RESEARCH ARTICLE OPEN ACCESS

A Comparative Analysis for Finding Shortest Path Problem in Computer Networks

Abdullah M. A. Alzafiri*
**(Industrial Institute -Shuwaikh, Public Authority of Applied Education and Training, Kuwait*

ABSTRACT
Network routing is an important issue in computer networks, as routingis used to transfer packets from source node to destination node by selecting the shortest path between these nodes to increase the network's performance. There are many routing protocols to accomplish this task and they are divided into two main types: distance vector routing protocols and link-state routing protocols which is the most commonly used. Link-state routing protocols use graph search algorithms to obtain the shortest route between two nodes. Dijkstra's algorithm (DA) is used as a classical graph search algorithm in many famous routing protocols such as Open Shortest Path First (OSPF). In this work, a genetic algorithm (GA) is used as an alternative graph search algorithm to find the optimal route. Both DA and GA are implemented on three network graphs with different sizes (small, medium, and large) to obtain the shortest route between the source and destination nodes. The results show that both algorithms lead to optimal route but DA is faster than GA in small networks, while GA is faster than DA in large networks.

Keywords–computer network, Dijkstra's algorithm, genetic algorithm, routing, shortest route.

Date Of Submission: 10-11-2019 Date Of Acceptance: 30-11-2019

I. INTRODUCTION

In the context of computer networks, network routing is the process of selecting paths and forwarding traffic from a source node to a destination node. These network nodes are usually presented by routers or switches. By exchanging information between routers in the network, a router usually maintains a list of Internet addresses (IPs) and their corresponding locations in the network. This list is called a routing table [1].A routing table may be configured in two ways: manually or automatically. Manual configuration for the routing table is done by a network administrator who should manually define the routes or paths that can reach the destination (static routes) and each time update it if there is a change in the network configuration or topology. The second way is better due to the routing table could be configured automatically using "routing protocols" (dynamic routing). Routing protocols such as Routing Information Protocol (RIP), Open Shortest Path First (OSPF), etc. create and maintain routing tables periodically for each router. It therefore updates its own routing table and constructs a picture of how to reach other parts of the network [1, 2].A routing protocol specifies how routers communicate with each other and distributes information that enables them to select routes between any two nodes on a computer network, with the choice of the route being done by routing algorithms [1, 2].There are many types of routing protocols which are divided into two main types depending on their functionality. The first type is distance vector routing protocols such as Routing Internet Protocol (RIP), Interior Gateway Routing Protocol (IGRP), and Enhanced Interior Gateway Routing Protocol (EIGRP). The second type is link-state routing protocols such as Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS). In distance-vector routing protocols, the best path is determined on the basis of how far the destination is, typically, in terms of the number of hops to the destination. Link-state routing protocols use more advanced methods by taking into consideration link variables or the total cost of the path. OSPF, for example, is one of the most popular routing protocols and uses a link-state algorithm [1, 2].When applying link-state algorithms, every node constructs a map of the network connectivity in the form of a graph, showing the nodes that are connected to other nodes. Using this graph, each router then independently determines the least-cost path from itself to every other node using a graph search algorithm such as Dijkstra's algorithm [3, 4, and 5].Dijkstra's algorithm is the most popular and it is considered as a graph search algorithm that uses to solve the single-source shortest path problem. In this work, a genetic algorithm (GA) [6] is suggested instead of this search graph algorithm (Dijkstra's algorithm) to obtain the shortest path between two nodes in a network graph. The performance of both algorithms will be compared in terms of speed and accuracy to decide which one is better in certain situations.The genetic algorithm (GA) is "a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA) that generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover" [7, ,8 ,9, 10, 11 and 12].

II. THE SUGGESTED GENETIC ALGORITHM FOR SOLVING SHORTEST PATH PROBLEM

The network topology can be represented as a graph G = (V, A), where V is vertices' set (e.g., routers or switches) and A is a set of arcs or links that connect these nodes together (e.g., wires). Another factor that describes the network is the cost of each link, where the cost between the two nodes represents the length between them. C_{ij} is the cost of transmitting a packet between node i to node j and the cost matrix C = [C_{ij}] represents the costs for our network graph. The source node denoted by S and the destination node denoted by D. Each arc or link connects two nodes in the graph denoted by I_{ij} that is a very important matrix used to inform if the link from node i to node j is included in the path or not. I_{ij} can be defined by the flowing equation [15].

$$I_{ij} = \begin{cases} 1, & \text{if the link } (i, j) \text{ exists in the routing path} \\ 0, & \text{otherwise.} \end{cases}$$

We note that the elements in the diagonal of matrix I_{ij} are zeros because the cost between any node and itself is zero.

2.1 Chromosome Representation

Each solution in the suggested GA is considered as a suggested network route that connects the source node to destination node in the network graph. The chromosome here consists of a sequence of integers. These integers represent the node number or the node ID that the path passes through to get to destination node. The first gene in the chromosome is always the source node ID and the last gene is the destination node ID. The length of the chromosome is variable depending on the number of visited nodes in the routing path that should not be more than the number of nodes in the network graph. Exceeding the total number of nodes means that there are repeated nodes in the routing path and this will create a loop that is not acceptable in our algorithm [13, 14, and 16].

The routing path or the chromosome is encoded by listing the visited nodes ID from the source node S to the destination node D depending on the information obtained from the routing table, which is built and maintained by a routing protocol such as OSPF [13, 14, and 16]. Figure 2.1 shows an example of chromosome representation (i.e., routing path) and the encoding scheme. The chromosome is an array of nodes represents a routing path that starts with the source node (S), passes through different nodes (N_1, N_2,…) in the network and ends with the destination node (D).

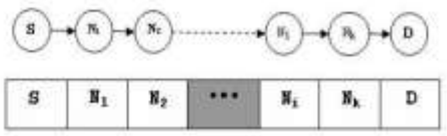


Figure 2.1. Example of a chromosome representation [14]

The first gene in the chromosome is always reserved for the source node (S), and the next gene represents the node randomly selected from one of the neighbors of the previous node (now it is S). To avoid creating loops in the path, any selected node will not be selected again in the routing path. This step will be repeated and the process continues until the destination node is reached and selected. So all suggested feasible chromosomes should have the destination node (D) at the last locus [13, 14, and 16].

2.2 Population Initialization

Usually the initial population is created randomly in GAs, but in this shortest path routing problem, it is better to use heuristic initialization because the topological information is already known from the routing table. Heuristic initialization makes the algorithm more complex, but it also pushes the population to higher fitness values and eliminates the infeasible solutions, so the algorithm will be more effective and the execution time will be minimized in general. First, the source node is chosen as the first gene in the chromosome, and then the next gene will be one of the nodes that connect to the previous gene that is selected randomly depending on the topological information and so on until the destination node is reached and selected. If the algorithm finds that all nodes that are connected to the previous gene in the chromosome are visited and the previous gene is not the destination node, the chromosome will be refreshed to avoid creating an infeasible chromosome [13, 14, and 16].

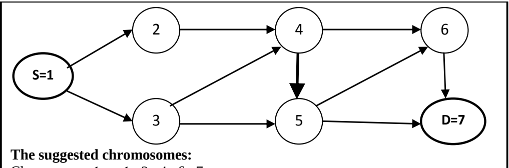
2.3 Fitness Function

Fitness function is the tool used to evaluate the different chromosomes (i.e., routing paths) to find the best of them, so it should be designed carefully to get the best results. In shortest path routing problems, the best solution should have the minimal route cost from the source node to the destination node in the network graph. So, the fitness function is defined as in the folowing equation [14].

$$F_i = \left[\sum_{j=1}^{n_i - 1} C_{g_i(j), g_i(j+1)} \right]^{-1}$$

where F_i is chromosome(i) fitness value, n_i is the chromosome (i) length or the number of genes in this chromosome, g_i (j) is the gene in locus number (j) for chromosome (i), and C_{ij} is the cost of the link that connects node i with node j.

The fitness function gives highest value for the best chromosome with the lowest cost. Also, the fitness function is important for the selection process to select the parents that will be used to produce the next generation. Figure 2.2 shows an example of a small network with some initial paths.



The suggested chromosomes:
Chromosome 1 = 1 2 4 6 7
Chromosome 2 = 1 2 4 5 7
Chromosome 3 = 1 2 4 5 6 7
Chromosome 4 = 1 3 4 6 7
Chromosome 5 = 1 3 4 5 7
Chromosome 6 = 1 3 4 5 6 7
Chromosome 7 = 1 3 5 7
Chromosome 8 = 1 3 5 6 7

Figure 2.2: An example of network and initial paths

2.4 Genetic Operators

Selecting suitable genetic operators (Selection, Crossover and Mutation) for shortest path routing problem is a very important step that makes the suggested GA give the best solution in minimum execution time[13, 14, and 16].

2.4.1 Selection

Different selection schemes can work fine with the suggested GA, but tournament selection without replacementseems to be more suitable because it gives the best-fitted chromosomes more chances to be selected for crossover operation and keeps the selection noise very low. Selection pressure is easily adjusted by changing the tournament size. If the tournament size is larger, weak individuals have a smaller chance to be selected. The pair-wise tournament selection is used. Two chromosomes randomly selected from the population and select the fitter one, noting that any chromosome selected as a parent for crossover operation should not be picked again [9, 13, 14, and 16].

2.4.2 Crossover

Crossover is a procedure of using solutions from the population as parents and producing child solutions from them. These offspring (i.e., child solutions) used for the next generation. A one-point crossover scheme is used in the suggested GA to solve the shortest path routing problem, so each parent is divided into two parts (partial route) and these parts are exchanged to create two new offspring. In the parent chromosomes, the first part connects the source node with an intermediate node and the second part connects the intermediate node to the destination node. A crossing point is selected randomly in each parent, so maybe we get two children with different sizes.In this procedure, the two-parent chromosomes selected for a crossover operation should have one common gene (node) at least, except the source node and the destination node. This common node is considered to be the intermediate node and is not required to be at same location in both chromosomes (i.e., routing paths). If there is more than one common node in both parents, then one of them is randomly selected to be the intermediate node used for the crossing site [9, 13, 14, and 16]. Figure 2.3 describes an example of the suggested crossover process. It shows two routing paths (chromosomes) from a source node (S) to a destination node (D), with different size chromosomes. Two common nodes are noticed in the chromosomes (N_2 and N_5), and these nodes are located in positions (3 and 5) for the first chromosome and (2 and 4) for the second chromosome. In other words, we noticed node N_2 is located in locus 3 in the first chromosome and in locus 2 in the second chromosome, so the pair (3, 2) represents a potential crossing site that represents the location of the crossing of both nodes. Similarly, we noticed N_5 is located in locus 5 in the first chromosome and locus 4 in the second one, so the pair (5, 4) is a second potential crossing site for these two chromosomes. Then, randomly one pair is selected (3, 2) or N_2, which represents the crossing points of the chromosomes. Note that the crossing points of any two chromosomes no need to be the same. After that, the partial routes are exchanged between the two chromosomes and reassembled to get two new chromosomes as shown in the figure. Sometime after crossover, the new chromosomes contain loops that lead to infeasible solutions, and to solve this problem we used a repair function to treat the chromosomes and convert the infeasible chromosomes to feasible chromosomes [9,13, 14, and 16].

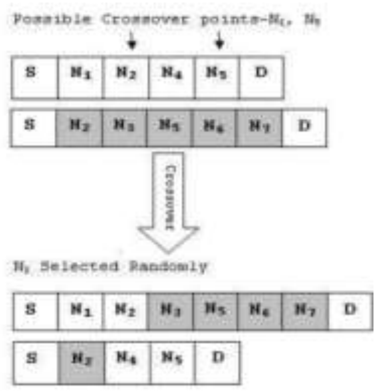


Figure 2.3: Example of the crossover process [14]

2.4.3 Mutation

Mutation is the process of flipping one gene or making a little change on a selected chromosome. This process is important to keep the solutions away from local optima. In this suggested GA, a chromosome (routing path) is selected randomly from the current population and one intermediate node is selected randomly also in this chromosome. Next, the first partial route from the source node to the mutation node is kept, but the second partial route that forms the mutation node to the destination node is rebuilt again, depending on the topological knowledge about the network (as we did in initial population step) [13, 14, and 16]. Figure 2.4 shows an example of the suggested mutation operation. This example shows a chosen chromosome with a randomly selected gene or mutation point (node N_2). The partial route from the source node (S) to the mutation point (N_2) is kept for the new chromosome and completed as follows: the next gene will be one of the neighbors of the mutation node. This is selected randomly, depending known network topology. Then the process continues as same as the initial population process to get a new feasible route has same upper partial route (before mutation point) but different lower partial route [13, 14, and 16].

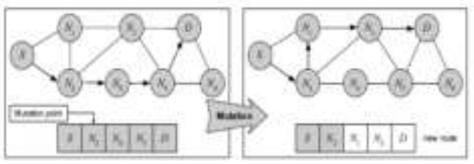


Figure 2.4. Example of the mutation operation [14]

2.5 Repair Function

In generating the initial population, all chromosomes are feasible without any repeated nodes in the chromosomes because once the node is selected for the path it will not selected again in the same path; also same idea happens with mutation process. But during crossover operation, an infeasible chromosome may be produced when a node is repeated in this chromosome (if this node was located in the first part of the first parent and in the second part of the second parent) and loops will be generated, which is not accepted in our algorithm. To keep the suggested GA effective and produce a feasible solution in an optimal time, we need a function to repair the infeasible chromosomes and convert them to feasible chromosomes. A repair function used in the suggested GA is easy and simple. The idea behind the repair function is finding and eliminating loops in the routing paths (i.e., chromosomes) [13, 14, and 16].Figure 2.5 shows an example of the repair function. The example shows an offspring produced from the crossover operation. It is infeasible because a loop appears when node (N_2) is repeated twice in the new routing path (i.e., chromosome). The repair function searches the offspring and finds this loop, then eliminates the genes (nodes) between the repeated nodes (N_2) and one of the repeated nodes. In this example, the repair function deletes nodes (N_3 and N_2) from that chromosome to eliminate the loop and make the chromosome is feasible [13, 14, 16].

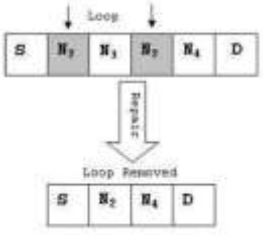


Figure 2.5 Example of the repair function [14]

III. RESULTS AND DISCUSSION

In this Section, Dijkstra's algorithm and the suggested GA will be tested on three networks with different sizes to obtain the shortest route between two nodes (source node and destination node). Both algorithms will be applied first on a small network (10 nodes) for 100 iterations, then on a medium size network (60 nodes) for 100 iterations, and then on a large network (100 nodes) for 100 iterations. The iterations can help in taking the average results by running the algorithms 100 times to insure accuracy for these results. In the suggested GA, the crossover probability is set to 1 and the mutation probability is set to 0.05. The population size is set to 100 chromosomes and the algorithm is terminated after 15 generations. Matlab is used to implement the algorithms and the results were exported to an Excel file to be compared. The experiment was done on a laptop with a core i5 CPU and 4 GB RAM.

3.1 GA and DA on a Small Size Network

In this section, a small network is used to test both graph search algorithms, Dijkstra's and GAs. This network as shown in Figure 3.1 consists of 10 nodes and 14 arcs connecting these nodes. Each node in the graph is denoted by a number that represents the node's ID. This is a weighted graph, so the numbers on the arc which connecting two nodes represents the cost for transferring a packet between these two nodes.

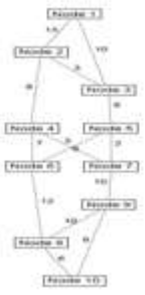


Figure 3.1: Small network [15]

In the small network, the node number 1 is used as the source node and node number 10 is used as the destination node. The shortest optimal path between source node and destination node is (1-3-5-7-9-10) with total costs equal (39).Dijkstra's algorithm and the suggested GA have been applied on this network to find the shortest path. Both algorithms are implemented for 100 iterations on this network. The results show that Dijkstra's algorithm always returns the optimal shortest path with the lowest cost, while the suggested GA returns the optimal shortest path in 97 iterations and in three iterations it gives longer paths. In other words, the accuracy of Dijkstra's algorithm is 100%, while the accuracy of the suggested GA is 97% here.

Table 3.1 concludes the results obtained from this experiment. GA accuracy is the percentage of the suggested algorithm iterations that returned the optimal shortest path. This table also shows the minimum, maximum, and average execution time needed for both DA and the suggested GA to find the shortest route in the 100 iterations. It is clear that the average execution time for running Dijkstra's algorithm on this small network (0.00215 seconds) is much lower than the average execution time for running the suggested GA (0.0628 seconds). So, in small networks, Dijkstra's algorithm is faster than GA.

Table 3.1: Concluded results of DA and GA applied on a small network

number of nodes = 10		
	DA	**GA**
results accuracy	100%	97%
minimum exe time	0.00210 sec	0.06110 sec
maximum exe time	0.00388 sec	0.06970 sec
average exe time	**0.00215 sec**	0.06280 sec

3.2 GA and DA on a Medium Size Network

In this section, a medium network is used to test both graph search algorithms, Dijkstra's algorithm and the suggested GA. This network is shown in Figure 3.2 and consists of 60 nodes and 92 arcs connecting these nodes.

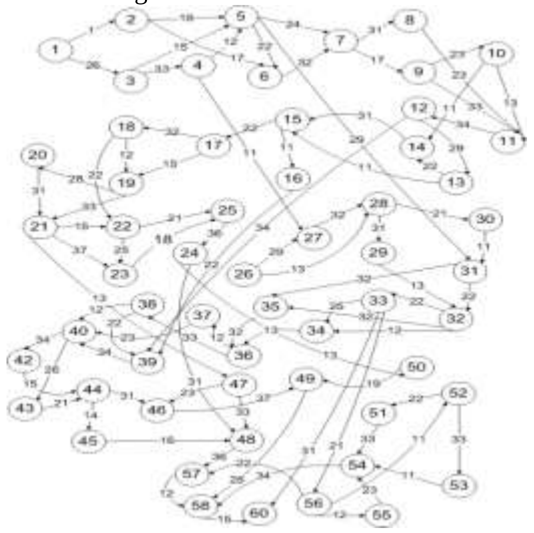


Figure 3.2: Medium network

In this implementation, node number 1 is used as the source node and node number 60 is used as the destination node. The shortest optimal path between source node and destination node is (1-2-5-31-32-33-60) with total costs equal (123).Dijkstra's algorithm and GA have been applied on this network to determine the shortest path. Both of them are implemented for 100 iterations, The obtained results indicate that Dijkstra's algorithm always returns the optimal shortest path with the lowest cost, while the suggested GA returns the optimal shortest path in 93 iterations and in 7 iterations it produces longer paths. In other words, the accuracy of Dijkstra's algorithm is 100%, while the accuracy of the suggested GA is 93% here.The results are summarized in Table 3.2. GA accuracy is the percentage of the suggested algorithm iterations that returned the optimal shortest path. This table also shows the minimum, maximum, and average execution time needed for both DA and the suggested GA to find the shortest route in the 100 iterations. We noticed that the average execution time for running Dijkstra's algorithm on this medium network (0.121357 seconds) exceeds the average execution time for running the suggested GA (0.096952 seconds). So in medium networks, GA is little faster than Dijkstra's algorithm.

Table 3.2: Concluded results of DA and GA applied on a medium network.

number of nodes = 60		
	DA	**GA**
results accuracy	100%	93%
minimum exe time	0.120402 sec	0.089947 sec
maximum exe time	0.123683 sec	0.121872 sec
average exe time	0.121357 sec	**0.096952 sec**

3.3 GA and DA on a large network

In this section, a large network is used to test both graph search algorithms. Such network is shown in Figure 3.3; it consists of 100 nodes and 158 arcs connecting these nodes.

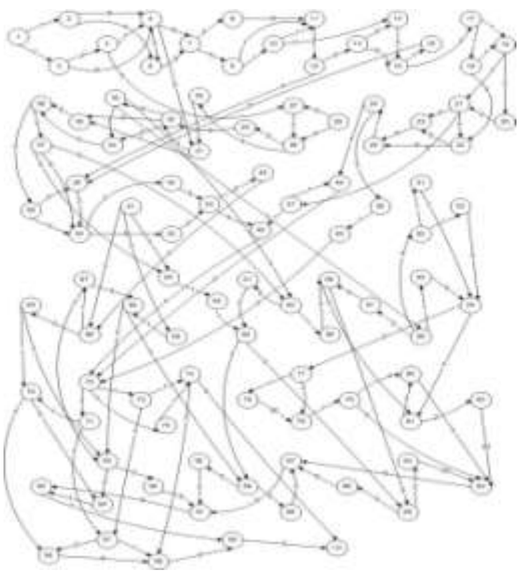


Figure 4.3. Large network

In this implementation node number 1 is used as the source node and node number 100 is used as the destination node. The shortest optimal path between source node and destination node is (1-2-6-7-9-11-12-39-40-42-44-46-72-73-74-100) with total costs equal (43). Dijkstra's algorithm and the suggested GA have been applied on this network to find the shortest path. Both algorithms are implemented for 100 iterations on this network.

The obtained results indicate that Dijkstra's algorithm always returns the optimal shortest path with the lowest cost, while the suggested GA returns the optimal shortest path in 94 iterations and in 6 iterations it produces longer paths. In other words, the accuracy of Dijkstra's algorithm is 100%, while the accuracy of the suggested GA is 94% here.

The results are summarized in Table 3.3. GA accuracy is the percentage of the suggested algorithm iterations that returned the optimal shortest path. This table also shows the minimum, maximum, and average execution time needed for both DA and the suggested GA to find the shortest route in the 100 iterations. We noticed that the average execution time for running Dijkstra's algorithm on this medium network (0.404134 seconds) is more than the average execution time for running the suggested GA (0.140085 seconds). So in large networks, GA is faster than Dijkstra's algorithm.

Table 3.3. Concluded results of DA and GA applied on a large network.

number of nodes = 100		
	DA	**GA**
results accuracy	100%	94%
minimum exe time	0.401095 sec	0.135918 sec
maximum exe time	0.454732 sec	0.153171 sec
average exe time	0.404134 sec	**0.140085 sec**

IV. CONCLUSION

Genetic algorithm (GA) is tested as an alternative algorithm used to obtain the lowest cost route from source node to destination node in computer networks instead of Dijkstra's algorithm, which is the classical graph search algorithm used by most routing protocols (such as OSPF).

The suggested GA uses Munetomo's techniques; it uses real numbers for chromosome encoding, heuristic initialization depending on the knowledge of network topology, tournament selection, one-point crossover, and the mutation method proposed by Munetomo. A repair function is used to treat the infeasible chromosomes and make them feasible.

Dijkstra's algorithm and the suggested GA are tested on three different networks: a small network containing 10 nodes, a medium network containing 60 nodes, and large network containing 100 nodes. DA always finds the correct shortest path, while GA finds it at a percentage of more than 92%. So both algorithms almost find the same path, but the difference is in the execution time needed by the algorithms to find the shortest path. The results obtained show that in a small network, DA is much faster than GA in finding the shortest path and in a medium network, GA is faster than GA but with little difference. While in a large network, GA is faster than DA in finding the shortest path with a clear difference in time.

It can be concluded that DA is better for use in the case of small networks because it finds the shortest path faster than GA, while GA is better in the case of large networks.

REFERENCES

[1]. B. A. Forouzan, "Data Communications and Networking," Fourth Edition, McGraw Hill, 2007.
[2]. W. Stalling, "High-Speed Networks: TCP/IP and ATM Design Principles," Englewood Cliffs, NJ: Prentice-Hall, pp. 131-214, 1998.
[3]. E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," Numerische Mathematlk, 1: 269–271, l 959.
[4]. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Section 24.3: Dijkstra's algorithm. Introduction to Algorithms (Second ed.), MIT Press and McGraw-Hill, pp. 595–601, 2001.

[5]. M. Sniedovich, “Dijkstra’s algorithm revisited: the dynamic programming connexion.” Journal of Control and Cybernetics 35 (3): 599–620, 2006.

[6]. J. H. Holland, Adaptation in Natural and Artificial Systems, Ann Arbor, MI, The University of Michigan Press, 1975.

[7]. D. E. Goldberg, Genetic Algorithms in Search, Optimization, andMachine Learning, Addison-Wesley Publishing Company, 1989.

[8]. Mitchell, M. “An introduction to Genetic Algorithms.” MIT Press, Cambridge MA, 1996.

[9]. S. N. Sivanandam and S. N. Deepa, “Introduction to Genetic Algorithms,” First Edition, Springer, 2008.

[10]. Michalewicz, Z., “Genetic Algorithm + Data Structure = Evolution Programs,” Third Edition, Springer-Verlag, New York, pp. 13- 55, 1996.

[11]. Randy L. Haupt and Sue Ellen Haupt, “Practical Genetic Algorithms,” second edition, Wiley, 2004.

[12]. Back, T., Fogel, D. B. and Michalewicz, Z., “Handbook of Evolutionary Computation.” New York: Institution of Physics Publishing and Oxford University Press, 1997.

[13]. M. Munetomo, Y. Takai, Y. Sato, “A migration scheme for the genetic adaptive routing algorithm,” IEEE International Conference on Systems, Man and Cybernatics, vol. 3, pp. 2774 – 2779, October 1998.

[14]. S.C. Nanayakkara, D. Srinivasan, L.W. Lup, X. German, E. Taylor, and S.H. Ong, “Genetic Algorithm based route planner for large urban street networks”; in Proc. IEEE Congress on Evolutionary Computation, 2007, pp.4469–4474.

[15]. Gihan Nagib and wahied G. Ali, “Network Routing Protocol Using Genetic Algorithms.” International Journal of Electrical & Computer Sciences, IJECS-IJENS vol:10 No: 02, March, 2010.

[16]. Yagvalkya Sharma, Subhash Chandra Saini, Manisha Bhandhari,” Comparison of Dijkstra’s Shortest Path Algorithm with Genetic Algorithm for Static and Dynamic Routing Network,” International Journal of Electronics and Computer Science Engineering, pp. 416–425, 2012

Abdullah M. A. Alzafiri "A Comparative Analysis for Finding Shortest Path Problem in Computer Networks" International Journal of Engineering Research and Applications (IJERA), vol. 9, no. 11, 2019, pp 32-38