V. and Pascoal, M. M. B. 2003 A new implementation of Yen's ranking loopless paths algorithm, 4OR, Springer Berlin, 1:121- 133.

[26] Dial, R., Glover, G., Karney, D., and Klingman, D. 1979 A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees. Networks, 9:215-348.

[27] Eppstein, D. 1998 Finding the k shortest paths. SIAM Journal on Computing 28:652-673.

[28] Yen JY. Finding the K shortest loopless paths in a network. Management Science 1971;17:712-716 [29]Dijkastra EW. A note on two problems in connexion with graphs. Numerische Mathematik 1959;1:269-271

[30] Lawler EL. A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem.

Management Science, Theory Series 1972;18:401-405

[31] Kaoth N, Ibaraki T, Mine H. An efficient algorithm for K shortest simple path. Networks 1982;12:411-427

[32] Houffman W, Pavley R. A method for the solution of the nth best path problem. Journal of the Association for Computing Machinery(ACM)1959;6:506-514

[33] www.ijest-ng.com/vol3_no1/ijest-ng-vol3-no1-pp41-51.pdf