Comparative Study On Bellman-Ford And Dijkstra Algorithms

Conference Paper · February 2020

CITATIONS
12

READS
5,299

2 authors:

Fadhil Mukhlif
Universiti Teknologi Malaysia
39 PUBLICATIONS **169** CITATIONS
SEE PROFILE

Abdu Saif
University of Malaya
62 PUBLICATIONS **923** CITATIONS
SEE PROFILE

Comparative Study On Bellman-Ford And Dijkstra Algorithms

Fadhil Mukhlif[1,*], Abdu Saif[1]

[1]Department of Electrical Engineering, Faculty of Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia

**Corresponding Author: fadhil.researcher@gmail.com*

Abstract: This paper introduces a brief introduction on both algorithms; bellman-ford and dijkstra. A comparison of these two algorithms is performed as well based on their efficiency on attenuation vs. distance and throughput vs. traffic load.

Keywords: Dijkstra, Bellman-Ford, Comparative, Development, Algorithms, Machine Learning, Deep Learning

1 Bellman-Ford Algorithm

The Bellman-ford algorithm solves the single source shortest path problem in which edge weights may be negative. The Bellman-Ford algorithm for shortest paths is almost completely intuitive and it returns a Boolean value indicating whether or not there is a negative weight cycle that is reachable from the source. So when there is a cycle the algorithm indicates that no solution exists but when there is not, the algorithm produces the shortest paths and their weight. Furthermore, the basis for it is as: Given a graph with n vertices the most arcs there can be in a shortest path is n-1. This is easily proven by induction. The shortest path have more than n-1 then it must have a negative circuit, similarly a graph with a negative circuit will have a shortest path of n or more arcs. All of these observations lead to the Bellman-Ford algorithm

2 Bellman-Ford Algorithm Conventions

The conventions here are nearly the same as for Dijkstra's algorithm explain as below. Among the difference is that the Bellman-Ford algorithm can handle arcs of negative length. In this project, the input is the same for Dijkstra's algorithm and bellman with 9 nodes that will used for sending data from source to destination. However, the algorithm is more efficient if it is written using a list of the arcs [1].

3 Pseudo Code Of Bellman-Ford Algorithm

Short pseudo code for the bellman-ford algorithm is summaries as given below:

1. Initialize all the vertices and set their distance from source to a high value such as infinity and set the value of distance for source to 0 values.

2.Repeat the following for number of (vertices – 1) times such as: for each edge (u,v) that belongs to the graph drawn if distance[v] > distance[u] +weight(u,v) then the distance[v] = distance[u] +weight(u,v), which means distance is updated

3. Repeat the following steps for each edge (u,v) that belongs to graph drawn if distance[v] > distance[u] +weight(u,v) then there is No negative - weight cycle exists between source and destination.

4. Return the distance values for all the nodes after all the iterations are completed.

4 Dijkstra's Algorithm

Dijkstra's algorithm is called the single-source shortest path. Almost certainly Dijkstra's algorithm is the most used algorithm for finding shortest paths. It is also known as the single source shortest path problem. It is the algorithm generally used for finding routes through the Internet. It computes length of the shortest path from the source to each of the remaining vertices in the graph. An explanation to the single source shortest path problem can be described as given: Let G= {V, E} be a directed weighted graph with V having the set of vertices. From the graph, the special vertex s in V where s is the source and let for any edge e in E, Edge-Cost (e) be the length of edge e in graph. All the weights in the graph should be non-negative or contain non-negative values. Before going in depth about Dijkstra's algorithm let's give some details about directed-weighted graph. The directed graph can be defined as an ordered pair G: = (V,E) with V is a set that its elements are called vertices or nodes and E is a set of ordered pairs of vertices called directed edges or arc. However, the directed graphs are also known as digraph as shown in the figure below [2].

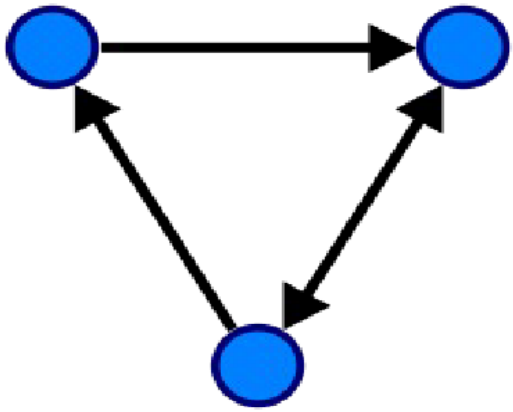

Figure 1: Direct Graph

The figure above is shown the direct flow in graphical model format. Furthermore, the directed-weighted graph is a directed graph with weight attached to each of the edge of the graph as in the following figure.

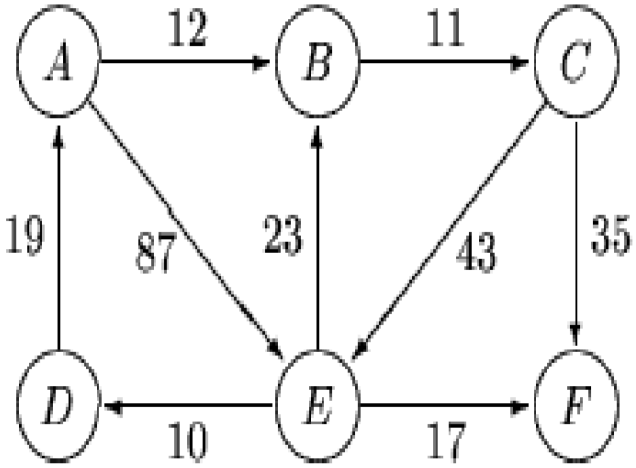

Figure 2: Direct-Weighted Graph

5 Description of the Algorithm

Before going into details of the pseudo-code of the algorithm it is important to know how the algorithm works or functions. The Dijkstra's algorithm works by solving the sub-problem k which computes the shortest path from the source to vertices among the k closest vertices to the source, from source to destination. For the Dijkstra's algorithm to work it should be directed weighted graph and the edges should be non-negative as stated previously. If the edges are negative then the actual shortest path cannot be obtained. However, at the k'th round there will be a set called Frontier of k vertices that will consist of the vertices closest to the source and the vertices that lie outside frontier are computed and put into New Frontier. As a result, the shortest distance obtained is maintained in sDist[w]. Furthermore, it holds the estimate of the distance from s to w [3]. The Dijkstra's algorithm finds the next closest vertex by maintaining the New Frontier vertices in a priority-min queue or nearest distance/ neighbor. The algorithm works by keeping the shortest distance of vertex v from the source in an array as sDist and the shortest distance of the source to itself is zero. The sDist for all other vertices is set to big value or to infinity to indicate that those vertices are not yet processed. Subsequently the algorithm finishes the processing of the vertices sDist will have the shortest distance of vertex w to s. in this algorithm two sets are maintained Frontier and New Frontier which helps in the processing of the algorithm. The frontier has k vertices which are closest to the source and already compute shortest distances to these vertices, for paths restricted up to k vertices [3]. Finally the vertex that resides outside of Frontier is put in a set called New Frontier.

6 Dijkstra's algorithm Conventions

Dijkstra's algorithm finds all shortest paths from a source vertex S to all other vertices in the graph. In dijkstra algorithm there are no negative arcs allowed to be. The graph is represented by the array M where M (i, j) is the direct distance from node i to node j. furthermore, the set of vertices of the graph is denoted by V symbol. Each vertex v in the graph will have a state value denoted State (v) value [4]. There are two states T for temporary state and P for permanent state. Similarly associated with each vertex v are two other functions given as: Dist (v) which stands for the distance to v from S and Prior (v) is the vertex visited just prior to v in the shortest path from S vertex. The idea of the algorithm is reasonably simple and easy to implement. The algorithm proceeds recursively through stages or through number of stages. At each stage a new vertex is marked as permanent P in its distance from S. At that moment the distances to the vertices marked temporary T are updated through the latest permanent vertex. So, if it is possible to shorten the current distance to a vertex through the most recently permanent vertex then its distance are updated. When it occurs, a new stage begins and the temporary vertex with least distance from S is marked permanent. Finally once all vertices are marked as permanent then the algorithm is terminated [5].

7 Efficiency

The complexity or the efficiency of the dijkstra algorithm can be expressed in terms of Big-O Notation or symbol. The Big-O notation gives another way of talking about the way input affects the algorithm's running time/ processing time. In fact it gives an upper bound of the running time.

However, in Dijkstra's algorithm the efficiency varies depending on |V|=n DeleteMins and |E| updates for priority queues that were used in designing the algorithm.

If during the design, Fibonacci heap used then the complexity is given as O (| E | + | V | log | V |) which is the best bound. However, the DeleteMins operation takes O (log|V|). At the same time, if a standard binary heap is used then the complexity is given in the form of O (| E |log |E|) and the | E | log |E| term comes from |E| updates for the standard heap.

Furthermore, if the set used is a priority queue then the complexity is given in the form of O (|E|+|V|2). The O (V|2) term comes from |V| scans of the unordered set New Frontier to find the vertex with the least sDist value [6].

8 Dis-advantages

The major disadvantage of the algorithm is the fact that it does a blind search there by consuming a lot of time waste of necessary resources or in other words it is time consuming. Another disadvantage is that it cannot handle negative edges where this leads to acyclic graphs and most often cannot obtain the right shortest path [6].

9 Applications

• Traffic information systems use Dijkstra's algorithm in order to track the source and destinations from a given particular source and destination.

• Open Shortest Path First (OSPF) used in Internet routing. Actually it uses a link-state in the individual areas that make up the hierarchy. However, the computation is based on Dijkstra's algorithm which is used to calculate the shortest path tree inside each area of the network [7].

10 Comparison

In order to find the shortest path from one node to all other nodes then Dijkstra's algorithm is usually more efficient that the Bellman-Ford algorithm. In the other side, Bellman-Ford algorithm is simpler than Dijkstra's algorithm. Also the Bellman-Ford algorithm can handle negative arcs whereas Dijkstra's algorithm can't handle it. It has a better running time than that of Dijkstra's algorithm. If on the other hand we want the shortest paths from all nodes we can run Dijkstra's algorithm on each node or we can do the same thing with the Bellman- Ford algorithm or we can use Floyd's algorithm [8].

11 Result and Discussion

The result obtained for this project is based on the specifications given: such as creating 9 nodes from source to destination in order to send data by using the two algorithms of bellman-ford and dijkstra. After designing the algorithms, part of the result is presented and discussed here.

A. Dijkstra's Algorithm

*1) **Short distance vs bath***

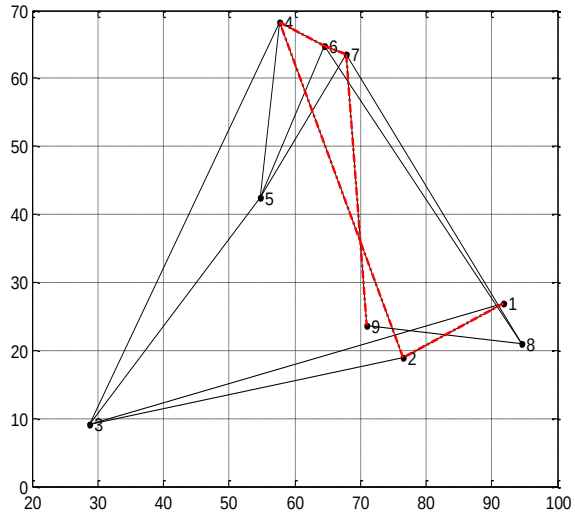

Figure 3: Dijkstra with 9 nodes.

The graph above is a representation of network with 9 nodes generated randomly. By using the dijkstra algorithm, short distance is calculated between two nodes and the shortest distance is return as index value. In this figure, we tried to send data from node 1 to node 9. In other words, short distance is calculated in the network and highlighted in red color. The shortest distance was obtained as d = 154.4305 and the bath of follow was taken as p =

[1 2 4 5 7 9].

*2) **Attenuation vs. distance***

Attenuation is a term used to describe the reduction of the adsl signal strength that occurs on the copper pair over distance and is measured in dB decibels.

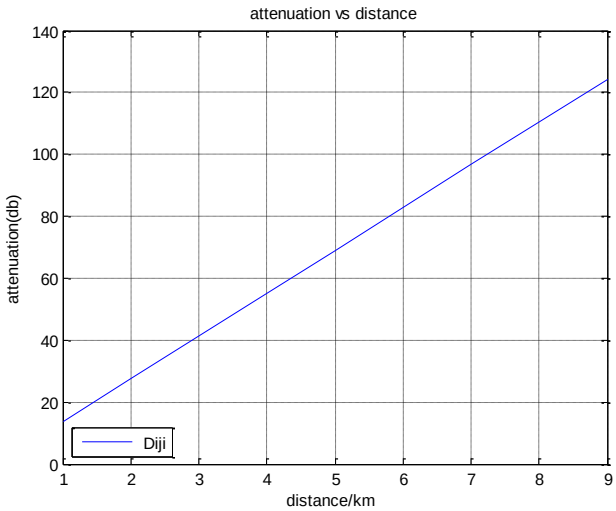

Figure 4: Dijkstra attenuation vs. distance

From the graph it can be seen that, the higher the distance is, the higher the attenuation [8].

3) ***Throughput vs. traffic load***

Throughput: This is the average number of packets delivered per unit time. Throughput of received bits is measured in kilobits per second.

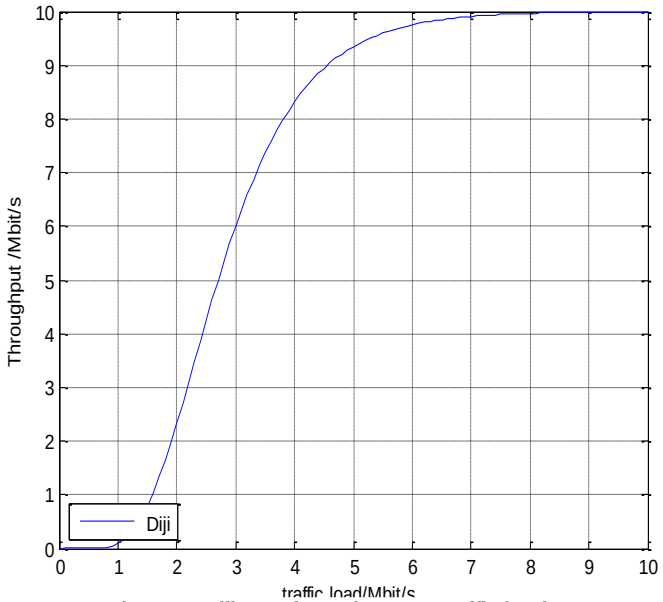

Figure 5: Dijkstra throughput vs. traffic load

From the graph above, we can clearly notice that the throughput is proportional to the traffic load. The throughput is increased with respect to the load up to certain value then it remains constant [8].

B. Bellman-ford algorithm

1) ***Short distance***

The algorithm of bellman is design based on the specifications given in the assignment and with the same number of nodes with dijkstra algorithm, 9 nodes are generated randomly. The distance between the nearest nodes is calculated and it was given as: d = 190.6490

p = [1 3 5 7 9]

Comparing to the dijikstra algorithm, the distance is higher.

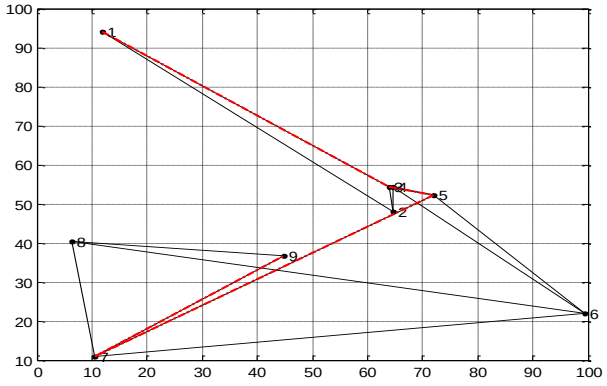

Figure 6: Bellman-ford with 9 nodes

2) ***Attenuation vs. distance***

As explained in the previous method, same goes for this method, except that the distance at dijkstra is better compare to bellman-ford.

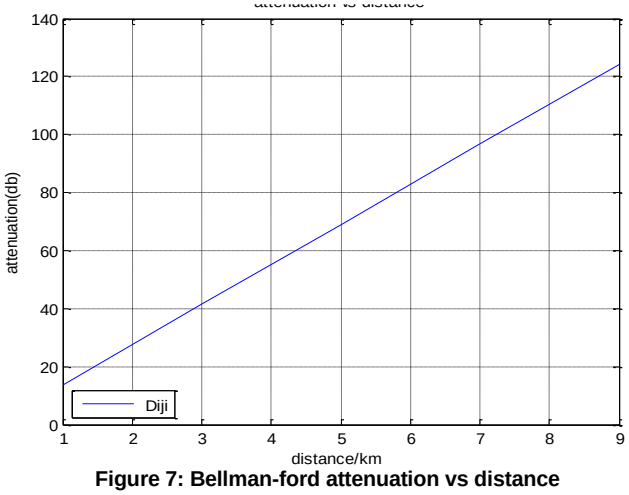

Figure 7: Bellman-ford attenuation vs distance

3) ***Throughput vs. traffic load***

Throughput is the average number of packets delivered to the destination per unit time.

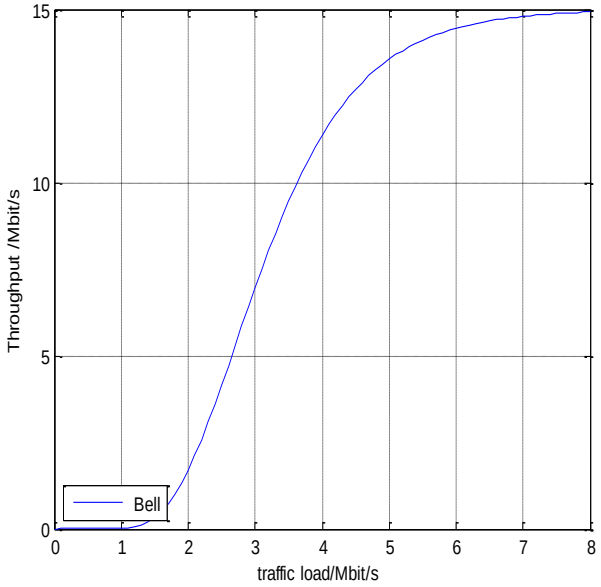

Figure 8: Bellman-throughput vs. traffic load.

From the graph obtained above, we can notice that the throughput is increasing with distance; however in bellman-ford algorithm the traffic load is bigger than that in dijikstra [9].

12 Conclusion

As a conclusion for this project, and from the graphs obtained for throughput vs. traffic load and attenuation vs. distance for both algorithms we see that the average packet delay increase for increase in number of nodes in the interface queue while routing protocols try to find valid route to the destination. Moreover, the actual delivery of data packets, the delay time is also affected by route discovery, which is the first step to begin a communication session. The source routing protocols have a longer delay because their route discovery takes more time as every intermediate node tries to extract information before forwarding the reply at the same time same thing happens

when a data packet is forwarded hop by hop. Hence, while source routing makes route discovery more profitable, it slows down the transmission of packets. Finally the result shows that dijikstra is better than bellman-ford in term of short distance and less traffic loads.

References

[1] B. Latré, P. De Mil, I. Moerman, N. Van Dierdonck, B. Dhoedt and P. Demeester, “Maximum Throughput and Minimum Delay in IEEE 802.15.4”, Proceedings of the International Conference on Mobile Ad-hoc and Sensor Network *(MSN’05)*, LNCS 3794, Wuhan, China, December2005.

[2] Soo Young Shin, Sunghyun Choi, Hong Seong Park, Wook Hyun Kwon, “Packet Error Rate Analysis of IEEE 802.15.4 under IEEE 802.11b Interference”, Wired/Wireless Internet Communications 2005, LCNS 3510,2005

[3] Y. Xiao and J. Rosdahl, “Throughput and delay limits of 802.11”, *IEEE* Communications Letter, Vol. 6, No. 8, August 2002.

[4] Jangeun Jun, Pushkin Peddabachagari and M.Sichitiu, "Theoretical Maximum Throughput of IEEE 802.11 and its Applications*," 2nd IEEE International Symposium on Network Computing and Applications (NCA'03),* Cambridge, 2003.

[5] B.V. Cherkassky, A.V. Goldberg, and T. Radizik. Shortest paths algorithms: theory and experimental evaluation. Mathematical Programming, Series A, (73):129-174, 1996.

[6] H. De Neve and P. Van Mieghem. A multiple quality of service routing algorithm for PNNI. Proc. of IEEE ATM workshop, Fairfax, USA, 1998.

[7] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra’s Algorithm. ACM Journal of Experimental Algorithmics 15 (2010).

[8] B.V. Cherkassky, L. Georgiadis, A.V. Goldberg, R.E. Tarjan, and Renato F. Werneck. Shortest Path Feasibility Algorithms: an Experimental Evaluation. In Proc. of 6th International Workshop on Algorithm Engineering and Experiments, SIAM, 2008.

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Introduction to Algorithms,

McGraw-Hill, 2001.