A New Algorithm for Finding Vertex-Disjoint Paths

Mehmet Kurt[1], Murat Berberler[2], and Onur Ugurlu[3]
[1]Department of Mathematics and Computer, Izmir University, Turkey
[2]Department of Computer Science, Dokuz Eylul University, Turkey
[3]Department of Mathematics, Ege University, Turkey

Abstract: *The fact that the demands which could be labelled as "luxurious" in the past times, have became requirements makes it inevitable that the service providers do new researches and prepare alternative plans under harsh competition conditions. In order to, provide the customers with the services in terms of the committed standards by taking the possible damages on wired and wireless networks into consideration. Finding vertex disjoint paths gives many advantages on the wired or wireless communication especially on Ad-Hoc Networks. In this paper, we suggest a new algorithm that calculates alternative routes which do not contain common vertex (vertex-disjoint path) with problematic route during a point-to-point communication on the network in a short time and it is compared to similar algorithms.*

Keywords: *Vertex-disjoint paths, multipath, Ad-Hoc wireless networks.*

Received September 2, 2013; accepted March 29, 2014; published online December 3, 2014

1. Introduction

Today, telecommunications technology continues to progress at an enthusing rate. Just a few decades ago, we were greatly pleased with audio communication despite noise on the wired telephone network, however, we are not pleased with the wireless network through which we cannot get the expected quality regarding both video and audio communications (when the audio, video and speed rate standards are taken into considerations)

That those needs which could be labelled as luxury turned out to be necessary needs today makes the service providers make new researches considering the possible damages on the wired or wireless networks that are used in order to, enable the service providers give their customers the services in the standards that they gave under harsh competitive financial environment.

In recent years, one of the subjects on which the researches have focused is the MultiPath that came out in line with alternative plan preparation. This system provides many advantages for both wired and wireless [2].

Wired systems, for instance through a complex communication network, data transmission from one point to another can occur through many different ways. These differences may cause a variety of possibilities. Problem such as damage on the route or a slowdown due to an unusual traffic can be overcame by putting alternative routes into action. Transmission rate of the data stack can be increased by parallel paths.

"Ad-Hoc Networks" which is related to the concept of multipath emerges as a highly active research topic due to intense interest in wireless systems in recent years. [1, 12, 18] Increasing use of Personal Digital Assistant (PDA) devices, advanced mobile phones and personal computers with the wireless adapter represents "Ad-Hoc Networks". The members of these networks are both receivers (client) also transmitters (server). A device, which is on the network, can transmit the data it received to other devices in range.

In the example given in Figure 1, uppercase letters indicate devices of the network and lowercase letters indicate the coverage area of the devices. For instance, the device that is indicated as C can send data to B, D, E and F devices and also receive data from B, D and E devices. This structure can be modelled by the directed graph indicates in Figure 2.

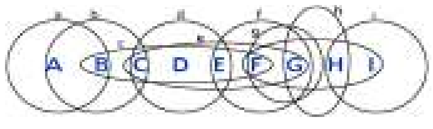


Figure 1. Members and their ranges.

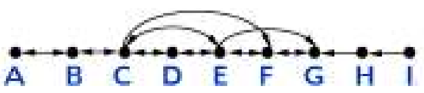


Figure 2. Graph model for Figure 1.

The fact that a device can transmit an incoming data to the other devices in its range enables the possibility of connection through various ways from any device to another.

Figures 1 and 2 show that it is possible to reach to device G from device C via these routes: C-D-E-F-G or C-E-F-G or C-F-G.

In some cases, there may be a lot of data request from device C to device G most densely on the route C-F-G. In such a case, if an alternative route without any common device is found then, the traffic can be directed to that route and transmission quality problem of the network can be overcame. C-E-G route can be chosen as an alternative of C-F-G.

An important issue in the concept of multipath is the necessity of independent alternative ways. The routes to be found due to a damaged member and available connected routes, should be vertex-disjoint (node-disjoint), a graph theory terminology, because any device of the former route have already been damaged.

2. Literature Review

Scientists have done many researches about multipath so far. A general analysis comparing multipath algorithms and well-chosen single path algorithms have been done by Cidon *et al*. [6]. They pointed out the necessity of multipath algorithms for the interactive network structures. The shortest path algorithms on the basis of the first studies about disjoint-path is chosen as the starting point [8, 19, 20]. However, these studies failed to consider the condition of vertex-disjoint path in the proposed algorithms.

In the following years, several studies had been conducted which ignored the condition of vertex-disjoint path but proposed methods that aimed at minimizing the lengths of disjoint paths. In same of these methods, constraints regarding the use of all vertex are stated same studies are about minimizing the energy consumed on the network [3, 4, 14, 15].

Another term of disjoint multipaths is edge-disjoint paths enables re-usage of nodes used in the graph. In fact vertex-disjoint paths include edge-disjoint paths. In this study Guo and etc., have made the definitions of edge-disjoint paths and offered an algorithm about finding them.

Studies regarding the algorithms about the number of vertex-disjoint paths are generally carried out for special graph structures. Methods about special graph types such as star graphs, n-k star graphs, hierarchical cubic graphs, fixed-clique graphs and circular-arc. [5, 9, 10, 11, 13, 16, 17].

The algorithm including vertex-disjoint paths which will be valid for all graph structures is the rearranged of Ford-Fulkerson algorithm through.

3. Modified Ford-Fulkerson Algorithm

Ford-Fulkerson algorithm is published in 1956. This algorithm is aims to find route that the most amount of flow from source vertex to target vertex on the graph model in which each edges have a flow capacity.

The working principle of the algorithm is as follows: Finds a path from source vertex to target vertex. It passes the small flow capacity on the path. Then, it finds a new path and passes the new small flow capacity on the new path. Process continues until total flow finished or there is no new path.

Distinctly, the algorithm will find all the paths if the edge capacities was equal to 1. Paths are disjoint although are not the vertex-disjoint.

In this study, we added to Ford-Fulkerson algorithm a condition that nodes could be used once and encoded it in C programming language.

4. A New Algorithm: MebDisjPath

MebDisjPath algorithm finds vertex-disjoint paths on the any kind of graph. The working principle of the algorithm is as follows:

- *Step* 1: Finds the shortest paths between any two nodes on the graph. In fact, the values required for the algorithm are the shortest path lengths from each vertices to target vertex. An example is illustrated in Figure 3. S is the source vertex and T is the target vertex. Numbers in other vertices are shortest path length from them to T.

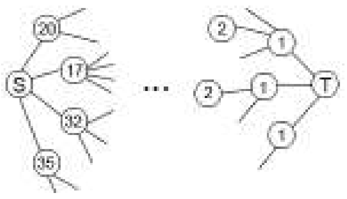


Figure 3. Final state example of step 1.

- *Step* 2: Starts walking with a neighbour of source vertex which is the closest vertex to target vertex via information on step 1. In Figure 4 we can see 4 neighbours and their shortest path lengths are 20, 17, 32 and 35. So, algorithm continuous to select the vertex has 17. For a special case, if the most close vertex's neighbours are more than 1, then it continuous to select a vertex that is the most close to target vertex among them. It is illustrated in Figure 5.

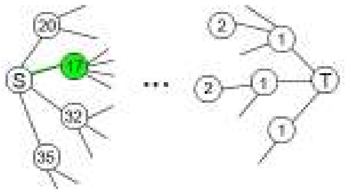


Figure 4. Final state example of step 2.

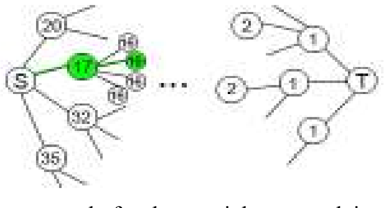


Figure 5. An example for the special case explained in step 2.

- *Step* 3: When the path has been completed, it marked (removed vertices and edges) vertices on this path. Because they could not be used again.
- *Step* 4: Continuous until a new path which could not be found.

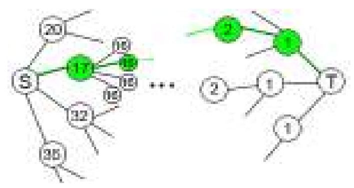


Figure 6. A path completed.

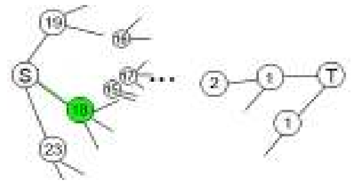


Figure 7. Finding path is removed and re-calculated shortest path lentghs.

The algorithm can find fewer paths than it should be because it gives priority to the shortest path between source and target vertices. To solve this problem, each edge on the missing paths are removed one by one and re-calculated paths again after the first step of the algorithm. If the new path number found is equal to the first then the edge is added to graph again. If the new path number found is more than the first, the record is updated. In this case, the edge is not added to graph. When the algorithm terminates, all of the vertex disjoint paths will have been found.

As a result, all cases which need record update are reduced in two groups, (a) and (b). They have been illustrated below in Figures 8, 9, 10, 11, 12 and 13. Other all cases match with group (a) or group (b).

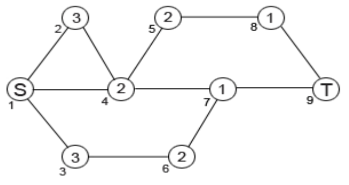


Figure 8. Group (a).1.

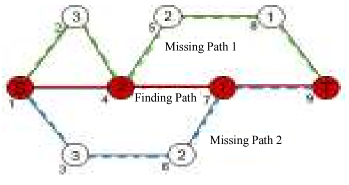


Figure 9. Group (a).2.

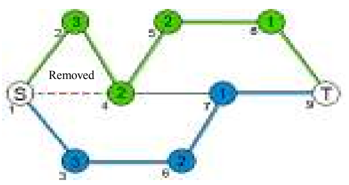


Figure 10. Group (a).3.

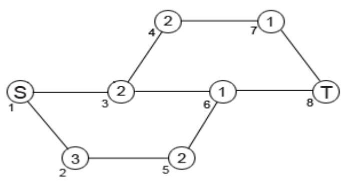


Figure 11. Group (b).1.

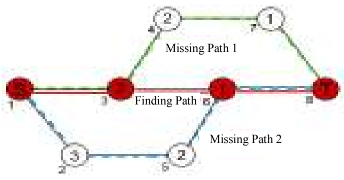


Figure 12. Group (b).2.

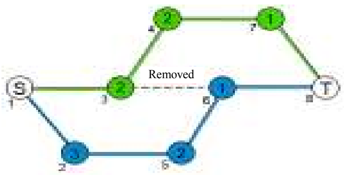


Figure 13. Group (b).3.

Meaning of index used in Figures 8 and 9 is:

x= a or b

- *a*: If the edge which must be erased was neighbor to source vertex.
- *b*: If the edge which must be erased was not neighbor to source vertex.
- (*x*).1: Simple graph.
- (*x*).2: Missing path, after the first result.
- (*x*).3: New paths found by removing edge.

BFS algorithm has been modified to finding the shortest paths in the first step of our algorithm. The complexity of modified BFS is $O(m+n)$ for unweighted graphs [7]. On the other part of our algorithm's complexity is $O(n2)$ because the largest cost is routing $n \times n$ matrix. Total complexity for our algorithm is $O(Epn2)$ as its steps have been run all of the paths. P is the total number of vertex-disjoint paths. E is the total number of edges found after the first step of the algorithm.

5. Analysis of Exam Results

They have been produced 90 kinds of matrix for testing and its analysis.

Matrices which are shown on the Table 1 contains 100, 200, …, 1000 vertices and vertex-disjoint paths about %25, % 50, %75, %95 and %100 ratio.

Table 1. For vertex-disjoint paths ratio (mm: ss. sp-minute: second. split second).

Number of Vertices	Path Ratio %25			%50			%75			%95			%100		
	Gams	Ffdisjpath	Mebdisjpath	Gams	Ffdisjpath	Mebdisjpath	Gams	Ffdisjpath	Mebdisjpath	Gams	Ffdisjpath	Mebdisjpath	Gams	Ffdisjpath	Mebdisjpath
100	0.344	0	0	0.344	0	0	0.344	0.01	0	0.35	0	0	0.35	0	0
200	1.016	0.03	0	1.125	0.04	0.01	1.031	0.04	0.01	1.28	0.05	0.01	1.28	0.04	0
300	2.266	0.07	0.02	2.859	0.12	0.04	2.672	0.18	0.02	3.313	0.16	0.01	2.93	0.10	0
400	4.891	0.13	0.04	5.875	0.27	0.08	5.312	0.39	0.05	7.125	0.27	0.01	5.67	0.24	0
500	9.,047	0.23	0.05	10.64	0.44	0.13	10.344	0.72	0.09	15.016	0.47	0.01	10.04	0.40	0
600	14.5	0.41	0.13	18.29	0.66	0.21	15.570	1.19	0.14	23.890	0.91	0.03	15.54	0.71	0
700	21.,40	0.63	0.18	28.42	1.07	0.35	24.485	1.88	0.24	38.688	1.41	0.04	23.35	1.15	0
800	32.07	0.97	0.28	38.53	1.66	0.49	35.406	2.85	0.30	1:02.73	2.13	0.06	31.76	2.55	0
900	39.93	1.38	0.35	54.75	2.43	0.70	53.094	4.14	0.43	1:36.31	3.14	0.06	42.59	2.61	0.01
1000	52.50	1.90	0.65	1:17.95	3.40	1.09	1:10.84	5.64	0.61	1:50.14	4.50	0.08	55.57	5.09	0.01

The total numbers of matrices are 50. Matrices which are shown on the Table 2 contains 100, 200, …, 1000 vertices and numbers elements of 1 about 25, 50, 75 and %95. The total numbers of matrices are 50.

Table 2. Number of 1's ratio (mm: ss. sp-minute: second. split second).

Number of Vertices	1's Ratio %25			%50			%75			%95		
	Gams	Ffdisjpath	Mebdisjpath	Gams	Ffdisjpath	Mebdisjpath	Gams	Ffdisjpath	Mebdisjpath	Gams	Ffdisjpath	Mebdisjpath
100	25 PATH			42 PATH			72 PATH			94 PATH		
	0.344	0	0	0.344	0	0	0.344	0	0	0.375	0	0
200	51 PATH			94 PATH			151 PATH			192 PATH		
	0.812	0.02	0.02	1.00	0.05	0.02	1.109	0.05	0.01	1.235	0.05	0
300	74 PATH			149 PATH			215 PATH			288 PATH		
	2.032	0.05	0.05	2.453	0.15	0.05	2.860	0.18	0.03	3.266	0.14	0.01
400	105 PATH			191 PATH			298 PATH			381 PATH		
	4.422	0.14	0.07	5.187	0.28	0.11	5.703	0.34	0.05	7.110	0.23	0.01
500	131 PATH			245 PATH			370 PATH			462 PATH		
	8.454	0.24	0.14	9.593	0.5	0.16	10.93	0.6	0.11	13.21	0.47	0.01
600	144 PATH			301 PATH			452 PATH			563 PATH		
	13.01	0.34	0.25	15.06	0.83	0.31	19.81	1.01	0.16	28.73	0.8	0.03
700	165 PATH			344 PATH			512 PATH			660 PATH		
	19.06	0.53	0.44	22.21	1.25	0.43	28.29	1.55	0.23	37.37	1.19	0.04
800	178 PATH			395 PATH			573 PATH			757 PATH		
	26.26	0.71	0.54	31.35	1.9	0.65	35.96	2.32	0.34	46.71	1.91	0.04
900	214 PATH			436 PATH			693 PATH			856 PATH		
	35.67	1.1	0.97	44.65	2.7	1.05	51.09	3.64	0.41	1:19.12	2.75	0.06
1000	243 PATH			486 PATH			741 PATH			957 PATH		
	47.48	1.54	1.40	1:0.03	3.71	1.51	1:10.17	6.96	0.94	2:21.09	5.39	0.07

- MebDisjPath: The new algorithm's results.
- FFDisjPath: The modified ford-fulkerson algorithm's results.
- Gams: general algebraic modelling system results.
- Computer system configuration: Intel 2.4GHz, 2GB RAM, linux mandriva 11
- Compiler and optimizer: Gcc 4.2 compiler for linux, -O2 optimizer.

Moreover, GAMS 22, 5 version is used to finding optimum solutions of problem. It is possible to more clearly examine the results on the Table 1 with following figures. Let compare to FFDisjPath and mebDisjPath methods with different paths ratios via Figures from 14 to 21.

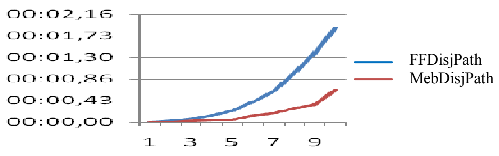


Figure 14. %25 path ratio.

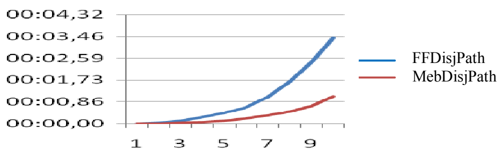


Figure 15. %50 path ratios.

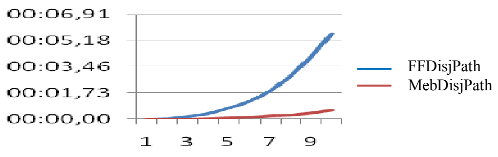


Figure 16. %75 path ratio.

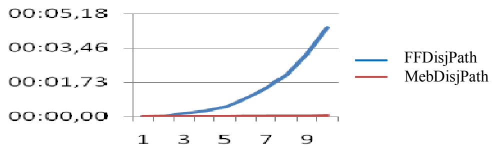


Figure 17. %95 path ratio.

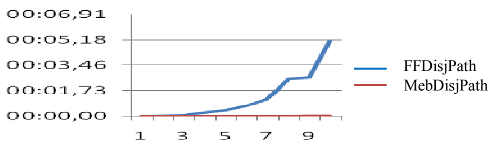


Figure 18. %100 path ratio.

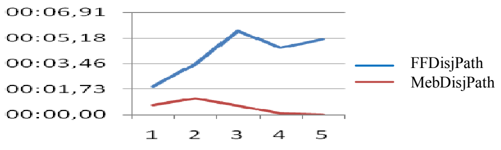


Figure 19. 1000 vertices (1: %25 path ratio, 2: %50 path ratio, …).

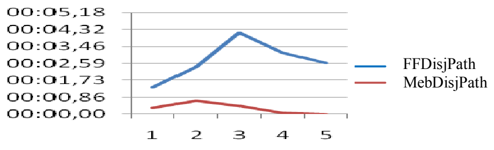


Figure 20. 900 vertices (1. %25 path ratio, 2. %50 path ratio …).

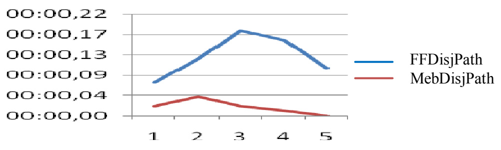


Figure 21. 300 vertices (1. %25 path ratio, 2. %50 path ratio, …).

We can make the following comments through these figures:

- In the same path ratios, although the work time increases, the speed of mebDisjPath algorithm is better than FFDisjPath.
- In almost all test mebDisjPath works in less than one second, whereas FFDisjPath can work in five or six seconds.
- For the same number of vertices, process time is maximum in some of the path rations. This ratio is about 75% for FFDisjPath, whereas it is about 50% for mebDisjPath.

It is possible to more clearly examine the results on the Table 2 with following figures. Let compare to FFDisjPath and mebDisjPath methods with different numbers of 1's ratio via Figures from 22 to 29.

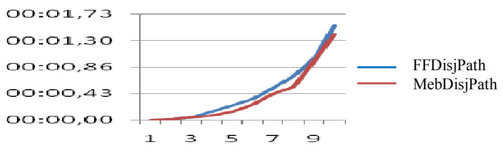


Figure 22. 1's ratio %25.

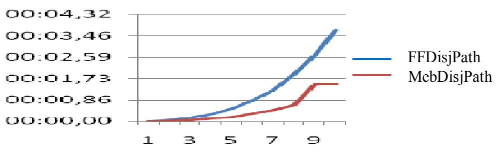


Figure 23. 1's ratio %50.

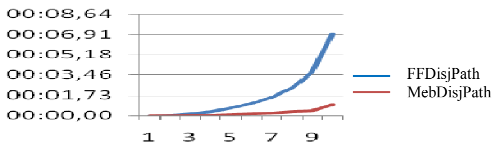


Figure 24. 1's ratio %75.

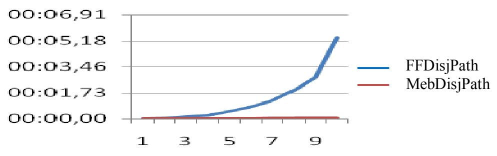


Figure 25. 1's ratio %95.

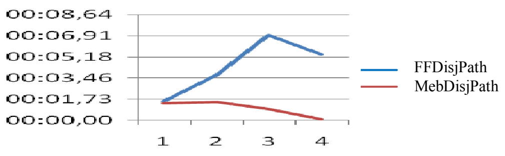


Figure 26. 1000 vertices (1. %25 1's ratio, 2. %50 1's ratio, …).

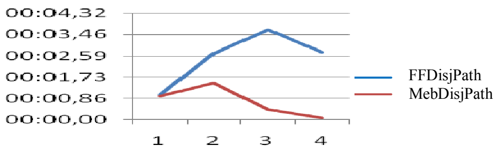


Figure 27. 900 vertices (1. %25 1's ratio, 2. %50 1's ratio, …).

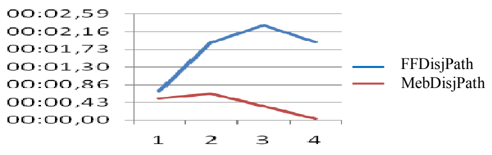


Figure 28. 800 vertices (1. %25 1's ratio, 2. %50 1's ratio, …).

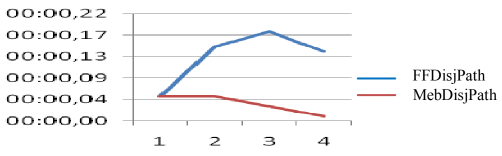


Figure 29. 300 vertices (1. %25 1's ratio, 2. %50 1's ratio, …).

We can make the following comments through these figures:

- In the same 1 density ratio, although the work time increases as the number of vertices increases, the speed of mebDisjPath is beter than FFDisjPath.
- In almost all test mebDisjPath works in less than one second whereas FFDisjPath can work approximately five or six seconds for the graphs which have got a lot vertices or 50% 1 density.
- In the same number of vertices, process time is maximum in some of the 1 density rations. This ratio is about 75% for FFDisjPath, whereas it is about 50% for mebDisjPath.
- mebDisjPaths performs much better than FFDisjPath in graphs which have got big 1 density ratio.

All of the programs and tested problems which are used to calculate algorithm's perform are published on http://kisi.deu.edu.tr/murat.berberler/ANAfFVDP/

6. Conclusions

According to results derived from the proposed algorithm finding the vertix disjoint paths in terms of the shortest paths towards the target vertex is a considerably rapid method.

MebDisjPath Algorithm is faster than FFDisjPath as the number of vertices increases in the same disjoint-vertex ratio.

MebDisjPath algorithm is faster than FFDisjPath as the 1 density ratio increases in the same vertices number.

It is thought that similar point of view with mebDisjPath Algorithm gives results to us for edge-disjoint paths. Also this point of view can give results for vertex-disjoint paths which they contain k=2, 3, … common vertices.

References

[1] Abbas A., "A Hybrid Protocol for Identification of a Maximal Set of Node Disjoint Paths in Mobile Ad-Hoc Networks," *the International Arab Journal of Information Technology*, vol. 6, no. 4, pp. 344-358, 2009.

[2] Ben-Asheri Y., Feldman S., and Feldman M., "Dynamic Multipath Allocation in Ad Hoc Networks," *the Computer Journal*, vol. 54, no. 2, pp. 197-212, 2011.

[3] Chang H. and Tassiulas L., "Energy Conserving Routing in Wireless Ad-Hoc Networks," *in Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Society*, pp. 22-31, 2000.

[4] Chang H. and Tassiulas L., "Fast Approximate Algorithms for Maximum LifeTime Routing in Wireless Ad-Hoc Networks," *in Proceedings of the European Commission International Conference on Networking Broadband Communications*, *High Performance Networking*, *and Performance of Communication Networks*, Paris, France, pp. 702-713, 2000.

[5] Chen C. and Chen J., "Nearly Optimal One-to-Many Parallel Routing in Star Networks," *IEEE Transaction on Parallel and Distributed Systems*, vol. 8, no. 12, pp. 1196-1202, 1997.

[6] Cidon I., Rom R., and Shavitt Y., "Analysis of Multi-Path Routing," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 885-896, 1999.

[7] Cormen H., Leiserson E., Rivest L. and Stein C., *Introduction to Algorithms*, MIT Press, 2001.

[8] Eppstein D., "Finding the k Shortest Path," *in Proceedings of the 35th Annual Symposium on Foundation of Computer Science*, New Mexico, USA, pp. 154-165, 1994.

[9] Fu S., Chen H. and Duh R., "Node-Disjoint Paths and Related Problems on Hierarchical Cubic Networks," *Networks*, vol. 40, no. 3, pp. 142-154, 2002.

[10] Fujita S., "Polynomial Time Algorithm for Constructing Vertex-Disjoint Paths in Transposition Graphs," *Networks*, vol. 56, no. 2, pp. 149-157, 2010.

[11] Gu P. and Peng S., "Node-to-Set Disjoint Paths Problem in Star Graphs," *Information Processing Letters*, vol. 62, no. 4, pp. 201-207, 1997.

[12] Guo Y., Kuipers F., and Van Mieghem P., "Link-Disjoint Paths for Reliable Qos Routing," *the International Journal of Communications Systems*, vol. 16, no. 9, pp. 779-798, 2003.

[13] Gurski F. and Wanke E., "Vertex Disjoint Paths on Clique-Width Bounded Graphs," *Theoretical Computer Science*, vol. 359, no. 1, pp. 188-199, 2006.

[14] Kang I. and Poovendran R., "Maximizing Static Network Lifetime of Wireless Broadcast Ad-Hoc Networks," *in Proceedings of International Conference on Communication*, Alaska, USA, pp. 2256-2261, 2003.

[15] Liang W. and Liu Y., "On-Line Disjoint Path Routing for Network Capacity maximization in Energy-Constrained Ad-Hoc Networks," *Ad-Hoc Networks*, vol. 5, no. 2, pp. 272-285, 2007.

[16] Lin C. and Duh R., "Constructing Vertex-Disjoint Paths in (n,k)-Star Graphs," *Information Sciences*, vol. 178, no. 3, pp. 788-801, 2008.

[17] Natarajan S. and Sprague P., "Disjoint Paths in Circular Arc Graphs," *Nordic Journal of Computing*, vol. 3, no. 3, pp. 256-270, 1996.

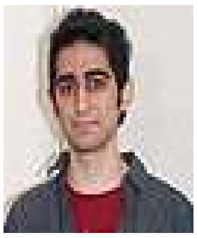
[18] Sesay S., Yang Z., and He A., "Survey on Mobile Ad-Hoc Wireless Network," *Information Technology Journal*, vol. 3, no. 2, pp. 168-175., 2004.

[19] Sidhu D., Nairand R., and Abdallah S., "Finding Disjoint Paths in Networks," *ACM SIGCOMM Computer Communication Review*, vol. 21, no. 4, pp. 43-51, 1991.

[20] Sidhu P., Abdauah S., and Nair R., "A Distance Vector Algorithm for Aletnate Path Routing," *Technical Report*, University of Maryland, 1990.

Mehmet Kurt assistant professor received his BE, ME and PhD degrees in mathematics and computer science from Ege University in 2000, 2004 and 2010. Currently, he is with mathematics and computer science, Izmir University, Turkey. He research interests include graph theory, computer networks and algorithms.

Murat Berberler received his BE, ME, PhD degrees in mathematics and computer science from Ege University in 2003, 2006 and 2009. Currently, he is with school of computer science, Dokuz Eylul University, Turkey. He research interests include computer networks, design and analysis of algorithms and mathematical modeling.



Onur Ugurlu received his BE degree in mathematics in 2011. Currently, he is is the ME degree candidate with school of computer science, Ege University, Turkey. His research interests include artificial intelligence and graph theory.