Global Journal of Computer Sciences: Theory and Research

Volume 8, Issue 3, (2018) 096-110

www.gjcs.eu

Finding and comparing shortest path solutions by using various methods on a detailed network of the Aegean Sea

Timur Inan*, Computer Programming Programme, Istanbul Arel University, Kucukcekmece, Istanbul 34537, Turkey

Ahmet Fevzi Baba, Electrics-Electronics Engineering, Marmara University, Kadikoy, Istanbul 34722, Turkey

Suggested Citation:

Inan, T. & Baba, A. F. (2018). Finding and comparing shortest path solutions by using various methods on a detailed network of the Aegean Sea. *Global Journal of Computer Sciences: Theory and Research*. *8*(3), 096-110.

Received from March 12, 2018; revised from July 12, 2018; accepted from November, 25, 2018.
Selection and peer review under responsibility of Prof. Dr. Doğan İbrahim, Near East University, Cyprus.

Abstract

Shortest path algorithms are frequently used in important sectoral areas such as maritime, aeronautical and land transport, and are still highly popular. The different side of our work is that it is on the ports of Greece and Turkey on the Aegean Sea and that the nodes are prepared on the actual map based on the actual coordinates and depths. Our study differs from previous studies in that it focuses on Greek and Turkish ports on the Aegean Sea and the nodes are prepared on an actual map based on actual coordinates and depths. The study is part of an intelligent system capable of actually planning the route of a sailing vessel. The Breadth-First, Bellman-Ford and Dijkstra's algorithms were used to calculate the shortest routes between ports and the results were evaluated in terms of the route, distance and calculation time used.

Keywords: Shortest path, Dijkstra, Bellman-Ford, breadth-first, aegean sea, decision support systems.

* **ADDRESS FOR CORRESPONDENCE**: **Timur Inan,** Computer Programming Programme, Istanbul Arel University, Kucukcekmece, Istanbul 34537, Turkey.
E-mail address: timurinan@arel.edu.tr / Tel.: +90-537-669-6627

1. Introduction

Shortest path algorithms are optimisation algorithms commonly used in sectoral areas such as maritime, aviation and land transport. Our work is based on ports in Greece and Turkey on the Aegean Sea. The actual coordinates of the harbours and the nodes are based on the World Port Index, the port authorities' websites, the Google Maps application and the port information (http://www.worldportsource.com, 2016; Ports.com, 2016; World Port Index, 2016). For in-depth knowledge, information requested from the website of the General Bathymetric Chart of Oceans (GEBCO) has been used (GEBCO, 2016; GEBCO2014_22.2269_34.7694_27.9551_41.517_30Sec_ ESRIASCII, 2016). The Mapping Toolbox functions of the MATLAB program were used for mapping. To create the Aegean Sea map, shapefile files containing Greek borders and Turkish borders were used. The QGis program has helped to create the map because there is no common map of the harbours of the two countries. The QGis program was used to create the map as there was no previous map that included the harbours of both countries. The two shapefiles were merged into a new shapefile. The reason for using shapefile is that shapefile does not require detailed harvesting of the map or Internet connection. The whole map of the application can be seen in Figure 1.

1.1. Ports

The harbours and coordinates and their coordinates along the coasts of Greece and Turkey that can be reached by sea were recorded and the ports were identified on the map.

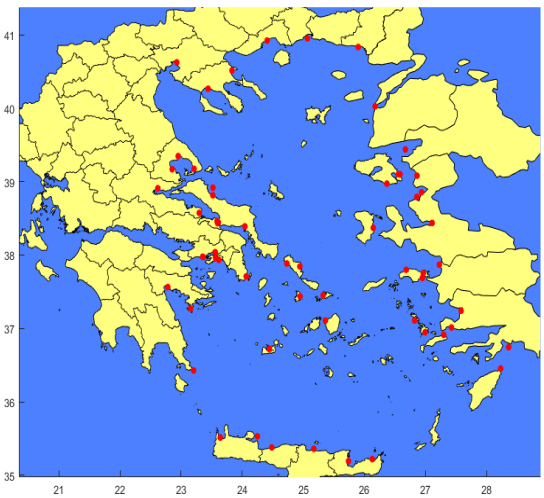

Figure 1. Ports on the aegean sea

The names of the ports registered and the coordinate information are shown in Table 1.

Table 1. Aegean sea ports and coordinates

Port Name	Latitude (North)	Longitude (East)	Port Name	Latitude (North)	Longitude (East)
Achladi	38,9166°	23,52183°	**Gulluk**	37,2407°	27,58553°
Akra Kavonisi	35,5145°	23,64159°	**Iraklion**	35,3613°	25,17378°
Alexandroupoli	40,8329°	25,90073°	**İzmir**	38,4370°	27,11123°
Aliağa	38,8552°	26,93722°	**Kalimnos**	36,9487°	26,99567°
Andros	37,8445°	24,94554°	**Kavala**	40,9211°	24,40924°
Aspropirgos	38,0370°	23,56076°	**Kuşadası**	37,8679°	27,22846°
Ayios Nikolaos	35,1940°	25,74136°	**Kymassi**	38,8135°	23,52061°
Ayvalık	39,3217°	26,63370°	**Lagos**	40,9497°	25,06806°

Bodrum	37,0135°	27,42766°	**Lakki**	37,1089°	26,82484°
Chalkis	38,4430°	23,60909°	**Larimna**	38,5731°	23,29796°
Dhiavlos Steno	38,4498°	23,59945°	**Lavrio**	37,7081°	24,06613°
Dikili	39,0806°	26,86314°	**Limin Kos**	36,9136°	27,30059°
Elevsis	38,0384°	23,55630°	**Limin Sirou**	37,4352°	24,94863°
Gavrio	37,8849°	24,73322°	**Marmaris**	36,7451°	28,36273°
Megara Oil Terminal	37,9770°	23,35355°	**Porthmos Evripou**	38,4618°	23,5884°
Mikonos	37,4499°	25,3260°	**Rethimnon**	35,3801°	24,4851°
Miliana	39,1740°	23,2131°	**Rodhos**	36,4548°	28,2330°
Milos	36,7211°	24,4438°	**Sitia**	35,2209°	26,1298°
Mitilini	39,1006°	26,5848°	**Soudha**	35,5318°	24,2495°
Navplio	37,5636°	22,7829°	**Spetses**	37,2695°	23,1561°
Nemrut Limanı	38,7919°	26,8606°	**Stilis**	38,9091°	22,6192°
Neon Karlovas	37,7993°	26,6855°	**Stratoni**	40,5130°	23,8329°
Nisos Naxos	37,1078°	25,3641°	**Thessaloniki**	40,6234°	22,9295°
Ormos Aliveriou	38,3899°	24,0450°	**Tsingeli**	39,1689°	22,8536°
Ormos Mikro Vathi	38,4343°	23,5993°	**Volos**	39,3464°	22,9525°
Pachi Oil Terminal	37,9725°	23,3628°	**Vrakhonisis Kallonis**	39,1010°	26,5577°
Perama	37,9571°	23,5663°	**Yerakini**	36,4279°	23,2045°
Piraievs	37,9344°	23,6174°	**Çanakkale**	40,2643°	23,4426°
Pithagorion	37,6867°	26,9507°	**Ekklisia Agios Georgios**	40,0254°	26,1766°
Plomarion	38,9723°	26,3694°			

1.2. Nodes

Due to large islands situated in the Aegean Sea, the connections between most ports cannot be expressed by a straight line. It is necessary to calculate the coordinates of these nodes. These nodes can be calculated by determining the node map, calculating the connections of the sea routes between the nodes and determining which nodes are connected to which ports. When determining the node map, the nodes are calculated by establishing the points of the lands as the shortest path as if the commercial vessels were planning the route for a possible cruise. The experience gained by the author while planning the route of commercial vessels was used during the determination of the nodes (Table 2).

The reason why the node map is so detailed is that it is going to be used for the operation of the intelligent system that we are working with. Since the intelligent system can dynamically change the route planning, any other node on the map or on the road can be navigated on any node or it can evaluate alternatives. The real need for determining the node map is that it is necessary to obtain a node map that will allow the intelligent system to consider the likelihood of a captain and to assess how these possibilities can be achieved, rather than calculating the shortest path between ports.

Table 2. Numbers and coordinates of nodes

Node	Latitude (North)	Longitude (East)	Node	Latitude (North)	Longitude (East)	Node	Latitude (North)	Longitude (East)
1	39,9409°	26,0538°	**11**	39,9935°	26,1701°	**21**	39,0330°	26,6396°
2	39,9363°	26,0932°	**12**	40,0131°	26,1927°	**22**	38,9238°	26,5898°
3	39,8420°	25,9578°	**13**	39,7605°	26,1303°	**23**	38,9514°	26,4051°
4	39,8508°	26,0548°	**14**	39,5903°	26,0940°	**24**	38,9591°	26,1613°
5	39,8309°	26,0848°	**15**	39,4776°	26,0500°	**25**	39,0274°	25,8922°

6	39,7978°	26,0843°	**16**	39,4491°	26,1160°	**26**	39,1808°	25,7473°
7	39,7861°	26,0488°	**17**	39,3879°	26,1849°	**…**	…	…
8	39,8203°	25,9662°	**18**	39,3983°	26,3446°	**604**	35,5382°	24,2472°
9	39,8408°	26,1285°	**19**	39,3366°	26,4320°			
10	39,9152°	26,1421°	**20**	39,1687°	26,5522°			

As the number of nodes is too long, the table is kept short. The nodes have a critical presumption of whether there is a connection by sea if there are distances between them and what the angle between them is.

1.3. Calculation of the angles between the nodes

In order to calculate the links between the nodes, it is necessary to record the angle of the link between the nodes. The angles are used to understand whether there is a real sea connection between the nodes, and also the intelligent system will be used in the course of route determination. The matrix of angles is formed to hold 604 nodes between each other as shown in Table 3.

Table 3. Matrix showing the angle between the nodes (angles in degrees)

Node	1	2	3	4	…	604
1	0°	99°	217°	180°	…	198°
2	279°	0°	228°	199°	…	199°
3	37°	48°	0°	83°	…	198°
4	360°	19°	263°	0°	…	198°
5	348°	4°	277°	311°	…	199°
6	351°	3°	294°	337°	…	199°
7	1°	13°	309°	4°	…	199°
…	…	…	…	…	0°	198°
604	17°	17°	17°	17°	…	0°

1.4. Calculation of distances between nodes

In order to calculate the connections between the nodes, it is necessary to record the distance between the nodes. The matrix of distances is formed so that the 604 nodes hold the distance between each other (Table 4).

Table 4. Matrix indicating distances between nodes (distance in kilometres)

Node	1	2	3	…	604
1	0	3,404	13,71	…	499,32
2	3,40	0	15,60	…	499,9
3	13,71	15,60	0	…	486,35
4	10,01	10,05	8,334	…	489,84
5	12,52	11,74	10,90	…	488,56
6	16,12	15,41	11,86	…	485,07
7	17,22	17,13	9,955	…	482,85
…	…	…	…	0	…
604	499,3	499,9	486,35	…	0

1.5. Calculation of links between nodes

During the calculation of the connections between the nodes, the distances between the nodes and the angles were calculated to check whether there were any blackouts on the path between the

nodes. Shapefile files can be matched to find out whether the specified point is on the sea or on land. This method calculates whether the existing 604 nodes are directly connected to each other. If the distance and the angle between two nodes are known, the existence of the connection link can be calculated step by step at certain distance intervals and can be determined whether the specified point is on land or sea. The distances between the points and the angles were recorded and two matrices were created for distances and angles at a size of 604 × 604. Next, a connection matrix of size 604 × 604 was created to record the connections (Table 5). In the matrix, the information was recorded as 1 for the connected links and 0 for the unconnected links. The step-by-step calculation is done as follows:

$$\Delta x = km * \sin\left(\text{angle}\left(\text{node1},\text{node2}\right)\right) \quad (1)$$

$$\Delta y = km * \cos\left(\text{angle}\left(\text{node1},\text{node2}\right)\right) \quad (2)$$

$$\Delta_{\text{longtitude}} = \Delta x / (11320 * \cos\left(\text{node1.latitude}\right) \quad (3)$$

$$\Delta_{\text{latitude}} = \Delta_y / 110540 \quad (4)$$

Δ*x*: The amount of travel required to travel at the desired mileage in the desired longitude

Δ*y*: The amount of travel required to travel at the desired mileage in the desired latitude.

km: Desired amount of roads to travel (kilometres).

$\Delta_{\text{longtitude}}$ = Degrees longitude value of the amount required to proceed on the road.

Δ_{latitude} = Degrees latitude value of the amount required to proceed on the road.

After calculating the step range, the coordinate is stored in link matrix 1 if it coincides with the land segment and 0 if it does not coincide with the land segment after the completion of all the steps.

Table 5. The matrix showing the link between nodes

Node	1	2	3	4	5	…	604
1	0	1	1	1	1	…	0
2	0	0	0	0	0	…	0
3	0	0	0	0	0	…	0
4	0	0	0	0	0	…	0
5	0	0	0	0	0	…	0
…	…	…	…	…	…	…	…
6 0 4	0	0	0	0	0		0

2. Methods

Three different methods have been used to calculate the shortest paths. These are the Breadth-First, Bellman-Ford and Dijkstra's algorithms.

2.1. Breadth-first algorithm

This algorithm, which is a depth-first algorithm, ends with finding the end node by traversing all connected nodes from the start node to the end node. The advantage of this algorithm is that the accuracy is high because all nodes are visited, the disadvantage, however, is that sometimes it is possible to find the end node longer compared to other algorithms due to all the nodes having been traversed.

2.2. Structure of the breadth-first algorithm

The Breadth-First algorithm keeps all connected nodes in a queue, starting from the start node, and initially marks all nodes as 'not visited'. After the first node has been connected to the queue, the algorithm registers the distances by traversing the connected nodes in order. Then, it repeats the same process for the other nodes. Each visit continues until the end node, marking the nodes as 'visited' (Kaur, 2014). The structure of the algorithm can be seen in Figure 2.

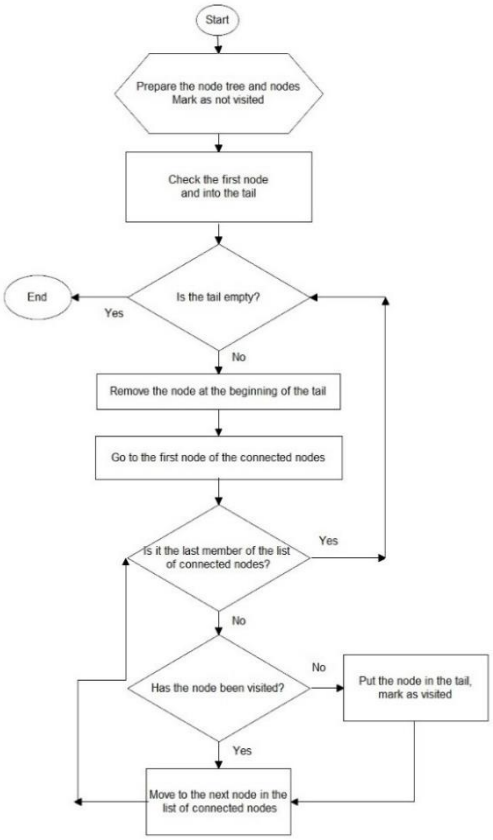

Figure 2. Breadth-first search algorithm flow chart

In order for this algorithm to be used for finding the shortest path, the connected nodes must be arranged in a certain order and the shortest paths must be calculated. The obtained tree structure information and the previously obtained distances matrix can be used together to obtain the shortest path calculation.

2.3. Bellman-Ford algorithm

The Bellman-Ford algorithm is a popular algorithm for shortest path computation. This algorithm moves the paths between nodes one at a time to find the shortest path. While it is advantageous to approach the right conclusion because it traverses all the paths, it can sometimes be regarded as a disadvantage to arrive slower than other short path algorithms, because all paths spend a lot of time during the traversal.

In the initial phase of the algorithm, the cost of the start node is 0 and the cost of the other nodes is infinite. The nodes that can be traversed from the start node are placed in order. The cost values of all connected nodes are updated by calculating the distance from the originating node to the connected nodes. The updated value is obtained by the cost of the initial node plus the distance of the node. The

cost information of the adjacent nodes is thus updated. Then, the nodes are examined one by one. The costs are updated by calculating the distance to the nodes connected to the nodes from the first node to the last node. If a shorter distance from the previously visited node is found during the update phase, the cost information of that node is updated to record which node has the lower cost reason (Bellman, 1958; Ford, 1956). This process continues until the end node has the lowest cost. A simple example is shown in Figure 3.

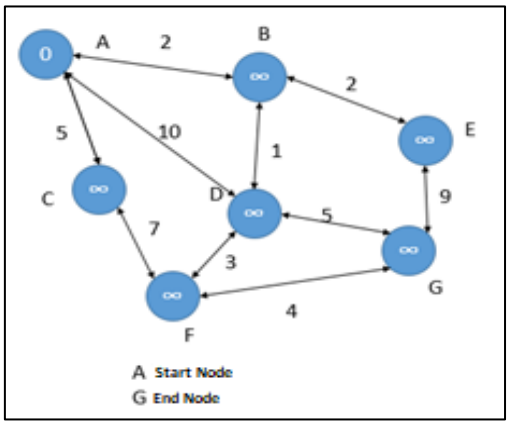

Figure 3. A simple shortest path proble

Step 1: The cost of the initial node is taken as 0 and the cost of the other nodes is recorded as infinite.

Step2: The cost of B node is recorded as 2 by calculating the cost of A node and the path between A and B.

Step 3: The cost of the D node, which is the node adjacent to A node, is calculated by summing the cost of A node with the path A-D.

Step 4: The cost of the C-node is determined as 0 + 5 = 5.

Step 5: The cost of the E-node is determined as 2 + 2 = 4.

Step 6: The cost of the D-node is updated to 2 + 1 = 3.

Step7: The cost of the G-node is updated to 8.

Step8: The cost of the F-node is updated to 6.

Step 9: The cost of the F-node is calculated by adding the cost of the C-node to the sum of C-F, but the value of the F-node is not updated because the cost is already 6.

Step 10: Since the first node connected to B node is the E node, the cost of the G node is calculated by including the cost from E to G, but the value is not updated because the cost of the G node is 8.

Step 11: The new cost of the G-node is calculated by adding the cost between the F-node and the G-node, but the value is not updated because the cost of the G-node is already lower.

As a result, the A-B-D-G sequence is found as the shortest path.

2.4. Dijkstra's algorithm

Dijkstra's algorithm differs according to the Bellman-Ford algorithm. Dijkstra's algorithm is faster than the Bellman-Ford algorithm as not all nodes are circled in the Dijkstra's algorithm. The distances to the nodes connected to the starting node are in order. These nodes are then recorded by directing the node that brings the total cost to the minimum value (Dijkstra, 1959).

The example shown in Figure 3 was repeated using Dijkstra's algorithm. The steps for this example are as follows. B node is the node that can be moved at the lowest cost by moving from A node. For the shortest path, the B node is stored in the calculated node array; the array becomes {A and B}. From here, there are nodes D and E that can be moved from B node to A node. The shortest path will cost 3 for the D node. Since this cost is lower than the cost from A node to D node or C node, D node is added to the array, taking the form of an array which becomes {A, B and D}. The nodes that can be traversed from D node are nodes F and G. The cost of going to the F node is 6, while the cost of going to the G node is 8; thus, the F node is selected. However, if the cost is lower than the C node, the C node is selected and the cost is reduced to 5. The sequence becomes {A, B, D and C} accordingly. From the C node, it is only possible to go to the F node, which increases the cost to 12. However, as the lower cost 6 has already been found, the sequence then becomes {A, B, D, C and F}. Only the G node, which can be moved from the F node, is left because the C node and the D node have also been visited.

Going to G node costs 10, but going from D node to G node is less costly, so this path is not preferred. The sequence becomes {A, B, D, C, F and G}. Since there is a direct connection from D node to G node, nodes C and F are removed from the array and the shortest path is found as {A, B, D and G}.

3. Results

The Breadth-First, Bellman-Ford and Dijkstra's algorithms were used to calculate the shortest route nodes, the shortest path distances and the time used in the calculation are recorded.

The shortest distance between each of the first four ports will be shown in a table because the information for 61 ports will be too much longer. The ports are located in nodes 539, which is the node number of the first port and 599, which is the node number of the last port.

3.1. Experimental results of the breadth-first algorithm

3.1.1. Result nodes obtained using the breadth-first algorithm

The nodes for the shortest paths computed by the Breadth-first algorithm were recorded for calculations.

An interface was prepared in order to show the calculated values for all ports one at a time. The results can be displayed by selecting the start and end ports and calculation methods for the interface. The calculation results between the Achladi and the Aya Nikolaos ports obtained using the Breadth-first algorithm are shown in Figure 4.

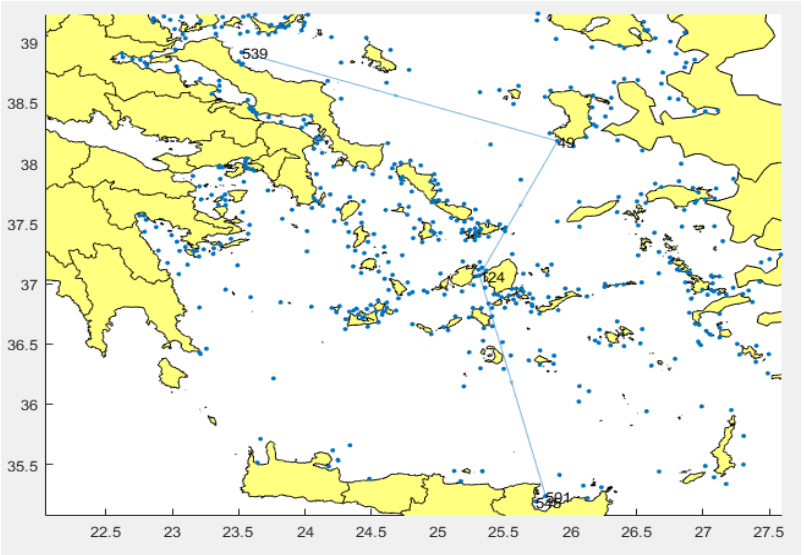

Figure 4. Calculation of the shortest path between Achladi and Aya Nikolaos by breadth-first algorithm

3.1.2. Distances

The distance calculated by the Breadth-first algorithm was recorded. The records in Table 6 show the distance information obtained for the first four ports and the distance information of the last port to the other four ports. The values are given in kilometres.

Table 6. Breadth-first algorithm's shortest distance between the ports

Node	539	540	541	...	599
539	0	1681,49	655,15	...	1264,44
540	1681,49	0	1805,58	...	566,413
541	655,15	1805,58	0	...	1245,28
...	...	...	...	...	...
599	1264,44	566,41	1245,28	...	0

3.1.3. Durations

With the Breadth-first algorithm, the time elapsed during short path results were recorded and some of the records are shown in Table 7. The values are given in ms.

Table 7. Time spent calculating the shortest paths of the breadth-first algorithm

Node	539	540	541	...	599
539	0	7,221	6,601	...	6,596
540	7,604	0	7,033	...	7,098
541	6,531	6,536	0	...	6,583
...	...	...	...	...	...
599	7,224	8,335	6,574	...	0

3.2. Experimental results of the Bellman-Ford algorithm

3.2.1. Result nodes obtained using the Bellman-Ford algorithm

The shortest path results obtained by the Bellman-Ford algorithm are recorded for calculations.

Calculated values are shown on the map in Figure 5, which also shows the calculation result between the Achladi and Aya Nikolaos ports using the Bellman-Ford algorithm.

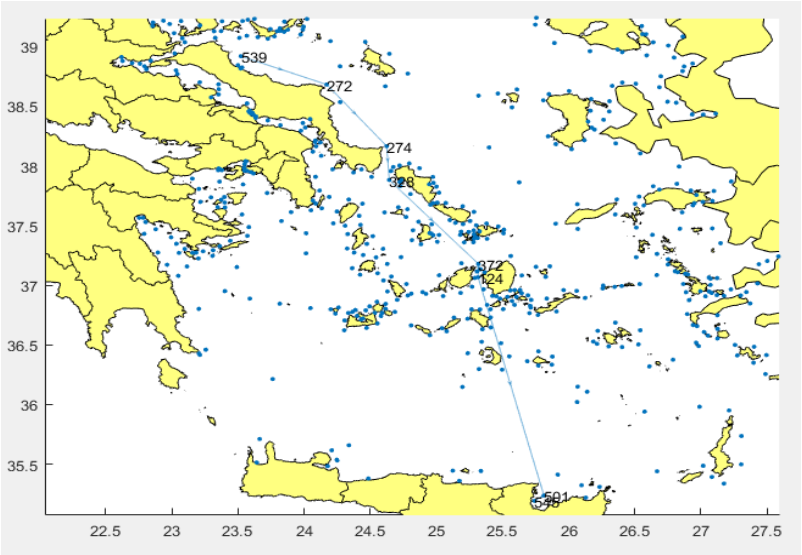

Figure 5. The result obtained by the calculation of the shortest route between the Achladi port and the Aya Nikolaos port by the Bellman-Ford algorithm

3.2.2. Distances

From the results obtained by the Bellman-Ford algorithm, the amount of distance the nodes will reach is recorded. Some of the distance information can be seen in Table 8. The values have been given in kilometres.

Table 8. Distances of the roads calculated by the Bellman-Ford algorithm

Node	539	540	541	…	599
539	0	962,12	533,9	…	664,5
540	962,12	0	1289,1	…	566,4
541	533,93	1289,1	0	…	991,9
…	…	…	…	…	…
599	664,56	566,4	991,9	…	0

3.2.3. Durations

The breadth-first algorithm calculates the shortest paths calculated during the time taken to record them. Some of the records are shown in Table 9. The values have been given in ms.

Table 9. The time spent calculating the shortest paths of the Bellman-Ford algorithm

Node	539	540	541	…	599
539	1,45	6,21	5,13	…	4,79
540	5,02	5,50	6,27	…	5,22
541	4,78	5,28	4,74	…	4,90
…	…	…	…	…	…
599	5,66	5,47	5,09	…	0

3.3. Experimental results of Dijkstra's algorithm

3.3.1. Result nodes obtained using Dijkstra's algorithm

The node sequence for the paths computed by the Dijkstra algorithm was recorded for calculations and is shown on the map. The shortest paths calculated by the Dijkstra's algorithm are also shown on the map. The calculation result between Achladi and Aya Nikolaos port is shown in Figure 6.

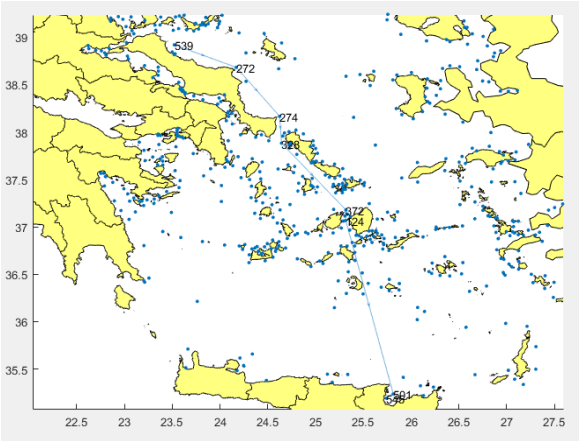

Figure 6. Calculation of the shortest path between Achladi and Aya Nikolaos by Dijkstra's algorithm

3.3.2. Distances

As can be seen, the Bellman-Ford algorithm and the Dijkstra's algorithm gave the same results for these ports. Detailed comparisons and results are given in the comparative section.

The amount of distance that must be taken when the nodes on the path obtained from the results obtained by Dijkstra's algorithm is recorded. Table 10 shows a portion of the distance information (Values in kilometres).

Table 10. Distances calculated by Dijkstra's algorithm

Node	539	540	541	…	599
539	0	962	533	…	664,56
540	962	0	1289	…	566,41
541	533	1289	0	…	991,97
…	…	…	…	…	…
599	664	566	991	…	0

3.3.3. Durations (Dijkstra's)

The time taken to obtain the short paths calculated by Dijkstra's algorithm was recorded and some of the records are shown in Table 11. The values are given in ms.

Table 11. The elapsed times when calculating the shortest paths of Dijkstra's algorithm

Node	539	540	541	…	599
539	0	6,2	5,2	…	5,95
540	5,43	0	5,17	…	5,30
541	5,71	5,71	0	…	5,58
…	…	…	…	…	…
599	4,83	4,59	4,95	…	0

The values calculated using the three algorithms are recorded. The advantages and disadvantages of each algorithm are explored and graphically presented in this section. Results are compared according to distance and time criteria.

3.4. Comparison of distances

The distances obtained by the algorithms according to the shortest paths were evaluated. For example, the distances of the Achladi port to the other 60 ports are shown in a bar graph (Figure 7) in accordance with the results obtained by the algorithms. In terms of distance, Bellman-Ford and Dijkstra's algorithms yield the same distance results for all ports. The Breadth-first algorithm produces longer distance results than the other two algorithms.

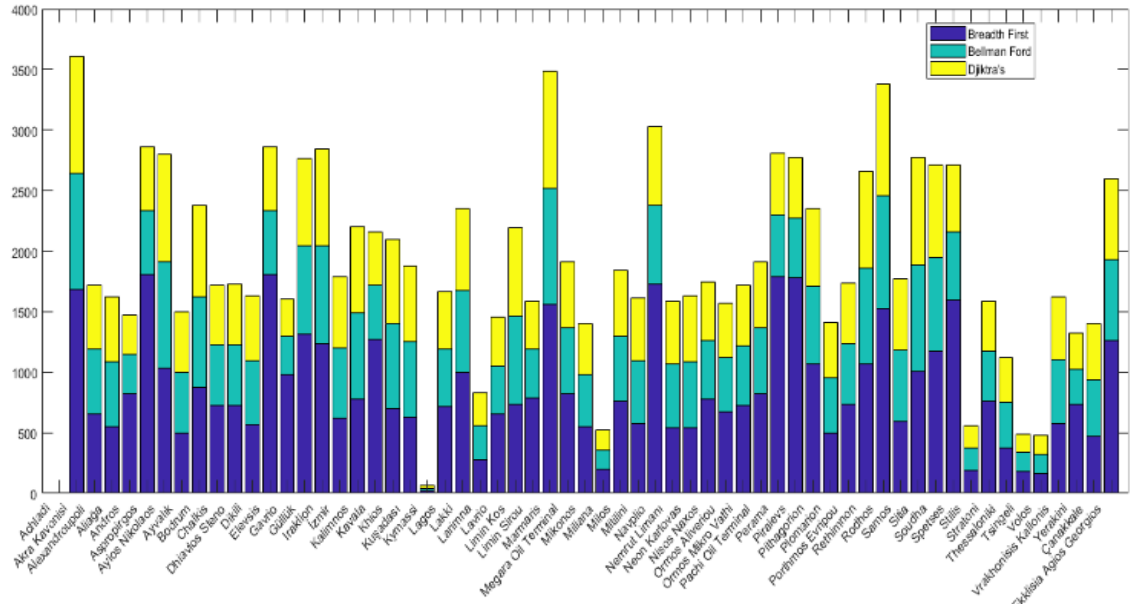

Figure 7. Comparison of distance values obtained by algorithms

3.5. Comparison of calculation times

The time that algorithms spent computing was recorded in terms of the speed of their algorithm. The comparison of the speeds of the algorithms is shown in Figure 8. The Breadth-First algorithm was found to calculate the port for a longer period of time than the others. When the value of the time spent while calculating distances from the Achladi port to the other 60 ports were compared with each other, it was found that the Breadth-first algorithm calculated the port for a longer period of time than to the other two algorithms the others. When the mean values of the Bellman-Ford algorithm and Dijkstra's algorithm are compared, it is observed that the Bellman-Ford algorithm has a value of 5.4 ms and Dijkstra's algorithm has a value of 5.2 ms.

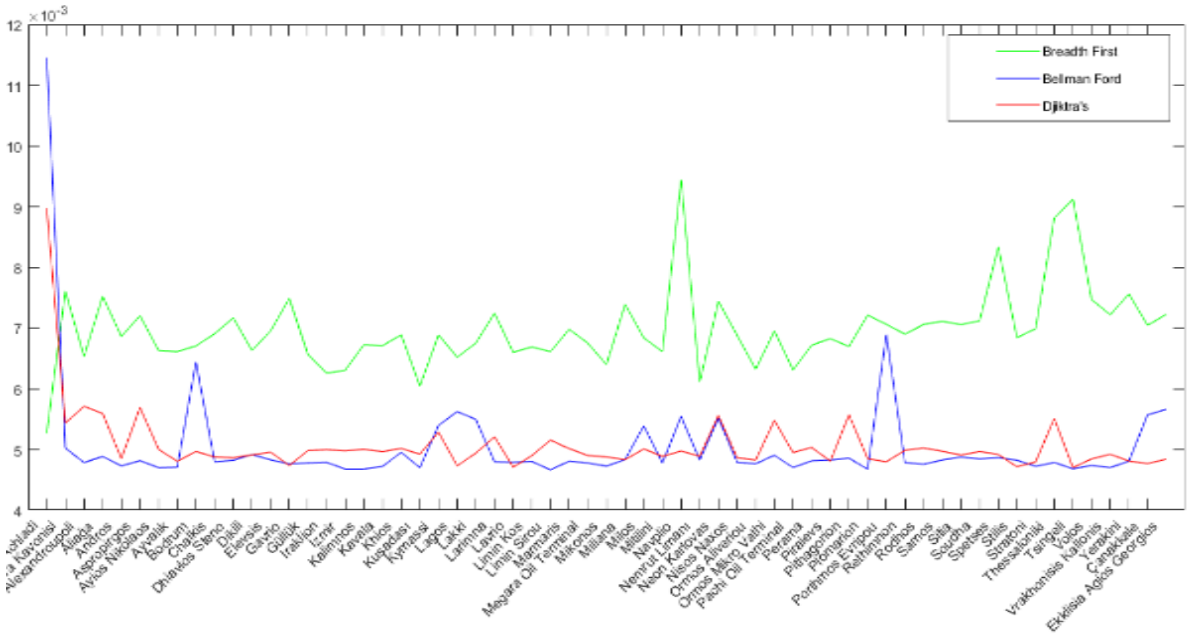

Figure 8. Distance results obtained by all algoritms from the Port of Achladi to all ports

4. Discussion

Given the short paths and time spent, it was observed that the Breadth-First algorithm was slow to find the end and did not give the shortest path in most cases. The Breadth-First algorithm guarantees to find a result because it is already a search algorithm but in order to use it as a shortest path algorithm, it is necessary to reach the shortest path from the road possibilities by looking at the node tree after finding the node. In this work, because of the cost of getting zero in the process of finding the target, the algorithm has not always found the shortest path. An algorithm such as the Bellman-Ford or Dijkstra's algorithm is needed after finding the result node to find the shortest path using the Breadth-First algorithm. On the whole, it is observed that the Bellman-Ford and Dijkstra's algorithms find the shortest path and give the same results but differ in calculation times.

In Figure 9(c), it is seen that in some places, Dijkstra's algorithm is faster than the Bellman-Ford algorithm. The said places have been circled.

In Figure 9(a, b) the related experiments can be seen on a map. Dijkstra's algorithm computes faster between the Achladi-Chalkis and Achladi-Rethymnon ports. The reason is that the Dijkstra's algorithm continues on the same line if it cannot find a less costly way to calculate the cost to find the result. The Bellman-Ford algorithm is slow in these two cases as it goes through individual node costs.

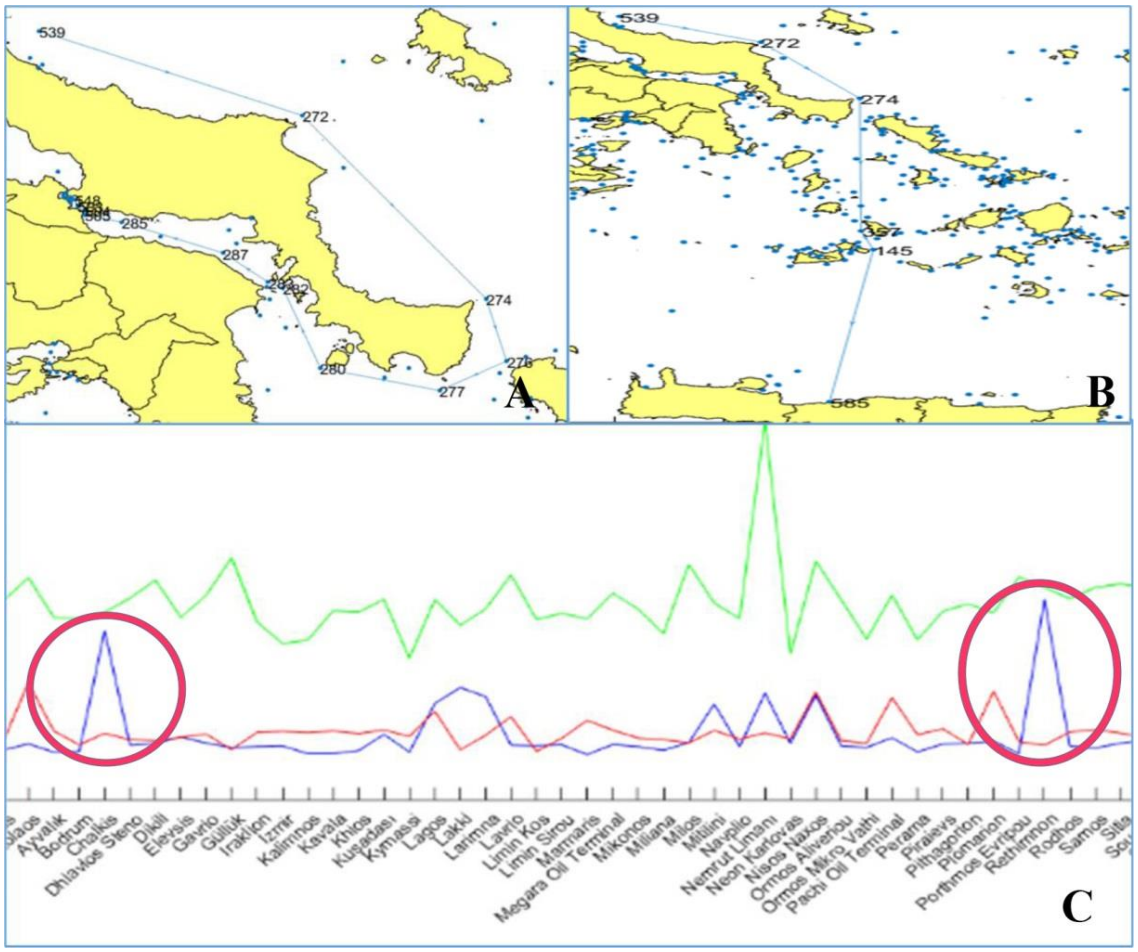

Figure 9. a) Result of Achladi–Chalkis computation with Bellman Ford and Dijkstra's algorithms. b) Result of Achladi–Rethymnon computation with Bellman Ford and Dijkstra's algorithms. c) Computation times of BellmanFord and Dijkstra's algorithms, difference shown in the circles and the blue line presents the Bellman Ford algorithm

References

Bellman, R. (1958). On a routing problem. *Quarterly of Applied Mathe-matics, 16*(1), 87–90.

British Ocenographic Data Centre. (2016). Retrieved from December 20, 2016 https://www.bodc.ac.uk https://www.bodc.ac.uk/my_account/get_basket/E2DH5DEF85D870C4E008G86129F5G512/SkeyGEBCONETE/

Dijkstra, E. W. (1959). A note on two problems in connexion withgraphs. *Numerısche Mathematik, 1*(1), 269–271.

Ford, L. R. (1956). Network flow theory. *Defense Documentation Center for Scientific and Technical Information, 923*.

GEBCO. (2016). Retrieved from December 13, 2016 http://www.gebco.net http://www.gebco.net/data_and_products/gridded_bathymetry_data/

GEBCO2014_22.2269_34.7694_27.9551_41.517_30Sec_ESRIASCII. (2016). Retrieved from December 20, 2016 GEBCO2014_22.2269_34.7694_27.9551_41.517_30Sec_ESRIASCII

http://www.worldportsource.com(2016). *World port source.* Retrieved from December 13, 2016 http://www.worldportsource.com/waterways/Aegean_Sea_8.php

Kaur, A. S. P. (2014). A appraisal paper on breadth-first search, depth-first. *International Journal of Scientific and Research Publications, 4*(3).

Oceans, G. (2016). *Gridded bathymetric data*.

Ports.com. (2016). Retrieved from December 12, 2016 http://ports.com: http://ports.com/sea/aegean-sea/

World Port Index. (2016). *National geospatial intelligence agency*: Retrieved from December 10, 2016 https://msi.nga.mil/NGAPortal/MSI.portal?_nfpb=true&_pageLabel=msi_portal_page_62&pubCode=0015