An Experiment on the Performance of Shortest Path Algorithm

Simon Chan Yew Meng, Nur'ayuni Adnan, Syazwan Syafiqah Sukri, and Wan Mohd Nazmee Wan Zainon

Universiti Sains Malaysia, Malaysia
{simon.ucom13; ayuni.ucom13; ssyafiqah.ucom13}@student.usm.my, nazmee@usm.my

ABSTRACT

Shortest path algorithms are one of the main algorithms used in most navigation system. By implementing these algorithm, the related overall costs such as time and work load can be minimized. The main objective of this paper is to study and experiment the different shortest path algorithm such as Dijkstra's algorithm, Symmetrical Dijkstra's algorithm, A* algorithm, Bellman-Ford algorithm, Floyd-Warshall algorithm and Genetic algorithm in solving the shortest path problem. In this paper, a brief review on each of the shortest path algorithm and its implementation method was discussed. Explanation on how the experiment was conducted and the sample data that involved in the experiment were also presented. The result of the experiment shows the overall performance of each algorithm in solving shortest path problem in term of running time and total distances. The analysis of result shows the performance of each algorithm in order to suggest the most efficient algorithm in solving the shortest path problem.

Keywords: Dijkstra's algorithm, Symmetrical Dijkstra's algorithm, Bellman-Ford algorithm, A* algorithm, Floyd-Warshall algorithm, Genetic algorithm

I INTRODUCTION

Shortest path problem is one of an interesting topic and widely researched until these days. The shortest path problem involve in finding shortest route from a starting point to a destination point (Magzhan & Jani, 2013). This problem is widely applied for GPS routing system, network routing system and logistic automation (Beker et.al., 2012). The aim of solving shortest path problem is to improve the productivity as well as save cost and time. Currently there are many shortest path algorithms that has been proposed by the researcher to solve the shortest path problem. Each of these proposed algorithms has its own method to solve the problem and each algorithm has its advantages and disadvantages over each other depends on the situation it is used. Thus, it is important to study about the characteristic of these algorithms and able to choose the right algorithm that suits each situation especially when users want to implemented it to solve the shortest path problem since using unsuitable algorithm could lead to time wasting and inaccurate result. In this paper, there are several shortest path algorithm that will be discussed;

1) Dijkstra's Algorithm
2) Symmetrical Dijkstra's Algorithm
3) A* Algorithm
4) Bellman-Ford Algorithm
5) Floyd-Warshall Algorithm
6) Genetic Algorithm

The goal of this research is compare these six algorithms in term of their performance, accuracy and understand their characteristic. In the process, it will try to determine the most efficient algorithm to solve the shortest path problem. To achieve this, we have conducted an experiment to test the performance of the algorithm in different situation.

II LITERATURE REVIEW

As mention earlier, every shortest path algorithm has its own unique characteristic and method in solving the shortest path problem. In this section, the brief description and implementations of six proposed shortest path algorithm will be presented.

1) Dijkstra's Algorithm: Dijkstra's algorithm is a shortest path algorithm discovered by E.W. Dijkstra (Morris, 2016) (Zhang et. al., 2005), used to solve the single-source shortest-path problem when all edges have non-negative weights. In a graph, the algorithm starts at the starting node and grows a tree that ultimately spans all nodes reachable from the starting node. The algorithm will works iteratively where in each iterative it visits the node with shortest distance path from the starting node and then revalue the path distance of remaining unvisited node. This process will keep repeated until the destination node was visited (Zhang et. al., 2005). In overall, the Dijkstra's algorithm has running complexity of $O(n^2)$. One of the advantage of Dijkstra's algorithm is the algorithm will be terminated once the destination node has reached and without need to visit the remaining unvisited node. In other hand, the disadvantage of Dijkstra's algorithm is difficult to be implemented on computer program when the number of node is very large because it will consume a lot of CPU memory in order to compute it (Aghaei et. al., 2009).

2) Symmetrical Dijkstra's Algorithm: Symmetrical Dijkstra's algorithm was invented by Pohl where the algorithm was derived from the Dijkstra algorithm (Zhang et. al., 2005) by implementing the bi-directional search method into it. The process of Symmetrical Dijkstra's algorithm was similar to the original one with addition of a forward search from the origin node to the destination node and a backward search from the destination node to the origin node. The process of algorithm will terminated when forward search and backward search meet at certain node. According to Pohl, this algorithm was invented in attempt to reduce the running complexity of Dijkstra's algorithm from $O(N^b)$ to $O(N^{b/2})$. But in worst case scenario, the running complexity of the algorithm could become two $O(N^b)$ searches.

3) A Algorithm*: A* algorithm was invented by Hart and Nilsson (Mitchell, 1999) where the algorithm implement the concept of integrating a heuristic into the search procedure. The A* algorithm was working as similar as the Dijkstra's algorithm except for its difference heuristic controls in choosing the node for every iteration. Rather than choosing the node with shortest distance path from starting node, the A* algorithm will choose the node based on its distance path from starting node with addition heuristic estimation of its proximity to the destination node (Beker et.al., 2012) (Cho et. al., 2013). The heuristic estimation was evaluated by one of two main evaluation functions, which were the Euclidean distance and the Manhattan distance (Zhang et. al., 2005). The Euclidean distance is calculated by the length of straight line between the evaluated node and the destination node, while the Manhattan distance evaluated by the sum of distance in the X and Y coordinates of both nodes. Through the usage of these heuristic, the A* algorithm will cause the shortest path tree expanded toward to the destination node instead of expand the tree radially using the Dijkstra's algorithm. As results, A* algorithm has reduce the search space require to reach the destination node compare to Dijkstra's algorithm. This shows that A* algorithm will have better performance compare to Dijkstra's algorithm unless its heuristic was less accurate.

4) Bellman-Ford Algorithm: Bellman-Ford algorithm was developed by Richard E. Bellman and Lester R. Ford, Jr (Stoimen, 2016). It is suitable to be implemented to solve the shortest path problem when the graph contains negative value edges (Beker et.al., 2012) (Schrijver, 2012) (Glabowski et. al., 2013). This algorithm works iteratively where its number of iteration was based on the number of edges path from starting node to destination node. For each iteration, every of the last visited node will transverse to its nearby node and label it with the most optimal distance path from the starting node. The running complexity of Bellman-Ford algorithm is $O(NA)$ where (N + 1) is the number of iterations and A is the number of edges in the graph.

5) Floyd-Warshall Algorithm: Floyd-Warshall algorithm was discovered by Bernard Roy and Stephen Warshall (Weisstein, 2016). It works by finding the shortest distance path between all of pairs of nodes in a graph (Beker et.al., 2012). The running complexity of Floyd-Warshall algorithm is $O(N^3)$. Besides that, Floyd-Warshall algorithm was also explained as one of the few algorithms that able to solve the shortest path problem in a graph that contains negative values edges and without the existed of negative edges cycle. The main advantage of Floyd-Warshall algorithm is able to obtain the shortest distance between any two nodes (Cho, 2013). In other hand, this algorithm is simple and easy to implement into the program but it was not suitable for solving shortest path problem in large network because its running complexity is too high for the calculation.

6) Genetic Algorithm: Genetic algorithm was invented by John Holland in the 1960s and then developed by him and his students and colleagues at the University of Michigan in the 1960s and the 1970s (Mitchell, 2016). This intelligent algorithm was invented to solve the shortest path problem in a flexible situation that has a very large search space and constant changing environment (Magzhan & Jani, 2013). In addition, it also defines as a stochastic search algorithm that based on the biological evolution and used to produce a most optimizes results. The genetic algorithm works by produce a set of solution which is known as the population where each of it was evaluated by its own fitness value. Then, the population will goes through several genetic operations such as selection, crossover and mutation in order to generate a new generation of population that supposed to have better fitness value compare to the previous one. After going through specific number of generations, the population with the most optimal fitness value will be chosen as the solution of the problem.

III EXPERIMENT IMPLEMENTATION

In this research, the experiment has been conducted using a special application developed in Java. The sample data that used to test the shortest path algorithm is the existing bus route of Penang area. This experiment stimulates the navigation system to

find the shortest path from the origin to the destination using the proposed algorithm. Through this experiment, the proposed algorithm will be tested in different situation such as using large sample data versus small sample data, traveling long journey versus short journey and implementation of genetic algorithm with different number of generation. Figure 1 and 2 show the large sample data and small sample data respectively.

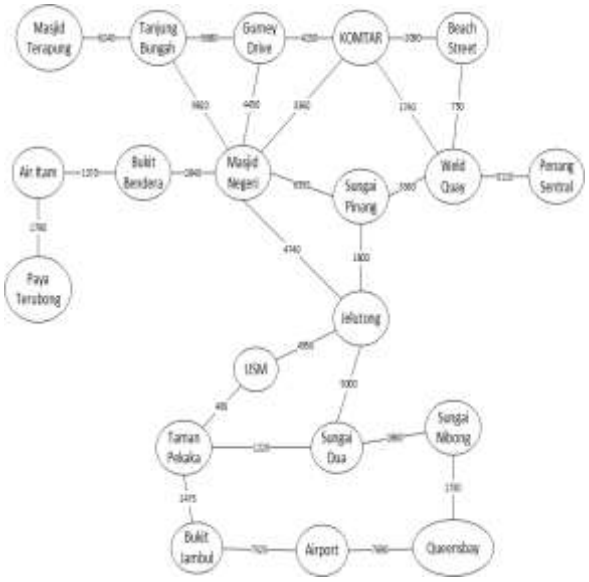

Figure 1. Large Sample Data

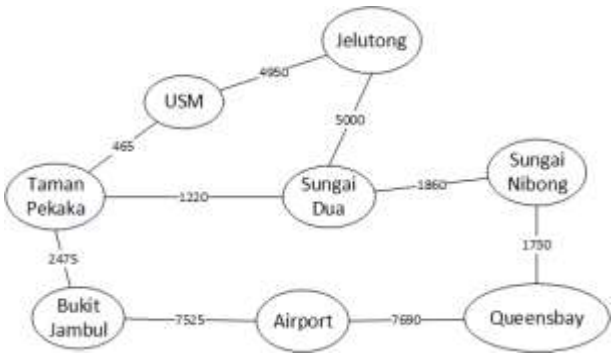

Figure 2. Small Sample Data

IV EXPERIMENT RESULT AND ANALYSIS

In order to achieve accurate results, each algorithms performed 20 times for each of the cases of the experiment. Then, the average value was calculated as the final result.

For large data versus small data cases, each algorithm was tested with same origin and destination (Jelutong to Airport) but using the different data set (large sample data and small sample data). The Table 1 shows the performance of each algorithm to solve shortest path problem in term of running time and result distance. The result was divided into two groups, which are large sample data and small sample data.

Table 1. Comparison Of Performance Of Algorithm To Solve Shortest Path Problem For Large Data Versus Small Data Cases

Algorithm	Small Sample Data		Large Sample Data	
	Running Time (nanosecond)	Total Distance (meter)	Running Time (nanosecond)	Total Distance (meter)
Dijkstra	55658	18240	321712	18240
Symmetrical Dijkstra	27632	18240	144080	18240
A*	43816	18240	322501	18240
Bellman-Ford	42237	18240	88816	18240
Floyd-Warshall	61974	18240	584213	18240
Genetic	863688	18240	865661	18240

The Figure 3 shows the chart to compare the running time of each algorithm to solve shortest path problem for large data and small data cases.

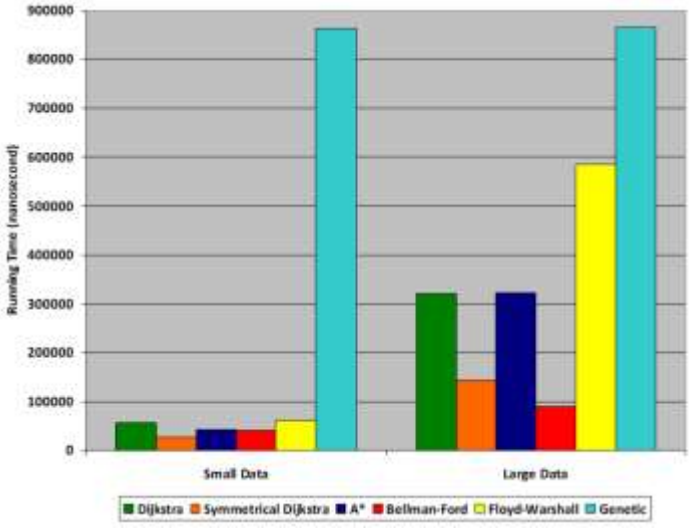

Figure 3. Chart Of Running Time Of Algorithm To Solve Shortest Path Problem For Large Data Versus Small Data Cases

The chart shows that the algorithm solve the shortest path problem using small sample data has much shorter running time compare to using large sample data. This is can be explain by the smaller the data size leads to the lesser the operations of algorithm require to be executing for solving the problem. The genetic algorithm is the exception because its number of operation was based on its generation number rather than the data size. This explains why the running time of genetic algorithm only has slightly different between using large sample data and small sample data.

In small data cases, the symmetrical Dijkstra's algorithm has the highest performance compare to others. This was follow-up by the Bellman-Ford algorithm, A* algorithm and Dijkstra's algorithm. In large data cases, the Bellman-Ford algorithm has the shortest running time which was follow-up by the symmetrical Dijkstra's algorithm, Dijkstra's algorithm and A* algorihm. In both cases, the genetic algorithm and Floyd-Warshall algorithm has the worst and second worst performance respectively. The performance of genetic algorithm can be explained by its complex operation while the Floyd-Warshall algorithm cases is due to its time complexity of $O(n^3)$. In term of accuracy, the result shows that all algorithms were able to produce the similar and most optimum solution for both small and large data cases.

For long journey versus short journey cases, the experiment was carried out for each algorithm was tested to start at the same origin location and travel to two different destinations, where one is the short journey (Airport to Jelutong) while another one is the long journey (Airport to *Masjid Terapung*). The Table 2 shows the performance of each algorithm to solve shortest path problem in term of running time and result distance. The result was divided into two groups, which are short journey cases and long journey cases.

Table 2: Comparison Of Performance Of Algorithm To Solve Shortest Path Problem For Long Journey Versus Short Journey Cases

Algorithm	Short Journey		Long Journey	
	Running Time (nanosecond)	Total Distance (meter)	Running Time (nanosecond)	Total Distance (meter)
Dijkstra	48553	18240	323291	35469
Symmetrical Dijkstra	37500	18240	219474	35469
A*	63947	18240	230527	22449
Bellman-Ford	100264	18240	89606	22449
Floyd-Warshall	572371	18240	581450	22449
Genetic	797766	18240	1100531	28999

The Figure 4 and 5 shows the chart to compare each algorithm to solve shortest path problem in term of running time for long journey and short journey cases and result distance for long journey cases respectively.

The chart shows that each algorithm except Floyd-Warshall algorithm and Bellman-Ford algorithm, solves the short journey cases of shortest path problem has better performance compare to long journey. The different in performance of algorithm was due to different in path distance between short journey and long journey which resulting the algorithm in short journey cases require to traverse less node in order to reach the destination compare to long journey. The Floyd-Warshall algorithm and Bellman-Ford has the similar running time for both cases due to its requirement to traverse all nodes before able to product the solution.

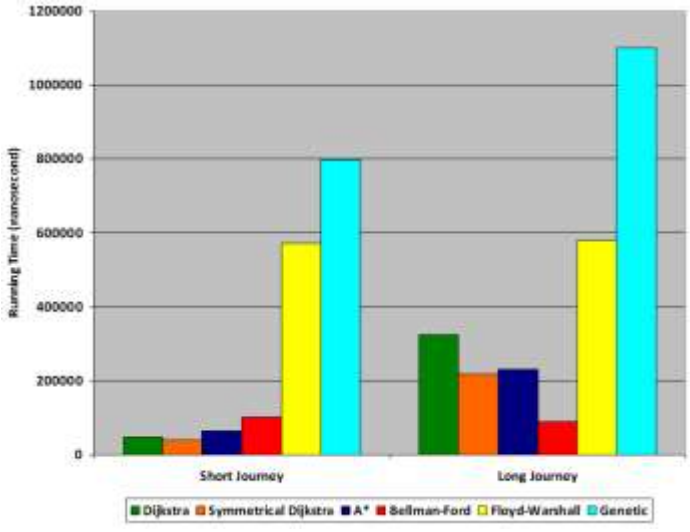

Figure 4. Chart of Running Time of Algorithm to Solve Shortest Path Problem for Long Journey Versus Short Journey Cases

In short journey cases, the symmetrical Dijkstra's algorithm was shows to have the shortest running

time while the second runner up was Dijkstra's algorithm. The rest is follow by the A* algorithm and Bellman-Ford algorithm. In long journey cases, the chart shows that Bellman-Ford algorithm has the best performance compare to others. The result was follow-up by symmetrical Dijkstra's algorithm, A* algorithm and Dijkstra's algorithm. In both cases, the genetic algorithm has the worst performance follow-up by the Floyd-Warshall algorithm. Similar to large data versus small data case, the performance of genetic algorithm and Floyd-Warshall algorithm was due to the complex operation and time complexity of $O(n^3)$ respectively.

Based on Table 2, the result shows that the solutions of all algorithms are similar and optimum for short journey cases. For long journey cases, the chart shows that A* algorithm, Bellman-Ford algorithm and Floyd-Warshall algorithm have produce the most optimum solution which is follow-up by the genetic algorithm. For Dijkstra's algorithm and symmetrical Dijkstra's algorithm, both have produced the least optimum solution. The reason Bellman-Ford algorithm and Floyd-Warshall algorithm able produce better solution was because both algorithms able to generate all the possible solution before making comparison to get the best solution. In other hand, the A* algorithm cases can be explained by its implementation of heuristic search.

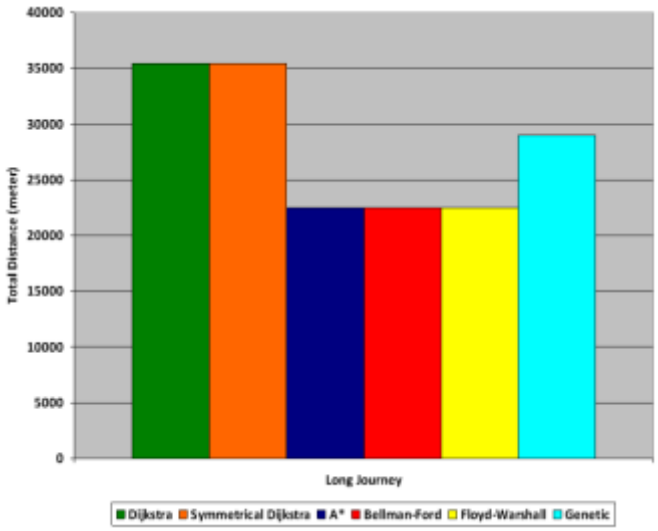

Figure 5. Chart of Result Distance of Algorithm to Solve Shortest Path Problem for Long Journey Cases

The next experiment was to test the performance of genetic algorithm to solve shortest path problem of same cases (Airport to KOMTAR) with implementation of different number of generation. The Table 3 shows the performance of genetic algorithm with implementation of different number of generation to solve the shortest path algorithm.

Table 3: Comparison Of Performance Of Genetic Algorithm With Implementation Of Different Number Of Generation To Solve Shortest Path Problem

Number of Generation	Running Time (nanosecond)	Total Distance (meter)
5	37106	32359
10	62369	28999
15	63158	22449
20	70263	28999
25	82500	22449
30	94342	19609

The Figure 6 shows the graph to compare the performance of genetic algorithm with implementation of different number of generation to solve the shortest path algorithm.

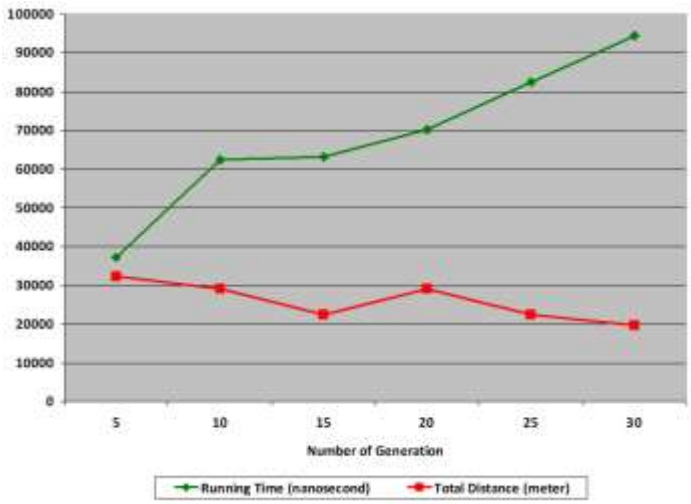

Figure 6. Graph Of Performance Of Genetic Algorithm With Implementation Of Different Number Of Generation To Solve Shortest Path Algorithm

The graph shows that the running time of genetic algorithm increase with its number of generation. This is due to increase in number of operation required to perform by genetic algorithm in order to solve the shortest path problem. In other hand, the graph also shows that the solution provided by the genetic algorithm is inconsistent for each number of generations because the solution produce by the genetic algorithm can be random sometime. As the number of generation increase further, the solution produce by the genetic algorithm was shown to improved and more optimum.

V CONCLUSION AND FUTURE WORK

Based on the experiment results, it shows that most algorithms will achieve better performance when it solves the short journey shortest path problem and using small data size. In exception, the data size will not affect the performance of genetic algorithm. In other hand, the Bellman-Ford algorithm and Floyd-Warshall algorithm will still retain its performance for both short journey and long journey cases. This show that the performance of algorithm can be different in various situations depends on the nature of data and method of algorithm to solve the shortest path problem. In overall, the experiment result shows that the Bellman-Ford algorithm was able to produce the optimum solution using short running time. The result also shows that the performance of Bellman-Ford algorithm was superior to other algorithm in most situations. This clearly indicates that the Bellman-Ford algorithm is the most efficient shortest path algorithm compare to others. In other hand, the genetic algorithm was shown to have highest running time but able to produce the optimum solution in most situation. The experiment shows that performance of genetic algorithm was affected by its number of generation where the larger the number of generation, the higher the running times as well as the better the solution. Thus, it is important to adjust the number of generation until the optimum running time to solution ratio was achieved so that the genetic algorithm can be used in the most efficient manner.

REFERENCES

Magzhan, K., Jani, H.M. (2013) "A Review and Evaluations of Shortest Path Algorithm", International Journal of Scientific & Technology Research Volume 2, Issue 6.

Beker, I., Jevtic, V., Dobrilovic, D. (2012) "Shortest-path algorithm as a tools for inner transportation optimization", International Journal of Industrial Engineering and Management (IJIEM), Vol.3 No 1, pp. 39-45.

Schrijver, A., (2012) "On The History Of The Shortest Path Problem", Documenta Mathematica, extra volume ISMP, pp. 155-167.

Glabowski, M., Musznicki, B., Nowak, P., and Zwierzykowski, P., (2013) "Efficiency Evaluation of Shortest Path Algorithms", AICT 2013: The Ninth Advanced International Conference on Telecommunications.

Cho, T., Kim, L., Yoon, W., and Choi, S., (2013) "A Hybrid Routing Algorithm for an Efficient Shortest Path Decision in Network Routing", International Journal of Multimedia and Ubiquitous Engineering, Vol. 8, No. 4.

Zhang, F., Qiu, A., and Li, Q., (2005) "Improve on Dijkstra Shortest Path Algorithm for Huge Data". Chinese academy of surveying and mapping

Aghaei, M.R.S., Zukarnain, Z.A., Mamat, A., Zainuddin, H., (2009) "A Hybrid Algorithm for Finding Shortest Path in Network Routing", Journal of Theoretical and Applied Information Technology.

Morris, J., (2016, February 15) 10.2 Dijkstra's Algorithm. Available from: https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html.

Engineer F. (2001) Fast Shortest Path Algorithms for Large Road Networks. In: Proceedings of 36th annual ORSNZ conference.

Mitchell, M. (1999) An Introduction to Genetic Algorithms. Cambridge, Massachusetts: Massachusetts Institute of Technology.

Stoimen. (2016, February 10) Bellman-Ford Shortest Path in a Graph. Available from: http://www.stoimen.com/blog/2012/10/22/computer-algorithms-bellman-ford-shortest-path-in-a-graph/

Weisstein, E.,. (2016, January 22) Floyd-Warshall Algorithm. Available from: http://mathworld.wolfram.com/Floyd-WarshallAlgorithm.html