西 南 交 通 大 学 学 报

第 55 卷 第 4 期
2020 年 8 月

JOURNAL OF SOUTHWEST JIAOTONG UNIVERSITY

Vol. 55 No. 4
Aug. 2020

ISSN: 0258-2724 **DOI：10.35741/issn.0258-2724.55.4.3**

Research article

Computer and Information Science

COMPARATIVE EVALUATION OF PATH FINDING IN ALL PAIR SHORTEST PATH PROBLEM FOR AIRLINE NETWORK

航空公司网络中所有最短路径问题中路径查找的比较评估

Nadia Moqbel Hassan Al-Zubaydi
Computer Engineering Department, College of Engineering, University of Mustansiriya
P.O. Box: 14022, Palestine St., Baghdad, Iraq, nadiahasan@uomustansiriyah.edu.iq

Received: April 8, 2020 ▪ Review: June 19, 2020 ▪ Accepted: July 9, 2020

Abstract

In the field of geoformation and transportation science, the shortest path is calculated on graph mostly found in road and transportation networks, especially the network of airports of some capitals around the globe. Against this backdrop, the present paper follows Geographic Information Systems to investigate the real distance for the airline in order to achieve an optimal aircraft routing within a dynamically stochastic transportation network. Routing is a procedure of forwarding the data from a known seed to the goal in which the data may travel through several intermediate routes. Consequently, the selection of the best possible shortest path nodes to forward the data using all pair shortest path is necessitated. Moreover, as a result of this approach, this shortest path selection of nodes will help facilitate the attainment of an efficient performance in the network. Copious amounts of work have been carried out to discover the shortest path in the network routing for the purpose of improving its efficiency and overcoming possible problems.

Keywords: Graph Algorithm, Programming Algorithm, Transportation Routing, Shortest Path, Geographic Information System

摘要 在地理构造和运输科学领域，最短路径是根据主要在公路和交通网络（尤其是全球一些首都的机场网络）中找到的图形计算得出的。在这**种**背景下，本文遵循地理信息系统研究航空公司的实际距离，以便在动态随机运输网络内实现最佳的飞机路线。路由是将数据从已知**种**子转发到目标的过程，在该目标中数据可能会通过多个中间路线传播。因此，必须选择最佳可能的最短路径节点以使用所有对最短路径转发数据。而且，作为这**种**方法的结果，节点的最短路径选择将有助于促进网络中有效性能的获得。为了提高网络效率并克服可能出现的问题，已经进行了大量工作以发现网络路由中的最短路径。

关键词: 图算法，编程算法，运输路线，最短路径，地理信息系统

I. INTRODUCTION

One of the essential algorithmic problems is the shortest path issue, wherein a shortest weight path is figured between two nodes of a weighted, coordinated chart. Having been explored for quite some time, this issue has remained the focal point of numerous specialists from different fields, for example, computer science software engineering, communication networks, geographic information systems (GIS), robotics and wire routing, and so on [1], [2], [3]. SPP is the place where the voyager may need to re process the briefest way while traversing in a dynamic transportation environment [4].

Several parameters define the problem that causes a taxonomy of shortest-path problems:

1. Objective function: The Euclidean measurement, a Lp metric, the quantity of connections, a blend of criteria, etc. might be utilized to gauge the length of the way.

2. Limitations along the way: The direction may need to go from s to t while visiting explicit regions en route.

3. Information geometry: The guide of the geometric area requires avoiding a wide range of obstacles while also imposing constraints on the path.

4. Type of moving item: The item to be moved along the way might be a solitary purpose of some predetermined geometry.

5. The dimension of the issue: The issue frequently lies in two or three measurements; be that as it may, higher measurements emerge in certain applications.

6. Single shot versus frequent mode inquiries must be considered.

7. Static versus dynamic environments: Occasionally, hindrances might be embedded or erased, or they might move in time.

8. Perfect versus approximate algorithms must be considered.

9. Known versus obscure guide: In the online variant, robot detect and determine the shape of the environment as it moves further. Only when the weight is negative can automated flight be used to reach the shortest path. The map may also feature a degree of uncertainty, requiring path planning with stochastic models [1].

The SPP illustrates chart G (V, E), where V is (n) vertices and E is the arrangement of (m) edges. Two vertices are associated with each edge, which is related to either a positive or a negative weight. The coordinated or undirected chart could conceivably contain cycles. A large portion of the answers for the SPP accept that the diagram contains no negative cycle. Different genuine models can be portrayed as diagrams, such as in transport systems. This is especially true of airport networks in capital cities, which represent edges and intersections and depict vertices as shown in Figure 1. Their weights may be distant or require travelling along a road, which takes far more time.

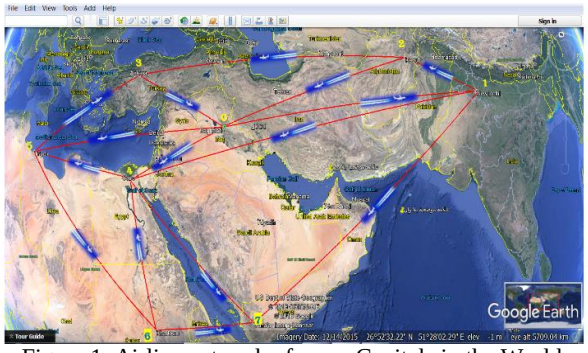


Figure 1. Airline network of some Capitals in the World

There are generally two main forms of SPP

1. (SSSP): An SP from a source node is connected to every single other node in the chart.

2. All pair shortest path (APSP): An SP is registered for each pair of nodes in the chart [1], [5].

This paper is concerned with APSP, as most genuine models need to determine the briefest path for each pair of vertices. In Section I, a writing audit investigates various methods of approaching SPPs. In Section II, both the execution investigation and the relative investigation of the shortest path calculations for each pair are described [6]. The test arrangement, results and algorithms comparison are depicted in Sections 4, 5 and 6, respectively. Section 7 presents the outcomes and conclusions. Section 8 represent the limitation and recommendation for future research.

II. LITERATURE SURVEY

The algorithms are part of a general indoor navigation model that is used to inform new indoor environments [7]. Researchers examining the shortest path problem in open transportation frameworks [8], [9] found that the concept of exchange required additional expenses from one edge to a nearby edge. Along these lines, direct contiguous weighted edges are coordinated within the chart as an open transportation information model, improving the capacity of an

adjacency matrix [7]. Several approaches have been used to address the shortest path problem, and many researchers have formulated sufficient algorithms to solve the issue. Some fundamental algorithms for assessing the shortest path are clarified below.

The Bellman–Ford algorithm (BFA) gives an answer for the single source shortest path issue [10]. It is utilized for weighted directed diagrams with the probability of having edges with negative weights and a time complexity of O (nm) (Big-O notation). It can also be used t explore negative weight cycles in the chart. Dijkstra's algorithm (DA) addresses the single source shortest path issue, working for diagrams with non-negative edge weights [11]. It has a running time of O (n2). In the event that the minimum need line used in Dijkstra's calculation is actualized with a double store, the running time accomplished is O (mlog (n)) [12]. By utilizing Fibonacci heaps, a far better execution than Dijkstra and Floyd Warshall's algorithms can be obtained, as in O (nlog(n) + m) [12]. The Floyd–Warshall calculation discovers answers for the all-pair shortest path issue [13]. The information diagram may incorporate contrarily weighted edges; however, it should not contain any negative weight cycles. It has a running time of Θ (n3).

Analysts apply every one of their endeavors to improve these calculations and create different calculations by consolidating these algorithms. For example, Gallo and Pallottino [14] present a well-executed investigation of classical calculations and their applications. Cherkassky et al. [15] propose a broad relative investigation on assorted SPAs. In Pettie's work [16], another all-pair shortest path calculation is presented for genuine weighted directed charts that run in O (mn + n2 log n) time. Hougardy [17] explains the mechanism of the Floyd–Warshall calculations, including diagrams with negative weight cycles. In [18], Dijkstra's calculations are explained in terms of finding the shortest paths in digrams with non-negative basic edge lengths that consider the very-large-scale integration (VLSI) directing issue.

[19] sum up Dijkstra's calculations in order to manage the shortest path issue with ambiguous parameters. Orlin [20] proposes a powerful technique in which he executes the single source shortest path calculations such that they run in straight time when the quantity of particular edge lengths is less than the thickness of the chart. Han [21] introduces a calculation to address the all-pair shortest path issue that features a time complexity of O (n3 (loglogn/logn) 5/4) [22].

III. ALL-PAIRS SHORTEST PATH ALGORITHMS (APSPA)

APSPAs are weighted, coordinated, and directed in chart G (V, E), where V is (n) vertices and E is (m) edges. They appear in an adjacency matrix, the components of which are w [p, q] (the weight of the edge) and [p, q] (p, q = 1, 2, ..., n). These loads might be negative, but the diagram must not have any negative cycles. To save the weights for SP, a distance matrix including components dim[p, q] (the heaviness of SP from p to q, where p, q = 1, 2, … , n) is utilized. An antecedent matrix is utilized for developing the SP. Its components are r[p, q], the ancestor of q on the SP from p, where p, q = 1, 2, ..., n. The SP between each pair of vertices of the chart must be found. The calculations utilized for comparison are clarified and explained below:

A. Floyd Warshall Algorithm

The Floyd Warshall algorithm (FWA) is a basic calculation for finding SP for each pair of nodes in a directed chart. It is a dynamic programming (DP) algorithm, created by Floyd [1] and dependent on a paper given by Warshall [23]. This calculation is firmly derived from middle vertices. In the event that $dist_{pq}0$ is the weight matrix, $dist_{pq}m$ is the SP from p to q with its middle vertices in the set {1, 2... m}, m > 1 as per the following:

$$dist_{pq}^{m}=\min(dist_{pm}^{m-1}, dist_{pm}^{m-1} + dist_{mq}^{m-1}) \quad (1)$$

Consequently, $dist_{pqn}$ will create the SP matrix for the input data for the graph. The calculation is expressed beneath. Subtleties of this calculation can be found in [24], [25].

The first step: set Dq and Rq as two squares n × n frameworks, where q is the stage number and n is the complete number of nodes of the system or network.

The second step: q = zero is calculated with D_0 and R_0:

$D_0 = [dist_{pm}]$, where

$$dist_{pm}=\begin{cases} dist_{pm} & \text{in the event that there is an immediate course associating node p to the node m} \\ \infty & \text{on the off chance that there is no immediate course associating the node p to the node m} \\ 0 & \text{if p = m.} \end{cases}$$

$R_0 = [r_{pm}]$, where

$$\begin{cases} m & \text{on the off chance that there is an immediate course interfacing node p to the node m} \end{cases}$$

r_{pm} = − in the event that there is no immediate course interfacing the node p to the node m
− if p = m.

The third step: The rest of the q = 1, . . . ,n compute the Dq and the Rq matrices contemplating that they derive the objects of the Dq and the Rq matrices dependent on the objects of the latest past matrices, for example the Dq−1 and the Rq−1 matrices - as follows:Dq = [$dist_{pm}$], where

D_q = [$dist_{pm}$], where

$$dist_{pm} = \begin{cases} Dist_{pm} & \text{if } p = m,\ p = q,\ m = q \\ \min(dist_{pm}, dist_{pq} + dist_{qm}) & \text{else.} \end{cases}$$

R_q = [r_{pm}], where

$$\text{if } \begin{cases} m & \text{if } p = m,\ p = q,\ m = q \\ & dist_{pm} \le dist_{pq} + dist_{qm} \\ q & \text{if } dist_{pm} > dist_{pq} + dist_{qm}. \end{cases} r_{pm} = m$$

The fourth step: Reiterate step 3 until the Dn and the Rn are created. The implemented algorithm utilizes one-distance and forerunner matrices rather than various matrices for each estimation of m. The explanation behind this is in the event that it is registered and stores $dist_{pm}$ or $dist_{mq}$ before utilizing these values to figure $dist_{pq}$, it may process one of the following:

$$dist_{pq}^{m} = \min(dist_{pq}^{m-1}, dist_{pm}^{m-1} + dist_{mq}^{m-1}) \quad (2)$$

$$dist_{pq}^{m} = \min(dist_{pq}^{m-1}, dist_{pm}^{m} + dist_{mq}^{m-1}) \quad (3)$$

$$dist_{pq}^{m} = \min(dist_{pq}^{m-1}, dist_{pm}^{m-1} + dist_{mq}^{m}) \quad (4)$$

In any of these cases, the weight of the SP from p to q with all intermediate nodes in {1, 2...m} is registered. If $dist_{pm}^{m}$ is utilized instead of $dist_{pm}^{m-1}$, in the calculation, at that point a sub way will be utilized from p to m with all intermediate nodes in {1, 2 . . . m}. Yet, m cannot be an intermediate node on the SP from p to m, since in any case there would be a cycle on this SP.

Thus, $dist_{pm}^{m} = dist_{pm}^{m-1}$.

A comparative variable is presented to display that $dist_{mq}^{k} = dist_{mq}^{m-1}$ without utilizing various distance matrices for various values of m. The time complexity of the algorithm is controlled by the triple settled for loops. It obviously demonstrates that the running time of this calculation is Θ(n3). The distance matrix involving dist[p, q] values gives the distance of SP from p to q, and, consequently, the path of the SP can be easily built from the predecessor matrix including r[p,q] values.

IV. Shortest Path Problem (SPP)

In this section, the (APSP) problem will be elaborated by providing an example for the Airline Network of some capitals in the world using dynamic programming (DP). The network N in Figure 2 will derive the minimum roads from each pair of nodes of this system. Firstly, it will provide the model utilizing the FWA, and afterward will apply the all-pair shortest path problem to find the solution [26], [27].

The FWA will assign some initial distance values and will try to improve them step by step.

1. V = 8 (No. of vertices)

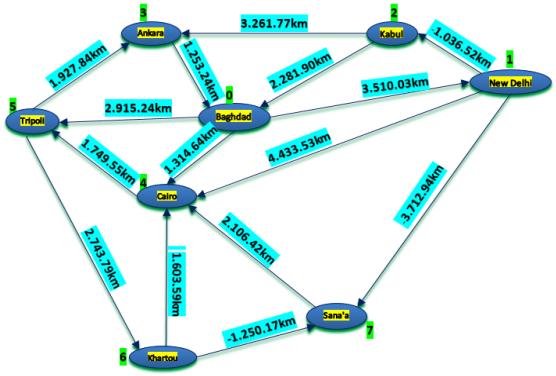


Figure 2. Directed graph with real distances for Capitals network

2. dist (v*v) = 8*8 adjacency matrix, as shown in Table 1.

Table 1.
Adjacency matrix representation of the diagram

W(p,q)	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								
6								
7								

3. if i = j then dist [v][v] = 0, as shown in Table 2.

Table 2.
Adjacency matrix representation when p = q

W(p,q)	0	1	2	3	4	5	6	7
0	0							
1		0						
2			0					
3				0				
4					0			
5						0		
6							0	
7								0

4. edge [u][v] = w(u,v)
for m from 1 to v
for p from 1 to v
for q from 1 to v, as shown in Table 3.

Table 3.
Adjacency matrix representation with weighted (+, -)

W(p,q)	0	1	2	3	4	5	6	7
0	0	3.150	∞	∞	1.314	2.915	∞	∞
1	∞	0	-1.036	∞	4.433	∞	∞	-3.712
2	2.281	∞	0	3.261	∞	∞	∞	∞
3	1.253	∞	∞	0	∞	∞	∞	∞
4	∞	∞	∞	∞	0	1.749	∞	∞
5	∞	∞	∞	1.927	∞	0	2.743	∞
6	∞	∞	∞	∞	1.603	∞	0	-1.250
7	∞	∞	∞	∞	2.106	∞	∞	0

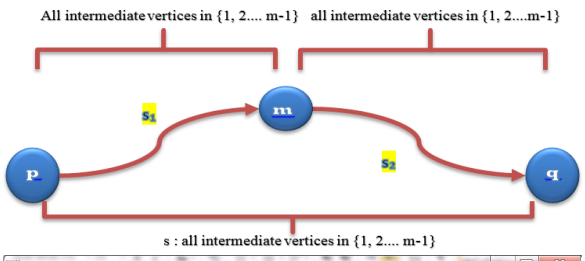
5. if dist[p][q] > dist[p][m] + dist[m][q]
dist[p][q] = dist[p][m] + dist[m][q]
After the analysis, adjacency matrix using APSP, as shown in Table 4.

Table 4.
Adjacency matrix representation the resulting implementation eligible to imitate the algorithm on a Capitals graph in the World using DP

W(p,q)	0	1	2	3	4	5	6	7
0	0	3.150	2.114	4.842	1.314	2.915	5.658	-0.562
1	1.245	0	-1.036	2.070	-1.606	0.143	2.886	-3.712
2	2.281	5.431	0	3.261	3.595	5.196	7.939	1.719
3	1.253	4.403	3.367	0	2.567	4.168	6.911	0.691
4	4.929	8.079	7.043	3.676	0	1.749	4.492	3.242
5	3.180	6.330	5.294	1.927	3.599	0	2.743	1.493
6	5.785	8.935	7.899	4.532	0.856	2.605	0	-1.250
7	7.035	10.185	9.149	5.782	2.106	3.855	6.598	0

V. Implementation and Experimental Results

The trials have been applied on a PC with Intel Core i3 CPU M 480 at 2.67 GHz x 4 processor with 3.7 GB memory, and the programs were written in C++ and assembled by a Visual C++ compiler. The charts for the experiments are weighted, directed, and have been created subjectively via the Airline network of certain capitals in the world. The goal is to achieve and implement practical. The model creates random charts of n nodes where every probable "edge" exists with potential s. The number of "edges" in the created diagram is an unsystematic argument with expected value sn (n–1)/2. Lower estimations of s demonstrate inadequate diagrams and higher values including thick charts. Along these lines, the arguments for the algorithms are n and s, as appeared in Figure 3.



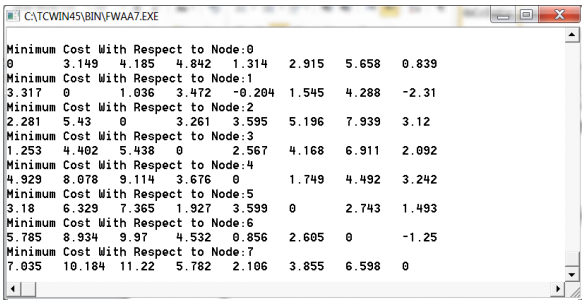
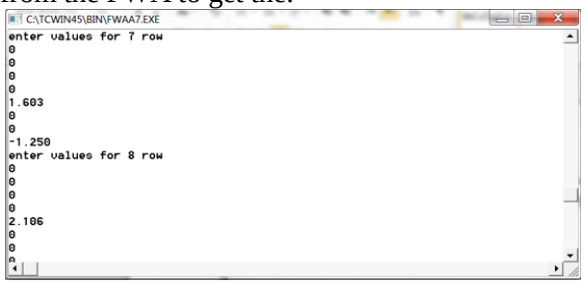
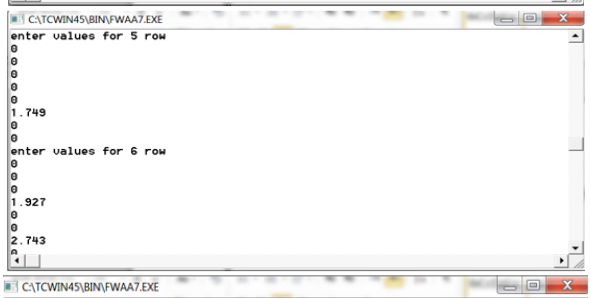
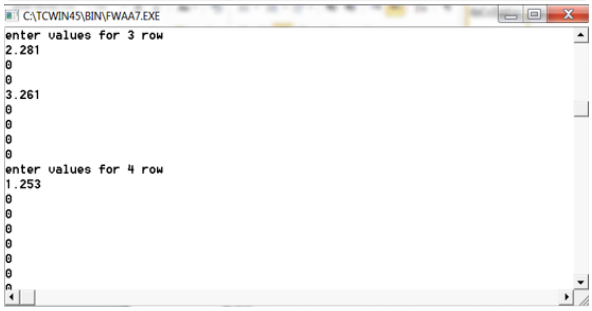


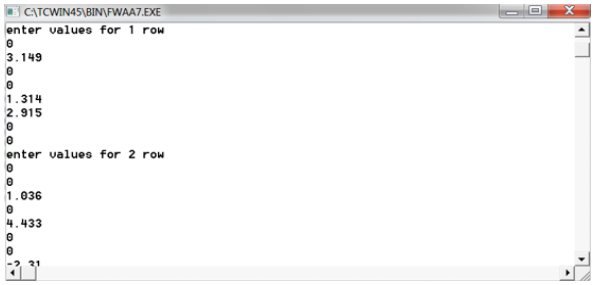
Figure 3. Path s is a SP from node p to node q, and m is the most noteworthy numbered middle of the road vertex of s. road s1, the part of road s from node p to node m, has all moderate nodes in the set {1, 2, … . m-1}. Similar holds for road s2 from node m to node q

Equations 2-4 accomplished n 3 times. Hence, the algorithm runs in $\Theta(n^3) = \Theta(V^3)$ time, noting that while the written algorithm requires $\Theta(V^3)$ space, only logical implementations require $\Theta(V^2)$ space: only four matrices are needed to be memorized at a time, namely $D^{(m)}$, $D^{(m/1)}$, $pq^{(m)}$ and $pq^{(m-1)}$. Define $dist^{m}_{pq}$ recursively by stage 3 from the FWA to get the:









Down to the best solution for shortest path (Figures 4 and 5).

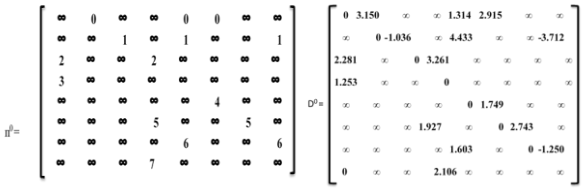


Figure 4. The resulting implementation eligible to imitate the algorithm on a Capitals graph in the World using DP in (Turbo) C++ program

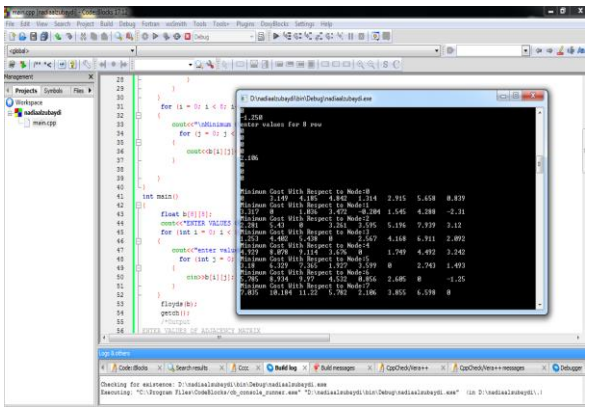


Figure 5. The resulting implementation eligible to imitate the algorithm on a Capitals graph in the World using DP in (CodeBlocks) C++ program

VI. Comparison of Algorithms

Compared to the best-performing algorithms (FWA and BFA), the reliable implementation of the DA is measured as follows:

1. Execution time
2. Space complexity

The time taken by the program to find the shortest path is called execution time. Execution time, which is measured in the range of seconds, provides a clear picture of the best algorithm. To identify the optimal path, the DA, FWA and BFA are carried out on a 4 GHz Core i3 PC machine with 3.7 GB memory, running Microsoft Windows 7. For comparing the performance, the execution time is observed by modifying the program of the three algorithms so that the execution time is also shown on the output screen. The results of the comparison are plotted on the graph, as shown in Figure 6.

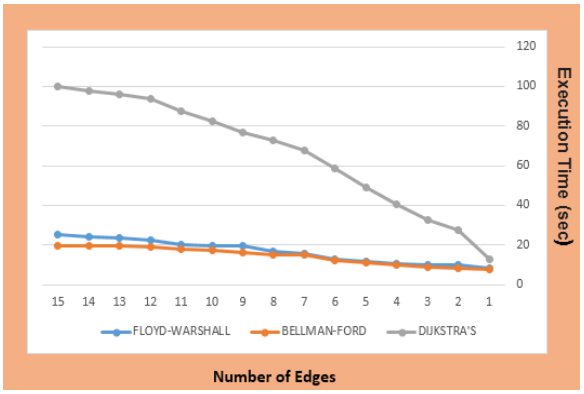


Figure 6. A comparison based on execution time of the FWA, BFA and DA

From Figure 6, it is observed that BFA has the lowest execution time when compared to DA and FWA for the same network. From the given results, it is apparent that the BFA performs more efficiently than the FWA. The proportion of the DA and FWA to BFA is calculated (Figures 6 and 7). It is clear that the BFA surpasses the DA by about three to four times, so the BFA is an efficient algorithm. The space complexity is described in the later part of paper.

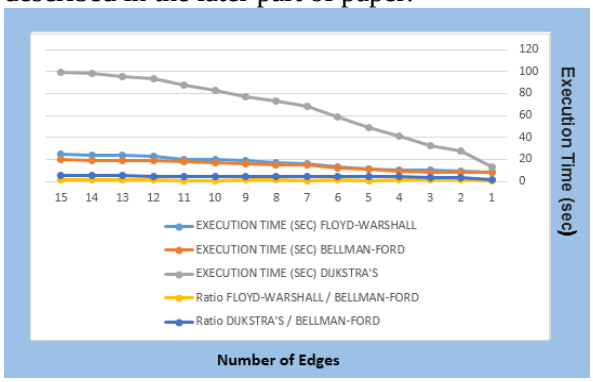


Figure 7. The ratio of Floyd Warshall and Dijkstra to Bellman Ford

VII. Conclusion

These observations infer that each of these three algorithms has specific applications; however, they can be appropriate for other approaches as well, as the problem of parameters can be adapted to the procedures of each algorithm (like using DA to find solutions to All-Pairs Shortest Paths, in which parameters have to be modified to make a call of DA for each vertex instead of only once). That is, the dijkstra algorithm does not handle negative weights. Although DA is the fastest, it isn't able to deal with negative weight. For dealing with such problems, Bellman-Ford's and Floyd-Warshall's are likely to be the best choices; furthermore, only Bellman-Ford's is appropriate when negative cycles occur. The programming contestant needs to devote part of his/her time on better understanding the language resources.

These language resources will be C++ STL, which will lead to a much better implementation of these algorithms, such as preserving development time and granting algorithm sufficiency, as well as avoiding minor obstacles during the contest. Dominating such notion (concepts of the Graph, Dijkstra, Bellman Ford, Floed Warshall Algorithms.), it is required to move forward and understand other graph obstacles, learning all ways and techniques of dealing with techniques.

VIII. LIMITATION AND RECOMMENDATION

The Detect the negative cycles Bellman-Ford and Floyd-Warshall algorithms in the graph but cannot solve it, and to solve the negative cycles one of the solutions is to use undirected graph with above algorithm.

ACKNOWLEDGMENTS

The authors would like to thank Mustansiriyah University (www.uomustansiriyah.edu.iq) Baghdad – Iraq for its support in the present work.

REFERENCES

[1] RAVINDRA, K., KURT, M., JAMES, B., and ROBERT, E. (1990) Faster algorithms for the shortest path problem. *Journal of the ACM*, 37, pp. 213-223.

[2] BANNISTER, M.J. and EPPSTEIN, D. (2012) Randomized Speedup of the Bellman-Ford Algorithm. In: HWANG, H.-K. and MARTÍNEZ, C. (eds.) *Proceedings of the 9th Workshop on Analytic Algorithmics and Combinatorics.* Philadelphia, Pennsylvania: Society for Industrial and Applied Mathematics, pp. 41-47.

[3] FREDMAN, M. (1976) New bounds on the complexity of the shortest path problem. *SIAM Journal on Computing*, 5 (1), pp. 83-89.

[4] ALHOULA, W. (2019) *Shortest Path Algorithms for Dynamic Transportation Networks.*

[5] WILSON, R.J. (1998) *Introduction to Graph Theory*. 4th ed. Addison Wesley.

[6] MALIK, D.S. (2010*) Data Structures Using C++*. 2nd ed. Boston, Massachusetts: Course Technology, Cengage Learning.

[7] MEGHANATHAN, D.N. (n.d.) *Review of Graph Theory Algorithms.* Jackson, Mississippi: Department of Computer Science, Jackson State University.

[8] CAO, L., ZHAO, X., ZHENG, H., and ZHAO, B.Y. (2011) Atlas: Approximating Shortest Paths in Social Graphs. Oakland, California: Department of Computer Science, University of California.

[9] GOYAL, S. (2010) *A Survey on Travelling Salesman Problem.* Grand Forks, North Dakota: Department of Computer Science, University of North Dakota.

[10] WADHWA, S. (2000) *Analysis of a Network Problem.* Bethlehem, Pennsylvania: Lehigh University.

[11] ZHAN, F.B. (2010) Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures. *Journal of Geographic Information and Decision Analysis*, 1 (1), pp. 69-82.

[12] BORGWARDT, K.M. and KRIEGEL, H.P. (2005) *Shortest Path Kernels on Graphs.* Munich: Institute for Computer Science, Ludwig Maximilian University of Munich.

[13] SADAVARE, A.B. and KULKARNI, R.V. (2012) A Review of Application of Graph Theory for Network. *International Journal of Computer Science and Information Technologies*, 3 (6), pp. 5296-5300.

[14] SHIVENDU, A., DHAKAL, D., and SHARMA, D. (2015) Emulation of Shortest Path Algorithm in Software Defined Networking Environment. *International Journal of Computer Applications*, 116 (1), pp. 47-49.

[15] FAKCHAROENPHOL, J. and RAO, S. (2006) Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72 (5), pp. 868-889.

[16] HASAN, N.M. (2015) *Implementation of Dijkstra's shortest path algorithm.*

[17] CORMEN, T.H., LEISERSON, C.E., RIVEST, R.L., and STEIN, C. (2009) *Introduction to Algorithms.* 3rd ed. Cambridge, Massachusetts: The MIT Press.

[18] HUANG, X. (2006) Negative-Weight Cycle Algorithms. In: *Proceedings of the International Conference on Foundations of*

Computer Science, Las Vegas, Nevada, June 2006. Available from https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.1981&rep=rep1&type=pdf.
[19] RUOHONEN, K. (2013) *Graph Theory*. [Online] Available from: http://math.tut.fi/~ruohonen/GT_English.pdf [Accessed 13/01/20].
[20] ACAR, U. (2013) *Parallel and Sequential Data Structures and Algorithms: Shortest Weighted Paths.*
[21] CABELLO, S., CHAMBERS, E.W., and ERICKSON, J. (2013) Multiple-Source Shortest Paths in Embedded Graphs. *SIAM Journal on Computing,* 42, pp. 1542-1571.
[22] HAJELA, G. and PANDEY, M. (2014) A Fine Tuned Hybrid Implementation for Solving Shortest Path Problems Using Bellman Ford. *International Journal of Computer Applications*, 99 (2), pp. 29-33.
[23] SHIVENDU, A., DHAKAL, D., and SHARMA, D. (2015) Emulation of Shortest Path Algorithm in Software Defined Networking Environment. *International Journal of Computer Applications,* 116 (1), pp. 47-49.
[24] ZHAN, F.B. and NOON, C.E. (1998) Shortest Path Algorithms: An Evaluation Using Real Road Networks. *Transportation Science,* 32, pp. 65-73.
[25] HOUGARDY, S. (2010) The Floyd-Warshall algorithm on graphs with negative cycles. *Information Processing Letters,* 110, pp. 279-281.
[26] HASSAN, N.M. (2015) *Design and Implementation of Shortest Path Algorithm for Network of Roads.*
[27] HASSAN, N.M. (2017) Analysis and Implementation of the Minimum Route Issues between Some Governorates of Iraq Using Bellman-Ford Algorithm. In: *Proceedings of the Annual Conference on New Trends in Information & Communications Technology Applications, Baghdad, March 2017*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, pp. 230-235.

参考文:

[1] RAVINDRA， K.， KURT， M.，JAMES，B. 和 ROBERT，E.（1990）最短路径问题的更快算法。ACM 杂志，37，第 213-223 页。
[2] BANNISTER，M.J. 和 EPPSTEIN，D.（2012）贝尔曼福特算法的随机加速。于：黄，H.-K.以及第 9 届分析算法与组合学研讨会的论文集。宾夕法尼亚州费城：工业和应用数学协会，第 41-47 页。
[3] FREDMAN，M.（1976）最短路径问题的复杂性的新界限。暹计算杂志，5（1），第 83-89 页。
[4] ALHOULA，W.（2019）动态交通网络的最短路径算法。
[5] WILSON，R.J。（1998）图形概论理论。第四版。艾迪生·韦斯利。
[6] MALIK，D.S.（2010）使用 C ++的数据结构。第二版。马萨诸塞州波士顿：课程技术，参与学习。
[7] 麦加纳顿（D.N. N.）图论算法综述。密西西比州杰克逊：杰克逊州立大学计算机科学系。
[8] 曹琳，赵新，郑和，赵炳炎。（2011）地图集：社交图谱中最短路径的逼近。加利福尼亚州奥克兰：加利福尼亚大学计算机科学系。
[9] GOYAL，S.（2010）关于旅行推销员问题的调查。北达科他州大福克斯：北达科他大学计算机科学系。
[10] WADHWA，S.（2000）网络问题分析。宾夕法尼亚伯利恒：里海大学。
[11] 詹，F.B。（2010）真实道路网络上的三种最快最短路径算法：数据结构和过程。地理信息与决策分析杂志，1（1），第 69-82 页。
[12] BORGWARDT，K.M。和 KRIEGEL，H.P.（2005）图上的最短路径核。慕尼黑：慕尼黑路德维希马克西米利安大学计算机科学学院。
[13] SADAVARE， A.B. 和库尔卡尼（R.V.）（2012）图论在网络中的应用综述。国际计算机科学与信息技术杂志，3（6），第 5296-5300 页。
[14] SHIVENDU，A.，DHAKAL，D. 和 SHARMA，D.（2015）软件定义网络环境

中的最短路径算法仿真。国际计算机应用杂志，116（1），第 47-49 页。
[15] FAKCHAROENPHOL，J. 和 RAO，S.（2006）平面图，负权重边缘，最短路径和接近线性时间。计算机与系统科学学报，72（5），第 868-889 页。
[16] HASAN，N.M。（2015）迪克斯特拉最短路径算法的实现。
[17] T.H. CORMEN，C.E。LEISERSON，R.L。RIVEST 和 C. STEIN.（2009）算法导论。第三版。马萨诸塞州剑桥：麻省理工学院出版社。
[18] HUANG X.（2006）负加权循环算法。于：2006 年 6 月在内华达州拉斯维加斯举行的国际计算机科学基础会议上的会议录。可从 https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.1981&rep=rep1&type=pdf 获取。
[19] RUOHONEN，K.（2013）图论。[在线]可从以下网站获得：http://math.tut.fi/~ruohonen/GT_English.pdf [访问时间：13/01/20]。
[20] ACAR，U。（2013）并行和顺序数据结构和算法：最短加权路径。
[21] CABELLO，S.，CHAMBERS，E.W. 和 ERICKSON，J.（2013）嵌入式图中的多源最短路径。暹计算杂志，42，第 1542-1571 页。
[22] HAJELA，G. 和 PANDEY，M.（2014）使用贝尔曼· 福特解决最短路径问题的微调混合实现。国际计算机应用杂志，99（2），第 29-33 页。
[23] SHIVENDU，A.，DHAKAL，D. 和 SHARMA，D.（2015）软件定义网络环境中的最短路径算法仿真。国际计算机应用杂志，116（1），第 47-49 页。
[24] 詹，F.B。和 NOON，C.E.（1998）最短路径算法：使用真实之路网络的评估。交通科学，32，第 65-73 页。
[25] HOUGARDY，S.（2010）带负周期图的弗洛伊德· 沃歇尔算法。信息处理快报，110，第 279-281 页。
[26] HASSAN，N.M。（2015）道路网最短路径算法的设计和实现。
[27] HASSAN，N.M。（2017）使用贝尔曼福特算法分析和实施伊拉克一些省份之间的最小路线问题。于：2017 年 3 月，巴格达，信息与通信技术应用新趋势年度会议论文集。新泽西州皮斯卡塔维：电气与电子工程师学会，第 230-235 页。