Scientific Journals
of the Maritime University of Szczecin

Zeszyty Naukowe
Akademii Morskiej w Szczecinie

2020, 62 (134), 61–68
ISSN 1733-8670 (Printed)
ISSN 2392-0378 (Online)
DOI: 10.17402/420

Received: 02.12.2019
Accepted: 08.06.2020
Published: 29.06.2020

Network routing method for ships and other moving objects using MATLAB

Vladimir V. Sakharov[1], Alexandr A. Chertkov[1], Igor B. Ariefjew[2]

[1] Admiral Makarov State University of Maritime and Inland Shipping
5/7 Dvinskaya St., St. Petersburg 198035, Russian Federation
e-mail: _saharov_@rambler.ru; chertkov51@mail.ru
[2] Maritime University of Szczecin
11 H. Pobożnego St., 70-507 Szczecin, Poland
e-mail: i.ariefiew@am.szczecin.pl
corresponding author

Key words: algorithm, path, optimization, transport, automation, moving object

Abstract
Task planning involves automating the creation of the routes for vessels with known coordinates in a confined space. The management of vessel release in a given area affects the time required for a vessel to complete its voyage, and maximizing vessel performance involves identifying the shortest route. A key issue in automating the generation of the optimal (shortest) routes is selecting the appropriate mathematical apparatus. This paper considers an optimization method based on a recursive algorithm using Bellman-Ford routing tasks for large dimensions. Unlike other optimization techniques, the proposed method enables the shortest path to be assessed in a network model with a complex topology, even if there are arcs with negative weights. The practical implementation of the modified Floyd algorithm was demonstrated using a sample automated build and using it to calculate a network model with a complex topology, using an iterative procedure for a program prepared in MATLAB. Implementation of the computer model is simple, and unlike existing models, it eliminates restrictions associated with the presence of negative weights and cycles on a network and automates search shortcuts in ground branch functional means in MATLAB. To confirm the accuracy of the obtained results, we performed an example calculation using the network. The proposed algorithm and recursive procedure are recommended for finding energy-efficient solutions during the management of mobile objects on waterways.

Introduction

The issues associated with choosing the movement routes of vessels using crewless technologies to organize movement have become extremely relevant, requiring operational solutions in case of situational changes, both along the route and in ports of destination. For example, to save fuel and energy along the route, it is necessary to assess a ship's time remaining to its port of destination without changing the volume of transport work performed by the ship. When ships move in stormy weather and during group ship management, schedule adjustments may be made, which also leads to the formation of running time reserves. Fuel reserves and energy savings can also be estimated when passing the gateway sections of a track, which requires dispatch services to have flexible operational solutions. The shortest or most efficient ship traffic routes are determined by taking into account the loading, freight cost, transportation costs, logistics characteristics, etc. – this class of tasks requires the creation of new operational technologies, models, and algorithms based on tools to make optimal decisions by computer systems. The evaluation and selection of the optimal (shortest) routes for organizing ship movements can be carried out using traffic management methods using operations research. It is customary to solve this class of problems using graph theory by introducing nodes and arcs with various weights (including

negative ones), which provides a convenient form to graphically interpret both objects and their models.

Numerous algorithms have been created to model graphs of stochastic and deterministic systems, which actually form tools to solve this type of problem. Fundamental works in this subject area include Dijkstra's algorithms (Chertkov, Vardomskaya & Dmitriev, 2015a), Bellman-Ford (Bellman, 1958), a modified Bellman-Ford algorithm (Chertkov, 2017), and Floyd-Warshell, which were considered in detail in (Saharov, Chertkov & Tormashev, 2014; Chertkov, Vardomskaya & Dmitriev, 2015b; Saharov, Chertkov & Dmitriev, 2016; Saharov, Sikarev & Chertkov, 2018; Saharov, Chertkov & Ochina, 2019). With the development of intelligent methods and systems, neural networks, genetic algorithms, as well as Fuzzy Systems, ANFIS models, etc., algorithms for decision-making in high-dimensional systems have appeared (Fonseca & Fleming, 1993; Sazonov & Derjabin, 2013), including operating spaces for coordinating actions to manage moving objects.

For optimization problems in operational spaces, it is important to provide a scientific justification and solve practical problems when choosing ship traffic routes. Mobile robots, video robots in service systems, and the provision of transport services in market demand are used (Castillo, Trujillo & Melin, 2007; Ariefiew, 2012; Chertkov, 2015; Kirsanov, 2016; Fedorenko & Olovyannikov, 2017). The scientific and practical significance of network models is significantly expanding. Using them to generate ship traffic routes that converge in nodes and operational subspaces on approach to ports, during pilotage, and during ebb and flow requires special measures to ensure the safety and trouble-free performance of all technical operations. A systematic solution to the above problems requires developing optimization algorithms on networks and the creation of an instrumental base for the crewless control of ships from the shore.

Methods and materials

Here, we consider the assessment and selection of the shortest routes for ship movement on selected operational subspaces using the example of an extensive network with a system of targets that simulate a real situation. We assume that the coordinates of the target location on subspaces are known. The task is to choose the route that will require the least amount of time to reach the goal. We will estimate the working time $T_{\max}$ when managing a group of ships according to the time necessary to arrive at the most distant target. The vessel speed (according to the accepted terminology in the routing of networks – "agents") was the same for all objects. We neglected the "processing" time of ships in our models. The accepted conditions allow the "agent" speed to equal 1. Then, the execution time of the operation can conditionally be taken equal to the length of the vessel route.

When solving a problem, we used the recurrent method of sequential operations, which is applicable to the parametric estimation of the route of each "agent". At the zeroth iteration, any "agent" can be arbitrarily selected for which its closest target is further away. Next, the location of the "agent" in the new position is fixed, and the decision is repeated until this "agent" passes all the goals assigned to it. According to recursion, these goals are excluded from the list of goals, and the next "agent" is selected, for which the problem is solved similarly. The computational algorithm can be attributed to the class of locally optimal (or "greedy") algorithms, which may involve a solution falling into a local minimum.

In contrast to the algorithms mentioned in (Bellman, 1958; Saharov, Chertkov & Tormashev, 2014; Chertkov, Vardomskaya & Dmitriev, 2015b; Saharov, Chertkov & Dmitriev, 2016; Chertkov, 2017; Saharov, Sikarev & Chertkov, 2018; Saharov, Chertkov & Ochina, 2019), a solution is proposed to not only estimate the shortest paths between given vertices of the graph but also to algorithmically construct it with a given configuration and digitize all vertices and edges. In addition, the solution to the shortest path problem is supplemented by simultaneously assessing the shortest paths between any two nodes of the network, as well as assessing the state variables on weighted graphs, both with positive and negative edge weights. This makes it possible to introduce corrective actions according to the calculation results by taking into account the convergence of the solution to the optimum. The negative weight of the individual edges of the graph was due to the influence of a changing external environment (fair wind, currents, waves, etc.); the fuel consumption in these sections will be lower than the average value. According to the algorithm, all subtasks are solved in inverse time, and the calculation data is saved at each step. In the well-known Dijkstra and Bellman-Ford algorithms, these operations are not performed, and the use of dynamic programming to evaluate the shortest paths is effective only when solving problems of low dimensions.

Formal description. The basis of the network routing method under consideration is the Floyd algorithm, considered in detail in (Saharov, Chertkov & Ochina, 2019). Therefore, we will consider the step-by-step implementation of the routing procedure of the graph in the sequence in which it is presented. Before considering it using a specific example, we show the possibility of algorithmically constructing the calculated graph using sparse adjacent matrices and its subsequent digitization. Sparse adjacent matrices, along with zero elements, contain elements with unit values, which are used to connect the vertices of a graph and create the corresponding figures.

In MATLAB code, we constructed a graph containing $n = 11$ vertices, given a sparse adjacent matrix of the following form:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

For a complete description of the graph, in addition to specifying the adjacent matrix, it is necessary to indicate the vector **XY** of the coordinates of the vertices of the constructed graph (with matrix **A**)

XY = [0 0; 1 0; 1 0.6; 1 –0.6; 2 0; 2 0.3; 2 –0.3; 3 0; 3 0.6; 3 –0.6; 4 0].

Then, using the graphic function `gplot` with the syntax `gplot (A,XY,'-o')` the third argument of which indicates the type of connection and the shape of the vertices, we obtain the given graph configuration. However, this graph does not digitize the vertices and weights of the connecting edges; therefore, at the next stage, we also introduced the vectors of the weights of the graph edges

E = [4 7 8 –9 –7 6 12 13 7 9 3 3 10 10 8 15 14 12 11 –3 –4 20 18 17]

and coordinates of the points of digitization of the weights of its edges **xy**

xy = [0.5 0; 0.5 0.3; 0.5 –0.3; 1 0.3; 1 –0.3; 1.5 0; 1.5 0.2; 1.5 –0.2; 1.5 0.5; 1.5 –0.5; 2 0.15; 2 –0.15; 2 0.6; 2 –0.6; 2.5 0; 2.5 0.2; 2.5 –0.2; 2.5 0.4; 2.5 –0.4; 3 0.3; 3 –0.3; 3.5 0; 3.5 0.3; 3.5 –0.3]

```
for j=1:n,text(XY(j,1),XY(j,2),
  int2str(j));
end.
```

We built the desired graph (Figure 1), subject; then, using a special `for-end` loop, we constructed the form to obtain optimal routing.

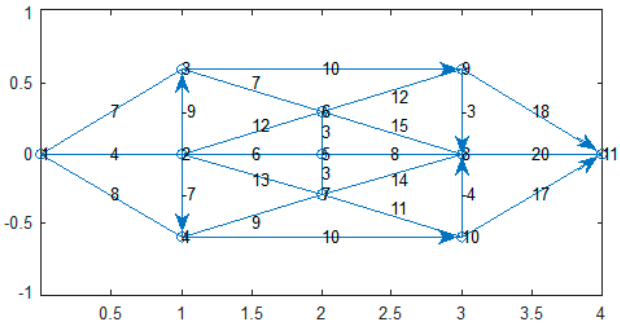

Figure 1. Source-weighted graph

The route of a moving object (vessel) with an estimate of the smallest sum of deviations in fuel consumption from a given average value in all sections of the track was taken as the optimal route.

As can be seen from Figure 1, edges (2,3), (2,4), (3,9), (4,10), (9,8), (10,8), as well as (9,11) and (10, 11) were oriented, while all other ribs allowed movement in both directions. The operation of the algorithm was considered in a step-by-step mode using fragments of a program compiled in MATLAB.

Step 0. The initial matrixes of distances (weights) $\mathbf{D}_0$ (Table 1) and the sequence of nodes $\mathbf{S}_0$ (Table 2) were constructed directly from the given network structure.

Table 1. The initial matrix of weights of the edges of the weighted graph

	1	2	3	4	5	6	7	8	9	10	11
1	0	4	7	8	100	100	100	100	100	100	100
2	4	0	–9	–7	6	12	13	100	100	100	100
3	7	100	0	100	100	7	100	100	10	100	100
4	8	100	100	0	100	100	9	100	100	10	100
5	100	6	100	100	0	3	3	5	100	100	100
6	100	12	7	100	3	0	100	15	12	100	100
7	100	13	100	9	3	100	0	14	100	11	100
8	100	100	100	100	5	15	14	0	100	100	20
9	100	100	100	100	100	12	100	–3	0	100	18
10	100	100	100	100	100	100	11	–4	100	0	17
11	100	100	100	100	100	100	100	20	100	100	0

Table 2. The initial matrix of the sequence of nodes S_0

	1	2	3	4	5	6	7	8	9	10	11
1	0	2	3	4	5	6	7	8	9	10	11
2	1	0	3	4	5	6	7	8	9	10	11
3	1	2	0	4	5	6	7	8	9	10	11
4	1	2	3	0	5	6	7	8	9	10	11
5	1	2	3	4	0	6	7	8	9	10	11
6	1	2	3	4	5	0	7	8	9	10	11
7	1	2	3	4	5	6	0	8	9	10	11
8	1	2	3	4	5	6	7	0	9	10	11
9	1	2	3	4	5	6	7	8	0	10	11
10	1	2	3	4	5	6	7	8	9	0	11
11	1	2	3	4	5	6	7	8	9	10	0

To form the matrix $\mathbf{S_0}$, the vertices of the graph were initialized according to the algorithm. A fragment of the graph vertex initialization program compiled in MATLAB codes is given below:

```
I=1;
while I<=n
for J=1:n
if J==I
S(I,J)=0;
else
S(I,J)=J;
end
end
I=I+1;
end
```

Step 1. At the first step $k = 1$. This means that the leading row and column will be the first row and the first column, highlighted on the matrix $\mathbf{D_0}$ with a light gray background. The frame and the darker background will highlight the elements whose values can be improved using the *triangular operator*:

- the third line includes the elements d_{32} and d_{34}, thus $d_{32} > d_{31} + d_{12}$ and $d_{34} > d_{31} + d_{14}$;
- the fourth line includes the elements d_{42} and d_{43} thus $d_{42} > d_{41} + d_{11}$ and $d_{43} > d_{41} + d_{13}$.

At the first step of the calculations in the cycle, we formed the matrices $\mathbf{D_1}$ and $\mathbf{S_1}$ based on the matrices $\mathbf{D_0}$ and $\mathbf{S_0}$ in the following order:

1) replace the value $d_{32} = 100$ with the sum of the values $d_{31} + d_{13} = 7 + 4 = 11$; in the matrix $\mathbf{S_1}$, a new value of element S_1 is set equal to 1;
2) replace the value $d_{34} = 100$ with the sum of the values $d_{31} + d_{14} = 7 + 8 = 15$; in the matrix $\mathbf{S_1}$, a new value of element s_3 is set equal to 1;
3) replace the value $d_{42} = 100$ with the sum of the values $d_{41} + d_{12} = 8 + 4 = 12$; in the matrix $\mathbf{S_1}$, a new value of element s_{42} is set equal to 1;
4) replace the value $d_{43} = 100$ with the sum of the values $d_{41} + d_{13} = 8 + 7 = 15$; in the matrix $\mathbf{S_1}$, a new value of element s_{43} is set equal to 1;

As a result, the matrices $\mathbf{D_1}$ and $\mathbf{S_1}$ will take the forms shown in Tables 3 and 4.

Table 3. Matrix D_1

	1	2	3	4	5	6	7	8	9	10	11
1	0	4	7	8	100	100	100	100	100	100	100
2	4	0	–9	–7	6	12	13	100	100	100	100
3	7	11	0	15	100	7	100	100	10	100	100
4	8	12	15	0	100	100	9	100	100	10	100
5	100	6	100	100	0	3	3	5	100	100	100
6	100	12	7	100	3	0	100	15	12	100	100
7	100	13	100	9	3	100	0	14	100	11	100
8	100	100	100	100	5	15	14	0	100	100	20
9	100	100	100	100	100	12	100	–3	0	100	18
10	100	100	100	100	100	100	11	–4	100	0	17
11	100	100	100	100	100	100	100	20	100	100	0

Table 4. Matrix S_1

	1	2	3	4	5	6	7	8	9	10	11
1	0	2	3	4	5	6	7	8	9	10	11
2	1	0	3	4	5	6	7	8	9	10	11
3	1	1	0	1	5	6	7	8	9	10	11
4	1	1	1	0	5	6	7	8	9	10	11
5	1	2	3	4	0	6	7	8	9	10	11
6	1	2	3	4	5	0	7	8	9	10	11
7	1	2	3	4	5	6	0	8	9	10	11
8	1	2	3	4	5	6	7	0	9	10	11
9	1	2	3	4	5	6	7	8	0	10	11
10	1	2	3	4	5	6	7	8	9	0	11
11	1	2	3	4	5	6	7	8	9	10	0

Step 2. At this step, $k = 2$. The second row and second column on a light-gray background become the leading row and column in matrix $\mathbf{D_1}$. The frame and the darker background highlight the elements whose values need to be improved using the *triangular operator*:

- the first line includes elements d_{13}, d_{14}, d_{15}, d_{16} and d_{17}, thus $d_{13} > d_{12} + d_{23}$, $d_{14} > d_{12} + d_{24}$, $d_{15} > d_{12} + d_{25}$, $d_{16} > d_{12} + d_{26}$ and $d_{17} > d_{12} + d_{27}$;
- the third line includes elements d_{34}, d_{35} and d_{37}, thus $d_{34} > d_{32} + d_{24}$, $d_{35} > d_{32} + d_{25}$ and $d_{37} > d_{32} + d_{27}$;
- the fourth line includes elements d_{43}, d_{45} and d_{46}, thus $d_{43} > d_{42} + d_{23}$, $d_{45} > d_{42} + d_{25}$ and $d_{46} > d_{42} + d_{26}$;
- the fifth line includes elements d_{51}, d_{53} and d_{54}, because $d_{51} > d_{52} + d_{21}$, $d_{53} > d_{52} + d_{23}$ and $d_{54} > d_{52} + d_{24}$;

- the sixth line includes elements d_{61}, d_{63}, d_{64} and d_{67}, thus $d_{61} > d_{62} + d_{21}$, $d_{63} > d_{62} + d_{23}$, $d_{64} > d_{62} + d_{24}$ and $d_{67} > d_{62} + d_{27}$;
- the seventh line includes elements d_{71}, d_{73}, d_{74} and d_{76}, thus $d_{71} > d_{72} + d_{21}$, $d_{73} > d_{72} + d_{23}$, $d_{74} > d_{72} + d_{24}$ and $d_{76} > d_{72} + d_{26}$;
- the eighth line includes elements d_{83} and d_{84}, thus $d_{83} > d_{82} + d_{23}$ and $d_{84} > d_{82} + d_{24}$;
- the ninth line includes elements d_{93} and d_{94}, thus $d_{93} > d_{92} + d_{23}$ and $d_{94} > d_{92} + d_{24}$;
- the tenth line includes elements $d_{10,3}$ and $d_{10,4}$, thus $d_{10,3} > d_{10,2} + d_{23}$ and $d_{10,4} > d_{10,2} + d_{24}$;
- the eleventh line includes elements $d_{11,3}$ and $d_{11,4}$, thus $d_{11,3} > d_{11,2} + d_{23}$ and $d_{11,4} > d_{11,2} + d_{24}$.

During the second step of the calculations, the values of the elements of the matrices $\mathbf{D_2}$ and $\mathbf{S_2}$ are formed during in the cycle and are obtained by replacing the elements of the matrices $\mathbf{D_1}$ and $\mathbf{S_1}$, whose new values require improvement:

1) in the first line, elements d_{13}, d_{14}, d_{15}, d_{16} and d_{17} receive new values equal to $d_{13} = d_{12} + d_{23} = -5$, $d_{14} = d_{12} + d_{24} = -3$, $d_{15} = d_{12} + d_{25} = 10$, $d_{16} = d_{12} + d_{26} = 16$, $d_{17} = d_{12} + d_{27} = 17$, and new element values are also set to $s_{13} = s_{14} = s_{15} = s_{16} = s_{17} = 2$;
2) in the third line, elements d_{34}, d_{35} and d_{37} receive new values equal to $d_{34} = d_{32} + d_{24} = 4$, $d_{35} = d_{32} + d_{25} = 17$, $d_{37} = d_{32} + d_{27} = 24$, and element values are also set to $s_{34} = s_{35} = s_{37} = 2$;
3) in the fourth line, elements d_{43}, d_{45} and d_{46} receive new values equal to $d_{43} = d_{42} + d_{23} = 3$, $d_{45} = d_{42} + d_{25} = 18$, $d_{46} = d_{42} + d_{26} = 24$, and element values are also set to $s_{43} = s_{45} = s_{46} = 2$;
4) in the fifth line, elements d_{51}, d_{53} and d_{54} receive new values equal to $d_{51} = d_{52} + d_{21} = 10$, $d_{53} = d_{52} + d_{23} = -3$, $d_{54} = d_{52} + d_{24} = -1$, and element values are also set to $s_{51} = s_{53} = s_{54} = 2$;
5) in the sixth line, elements d_{61}, d_{63} and d_{64} receive new values equal to $d_{61} = d_{62} + d_{21} = 16$, $d_{73} = d_{72} + d_{23} = 3$, $d_{74} = d_{72} + d_{24} = 5$, and element values are also set to $s_{71} = s_{73} = s_{74} = 2$;
6) in the seventh line, elements d_{71}, d_{73} and d_{74} receive new values equal to $d_{71} = d_{72} + d_{21} = 17$, $d_{53} = d_{52} + d_{23} = 4$, $d_{54} = d_{52} + d_{24} = 6$, and element values are also set to $s_{51} = s_{53} = s_{54} = 2$;
7) in the eighth line, elements d_{83} and d_{84} receive get new values equal to $d_{83} = d_{82} + d_{23} = 91$, $d_{84} = d_{82} + d_{24} = 93$, and element values are also set to $s_{83} = s_{84} = 2$.

Similarly, the same values will be obtained by the elements of the third column $d_{93} = d_{10,3} = d_{11,3} = 91$, as well as the fourth column $d_{94} = d_{10,4} = d_{11,4} = 93$ in the ninth, tenth, and eleventh rows of matrix $\mathbf{D_2}$ (Table 5). The corresponding elements of matrix $\mathbf{S_2}$ will also be equal to 2 (Table 6).

As a result, the matrices $\mathbf{D_2}$ and $\mathbf{S_2}$ will take the forms shown in Tables 5 and 6.

Table 5. Matrix D_2

	1	2	3	4	5	6	7	8	9	10	11
1	0	4	−5	−3	10	16	17	100	100	100	100
2	4	0	−9	−7	6	12	13	100	100	100	100
3	7	11	0	4	17	7	24	100	10	100	100
4	8	12	3	0	18	24	9	100	100	10	100
5	10	6	−3	−1	0	3	3	5	100	100	100
6	16	12	3	5	3	0	25	15	12	100	100
7	17	13	4	6	3	25	0	14	100	11	100
8	100	100	91	93	5	15	14	0	100	100	20
9	100	100	91	93	100	12	100	−3	0	100	18
10	100	100	91	93	100	100	11	−4	100	0	17
11	100	100	91	93	100	100	100	20	100	100	0

Table 6. Matrix S_2

	1	2	3	4	5	6	7	8	9	10	11
1	0	2	2	2	2	2	2	8	9	10	11
2	1	0	3	4	5	6	7	8	9	10	11
3	1	1	0	2	2	6	2	8	9	10	11
4	1	1	2	0	2	2	7	8	9	10	11
5	2	2	2	2	0	6	7	8	9	10	11
6	2	2	2	2	5	0	2	8	9	10	11
7	2	2	2	2	5	2	0	8	9	10	11
8	1	2	2	2	5	6	7	0	9	10	11
9	1	2	2	2	5	6	7	8	0	10	11
10	1	2	2	2	5	6	7	8	9	0	11
11	1	2	2	2	5	6	7	8	9	10	0

Step 3. At this step, $k = 3$. The leading row and column are the third row and third column, highlighted in matrix $\mathbf{D_2}$ with a light gray background. The frame and the darker background highlight the elements whose values can be improved using the *triangular operator*, whose method of application was discussed in the previous step. In the third step of the calculation, the matrices $\mathbf{D_2}$ and $\mathbf{S_2}$ are formed based on matrices $\mathbf{D_3}$ and $\mathbf{S_3}$ (Tables 7, 8).

The MATLAB code below is a fragment of a program for evaluating the numerical values of a triangular operator:

```
k=1
while k<=n
for i=1:n
for j=1:n
if D(i,j)>D(i,k)+D(k,j)
D(i,j)=D(i,k)+D(k,j);
S(i,j)=k;
```

Table 7. Matrix D_3

	1	2	3	4	5	6	7	8	9	10	11
1	0	4	−5	−3	10	2	17	95	5	95	95
2	−2	0	−9	−7	6	−2	13	91	1	91	91
3	7	11	0	4	17	7	24	100	10	100	100
4	8	12	3	0	18	10	9	100	13	10	100
5	4	6	−3	−1	0	3	3	5	7	97	97
6	10	12	3	5	3	0	25	15	12	100	100
7	11	13	4	6	3	11	0	14	14	11	100
8	98	100	91	93	5	15	14	0	100	100	20
9	98	100	91	93	100	12	100	−3	0	100	18
10	98	100	91	93	100	98	11	−4	100	0	17
11	98	100	91	93	100	98	100	20	100	100	0

Table 8. Matrix S_3

	1	2	3	4	5	6	7	8	9	10	11
1	0	2	2	2	2	3	2	3	3	3	3
2	3	0	3	4	5	3	7	3	3	3	3
3	1	1	0	2	2	6	2	8	9	10	11
4	1	1	2	0	2	3	7	8	3	10	11
5	3	2	2	2	0	6	7	8	3	3	3
6	3	2	2	2	5	0	2	8	9	10	11
7	3	2	2	4	5	3	0	8	3	10	11
8	3	2	2	2	5	6	7	0	9	10	11
9	3	2	2	2	5	6	7	8	0	10	11
10	3	2	2	2	5	3	7	8	9	0	11
11	3	2	2	2	5	3	7	8	9	10	0

Table 9. Matrix D_{10}

	1	2	3	4	5	6	7	8	9	10	11
1	0	4	−5	−3	5	2	6	10	5	7	22
2	−2	0	−9	−7	1	−2	2	−2	1	3	18
3	7	11	0	4	10	7	13	7	10	14	27
4	8	12	3	0	11	10	9	6	13	10	26
5	4	6	−3	−1	0	3	3	4	7	9	24
6	7	9	0	2	3	0	6	7	10	12	27
7	7	9	0	2	3	6	0	7	10	11	27
8	9	11	2	4	5	8	8	0	12	14	20
9	6	8	−1	1	2	5	5	−3	0	11	17
10	5	7	−2	0	1	4	4	−4	8	0	16
11	29	31	22	24	25	28	28	20	32	34	0

Table 10. Matrix S_{10}

	1	2	3	4	5	6	7	8	9	10	11
1	0	2	2	2	6	3	4	9	3	4	9
2	3	0	3	4	6	3	4	9	3	4	9
3	1	1	0	2	6	6	4	9	9	4	9
4	1	1	2	0	10	3	7	10	3	10	10
5	3	2	2	2	0	6	7	9	3	4	9
6	5	5	5	5	5	0	5	9	5	5	9
7	5	5	5	5	5	5	0	9	5	10	9
8	5	5	5	5	5	5	5	0	5	5	11
9	8	8	8	8	8	8	8	8	0	8	8
10	8	8	8	8	8	8	8	8	8	0	8
11	8	8	8	8	8	8	8	8	8	8	0

```
end
end
end
D
S
k=k+1
end
```

Similarly, the calculations are performed in the fourth and subsequent steps. At the tenth step (k = 10), the calculations are completed.

Results

The final matrices $\mathbf{D_{10}}$ and $\mathbf{S_{10}}$ (Tables 9 and 10) contain all the information necessary to determine the shortest paths between any two network nodes. Recall that, in relation to a ship, the shortest distance in the presence of edges with negative weights in the graph is the sequence that passes the nodes with the minimum sum of weights. Moreover, the weight of each rib may deviate from the normalized average fuel consumption per trip per mile of travel. For example, the shortest distance between nodes 1 and 11 is 22.

A route segment (i, j) consists of an edge (i, j), only if $s_{ij} = j$. The sequence of nodes of the shortest path from the first vertex to the 11th vertex was found, i.e., route 1→11 with a distance of $d_{1,11} = 22$ (Table 9). Turning to the matrix of the sequence of traversed nodes (Table 10), we found that $s_{1,11} = 9$ ($s_{1,11} \neq 11$). Nodes 1 and 11 were not connected by one edge; therefore, we determined the intermediate node. As can be seen from Table 10, the eighth node serves as the first intermediate node, since $s_{8,11} = 11$. Thus, the shortest route between nodes 1 and 11 takes the form:

$$1 \rightarrow 8 \rightarrow 11.$$

As seen in Table 10, the second intermediate node may be three nodes: the ninth, tenth, and eleventh, since for them $s_{9,8} = 8$, $s_{10,8} = 8$, and $s_{11,8} = 8$. According to the algorithm and using Table 10, we obtain two possible routes:

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 9 \rightarrow 8 \rightarrow 11 \text{ and } 1 \rightarrow 2 \rightarrow 4 \rightarrow 10 \rightarrow 8 \rightarrow 11.$$

Of these, option 1→2→3→9→8→11 is the shortest route, with an estimate of $d_{1,11} = 22$. The assessment is made by taking into account the negative "weights" of two ribs.

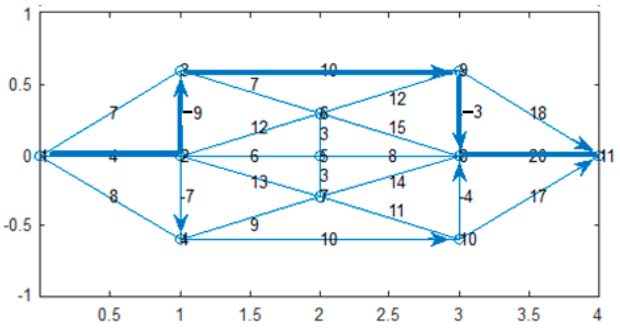

Figure 2. The shortest path from peak 1 to peak 11

Discussion

As shown by the step-by-step implementation of the algorithm, to correctly estimate the lengths of all shortest paths in the graph, it is necessary to perform n–1 iterations. If the graph contains cycles with a negative weight, then the Floyd algorithm is not formally applicable to such a graph. The proposed algorithm works correctly with negative weights, since when processing such a graph in the diagonal of the matrix of shortest paths, negative numbers were analyzed. Their appearance was associated with the processing of "negative" cycles.

Conclusions

1. Based on the Floyd algorithm, a universal computer model was developed that allows the automation of the operations for determining and constructing the shortest routes between any pair of transit nodes in transport and telecommunication networks. The model is suitable for rapidly assessing the energy efficiency of the selected routes for the movement of ships and other moving objects. It takes into account the variable sailing conditions during a voyage.
2. A recursive procedure was proposed to determine the sequence of edges comprising the shortest path using the functions in the Optimization Toolbox of MATLAB.
3. The correctness and effectiveness of the proposed model and its software implementation were confirmed using a specific example of the algorithmic design and digitization of a transport network graph with a complex configuration and the presence of edges with negative weights.
4. The advantages of the proposed procedure for automating the calculation of shortest paths based on the Floyd algorithm are the simplicity of the software implementation and the ability to use the universal tools of the MATLAB computing environment.

References

1. Ariefiew, I. (2012) *Forecasting and control object of management in the environment of system pert (the nod of integrated characteristics)*. Maritime University of Szczecin, Digital Library.
2. Bellman, R. (1958) On a routing problem. *Quarterly of Applied Mathematics* 16, 1, pp. 87–90.
3. Castillo, O., Trujillo, L. & Melin, P. (2007) Multiple Objective Genetic Algorithms for Path-planning Optimization in Autonomous Mobile Robots. *Soft Computing* 3 (11), pp. 269–279.
4. Chertkov, A.A. (2015) An iterative algorithm for choosing the optimal strategy of group interaction for moving objects. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota Imeni Admirala S.O. Makarova* 4 (32) (in Russian), pp. 207−215.
5. Chertkov, A.A. (2017) Automation selection shortcuts routes of ships on the basis of modified Bellman. Ford. Algorithm. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota Imeni Admirala S.O. Makarova* 9, 5, pp. 1113–1122. DOI: 10.21821/2309-5180-2017-9-5-1113-1122 (in Russian).
6. Chertkov, A.A., Vardomskaya, A.A. & Dmitriev, A.A. (2015a) A Recursive method of optimization of the logistic ways by means of MATLAB. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota Imeni Admirala S.O. Makarova* 6(34), pp. 196−204 (in Russian).
7. Chertkov, A.A., Vardomskaya, A.A. & Dmitriev, A.A. (2015b) Automation to define critical way in logistical system. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota Imeni Admirala S.O. Makarova* 5 (33), pp. 194–200 (in Russian). DOI: 10.21821/2309-5180-2015-7-5-194-200.
8. Fedorenko, K.V. & Olovyannikov, A.L. (2017) Research of the main parameters of the genetic algorithm for the problem of searching the optimal route. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota Imeni Admirala S.O. Makarova* 9, 4, pp. 714–723 (in Russian). DOI: 10.21821/2309-5180-2017-9-4-714-723.
9. Fonseca, C.M. & Fleming, P.J. (1993) Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: Forrest, S. (ed.) *Genetic Algorithms: Proceedings of the Fifth International Conference*, San Mateo, CA: Morgan Kaufman, pp. 416–423.
10. Kirsanov, M.N. (2016) Analysis of algorithms for the selection of optimal routes the groups vessels. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota Imeni Admirala S.O. Makarova* 2 (36), pp. 183–190 (in Russian). DOI: 10.21821/2309-5180-2016-8-2-183-190.
11. Saharov, V.V., Chertkov, A.A. & Dmitriev, A.A. (2016) Ensuring minimum transport work algorithm for goods traffic. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota Imeni Admirala S.O. Makarova* 1, 35, pp. 180–188 (in Russian). DOI: 10.21821/2309-5180-2016-8-1-180-188.

12. Saharov, V.V., Chertkov, A.A. & Ochina, L.B. (2019) Routing the networks with negative weights of links in the MATLAB optimization package. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota Imeni Admirala S.O. Makarova* 11, 2, pp. 230–242 (in Russian). DOI: 10.21821/2309-5180-2019-11-2230-242.
13. Saharov, V.V., Chertkov, A.A. & Tormashev, D.S. (2014) Algoritm optimalnogo planirovanija gruppovogo vzaimodejstvija robotov. *Morskoj Vestnik* 4 (52) (214): pp. 119–122.
14. Saharov, V.V., Sikarev, I.A. & Chertkov, A.A. (2018) Automating search optimal routes and goods flows in transport networks means the integer linear programming. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota Imeni Admirala S.O. Makarova* 10, 3, pp. 647–657 (in Russian). DOI: 10.21821/23095180-2018-10-3-647-657.
15. Sazonov, A.E. & Derjabin, V.V. (2013) Forecasting to paths of the motion ship with the help of neyronnoy network. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota Imeni Admirala S.O. Makarova* 3, 22, pp. 6–13 (in Russian).