Journal of Innovative Technology Convergence
Vol. 2, No. 1 June 2020, pp. 1-10

The A Study of the Multidirectional Dijkstra's Algorithm in Solving the Shortest Route Problem

Jeacar D. Zamora [1], Ace Marco Floro E. Angeles [2], Marnei A. Manalo [3]

Abstract: This study primarily aims to save priority routes for quick emergency response in all kinds of emergency systems using the Multidirectional Dijkstra's algorithm. This is particularly focused on finding the shortest emergency route for a vehicle traveling from the starting node to its nearest possible destination. A fictional map will also be presented which was designed to simulate a process of Multidirectional Dijkstra's algorithm. It sorts out the entire fastest route in the shortest amount of time that could travel multiple destinations, and then vary the destination in a short period of time. This study will benefit the government or emergency departments by giving them immediate solutions for emergency problems.

Keywords: Multidirectional Dijkstra's algorithm, Shortest route problem, Fictional map, Quick emergency response

1. Introduction

The shortest route problem is generally associated with a route or map navigation. This study will be focused on emergency routing services provided by the hospitals, fire stations, police departments and also by other emergency route-related departments using proposed Multidirectional Dijkstra's Algorithm (MDA). The MDA refers to a hybrid algorithm that addresses the shortest route problem based on natural Dijkstra's algorithm and bidirectional search. This study utilizes a fictional map with two different traffic percentages and also different stages of emergency department's capabilities to choose to as multiple destinations.

Emergency vehicles are used for emergency transportation of sick or injured people, search and rescue teams, police and other emergency response-giving departments between places. The various emergency cases can be identified as: (1) sick and injured people; (2) crimes; (3) vehicular accidents; (4) burning houses and building; (5) search and rescue operations; and (6) natural disasters. The importance of these services is immense in saving of lives, as the slightest delay could be fatal. It also

[1] College of Computer Studies, University of Antique, Sibalom, Antique, Philippines
Email: jeacarzamora@yahoo.com
[2] College of Computer Studies, University of Antique, Sibalom, Antique, Philippines
Email: eysangeles@gmail.com
[3] College of Computer Studies, University of Antique, Sibalom, Antique, Philippines
Email: marnei.manalo@gmail.com

Received [January 11, 2020]; Revised [March 15, 2020]; Accepted [April 20, 2020]

ensures that citizens' lives are saved and protected, regardless with their location and time of day when an emergency call is made. Table 1 outlines the hospital capability levels.

Table 1. Hospital Capability Levels

Hospital Capability Level	**Description**
Level 1	Capable of initial treatment for cases that require immediate treatment. Minor surgeries, and non-surgical gynecology, surgery, anesthesia, obstetrics, first level radiology.
Level 2	Capable of all clinical services provided by Level 1 hospitals. Specialty clinical care tertiary clinical laboratory, pharmacy, second level radiology. Nursing care for patients needing total and intensive care.
Level 3	Capable of all clinical services provided by Level 1 and Level 2 hospitals. Specialized forms of treatments, intensive care and surgical procedures. Tertiary clinical laboratory, third level radiology, pharmacy. Nursing care for patients needing continuous and specialized critical care.

The MDA is essential in simulating the shortest route from the origin to any single or multiple destinations where traffic can be considered which affects the speed and time of travel. This means, in order to travel faster, there is a need to lessen the time consumed by avoiding paths that are prone to traffic jams. Most of the private, public, and local vehicles travel on national roads than on local roads. This makes the national roads more congested than the local roads. Traffic, time of travel, and distance are the variables that will be considered. The stages of the hospital capability (*i.e.*, based on Health Philippines) are also a variable on this problem since not all hospitals are capable of treating patients with higher level of emergency needs.

This study aims to show how to find the shortest emergency route for an emergency vehicle to travel from a starting place to the nearest destination using Multidirectional Dijkstra's algorithm as depicted in the theoretical framework shown in Figure 1. The specific objectives of this study includes: (1) to solve the mean running time between natural Dijkstra's algorithm, bidirectional search and MDA in solving the shortest route in a graph when grouped as number of nodes, number of edges, and number of weights; (2) to prove the reliability of MDA in solving the shortest route in terms of searching for multiple destinations simultaneously; and (3) to determine the significant difference between, natural Dijkstra's algorithm, bidirectional search, and MDA.

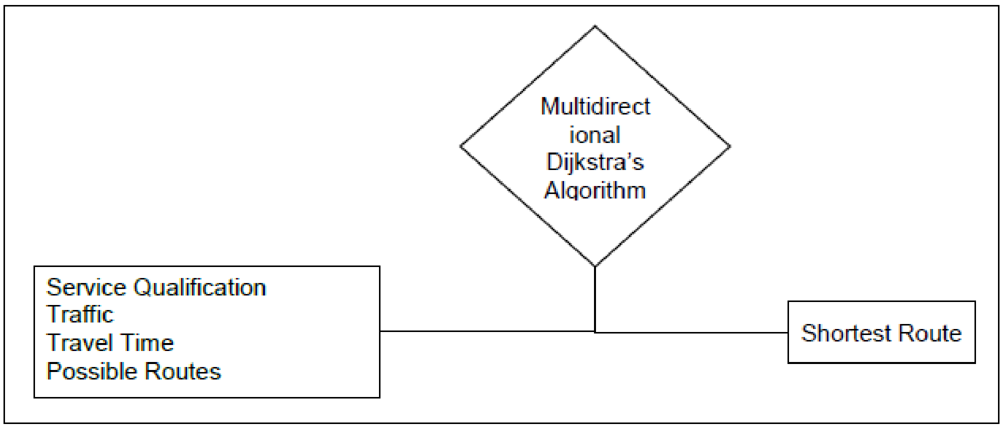


Figure 1. Theoretical Framework

The remainder of this paper is organized as follows: Section 2 outlines the different related literature as basis for the design of the proposed algorithm in solving the shortest route problem; Section 3 details the theoretical framework for the modified Multidirectional Dijkstra's algorithm; and Section 4 concludes the study.

2. Related Literature

This section outlines the various literature that was helpful and significant in showing the connection of concepts in this study.

2.1 Dijkstra's Algorithm

In 1959, algorithms were proposed by Edsger W. Dijkstra, a 29 years old computer scientist, to address the two fundamental graph theoretic problems such as the minimum weight spanning tree problem and the shortest path problem [1]. Nowadays, Dijkstra's algorithm in addressing the shortest path problem is one of the most distinguished algorithms in the fields of computer science (CS) and has become a very popular algorithm in operations research (OR) [2].

The Dijkstra's algorithm is often described as a greedy algorithm because it tests all possible routes in a location or graph. The algorithm is generally considered as a "computer science method" that is since its author was a computer scientist and its association with special data structures [2]. The concept of the algorithm is that for a given source vertex (node) in the graph, the algorithm finds the path with the shortest path between that vertex and every other vertex. It can also be used for finding the cost of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined [3].

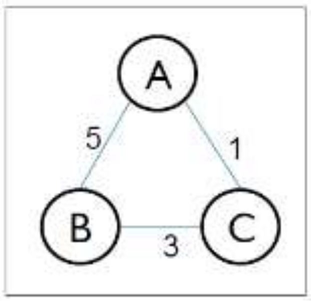


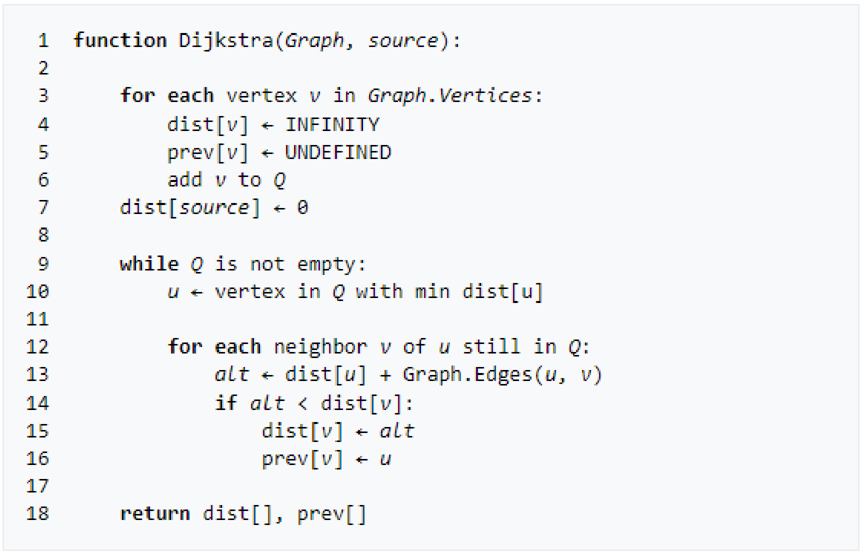
Figure 2. An Example of the Dijkstra's Algorithm

In this study, the Dijkstra's Algorithm will be adapted to create a simulation to find the fastest possible route. Figure 2 depicts an example of how Dijkstra's algorithm work and is described in Equation 1.

From A (*i.e.*, starting point), the shortest path to B (*i.e.*, destination) is ACB, because

$$AB = 5 \text{ and } ACB = 4 \quad (5 > 4) \tag{1}$$

Dijkstra's algorithm is an efficient and distinguished algorithm for finding the shortest paths between vertices (nodes) in a graph, which may represent, for example, road networks [1][4]. The algorithm generally fixes a single vertex as the "source" (*i.e.*, starting place) and finds shortest paths to all other vertices in the graph (*i.e.*, destinations), producing a shortest-path tree. For example, if the vertices of the graph represent cities and edge path costs represent the driving distances between pairs of cities connected by a direct road. Then, the Dijkstra's algorithm can be utilized in finding the shortest route between one city and all other cities.

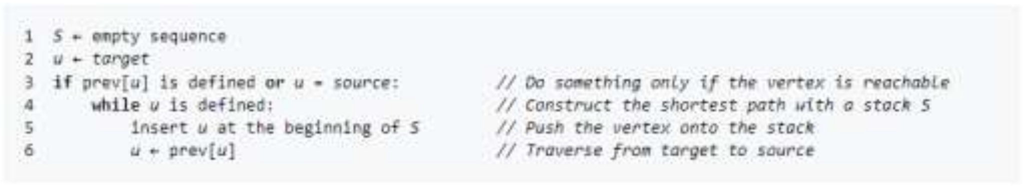


```
1   function Dijkstra(Graph, source):
2
3       for each vertex v in Graph.Vertices:
4           dist[v] ← INFINITY
5           prev[v] ← UNDEFINED
6           add v to Q
7       dist[source] ← 0
8
9       while Q is not empty:
10          u ← vertex in Q with min dist[u]
11
12          for each neighbor v of u still in Q:
13              alt ← dist[u] + Graph.Edges(u, v)
14              if alt < dist[v]:
15                  dist[v] ← alt
16                  prev[v] ← u
17
18      return dist[], prev[]
```

Figure 3. Dijkstra's Algorithm Main Pseudocode

The Dijkstra's algorithm main pseudocode is depicted in Figure 3. The pseudocode can be summarized as follows: (1) the code *u* ← vertex in *Q* with *min dist*[*u*] searches for the vertex *u* in the vertex set *Q* that has the least or minimum distance *dist*[*u*] value; (2) length(*u*, *v*) returns the length of the edge or the distance between vertices *u* and *v*; (3) the variable *alt* on denotes the length of the path from the root vertex to the neighbor vertex *v* if it were to go through vertex *u*; (4) if this path is shorter than the current shortest path recorded for *v*, then, that current path is replaced with this *alt* path; (5) the *prev* array is populated with a pointer to the "next-hop" vertex on the source graph to get the shortest route to the source.

To limit the search to the shortest path between vertices source and destination only, then the search can be terminated after line 13 if *u* = destination. Then, through the reverse iteration depicted in Figure 4, the shortest path from the source to destination can be determined.



```
S ← empty sequence
u ← target
if prev[u] is defined or u = source:         // Do something only if the vertex is reachable
    while u is defined:                      // Construct the shortest path with a stack S
        insert u at the beginning of S       // Push the vertex onto the stack
        u ← prev[u]                          // Traverse from target to source
```

Figure 4. Dijkstra’s Subroutines Algorithm

The Dijkstra’s algorithm search path has been described in Figure 4. It starts by fixing vertex *s* as the source as shown in Figure 5(a). Figure 5(b) shows the distance from the source to its neighboring vertices *t* and *y*, with values 10 and 5, respectively. Since the distance through vertex *y* is shorter as shown in Figure 5(c), the distances of all paths from vertex *y* has been determined and summarized as: (1) the computed distance value from the source going to vertex *z* is 7, (2) the computed distance value from the source going to vertex *x* is 14, and (3) the computed distance value from the source going to vertex *t* is 8. In this stage of the computation, it was seen that instead of going directly from the source to vertex *t* (*i.e.*, distance value is 10), the shorter path can be through vertex *y* (*i.e.*, distance value is 8). In Figure 5(d), the shorter path going to vertex *x* is via vertex *z* in which the distance value was changed from 14 to 13. In Figure 5(e), the distance value via vertex *t* has been computed resulting to a shortest path from the source going to vertex *x* which was changed from 13 to 9. The search path to all the neighboring nodes of the source has been summarized in Figure 5(f).

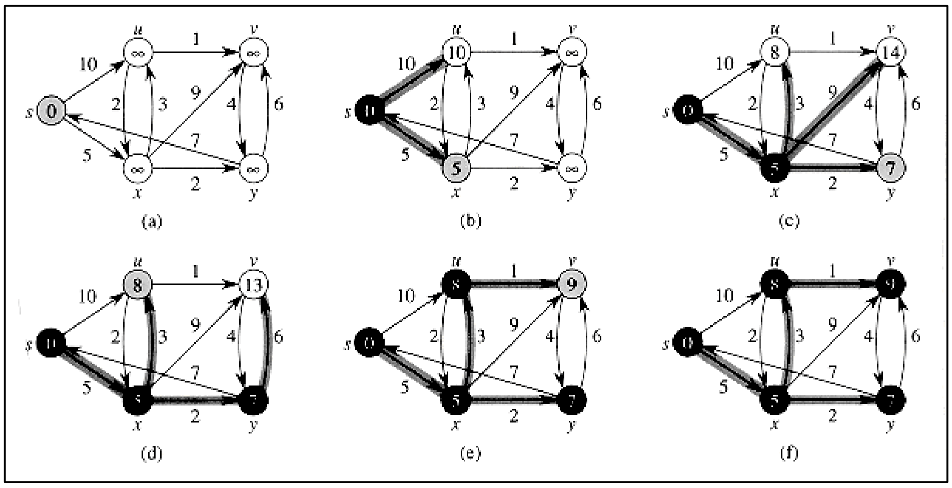


Figure 5. Dijkstra’s Algorithm Search Path [5]

The Dijkstra’s algorithm has been with a variety of enhancements in order to extend its functionalities. There are cases that requires solutions which are less mathematically optimized, while other applications may require more optimal solutions.

For example, Dijkstra’s algorithm can be adapted and integrated with the Bellman-Ford algorithm to handle negative weight edges. The Bellman-Ford algorithm computes for the shortest paths from a single source vertex (or node) to all of the other vertices in a weighted digraph [6]. It can be utilized on graphs with negative edge weights, as long as the graph contains no negative cycle that is reachable from the source vertex (or node). The presence of negative cycles means that there is no shortest path, since the total weight becomes lower each time the cycle is traversed [7]. Thus, the combination of Dijkstra’s algorithm and the Bellman-Ford algorithm is an efficient method in removing negative edges and detecting negative cycles.

Another enhancement of the Dijkstra's algorithm is the A* algorithm which also refers an informed search or a best-first search algorithm. This algorithm is conveyed in terms of weighted graphs that start from s fixed starting vertex (*i.e.*, source) and aimed to find a path to a given destination vertex with the smallest (*e.g.*, least distance travelled, shortest time) [8]. The algorithm achieves better performance through the use of heuristics to guide its search.

The Prim's algorithm refers to a greedy process that finds a minimum spanning tree for all nodes in a weighted undirected graph. This algorithm is more concerned with only the individual paths and does not compute for the total weight of the path from the source [9]. The algorithm is also known as the Jarník's algorithm [10], Prim-Jarník algorithm [11], Prim–Dijkstra algorithm [12] or the DJP (Dijkstra-Jarník-Prim) algorithm [13].

The breadth-first search (BFS) algorithm is regarded as a special-case of Dijkstra's algorithm on unweighted graphs where the priority queue degenerates into a FIFO (first-in, first-out) queue. It is used for searching tree or graph data structures. The algorithm starts at the root (*i.e.*, search key) and explores all of the neighboring vertices at the present depth prior to moving on to the nodes at the next depth level [14]. The fast marching method is regarded as a continuous version of Dijkstra's algorithm that computes the geodesic distance on a triangle mesh [15].

2.2 Bidirectional Search

A bidirectional search depicted in Figure 6 refers to a graph search algorithm that finds the shortest path from an initial vertex (*i.e.*, starting point) to a target vertex (*i.e.*, destination) in a directed graph [16]. This algorithm performs two simultaneous searches: (1) a forward search from the initial vertex, and (2) a backward search from the target vertex. The algorithm stops its operation whenever the two searches meet in the middle. The bidirectional search algorithm can be faster in many cases as the sum of the two search times can be much less the complexity that would result from a single search.

The bi-directional search, just like in an A* search, can be guided by a heuristic estimate of the remaining distance to the destination vertex (*i.e.*, in the forward tree) or from the initial vertex (i.e., in the backward tree) [16]. It aims to reduce the search time through performing two simultaneous searches (*i.e.*, forward and backward searches). It then reconstructs a single path from the initial vertex to the destination after the two searches intersects. The graph diagram for the bidirectional search algorithm is depicted in Figure 6 [17].

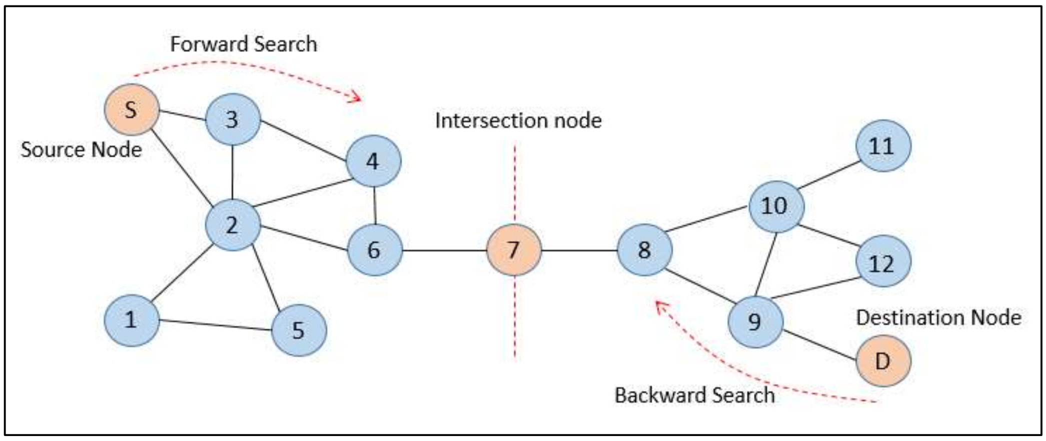


Figure 3. Search Paths from Start Address and Goal Address Connect to Finish the Search

One main issue for the bidirectional search is that when the two searches actually meet. That is, it must be guaranteed that both the forward and backward searches intersects in the middle (i.e., intersection node). In a depth-first search, searches are performed in both directions and is not likely to meet as its small search boundaries might pass by each other. But, with the use of both depth-first search in one direction while using the breadth-first search in the other direction can guarantee that the two searches will meet [18].

To address the issues regarding the intersection of the two searches for the bidirectional search algorithm, strategies to influence the two search wavefronts in order for them to meet quickly must be developed. The integration of the bidirectional search algorithm with other search algorithms can address these issues and limitations as well as to enable a faster and robust shortest path search from the source to destination locations.

3. The Modified Multidirectional Dijkstra’s Algorithm

The shortest route problem determines the shortest possible paths from one node to another. There are various algorithms as discussed in the previous sections that can be used to determine the shortest route between two nodes in a network. The term shortest route refers to the process of calculating the shortest path distance between two points. The start and the destination points are represented as nodes or vertices in graphs.

The multidirectional Dijkstra’s algorithm refers to a hybrid algorithm which is based on the well-known shortest path algorithm, the Dijkstra’s algorithm, with the combination of a bidirectional search algorithm. The features of these algorithms were combined in order to provide a better search outcome. The multidirectional Dijkstra’s algorithm is a greedy algorithm that searches all possible routes, a characteristic came from the generic Dijkstra’s algorithm. The algorithm finds the shortest route by starting on the origin and on the multiple capable destinations in order to minimize the time of searching. The search stops after the path from the starting point meets the path from the destination point which results to a shortest capable destination after the paths meet from both searches. The algorithm refers to a multidirectional searching which is also based on the bidirectional search algorithm.

For example, there is a need to find the shortest path between two locations in a city map. The starting point can be any location in city map where an emergency or accident has occurred, and the destination can be the nearest possible healthcare station. The multidirectional Dijkstra's algorithm starts its process by marking the distance from the starting point to every other intersection on the city map. The starting point is considered as the current intersection and its distance is set to zero, that is in the first iteration. The closest intersection to the starting point will be the next current intersection for the next iterations. The distance is updated for every unvisited intersection that is directly connected to the current intersection by adding its distances. Whenever the distances have been updated or labeled to each neighboring intersections, they will be marked as visited. The neighboring intersections with the shortest distance will be considered for the shortest route from the starting point. The process is continued until the destination intersection has been labeled as visited.

In the previous studies about shortest route problem, Dijkstra’s Algorithm and Bidirectional search algorithm are used to get the shortest route of a given area. Dijkstra’s algorithm is used to find the shortest route by searching all possible routes to destination until it concludes a shortest route, as seen in Figure 7. The bidirectional search is a searching algorithm that enables the search to start searching not only from the origin but also from the destination until both paths meet each other and concluding a shortest path.

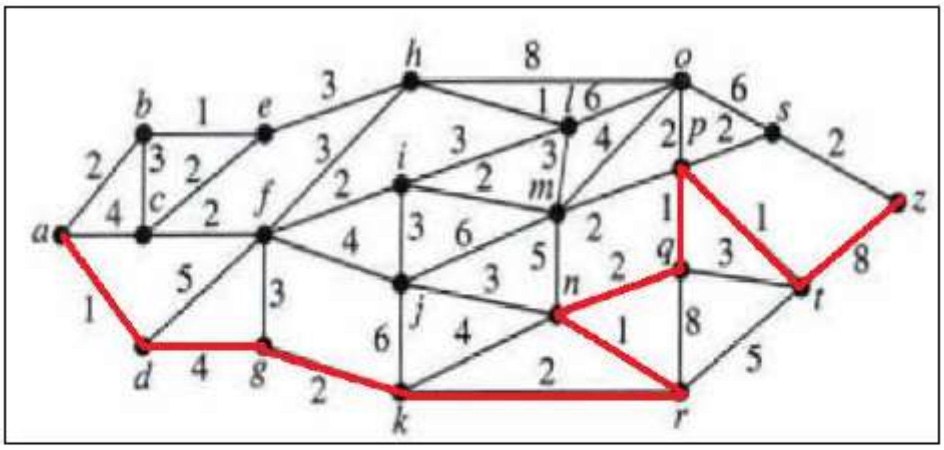


Figure 7. Shortest path in applied Multidirectional Dijkstra's Algorithm

This study determines the highest priority route for emergency transportations in a particular area or a graph. This includes the emergency services or departments such as hospitals, fire departments, police departments, and search & rescue operations, all with different distances in every road to choose from. Both the light and heavy traffic roads are considered for possible routes as well as the levels of hospitals capabilities. This is a paradigm of independent, intervening and dependent variable/s in solving the shortest route using Multidirectional Dijkstra's Algorithm.

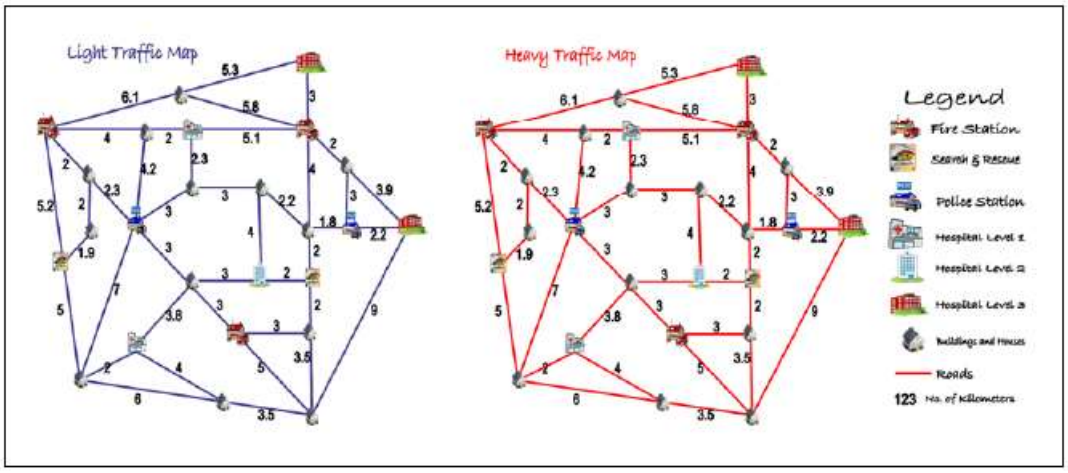


Figure 8. Fictional Map

This study will find a time saving priority route for quick emergency response in any aspects of emergency situation such as hospitals, fire department, police department, and other life-saving organizations. A simulation using Multidirectional Dijkstra's Algorithm applied into a fictional map is designed and developed in this study, as shown in Figure 8. This is a multi-tasking algorithm that sorts out the fastest route to travel from starting place to any single or multiple destinations.

4. Conclusion

The Multidirectional Dijkstra's algorithm can be used in finding the shortest route from one place to another or to multiple destinations. In this study, the characteristics of Dijkstra's Algorithm and Bidirectional Search algorithm were adopted. Combining these two algorithms is another way of enhancing the speed in finding the solution of searching the shortest path. The modified multidirectional Dijkstra's algorithms is a greedy approach and faster computation of distance because it eliminates the unnecessary destinations and search for the shortest route as it checks all possible routes to determine the shortest emergency route. This study will benefit the emergency departments by helping them to find the shortest route for them to give an immediate response and service. The result of this study could also be helpful for future researchers who would like to conduct a related study. This could provide information that they can use in their study.

References

[1] E. W. Dijkstra, "*A Note on Two Problems in Connexion with Graphs*", Numerische Mathematik, vol. 1, December 1959, pp.269-271, doi: 10.1007/BF01386390.

[2] M. Sniedovich, "*Dijkstra's Algorithm Revisited: the OR/MS Connexion*", ifors.ms.unimelb.edu.au, www.ifors.ms.unimelb.edu.au/tutorial/dijkstra_new/ (Accessed November 12, 2019).

[3] K. Mehlhorn and P. Sanders, "*Chapter 10: Shortest Paths*", in Algorithms and Data Structures: The Basic Toolbox, Berlin, Heidelberg, Germany, Springer, 2008, pp.191-215, doi: 10.1007/978-3-540-77978-0, ISBN 978-3-540-77977-3.

[4] P. L. Frana and T. J. Misa, "*An Interview with Edsger W. Dijkstra*", Communications of the ACM, vol. 53, no. 8, August 2010, pp.41-47, doi: 10.1145/1787234.1787249.

[5] Programmer All, "*Dijkstra Algorithm*", www.programmerall.com/article/224531575/ (Accessed November 12, 2019).

[6] J. Bang-Jensen and G. Gutin, "*Section 2.3.4: The Bellman-Ford-Moore algorithm*", in Digraphs: Theory, Algorithms and Applications (First edition), London, UK, Springer-Verlag, 2000, ISBN 978-1-84800-997-4.

[7] W. Zhang, H. Chen, C. Jiang, L. Zhu, "*Improvement and Experimental Evaluation Bellman-Ford Algorithm*", International Conference on Advanced Information and Communication Technology for Education (ICAICTE 2013), Hainan, China, September 20-22, 2013.

[8] W. Zeng and R. L. Church, "*Finding Shortest Paths on Real Road Networks: The Case for A**", International Journal of Geographical Information Science, vol. 23, no. 4, June 2009, pp.531-543, doi: 10.1080/13658810801949850.

[9] R. C. Prim, "*Shortest Connection Networks and Some Generalizations*", Bell System Technical Journal, vol. 36, no. 6, November 1957, pp.1389-1401, doi: 10.1002/j.1538-7305.1957.tb01515.x.

[10] R. Sedgewick and K. D. Wayne, "*Algorithms (4th edition)*", Boston, Massachusetts, USA, Addison-Wesley, 2011, p. 628, ISBN 978-0-321-57351-3.

[11] K. Rosen, (2011), "*Discrete Mathematics and Its Applications (7th edition)*", McGraw-Hill Science, New York, USA, 2011, p. 798, ISBN-13: 978-0073383095.

[12] D. Cheriton and R. E. Tarjan, "*Finding Minimum Spanning Trees*", SIAM Journal on Computing, vol. 5, no. 4, 1976, pp.724-742, doi: 10.1137/0205051.

[13] S. Pettie and V. Ramachandran, "*An Optimal Minimum Spanning Tree Algorithm*", Journal of the ACM, vol. 49, no. 1, January 2002, pp.16-34, doi: 10.1145/505241.505243.

[14] Khan Academy, "*The Breadth-First Search Algorithm*", www.khanacademy.org/computing/computer-science/algorithms/breadth-first-search/a/the-breadth-first-search-algorithm (Accessed November 12, 2019).

[15] J. A. Sethian, "*Fast Marching Methods: A Boundary Value Formulation*", www.math.berkeley.edu/~sethian/2006/Explanations/fast_marching_explain.html (Accessed November 12, 2019).

[16] A. V. Goldberg, "*Point-to-Point Shortest Path Algorithms with Preprocessing*", in SOFSEM 2007: *Theory and Practice of Computer Science*, J. van Leeuwen, G. F. Italiano, W. van der Hoek, C. Meinel, H. Sack, F. Plášil, (Eds.), Lecture Notes in Computer Science, vol. 4362, Berlin, Heidelberg, Germany, Springer, 2007, doi: 10.1007/978-3-540-69507-3_6.

[17] Wikipedia, "*Artificial Intelligence/Search/Heuristic search/Bidirectional Search*", www.en.wikibooks.org/wiki/Artificial_Intelligence/Search/Heuristic_search/Bidirectional_Search (Accessed November 12, 2019).

[18] H. Kaindl and G. Kainz, "*Bidirectional Heuristic Search Reconsidered*", Journal of Artificial Intelligence Research, vol. 7, December 1997, pp. 283-317, doi: 10.1613/jair.460.