International Journal of Computer Trends and Technology
ISSN: 2231 – 2803 / https://doi.org/10.14445/22312803/IJCTT-V69I5P102

Volume 69 Issue 5, 6-12, May 2021

Original Article

Designing and Implementing Shortest and Fastest Paths; A Comparison of Bellman-Ford algorithm, A*, and Dijkstra's algorithms

Al Bager A. Al Bager. R[1], Al Samani A. Ahmed[2]

[1,2]*Faculty of Computer Science and Information Technology, Neelain University, Khartoum, Sudan.*

Received Date: 04 April 2021
Revised Date: 10 May 2021
Accepted date: 12 May 2021

Abstract *- This paper compares the performances of three path algorithms, including the Bellman-Ford algorithm, A*, and Dijkstra's algorithm. These algorithms have found the paths in a map of Riyad,h, and the run times were compared of these algorithms. The experimental findings revealed the effectiveness of A* A search algorithm, followed by Dijkstra algorithm and Bellman-Ford algorithm. This shows that Dijkstra's algorithm can be extended into various fields to solve problems involving the computation of the shortest distance between various locations.*

Keywords *- Computation, Geographic Information System, Riyadh, Road time, Shortest path.*

I. INTRODUCTION

Information is effectively gaining its importance instantly. They become complex and excessive due to the intensity and size of information volume. This information should be adequately managed by organizing (Chen and Xu, 2019). The notion of an information system emerges as the outcome of this requirement. Geographic Information System (GIS) is a type of information systems, having broader area applications (Pramudita et al., 2019). GIS is best utilized for obtaining the information of a product and ca; thus, it perform culture, life and security, environment, and health procedures (Kim, 2019). For example, reaching to accident area, in a traffic accident on a highway, at the shortest time by an ambulance, and making the first aid rely on numerous parameters. Precaution, for reaching the accident areas, are taken into account for road information, traffic intensity, hospital location, life safety, and transportation network, must be efficiently managed (Parvin et al., 2020). Assessing the time information is the best practice for identifying the above information. GIS can assist in collecting such information types.

One of the most critical challenges is finding the best route for reaching a destination when it comes to emergency situations. Finding the shortest path on a map is appropriate and essential issue to be explored and solved (Schröder and Cabral, 2019). Much initiative and research has been done to find the best approach for solving this classic problem. These research initiatives have resulted in the advancement of different algorithms and experimental outcomes regarding their performances. Initially, a class of modified A^* search algorithms were identified and their performances were compared to existing state-of-the art shortest path finding algorithms (Ak, Bahrami and Bozkaya, 2020). Afterward, the possibility of utilizing genetic algorithms for finding solutions to shortest path issues was undertaken in previous studies. Lastly, real road network data was used to compare with the performances of several different cutting-edge algorithms.

In this experiment, three of the more common shortest path algorithms were compared including Bellman-Ford algorithm, A^*, and Dijkstra's algorithm. Road data was used for comparing the performances of these algorithms for finding the shortest node-to-node paths in the map of Riyadh (Abdulaziz, Adewale & Man-Yahya, 2017). These algorithms were integrated via JAVA, and their runtimes were monitored for different test cases. The contribution and motivation in conducting this experiment is to acquire a better comprehension of how different algorithms perform on the real data of Riyadh. These algorithms vary with respect to their trade-off between speed and precision while they are usually used in shortest issues. In this regard, it is expected to find the algorithm that will explore the optimal shortest path from a beginning point to an ultimate point in the shortest time period.

Fig. 1 Map of Riyadh

II. RELATED WORK

The most common shortest path issue is to find out the shortest path from one node to another node in a focused platform. The core objective of Goldberg and Harrelson (2005) was to identify the fastest algorithm for computing the solution for this node-to-node issue. The solution to this issue can be explored by examining merely a segment of the graph, and that shows that the algorithm's run time was relied on visiting nodes (Goldberg and Harrelson, 2005). Thereby, the performances of algorithms were computed as a function of the number of vertices in the solution platform. Distance bounds are implicit in the domain description when the classic A^* search is utilized for solving the node-to-node issue, and thus no preprocessing was needed. On the contrary, a new pre-processing technique was developed to compute the distance bounds rather than just allowing them to be understood in the domain description. For this approach, a number of landmarks are selected for computing the shortest path distances between all apexes of each of such landmarks. Afterward, they utilized such lower bounds, the triangle inequality, and A^* search for developing new algorithms, which were named ALT algorithms.

It is of no surprise that several differences of this algorithm have been developed within the years since A^* search is one of the popular path-finding algorithms. For instance, near-optimum path-finding algorithm namely Hierarchical Path Finding (HPA) A^* was presented by Botea, Muller and Schaeffer (2004), which was a variation of the conventional A^* algorithm. The speed of HPA* was 10 times faster as compared to A^* whereas exploring paths that are throughout 1% of the optimal solution. This approach classifies a map into associated local clusters. At the local levels, the optimal distances are pre-computed to cross the cluster whereas these are traversed in a single big step globally. On the contrary, HPA* returns a complete platform of sub-issues. This is beneficial for changing the destination, not all initiatives will be wasted. Therefore, this approach fits to dynamically modifying environments.

Su, Li and Shiu (2013) have proposed another variation of A^* namely the Genetic Convex A^* (G-CA^*) algorithm. This variation automatically crops the original map into different convex maps. The distance of the shortest path is proven to equal their Manhattan distance between any two vertices throughout a convex map where this distance is formed between two grid points measured along vertices at right angles. The genetic algorithm was employed in Genetic Convex A^* for merging adjacent convex maps and crop the number of chosen key nodes.

A modified Dijkstra's algorithm was investigated in Noto and Sato (2000) as the experimental outcomes of Goldberg and Harrelson (2005) indicated that the Dijkstra's algorithm was one of the core competitors of modified A^* search algorithms. Noto and Sato (2000) extended the traditional Dijkstras algorithm for reducing the search time for acquiring a near-optimal solution. A new algorithm was proposed for applying the Dijkstra method from both directions since the traditional method needs a very long search time considering long path. Despite this algorithm takes merely one out of five of the search time of the traditional Dijkstra method, it does not usually return the optimal solution. Genetic algorithms can further be utilized for solving the shortest path problem. The likelihood of using a genetic algorithm was presented in Gen, Cheng and Wang (1997) for solving the shortest path problem. The most complex activity experienced by the researchers while experimenting was to encode the path in a map into a gene. They utilized a preferred-based encoding method for representing all the map paths. In this approach, the gene position on a chromosome was represented through a node ID, and the value of this ID was utilized for representing the preference of this node to create a unique path with all the nodes (Gen, Cheng and Wang, 1997).

Efficient solutions were explored for shortest path on the basis of optimization issues rather than comparing this algorithm to existing state-of-the-art algorithms. This was due to the performance of genetic algorithms cannot presently outperform any of the traditional algorithms (Gen, Cheng and Wang, 1997). Genetic algorithms and their performances were implemented by Ismail, Sheta and Al-Weshah (2008) and Machado et al. (2011). A genetic algorithm was implemented by Ismail, Sheta and Al-Weshah (2008) for solving a mobile robot path planning issue in a static environment with predictable terrain. Three different environments were proposed with different barriers, which include a moderately scattered environment, a more difficult scattered environment, and an indoor environment.

Routing involves moving packets of data across networks from a source to a destination. Routing involves two phases. The first phase is to find the optimal routing paths. The second is the transport of the data packets in the network using the path that has been established. The most important part of the process of routing is the selection of the optimal path for routing (Garcia et al., 2007). This is because there are many constraints and rules to be met and also the complexity of network topologies. Routing algorithms are very important in the selection of the optimal path. Dijkstra's Algorithm is the standard for selecting the shortest path because it is both simple and efficient (Arisoylu, 2016). Many popular network simulators like OWns depend on Dijkstra's Algorithm to come up with static paths to be used in the process of simulation.

However, in network simulations, the Dijkstra's algorithm "tends to introduce unnecessary link overload' and thus induces false conclusions" (Garcia et al., 2007). This shows that it is important to come up with an extension to this algorithm that will be more efficient in network routing.

The design of Dijkstra's Algorithm is such that it solves only single-source shortest path problems where there are no negative weighs. Telecommunication networks are in this class (Arisoylu, 2016). However, sometimes links are

bidirectional. Therefore, they have to be viewed as a pair of graph edges with opposite directions. One of the most popular applications of the Dijkstra's Algorithm in network routing is the Open Shortest Path First (OSPF) (Arisoylu, 2016). In many networks, the Spanning-Tree Protocol (STP) runs before the OSPF on the network (Figure 2).

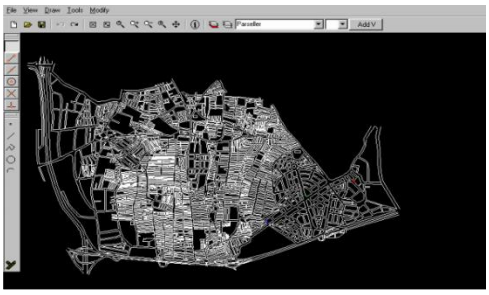

Fig. 2 Software's interface

Thus, the spanning tree is a sub-graph which has all the nodes. Therefore, network environments with redundant links appear closed for the operation of network elements and this is caused by the STP. This eliminates duplicate messages like neighbor discovery messages (Garcia et al., 2007). Rings allow additional levels of connectivity for every node with the cost of just one additional link. The iterative nature of Dijkstra's algorithm causes it to place more paths in particular links than in others in networks links that create rings. Thus, one of the links is likely to be chosen more often than the others (Garcia et al., 2007).

Dijkstra's algorithm is used in finding the shortest path from a source vertex to the destination vertex (Zhou, 2018). The algorithm keeps two sets. One set stores vertex with the shortest distance from the source while the other set has the vertices that are yet to be visited (Broumi et al., 2016). The property that makes Dijkstra's Algorithm to be considered a greedy algorithm is that it picks the vertex that has minimum distance from the source vertex to give the shortest path.
The algorithm is as follows:

A. The algorithm is as Follows

- Create set spSet or the shortest path tree set that will maintain track of all the vertices that are included in the shortest path tree (Zhou, 2018).
- Mark all the vertices that have not yet been visited with INT_MAX (infinity) and the source node mark as zero (Zhou, 2018).
- The spSet does not include all of the graph vertices.
- Pick one vertex that is not in the spSet and label it "u". This vertex must have the lowest weight or value.
- Add the vertex to the spSet.
- Update distances for the other vertices adjacent to "u". For all the adjacent vertices "v", if the sum of the distance of the edge from "u" to "v" and the distance from the source of vertex "u" is less than the distance value of "v", update the value of "v" (Figure 3)

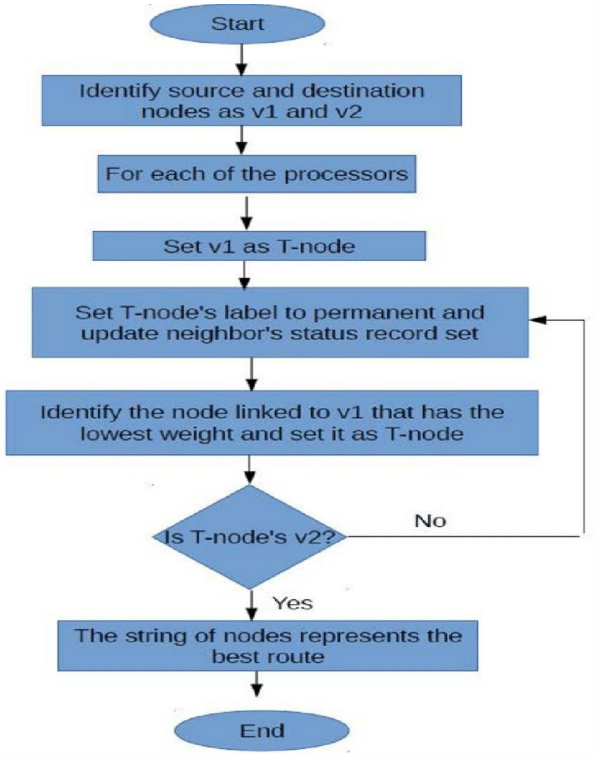

Fig. 3 Dijkstra algorithm flowchart

The Dijkstra's algorithm can be extended to solve these limitations. In a graph, each node can be identified by a unique node identification number (Arisoylu, 2016). The initial Dijkstra's Algorithm can be extended to detect any possible equal cost routes, and make use of more conditions on the bases of the node identification numbers to choose between the routes. Since the additional conditions only run in the presence of over one path candidates for the shortest path, this extended Dijkstra's Algorithm will still provide the shortest path (Garcia et al., 2007).

Roads are one of the most frequently used modes of transportation. In fact, the use of roads can be sad to be indispensable in the world today (Sivakumar and Chandrasekar, 2014). This means that computing the shortest path between various locations is an important issue which happens to be a key problem when it comes to road networks. The many applications of the need to find the shortest distance between places led to the creation of various shortest path algorithms with the aim of overcoming the problem. However, the problem still persists in road networks (Sivakumar and Chandrasekar, 2014). Finding the shortest path between two places is a fundamental problem in road networks because many people face problems when it comes to planning trips in which they will use their own vehicles.

Since the road links in a city possess different congestion levels at various times of the day, making it difficult to determine the shortest path. Therefore, the shortest path can only be determined at the time when need comes, and using an algorithm that is able to take into account all the factors that are involves in determining the length of time that can be taken between two locations (Sivakumar and Chandrasekar, 2014). There are also cases in which large networks of roads are involved in the application, making the computation of the shortest path quite difficult because of the many applications that have to be involved in finding the shortest path over road networks. The shortest path algorithms that have been developed are divided into three categories. There are single-source shortest path algorithms, single-destination shortest path algorithms, and all-pairs shortest path algorithms (Sivakumar and Chandrasekar, 2014).

Dijkstra's shortest path Algorithm is one of the first algorithms to be developed and has led to the creation of many different shortest path algorithms. It is also the most frequently used algorithm when it comes to problems of solving the shortest path between different locations (Sivakumar and Chandrasekar, 2014). According to Bauer et al. (2010), there is need to come up with an efficient shortest path route specifically for the road network. They came up with a new algorithm for calculating the shortest path. This they did by modifying Dijkstra's Algorithm by using goal-directed and combining hierarchical techniques (Bauer et al., 2010). Although the algorithm that they developed is better than Dijkstra's Algorithm when it comes to the results of computation, it takes quite a lot of time to carry out computation than the existing Dijkstra's Algorithm.

Swathika et al. (2016) also carried out an analysis of the various available shortest path algorithms. Through this, they found that the already existing algorithms abound in problems. Therefore, they developed another shortest path algorithm which they called the A* shortest path algorithm, an extension of Dijkstra's shortest path algorithm (Swathika et al., 2016). The main difference is that the new algorithm included such parameters as modified weights and cost which are not there in Dijkstra's shortest path algorithms. The results that this algorithm gives are better than those that are given by the original Dijkstra's shortest path algorithm. The complexity of the computation is quite high (Swathika et al., 2016). To reduce the computational time, they used the partitioning graphs technique used in the original Dijkstra's algorithm so as to reduce computation.

Sivakumar and Chandrasekar (2014) state that on road networks, the computation of the shortest distance between any two points is a big problem. They carried out an analysis of the various shortest path algorithms that are already in existence with the aim of determining the most efficient one in calculating the shortest distance between different points in road network (Sivakumar and Chandrasekar, 2014). Dijkstra's algorithm was found to be the most appropriate in the calculation of the shortest path. However, they found out that the existing Dijkstra's algorithm needs some modifications to be highly appropriate and become more efficient in finding the shortest path and to ensure that computation is not very complex (Sivakumar and Chandrasekar, 2014). They proposed a new algorithm which they called the Modified Dijkstra's shortest path algorithm. In the proposed algorithm, different parameters rather than just one are used in finding the shortest path between different locations. They measured the efficiency of the new model by measuring its time complexity and nodes (Sivakumar and Chandrasekar, 2014). The proposed algorithm was compared with the Dijkstra's algorithm, which showed that MDSP took a smaller number of nodes compared to Dijkstra's algorithm. It also takes lesser time in computing the shortest path than the already existing algorithm.

Ananta, Jiang and Muslim (2014) proposed a multicast algorithm for SDN on the basis of the extended algorithm that was proposed by Jiang et al (2014). Pyretic was used in implementing the algorithm proposed and compare it with other basic algorithms related to it. Comparisons of the algorithm they proposed with other algorithms shows it as the most efficient. The proposed multicast algorithm has its basis on the multicast tree construction algorithm. It makes use of the extended Dijkstra's algorithm for multicast group publisher which sends data packets to every member in the multicast group of subscribers (Ananta, Jiang & Muslim, 2014). "The multicast tree construction algorithm for the proposed multicast algorithm is called the EDSPT (Extended Dijkstra's Shortest Path Tree) algorithm" (Ananta, Jiang & Muslim, 2014). Simulation results showed that the proposed multicast algorithm was more efficient than the algorithms that existed before it in measuring the shortest distance between locations.

Abdulaziz, Adewale and Man-yahya (2017) proposed an improved extended Dijkstra's algorithm for software-defined networking (mED-SDN). The man feature of mED-SDN is REST. The application must be authenticated against the controller for REST API calls to be made to the controller. To control congestion, the new algorithm uses bandwidth as the evaluation criterion for the improvement of congestion in software-defined networking typologies. When the utilization of the bandwidth goes above the threshold that has been set by the algorithm, it reverts to the controller in search for a new path. In order to get the bandwidth usage link, the congestion component measures he topology's bandwidth and uses the REST AP present on the controller so as to collect the cumulative bytes transmitted through the openFlow switches port.

III. ANALYSIS

All three algorithms were tested on 10 different pairs (30 in total). These paths were selected, as mentioned in the experimental design section, that the Euclidean distances were short and some of them were longer between some of them. The Euclidean distances of each of these paths was computed and presented in Table 1. It can be observed from the run times for A^* search for each of the pairs remains consistent within all of the trials (Table 2). Likewise, the run times, as shown in Table 3 and 4 are moderately consistent within all three trials for the Bellman-Ford and Dijkstra algorithm. Before prior testing, it was essential to identify that the paths were not computed on the basis of rising distances between them. Therefore, findings plot a better visual illustration of the association between the end node, the run time, and the start node.

For the A^* search algorithm, the run time for path finding was between 4.0-15.7 seconds for the paths tested. The run time has further adequately returned null for the path 7 because there was no existence of path, leading from its start node to its end node. The run time was not often elevating when the distance rose although there was a slight rise in the association between the run time and distance of the paths. These might have occurred because the distances were larger for few paths, some might have needed going via several road segments. In the context of Dijkstra algorithm, the association between the run time and distance was stronger as can be witnessed. This algorithm, like previous one, returned null for a non-existing platform. In addition, the run times for path finding were greater in Dijkstra algorithm as compared to the A^* search run times. The run times rose more rapidly with the Dijkstra algorithm since it works effectively when the goal nodes were closer to the commencing nodes.

The Dijkstra algorithm ends as soon as the destination node is in reachability for this behavior, and this shows that computation time was a lot rapider. Lastly, the findings have indicated that the run times are higher for all the paths from the Dijkstra algorithm to the A^* search algorithm. It was observed that A^* search was faster as it utilizes a beat first search approach and uses a heuristic. On the contrary, a greedy approach was used in Dijkstra algorithm in order to search, and does a blind search, which is a drawback in a data set such as the Riyadh data as it was huge. Lastly, the Bellman-Ford algorithm performed adversely out of the three tested experiments. Higher run times were witnessed for the respective algorithm as compared to A^* search and Dijkstra algorithms for every tested path. On the contrary, the correct outputs were returned; for instance, returning null for a path that did not exist. The Riyadh road data was devised for handling adverse edge weights since but it performs adversely as compared to the other algorithms in the situation where all the edges were non-negative.

IV. EXPERIMENT, DESIGN, AND RESULTS

A total of ten sets of start-end nodes were selected, for the experiment, and texted on all three algorithms for calculating the run times. A combination of both short and long paths was selected for ensuring that the algorithm works perfectly. The longest path selected has a distance of 4394.4 ((1370, 5065), (1037, 9084)), whereas the shortest path selected has a distance of 503.4 ((1160, 7685), (0185, 7294)). A set was further encompassed where it will be unlikely for the algorithms to perfectly explore a path as it does not exist ((0540, 7237), (2259, 7064)). It can be observed, that all algorithms return null for path seven as there was no path leading from that start to end node (Figure 3). Three trials were conducted for each algorithm and the averages were computed for better accuracy. The ten paths as well as their respective distances were presented in Table 1. The run times of the A^*, Bellman-Ford, and Dijkstra algorithms were shown in Tables 2-4, respectively from all trials and their averages. A plot of distance versus run time was also illustrated for all three algorithms.

Table 1. Paths

Path	Distance
1	1756.1
2	824.1
3	1379.1
4	2579.6
5	4493.5
6	504.1
7	2514
8	589
9	2015.1
10	2524

Table 2. A^* Algorithm

Path	Trial 1 (ms)	Trial 2 (ms)	Trial 3 (ms)	Average (ms)
1	6	6	6	6
2	7	7	7	7
3	15	17	17	16.3
4	9	9	10	10.5
5	20	21	22	23
6	3	2	2	2.5
7	Null	Null	Null	Null
8	5	6	6	6.8
9	10	11	12	12.5
10	9	10	11	10.5

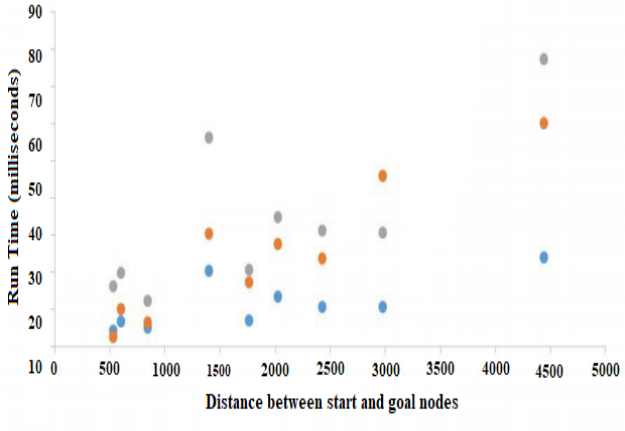

Fig. 4 Distance between Start and Goal Nodes

Table 3. Dijkstra's algorithm

Path	Trial 1 (ms)	Trial 2 (ms)	Trial 3 (ms)	Average (ms)
1	15	15	20	16.5
2	5	5	6	7.1
3	29	30	29	29.5
4	44	46	45	45.1
5	60	58	57	58
6	2	2	2	2.5
7	Null	Null	Null	Null
8	11	9	10	11.1
9	24	25	29	26.1
10	22	23	22	23.1

Table 4. Bellman-Ford Algorithm

Path	Trial 1 (ms)	Trial 2 (ms)	Trial 3 (ms)	Average (ms)
1	20	19	20	20.5
2	12	11	11	11.5
3	56	52	56	54.1
4	26	32	31	29.4
5	78	79	71	73.4
6	14	16	16	15.6
7	Null	Null	Null	Null
8	18	18	17	18.1
9	34	35	32	33.1
10	30	29	29	29.4

The selection of the shortest path using Dijkstra's algorithm is very important in various scenarios to cut on costs and time spent. This is because there are many emergency situations that require the shortest path between locations to be found for the shortest time to be taken between various locations. This algorithm is the fastest "single-source shortest path algorithm for arbitrary directed graphs having unbounded non-negative weights" (Mathur, Jakhotia & Lavalekar, 2014). When used, it is not required to keep investigating the paths because once it runs, the shortest that that can be used is discovered without having to draw any more diagrams. As such, it makes the getting of results faster and reduces the cost of computation even for large problems. The only limitation of this algorithm is that it does not support negative weights on the edges.

The paper has described the various areas into which Dijkstra's algorithm can be applied. In road networks, the algorithm has been extended to calculate the locations between various locations taking into account such factors as traffic jam at various time of the day. It has also been used in network routing protocols to help in finding the shortest path between routed devices. In software-defined networking, it has been extended to consider the weights not only at the edges but also on the nodes and also to consider negative weights on the edges. In autonomous evacuation navigation system, it has been extended so that it can detect not only the shortest but also the safest path. This shows that Dijkstra's algorithm can be extended into various fields to solve problems involving the computation of the shortest distance between various locations.

V. CONCLUSION

By concluding, the real road data of Riyadh was used to test the performances of the Dijkstra, A^* search, and Bellman-Ford algorithms. The experimental findings indicated that the A^* search algorithm performed effectively of all the tested algorithms. On the contrary, the Dijkstra algorithm had its benefits in that it was less complicated for programming as compared to the A^* search. If Bellman-Ford was used with negative edges even, it could be a very beneficial algorithm.

There are several interesting probabilities that can be investigated with respect to future work in this field. Firstly, the implemented algorithms can be changed for performing more efficiently. For instance, A^* search algorithm could have had a better heuristic function. On the other hand, it could be changed for performing effectively such as the algorithms proposed in previous studies. Future work could encompass experiments that would be performed for determining how effectively these modified algorithms were than to the A^* search algorithm implemented in this study. Genetic algorithms are also potential area for future work in the field of finding path algorithms.

In road networks, the algorithm has been extended to calculate the locations between various locations taking into account such factors as traffic jam at various time of the day. It has also been used in network routing protocols to help in finding the shortest path between routed devices. In software-defined networking, it has been extended to consider the weights not only at the edges but also on the nodes and also to consider negative weights on the edges. In autonomous evacuation navigation system, it has been extended so that it can detect not only the shortest but also the safest path.

REFERENCES

[1] Abdulaziz, A. H., Adewale, E., and Man-Yahya, S. Improved Extended Dijkstra's Algorithm for Software Defined Networks. International Journal of Applied Information Systems, 12 (2017) 22-26.

[2] Ak, R., Bahrami, M. and Bozkaya, B. A time-based model and GIS framework for assessing hazardous materials transportation risk in urban areas. Journal of Transport & Health, 19 (2020) 100943.

[3] Ananta, M. T., Jiang, J. R., & Muslim, M. A. Multicasting with the extended Dijkstra's shortest path algorithm for software defined networking. International Journal of Applied Engineering Research, 9(23) (2014) 21017-21030.

[4] Arisoylu, M. An initial analysis of packet function-aware extension to Dijkstra algorithm for wireless networks. EURASIP Journal on Wireless Communications and Networking, 2016(1)(2016) , 65.

[5] Bauer, R., Delling, D., Schieferdecker, P., Schulters, D., and Wagner, D. Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm. ACM Journal of Experimental Algorithmics, (2010) 33-42.

[6] Botea, A., Müller, M. and Schaeffer, J.,. Near optimal hierarchical path-finding. J. Game Dev., 1(1) (2004) 1-30.

[7] Broumi, S., Bakal, A., Talea, M., Smarandache, F., & Vladareanu, L. Applying Dijkstra algorithm for solving neutrosophic shortest path problem. In 2016 International Conference on Advanced Mechatronic Systems (ICAMechS) (2016) 412-416. IEEE.

[8] Chen, Q. and Xu, N., 2019, December. Research on the Shortest Path Analysis Method in Complex Traffic Environment Based on GIS. In 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC) 1, 208-212. IEEE.

[9] Garcia, N. M., Lenkiewicz, P., Freire, M. M., and Monteiro, P. P. 2007. On the Performance of Shortest Path Routing Algorithms for Modeling and Simulation of Static Source Routed Networks--an Extension to the Dijkstra Algorithm. In 2007 Second international conference on systems and networks communications (ICSNC 2007) 60-60. IEEE.

[10] Gen, M., Cheng, R. and Wang, D., April. Genetic algorithms for solving shortest path problems. In Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)(1997) 401-406. IEEE.

[11] Dr. Kavita, Dr. M. Anji Reddy, Geospatial Database Creation for Town Planning Using Satellite Data under GIS Environment SSRG International Journal of Civil Engineering 4(6) (2017) 98-102.

[12] Goldberg, A.V. and Harrelson, C., January. Computing the shortest path: A search meets graph theory. In SODA 5,(2005) 156-165).

[13] Ismail, A.T., Sheta, A. and Al-Weshah, M.. A mobile robot path planning using genetic algorithm in static environment. Journal of Computer Science, 4(4) (2008) 341-344.

[14] Jiang, J. R., Huang, H. W., Liao, J. H., and Chen, S. Y. Extending Dijkstra's shortest path algorithm for software defined networking. In The 16th Asia-Pacific Network Operations and Management Symposium (2014) 1-4. IEEE.

[15] Kim, H., 2019. Using Geographic Information Systems (GIS) to optimize delivery system services for BCCLS libraries.

[16] Machado, A.F.D.V., Santos, U.O., Vale, H., Gonçalvez, R., Neves, T., Ochi, L.S. and Clua, E.W.G., November. Real time pathfinding with genetic algorithm. In 2011 Brazilian Symposium on Games and Digital Entertainment (2011) (215-221). IEEE.

[17] Noto, M. and Sato, H., October. A method for the shortest path search by extended Dijkstra algorithm. In Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'cat. no. 0 3 (2000) 2316-2320. IEEE.

[18] Parvin, F., Ali, S.A., Hashmi, S.N.I. and Khatoon, A.. Accessibility and site suitability for healthcare services using GIS-based hybrid decision-making approach: a study in Murshidabad, India. Spatial Information Research,(2020) 1-18.

[19] Pramudita, R., Heryanto, H., Handayanto, R.T., Setiyadi, D., Arifin, R.W. and Safitri, N., October. Shortest Path Calculation Algorithms for Geographic Information Systems. In 2019 Fourth International Conference on Informatics and Computing (ICIC) (2019) 1-5. IEEE.

[20] Schröder, M. and Cabral, P.. Eco-friendly 3D-Routing: A GIS based 3D-Routing-Model to estimate and reduce CO2-emissions of distribution transports. Computers, Environment and Urban Systems, 73 (2019) 40-55.

[21] Sivakumar, S., and Chandrasekar, C. Modified Dijkstra's Shortest Path Algorithm. International Journal of Innovative Research In Computer And Communication Engineering, 2(11) (2014) 1-7.

[22] Su, P., Li, Y., Li, Y. and Shiu, S.C.K. An auto-adaptive convex map generating path-finding algorithm: Genetic Convex A. International Journal of Machine Learning and Cybernetics, 4(5)(2013) 551-563.

[23] Zhou, T. Deep learning models for route planning in road networks. (2018).