ISSN 2277-3061

Comparison Studies for Different Shortest path Algorithms

Hanaa M. Abu-Ryash, Dr.Abdelfatah A. Tamimi
Department of Computer Science
Faculty of Science and Information technology, Al-Zaytoonah University of Jordan,
hanaa-abu-ryash@hotmail.com
drtamimi@zuj.edu.jo

ABSTRACT

While technological revolution has active role to the increase of computer information, growing computational capabilities of devices, and raise the level of knowledge abilities, and skills. Increase developments in science and technology. In graph used the shortest path algorithms for solving the shortest path problem. The shortest path can be single pair shortest path problem or all pairs shortest path problem. This paper discuss briefly the shortest path algorithms such as Dijkstra's algorithm, Bellman-Ford algorithm,Floyd- Warshall algorithm, and johnson's algorithm. It describes the previous algorithms for solving the shortest path problem. The goal of this paper is to investigate and comparison the impacts of different shortest path algorithms. The study shows that the efficiency varies among algorithms, helps to suggest which one of them ought to be used to solve a specific variant of the shortest path problem.

Keywords

Shortest path, Dijkstra's, Bellman-ford, johnson's, Floyd-Warshall, Routing.

Academic Discipline And Sub-Disciplines

Computer networking, Computer Communication

SUBJECT CLASSIFICATION

Computer Science

TYPE (METHOD/APPROACH)

Shortest Path Algorithms

Council for Innovative Research
Peer Review Research Publishing System

Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY

Vol.14, No.8

www.ijctonline.com, editorijctonline@gmail.com

ISSN 2277-3061

INTRODUCTION

Today, we live in a rapid technological revolution and rapid development in the technical age. Technological revolution have active role to the increase of computer information. Raise the level of knowledge abilities, and skills. Increase developments in science and technology. Computer networks are considered one of the important elements that broke all barriers and develop many communication systems. Therefore, high speed routing has become more important in a process transferring packets from source node to destination node with minimum cost. Cost factors may be representing the distance of a router. The simplest link-weight strategy is to give each link a cost of 1; link costs can also be based on bandwidth, propagation delay, financial cost, or administrative preference value. Routing is the act of moving information across an internetwork from a source to a destination [3]. There are two main concepts of the routing: routing protocols and routing algorithms. The protocol defines rules and conventions for communication between network devices. Protocols for computer networking generally use packet switching techniques to send and receive messages in the form of packets. While algorithm is a procedure or formula for solve problem. Algorithm usually means a small procedure that solves a recurrent problem.

Routing protocols

Create to response to the demand for dynamic routing tables. A routing protocol is a combination of rules and procedures that allow routers in the internet deal with each other and adapt with changes [5]. The purpose of any routing protocol is a dynamically communicate information about all network paths used to reach to destination network. Routing protocols is determine the path that the packet are to follow when the packet transmitting from a source node to distention node. Routing protocols can be either an interior protocol or an exterior protocol. An interior protocol handles intra domain routing [8,9]. We discuss two intra-domain routing protocols: distance vector protocol and link state protocol.

Distance Vector Protocol (DVP):

Is designed in 1969 [3]. DVP uses the Bellman-Ford algorithm (also called Ford-Fulkerson) to calculate paths [3]. In DVA, needs to build table for each node to keep a distance vector to all other nodes. Which each node knows the distance of the link to each of its directly connected neighbors. DVP requires that a node informs its neighbors of topology changes periodically. When any change in the topology of a network is detected, must be a router to inform all the nodes in this change. When each node needs to transmit packet information to some destination, it needs transmit packet to the next hop node based on the best shortest path to the destination. Each router should keep at least three pieces of information for each route: destination network, the distance, and the next hop. Consist of the better path usable to pass to the destination and the next hop in the path. The better path can change based on administrative policies like shortest path or lowest cost path [3]**.** Routing Information Protocol (RIP) is the implementation of the distance vector protocol.

Link State Protocol (LSP):

Link-state routing is an alternative to DVP. In LSA, each node must have whole information about other nodes in the network. It stores information around each topology at every node. It computes the route from the source to the destination at the center. While, in DVA all nodes have information about neighboring nodes only. It computes the route on distributed mode [3]. Which each node keeps real picture of the network topology with a distance for each link. When any change occurs in each node, the node is broadcast the local case to every other node in the network. This broadcast process is called flooding, which link-state packets (LSPs) that flood the network. LSP does not distribute any routes, but exchange topology information that describes the network. Open Shortest Path First (OSPF) is the implementation of the link state protocol. The difference between DVP and LSP are: DVP based routers exchange their routing table on a periodic basis, while LSP based routers exchange only routing table updates. DVP is established path on a distributed version of the classical bellman-ford algorithm. While LSP, A node can then make the chart of the network and can compute path from itself to each other node independently [3].

Routing algorithm

Is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on [8]. Routing algorithm that determines the best path for a packet is the heart of any routing protocol. Each algorithm has a different impact on network and router resources. In summary, the routing algorithm is performing tasks according of the routing protocol rules. Routing algorithm can be divided into two main classes: non- adaptive and adaptive algorithms.

Non-adaptive algorithms

Do not dependent on any measurements or estimates of the current topology and traffic for routing decisions. Its choice of the route to use to get from I to J (for all I and J) is computed in advance, offline, and downloaded to the routers when the network is booted [8]. When there is a change in the network or a failure occurs between two nodes, traffic will not be rerouted and does not respond for this change. These procedures sometimes called static routing algorithms. Static routing is often useful for situations in which the routing choice is clearly.

Adaptive algorithm

In contrast, any changes in the topology of network to reflect changes in routing decisions, and sometimes changes in the traffic [8]. These procedures called Dynamic routing algorithms. Dynamic routing adapts well to changes in network topology, such as node failures and network expansion.

SHORTEST PATH ALGORITHMS (SPA)

The shortest path finding algorithms are used to find the minimum weighted or most efficient path in the network. The shortest paths from all vertices in the graph to a single destination vertex is called single shortest path problem. The shortest path between every pair of vertices is called all pairs shortest path problem. There are many algorithms for computing the shortest path such as: Dijkstra's algorithm, Floyd–Warshall algorithm, Bellman Ford Algorithm, and Johnson's algorithm. This paper used these shortest path algorithms for finding shortest path between source node and destination node. This paper is analysis for the results from algorithms, and compare between them. Find the best algorithm according to the time efficiency and number of nodes.

Dijkstra's algorithm:

Conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959 [5,6]. Dijkstra's algorithm is used in search graph algorithm for solve the single-source shortest path problem for a weighted graph with non-negative edge path costs, producing a shortest path tree [1]. This algorithm is often used in routing and as a subroutine in other graph algorithms. The Dijkstra's Algorithm finds the shortest path between two nodes on a network by use the greedy strategy where an algorithm that always takes the best immediate solution when finding an answer. Greedy algorithms find the overall optimal solution for some optimization problems, but may find less-than-optimal solutions for some instances of other problems. In Dijkstra's algorithm firstly, no path is known. Dijkstra's algorithm divides the nodes into two subset groups: temporary set (t) and permanently set (p). Then this algorithm assigns the zero distance value to source node s, and label it as permanent [The state of node s is (0, p)], and Assign to the remaining nodes a distance value of (∞) and label them as temporary. [The state of every other node is (∞, t)]. At each iteration, updates its distance label, and puts the node into a permanently set as permanently labeled nodes p. The permanently labeled distance associated with each examined node is the shortest path distance from the source node to the destination node. The source node is node s and neighbor's nodes are v. At each iteration, the node s is selected and marked, then update distance and label for each node. Selected nodes neighbors for node s and update distance values for these node v by formula the following.

$$Dv = min\{d_v , d_s + (s, v)\} \quad (1)$$

During the above formula, If the labeled distance of node u plus the weight of link (s, v) is shorter than the labeled distance of node v, then the estimated shortest distance from the source node to node v is updated with a value equal to Dv. The algorithm continues the node examination process and takes the next node as source node [2]. The algorithm terminates when is reached to the destination. This is process is clearly in a Figure1.

The computational complexity of the implementation of the Dijkstra's algorithm big-O notation is $O(n^2)$, where n = number of vertices [13]. Big-O notation is frequently used in the computer science and mathematics domain to describe an upper bound on the growth rate of the algorithm [4].

Bellman-ford algorithm

Is an algorithm that computes shortest paths from a single source node to all of the other nodes in a weighted graph. It was conceived by two developers Richard Bellman and Lester Ford. Who published it in 1958 and 1956, respectively; however, Edward F. Moore also published the same algorithm in 1957, and for this reason it is also sometimes called the Bellman–Ford–Moore algorithm. [6]. Bellman-Ford algorithm solves the single-source problem where some of the edge weights may be negative [6]. Dijkstra's algorithm cannot be used to solve the graphs with negative edge weights. The Bellman-Ford Algorithm finds the shortest path between two nodes on a network. This algorithm returns a boolean value representing whether or not there is a negative weight cycle that is reachable from the source, If there is no such a cycle ,the algorithm returns the shortest path, if there is negative cycle then the algorithms tells that no shortest path. A solution exist the Bellman–Ford algorithm can detect negative cycles and report their existence. The algorithm reducing an estimate d(v) on the weight of a shortest path from the source S to each vertex v ∈ V until it obtains the best the shortest path weight. The algorithm returns true if and only when the graph contains no negative weight cycles that are reachable from the source [7]. The bellman-Ford algorithm is executed in the simple example for 6 nodes as shown in figure 2. In step 1, Bellman-ford algorithm assigns every vertex distance to infinity, except the source vertex that gets distance 0. The step 2 relax each edge for (n − 1) times where n are the number of nodes . Relaxing an edge means checking to see if the path to the node to which the edge is pointing can be shortened, and if so, replace the path to the node with the found path [7]. Relax the edge with only 2 nodes starting from the source node, the E(1,2) of cost 7, the cost of the source node plus 7 is less than infinity. So, we replace the

ISSN 2277-3061

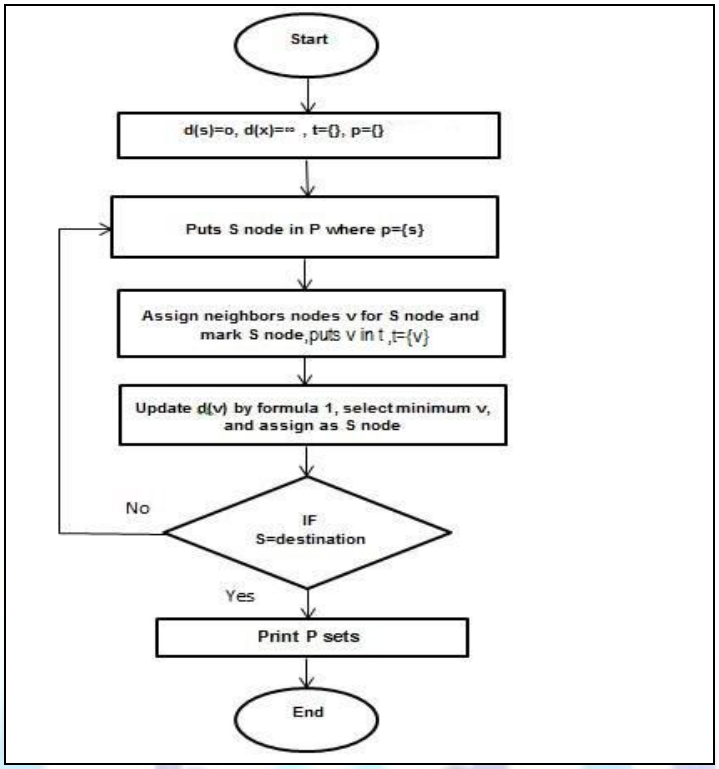

Figure 1. Dijkstra's algorithm

cost of the node 2 d(2)= 7. As well the E(1,6) of cost 6 which is also less than infinity, then d(6)=6 As figure 3. So relax edges for 5 times since n is 6. The step 3, consider the path with 3 nodes and relax the edge E(1,3) through 1→2→3, relax E(1,4) through 1→2→4 or 1→6→4 etc. The step 4, consider the path with 4 nodes. Thus, the process will continue. Hence, the final step gives the shortest path between each node and the source node. Thus all edges are relaxed, checked the negative cost cycle, and the appropriate boolean value is returned. Hence it is called the single-source shortest path algorithm.

```
INITIALIZE-SINGLE-SOURCE (G, S)
For i ←1 to |V[G]|− 1
    do for each edge (u, v) ∈ E[G]
        do RELAX (u, v, w)
for each edge (u, v) ∈ E[G]
    do if d[v] > d[u] + w(u, v)
            then return FALSE
return TRUE
```

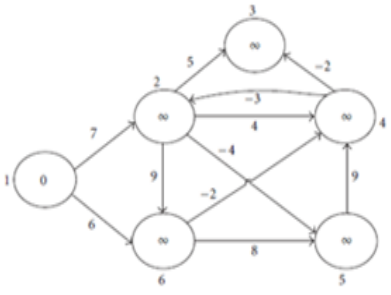

Figure 2 representation step1

Algorithm1. Bellman-Ford [7].

Floyd-Warshall algorithm

Is an algorithm using to computing of the shortest paths between all pairs of vertices in a weighted graph with positive or negative edge weights . The Floyd–Warshall algorithm also known as Floyd's algorithm, Roy–Warshall algorithm, Roy–Floyd algorithm, or the WFI algorithm [1]. The Floyd–Warshall algorithm was published in its currently recognized form by Robert Floyd in 1962 [1]. Floyd-Warshall algorithm uses a matrix (n*n) of lengths D_0 as its input. This matrix represents lengths of all paths between nodes that do not contain any intermediate node is called distance matrix D_{ij}. If there is an edge between nodes i and j, than the matrix D_0 contains its length at the D_{ij} matrix. The diagonal of the matrix contains only zeros. If there is no edge between edges i and j, than the position (i,j) contains positive infinity [2]. This matrix recalculate at every

ISSN 2277-3061

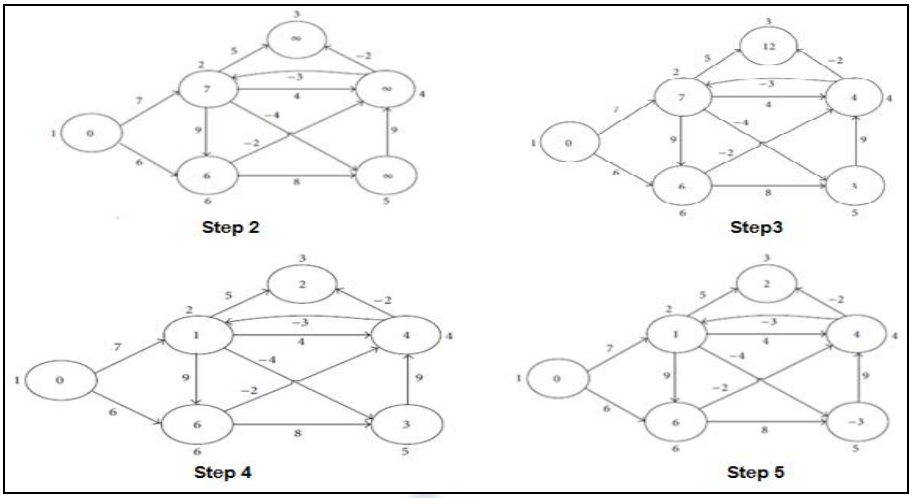

Figure 3. Presentation the steps from 2 to 5 [7].

Bellman–Ford runs in $O(V.E)$ time, where V and E are the number of vertices and edges respectively [10].

iteration of the Floyd-Warshall algorithm. So, it's keep track of the shortest path between any two vertices, using only some subset of the entire collection of vertices as intermediate steps along the path. The matrix D_1, which is created by the first iteration of the procedure, contains paths among all nodes using exactly one (predefined) intermediate node. D_2 Contain lengths using two predefined intermediate nodes. Finally the matrix D_n uses n intermediate nodes. This process can be described using the following recurrent formula [2]:

$$D_{ij}^{n} = min\,(D_{ij}^{n-1}, D_{ik}^{n-1} + D_{Kj}^{n-1}) \qquad (1)$$

The above formula is the heart of the Floyd–Warshall algorithm. The algorithm works by first computing shortest path (i, j, k) for all (i, j) pairs for k = 1, then k = 2, etc. This process continues until k = n, Then find the shortest path for all (i, j) pairs using any intermediate vertices. The pseudocode as shown in algorithm 2:

```
Procedure FLOYD(A, SD, SP)
for i = 1 to n
    for j = 1 to n
    SP(i,j) = j
    SD(i,j)=A(i,j)
    endfor (i)
endfor (i)
for k = 1 to n
          for i = 1 to n
              for j=1 to n
                  if (SD(i,j) > SD(i,k)+SD(k,j))
                      SD(i,j) = SD(i,k)+SD(k,j)
                      SP(i,j) = SP(i,k)
                  endif
              endfor (j)
          endfor (i)
endfor (k)
return SD,SP
End Floyd
```

Algorithm 2. Floyd-Warshall algorithm[2].

In order to return shortest paths among all pairs of nodes, must build another matrix during work of matrix D_k, this matrix is called Sequences matrix(P). In firstly must built initial matrix (n*n) is called A, which that columns equal i. For example, in column 1 all the rows are equal 1, column 2 all the rows are equal 2 etc, and the diagonal of the matrix contains only zeros. The matrix P, initially is equal for matrix A. Update P matrix as algorithm 2. Can be read P matrix as follows: if we want to reconstruct the SP between nodes i and j, we look at the element D_{ij}. If its value is 0, than there is no path between these nodes. Otherwise, the value of the element in D_{ij} of j on the path from i to j. So we repeat this procedure, while the preceding node is not equal to i as Algorithm 2.1. The Floyd-Warshall algorithm runs in $O(n^3)$ where N is number of nodes of the graph [4].

ISSN 2277-3061

```
Function getPath(P, i, j)
   If i = j
      print(i)
   else
    If P(i, j) = 0 then
       print("Path does not exist")
   else
       getPath(P, i, P(i, j))
        print(j)
   end
   end
end \\ function
```

Algorithm 2.1 Floyed-warshall for finding the shortest path.

Johnson's algorithm:

Is a way to find the shortest paths between all pairs of vertices in a graph, whose edges may have positive or negative weights. But no negative-weight cycles may exist. It combines the Bellman-Ford algorithm and Dijkstra's algorithm to quickly find shortest paths [9]. It is named after Donald B. Johnson, who first published the technique in 1977 [11]. The algorithm either returns (n*n) matrix of shortest-path weights for all pairs of vertices $D = dij$, or reports that the input graph contains a negative-weight cycle. The below algorithm is simply perform the actions for the Johnson's algorithm. Johnson's algorithm works as follows; firstly, produce s G' contains new vertex s with zero weight edges from it to all other nodes as algorithm 4. Then runs the Bellman-Ford algorithm on G' with source vertex s as line 2 at algorithm 4. The Bellman-Ford algorithm used to check for negative weight cycles. If this step detects a negative cycle, the algorithm reports the problem and terminated as line 3 at algorithms 4. Lines 4–12 at algorithm 4 assume that G' contains no negative-weight cycles. Line 4-5 The bellman algorithm find the minimum weight h(v)=p(s,v) for each vertex v of a path from s to v. Line 6-7 compute the new weights by the formula following[11,12]:

$$W'(u,v) = W(u,v) + h(u) - h(v) \qquad (2)$$

The for loop in line 9-12 compute the shortest paths weight p'(u,v) by using the Dijkstra's algorithm from vertex in v. Line 12 The correct shortest path stores in matrix d_{uv} as shown the formula 3.The final line is return the completed D matrix [11,12].

$$d_{uv} = P'(u,v) + h(u) - h(v) \qquad (3)$$

```
JOHNSON(G, w)
1   compute G', where G'.V = G.V ∪ {s},
        G'.E = G.E ∪ {(s, v) : v ∈ G.V}, and
        w(s, v) = 0 for all v ∈ G.V
2   if BELLMAN-FORD(G', w, s) == FALSE
3       print "the input graph contains a negative-weight cycle"
4   else for each vertex v ∈ G'.V
5           set h(v) to the value of δ(s, v)
                computed by the Bellman-Ford algorithm
6       for each edge (u, v) ∈ G'.E
7           ŵ(u, v) = w(u, v) + h(u) − h(v)
8       let D = (d_uv) be a new n × n matrix
9       for each vertex u ∈ G.V
10          run DIJKSTRA(G, ŵ, u) to compute δ̂(u, v) for all v ∈ G.V
11          for each vertex v ∈ G.V
12              d_uv = δ̂(u, v) + h(v) − h(u)
13      return D
```

Algorithm 4. Johnson's algorithm [11].

If we implement the min-priority queue in Dijkstra's algorithm by a Fibonacci heap, Johnson's algorithm runs in $O(V^2 \log V + VE)$ time. The simpler binary min-heap implementation yields a running time of $O(VE \lg V)$ [9,11].

CONCLUSION

SPA is an important problem in graph theory and has applications in communications, transportation, and electronics problems**.** In graph theory, used many algorithm that solve the shortest path algorithms. Dijkstra's algorithm finds solution in the single-pair, single-source, and single-destination shortest path problem. Johnsons's algorithm identifies the solution in all pairs shortest path problem. The Floyd Warshall algorithm is assign the shortest path between all pairs of vertices by a graph analysis algorithm. It's an example of dynamic programming. Bellman Ford algorithm obtains solution in the single-source problem if the edge weights are negative too. Tabel 1 described the different SPA.

Tabel.1 Described the different SPA.

ALGORITHM	Negative edge	Single Source	All Sources	Time Complexity	Space Complexity
Dijkstra		√		O(|E| + |V|Log|V|)	$O(v^2)$
Bellman-Ford	√	√		$O(VE)$	$O(v^2)$
FloydWarshall	√		√	$O(n^3)$	$O(n^3)$
Johnson	√		√	$O(V^2 \log V + VE)$	

In [6], Johnson's algorithm may be faster than Floyd–Warshall on sparse graphs. , but the Floyd–Warshall algorithm is faster when the graph is dense [11]. In [4] the Floyd-Warshall algorithm has better cache performance than the sparse matrix implementation because dense matrix computations typically have a higher ratio of floating-point operations to memory. The Dijkstra algorithm does not require the distance matrix to be represented as a dense matrix, thus making the algorithm more memory efficient for sparse graphs. Dijkstra's algorithm time complexity is faster than Bellman-Ford algorithm, but Dijkstra's algorithm can't be used to solve the graphs with negative edge weights [2].

REFERENCES:

[1] Peter Hofner, and Bernhard Moller. 2012." Dijkstra, Floyd and Warshall meet Kleene". Formal Aspects of Computing, 459–476.

[2] Huang, B, Wu, Q, and Zhan, F.B. 2007. "A shortest path algorithm with novel heuristics for dynamic transportation networks. International Journal of Geographical Information Science. Vol. 21, No. 6, 625–644.

[3] Asmaa Shaker Ashoor.2015. "Performance Analysis between Distance Vector Algorithm (DVA) & Link State Algorithm (LSA) For Routing Network".International Journal of Scientific & Technology Research. Vol 4, ISSN 02,101-105.

[4] Anu Pradhan, and Kumar, G. 2013. "Finding All-Pairs Shortest Path for a Large-Scale Transportation Network Using Parallel Floyd-Warshall and Parallel Dijkstra Algorithm". Journal of Computing in Civil Engineering ASCE. 263-273.

[5] Forouzan, B. A. "TCP/IP Protocol Suite". 2010. Prentice Hall. ISBN-13: 978-0-13-2126953. 282-333.

[6] Wei Zhang , Hao Chen , Chong Jiang , Lin Zhu. 2013. "Improvement And Experimental Evaluation Bellman-Ford Algorithm". International Conference on Advanced Information and Communication Technology for Education ICAICTE , 150-153.

[7] Hemalatha, S, and Valsalal, P. 2012. "Identification of Optimal Path in Power System Network Using Bellman Ford AlgorithmS". Hindawi Publishing Corporation Modelling and Simulation in Engineering.

[8] Andrew, S, and David, J. 2011. "Computer network". ISBN-13: 978-0-13-212695-3. 350-366.

[9] Mariusz Głabowski, Bartosz Musznicki, Przemysław Nowak , and Piotr Zwierzykowski. 2013. "Efficiency Evaluation of Shortest Path Algorithms". The Ninth Advanced International Conference on Telecommunications (IARIA). ISBN: 978-1-61208-279-0.

[10] Kevin,R. Hutson, Terri L, Schlosser, and Douglas, R. Shier.2007. "On the Distributed Bellman-Ford Algorithm and the Looping Problem" .Informs Journal on Computing. Vol 19, No 4, 542-551.

[11] Cormen, Thomas H, Leiserson, Charles E, and Rivest, Ronald L.Stein Clifford . 2001. "Johnson's algorithm for sparse graphs". Introduction to Algorithms. ISBN 978-0-262 03293-3. 636–640.

[12] Phillip, G. Bradford , and David , A.Thomas.2009." Labeled Shortest Paths in Digraphs With Negative and Positive Edge Weights". EDP Sciences.567–583.

[13] Nizami Gasilov, Mustafa Doğan, and Volkan Arici. 2011. "Two-stage Shortest Path Algorithm for Solving Optimal Obstacle Avoidance Problem" JOURNAL OF RESEARCH (IETE). VOL 57. ISSUE 3, 278-185.

ISSN 2277-3061

Author' biography with Photo

Hanaa M. Abu-ryash born in Amman in Jordan, on May, 17, 1991. She completed her B.Sc. in Managemen Information Technology from Al-Zaytoonah University. Jordan in 2013. Currently, she is a Master student in the department of Computer Science , Faculty of Science and Innformation Technology , Al-Zaytoonah University of Jordan.

Dr. Abdelfatah A. Tamimi has been a member of the Faculty of Science and Information Technology at Al-Zaytoonah University since 1996, where he held different positions including the Dean of the Faculty and the Chair of the Department of Computer Science. He received his Ph.D. in computer science from the City University of New York, NY, USA; his Master's degree in computer science from Montclair State University, NJ, USA; and his Bachelor's degree in mathematics from Jordan University, Amman, Jordan. In addition to his research and teaching experience, he has a 15 year experience in information technology design, development and implementation in United States companies