Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 54 (2021) 1–11

Transportation Research Procedia

www.elsevier.com/locate/procedia

TransSiberia 2020 Conference

Automatization Search for the Shortest Routes in the Transport Network Using the Floyd-warshell Algorithm

Vladimir Sakharov[a], Sergei Chernyi[a]*, Sergey Saburov[a], Alexandr Chertkov[a]

[a]*Admiral Makarov State University of Maritime and Inland Shipping, 198035, Saint-Petersburg, Russia*

Abstract

The operational problem of determining the shortest routes for a group of vessels that have received the task of achieving hypothetical target with certain coordinates located in a confined space is being solved. When controlling the process of vessels entering a desired area, time is an important indicator. The practical implementation of the modified Floyd-Warshall algorithm is demonstrated by calculating a network model with a complex topology, using an iterative procedure by means of a program in MATLAB codes. The computer model implemented on its basis is simple and compact. Unlike existing models, the suggested model allows one to eliminate the restrictions associated with the presence of negative weights and cycles in the network, to automate the calculation of the shortest paths at the nodes by means of the computer technology implemented in MATLAB.

Keywords: Route; Target; Algorithm; Optimization; Procedure; Transport.

1. Introduction

The introduction should briefly place the study in a broad context and highlight why it is important. It should define the purpose of the work and its significance. The current state of the research field should be reviewed carefully and key publications cited. Please highlight controversial and diverging hypotheses when necessary. Finally, briefly mention the main aim of the work and highlight the principal conclusions. As far as possible, please keep the introduction comprehensible to scientists outside your particular field of research. References should be numbered in order of appearance and indicated by a numeral or numerals in square brackets, e.g., (Bellman, 1958) or (Chertkov, 2017; Saharov, 2014). See the end of the document for further details on references.

* Corresponding author. Tel.: +7-911-234-0101.
E-mail address: sergiiblack@gmail.com

10.1016/j.trpro.2021.02.041

The selection of the shortest route for the traffic of vessels and other mobile objects determines their effectiveness and economic efficiency. Such problems for individual vessels, or in the terms of robotics - agents, are solved by methods of studying operations on graphs, in particular, on the basis of the well-known Dijkstra algorithms, considered in detail in (Chertkov, 2017), Bellman-Ford in (Bellman, 1958) and (Saharov, 2014), Floyd et al. in (Saharov, 2018). Artificial intelligence methods based on neural networks are not less effective. They allow one to calculate conditions changing over time (Sazonov, 2013), and genetic algorithms (Castilo, 2007; Fedorenko, 2017; Fonseca, 1993). More complex tasks for a group of agents, including tasks on the collective coverage of operating spaces, are solved using probabilistic methods and coordination of agents on pre-allocated subspaces. The scope of solutions for such optimization problems is very wide namely from the choice of the routes of ships (Kirsanov, 2016) and mobile robots in the service system and military sphere (Dedkov, 2013; Kirsanov, 2013), to three-dimensional problems of planning the paths of underwater vehicles (Kirsanov, 2013; Sokolov, 2020).

Within the theory of study of operations, a large number of practical problems are considered, which can be formulated as problems of finding the shortest path. They may be solved by means of network models, in particular:

- the formation of safe routes in the traffic separation zones with different directions, converging in a small location or at clusters, which may be the approaches to ports, pilot stations, areas of establishing approach buoys or beacons, entrances to fairways, canals, etc.
- designing a pipeline connecting offshore boreholes to a shore-based collector station;
- connection of all network nodes (cable, road, pipeline, communication) using the paths of the smallest length;
- designing the most reliable traffic routes in a network of roads (waterways);
- formation of ship routes, optimal in transit time;
- replacement of equipment when updating the fleet for a certain period;
- determining the maximum throughput of the pipeline;
- development of the scheme for the transportation of oil and oil products from production sites to oil refineries and consumers with a minimum cost of transportation;
- drawing up a time schedule for construction work (evaluating the start and final dates of individual stages of operations);
- scheduling using difference constraints.

The solution of these problems requires the use of various network optimization algorithms.

2. Methods and Materials

Among many problems of finding the shortest routes, we consider the task of planning ship traffic routes in an extensive system of targets that simulate a real situation. It is assumed that the locations of the targets (location coordinates) are considered known. The task is to select routes for which the passage time will be minimal. The operating time is evaluated by the time T_{max} to reach the most remote target (destination). The speed of movement of the agents is assumed to be the same; we neglect the time of "processing" targets. Therefore, if the agent's speed is taken as a unity, then the task execution time can be roughly considered equal to the length of the entire route.

The minimum route problem is solved sequentially for each agent. An arbitrary agent is selected and the target closest to it is determined. Then the agent is placed in a new position, and the task is repeated again until the agent passes N targets assigned to it. Then these targets are deleted from the list of targets, and the task is being solved for another agent. Obviously, this algorithm belongs to locally optimal (or "greedy") algorithms and tends to fall in a local minimum. If one does not resort to neural network or genetic algorithms, then the only way to avoid falling in a local minimum is to completely enumerate all variants of agent sequences.

In this paper, the Floyd-Warshall algorithm is presented, which, unlike the Dijkstra and Bellman-Ford algorithms, which allow solving the shortest path problem between a certain fixed node and any other network node, is more general and provides the simultaneous determination of the shortest paths between any two network nodes. Other advantages include ease of implementation, work on weighted graphs not only with positive but also with negative weights of edges without negative cycles, as well as the possibility of visual reconstruction of simultaneously shortest paths between any pairs of nodes immediately after the end of the iteration cycle.

The algorithm is named after two American scientists. The Floyd-Warshall algorithm is a dynamic algorithm for finding the shortest distances between all points of a weighted directed graph without cycles with negative weights.

In the Floyd-Warshall algorithm, in order to find the shortest paths between all points of a graph, upward dynamic programming rather than enumeration of all the capabilities are applied. According to the algorithm, all the subtasks that will later be required to solve the original problem are calculated in advance.

3. Formal Definition

Suppose one wants to find the lengths of the shortest paths between any two nodes of a weighted graph. Moreover, arc distances can be both positive and negative, but there should be no cycles with negative length. A theoretically absent arc in the graph is assigned a weight equal to infinity. However, in practice, they are limited to a weight equal to a finite (sufficiently large) number, which could be greater than the length of any path in this graph.

First, let us explain the content-related component of the Floyd-Warshall method in a simple graph. Let there be three nodes i, j and k, and the distances between them are given as well (Figure 1).

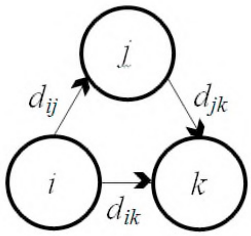

Fig. 1. Triangle Operator Attribution.

If the inequality $d_{jj}+d_{jk}<d_{ik}$ holds, then it is advisable to replace the path $i \rightarrow k$ with $i \rightarrow j \rightarrow k$. Such a replacement (hereinafter, we will arbitrarily call it a triangular operator) is carried out systematically in the process of implementing the Floyd-Warshell algorithm.

The Floyd-Warshall algorithm requires the following actions.

The initial step. Let us evaluate the initial distance matrix D_0 and the matrix of the sequence of nodes S_0. The diagonal elements of both matrices (see Tables 1 and 2) are marked with a «–» sign, which indicates that these elements are not involved in the calculations. Suppose k=1.

Table 1. Initial Distance Matrix.

		1	**2**	…	**j**	…	n
	1	—	$\mathbf{d_{12}}$	…	$\mathbf{d_{1j}}$	**...**	d_{1n}
D_0=	**2**	$\mathbf{d_{21}}$	—	…	$\mathbf{d_{2j}}$	**...**	$\mathbf{d_{2n}}$
		…	…	…	…	…	…
	i	$\mathbf{d_{i1}}$	$\mathbf{d_{i2}}$	…	$\mathbf{d_{ij}}$	**...**	d_{in}
		…	…	…	…	…	…
	n	$\mathbf{d_{n1}}$	d_{n2}	…	d_{nj}	**...**	—

Table 2. Matrix of the knot sequence.

		1	**2**	…	**j**	…	n
	1	—	2	…	**J**	…	n
S_0=	**2**	**1**	—	…	j	…	**n**
		…	…	…	…	…	…
	i	**1**	2	…	**J**	…	n
		…	…	…	…	…	…
	n	**1**	2	…	j	…	—

The main step k. Set row k and column k to be the leading row and leading column. Let us consider the feature of applying the triangular operator to all elements d_{ij} of the matrix D_{k-1}. If the inequality holds

$$\boldsymbol{d_{ik}+d_{kj}<d_{ij}(i\neq k,j\neq k} \text{ and } i\neq j), \tag{1}$$

then do the following actions:

- create the matrix D_k by replacing the element d_{jj} with the sum $d_{i,k}+d_{kj}$ in the matrix D_{k-1};
- form the matrix S_k by replacing element $s_{i,j}$ with k in the matrix S_{k-1}. We set k=k+1 and repeat step k.

In step k=n the leading row and column are the last row in the matrix D_n and the last column in the matrix S_n. We do not perform any actions at this step. The calculations are finished.

Example. For the network shown in Figure 2, we find the shortest paths between any two network nodes.

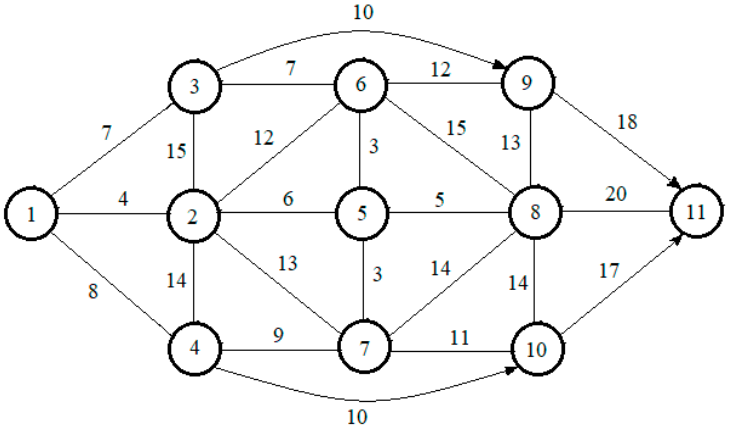

Fig. 2. Initial weighted graph.

As it is shown in Fig. 2, the edges (3,7), (5,2), (5,9) and (12,14) are oriented, and all other edges allow movement in both directions.

Let us consider in details the operation of the algorithm in a step-by-step mode using fragments of a program compiled in MATLAB codes.

Step 0. The initial matrixes of distances (weights) D_0 (Table 3) and the sequence of nodes S_0 (Table 4) are constructed directly from the specified network structure.

Table 3. Initial weight matrix of the weighted graph edges.

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

Table 4. Initial matrix of the node sequence 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 S_0, the graph points are initialized according to the algorithm. A fragment of the graph point initialization program compiled in MATLAB codes is given below:

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

4. Algorithms

Step 1. In this step k=1. This means that the leading row and column will be the first row and the first column, highlighted with a weak hue in the D_0 matrix. The frame and the darker color highlight the elements whose values can be improved using the triangular operator:

- In the second row - elements d_{23} and d_{24}, as $d_{23}>d_{21}+d_{13}$ and $d_{24}>d_{21}+d_{14}$;
- In the third row - elements d_{32} and d_{34}, as $d_{32}>d_{31}+d_{12}$ and $d_{34}>d_{31}+d_{14}$;
- In the fourth row - elements d_{42} and d_{43}, as $d_{43}>d_{41}+d_{14}$ and $d_{43}>d_{41}+d_{13}$.

At the first step of the calculations in the cycle (Zhilenkov, 2020), we form the matrices D_1 and S_1 based on the matrices D_0 and S_0 in the following order:

- replace $d_{23}=15$ (from matrix D0) with the sum of values $d_{21}+d_{13}=4+7=11$ and set the new value of the element $s_{23}=1$, equal to the step k;

- replace d_{24}=14 with the sum of values $d_{21}+d_{14}$=4+8=12, and in matrix S_1 we set the new value of the element s_{24}, equal 1;
- replace d_{32}=15 with the sum of values $d_{31}+d_{13}$=7+4=11, and in matrix S_1 set the new value of the element s_{32}, equal 1;
- replace d_{34}=100 with the sum of values $d_{31}+d_{14}$=7+8=15, and in matrix S_1 set the new value of the element s_{34}, equal 1;
- replace d_{42}=14 with the sum of values $d_{41}+d_{12}$=8+4=12, and in matrix S_1 set the new value of the element s_{42}, equal 1;
- replace d_{43}=100 with the sum of values $d_{41}+d_{13}$=8+7=15, and in matrix S_1 set the new value of the element s_{43}, equal 1.

As a result, matrices D_1 and S_1 take the form shown in Tables 5 and 6.

Table 5. 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	11	12	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	13	14	20
9	100	100	100	100	100	12	100	13	0	100	18
10	100	100	100	100	100	100	11	14	100	0	17
11	100	100	100	100	100	100	100	20	100	100	0

Table 6. 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	1	1	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. In this step k=2. This means that the leading row and column will be the second row and the second column, highlighted with a grey hue in the D_1 matrix. The frame and the darker color highlight the elements whose values can be improved using the triangular operator:

- in the first row - elements d_{15}, d_{16} and d_{17}, as $d_{15}>d_{12}+d_{25}$, $d_{16}>d_{12}+d_{26}$ and $d_{17}>d_{12}+d_{27}$;
- in the third row - elements d_{35} and d_{37}, as $d_{35}>d_{32}+d_{25}$ and $d_{37}>d_{32}+d_{27}$;
- in the fourth row - elements d_{45} and d_{46}, as $d_{45}>d_{42}+d_{25}$ and $d_{46}>d_{42}+d_{26}$;
- in the fifth row - elements d_{51}, d_{53} and d_{54}, as $d_{51}>d_{52}+d_{21}$, $d_{53}>d_{52}+d_{23}$ and $d_{54}>d_{52}+d_{24}$;
- in the sixth row - elements d_{61}, d_{64} and d_{67}, as $d_{61}>d_{62}+d_{21}$, $d_{64}>d_{62}+d_{26}$ and $d_{67}>d_{62}+d_{27}$;
- in the seventh row - elements d_{71}, d_{73} and d_{76}, as $d_{71}>d_{72}+d_{21}$, $d_{73}>d_{72}+d_{23}$ and $d_{76}>d_{72}+d_{26}$

At the second step of calculations in the cycle we form the values of the matrices D_2 and S_2 based on the matrices D_1 and S_1 in the following order:

- replace values d_{15}=100, d_{16}=100 and d_{17}=100 with sums of values respectively $d_{12}+d_{25}$ =10, $d_{12}+d_{26}$=16, $d_{12}+d_{27}$ =17 and set the values of the elements $s_{15}=s_{16}=s_{17}$=2;
- replace values d_{35}=100, d_{37}=100 with sums of values respectively $d_{32}+d_{25}$=17, $d_{32}+d_{27}$=24 и set the values of the elements $s_{35}=s_{37}$=2;
- replace values d_{45}=100, d_{46}=100 with sums of values respectively $d_{42}+d_{25}$=18, $d_{42}+d_{26}$=24 and set the values of the elements $s_{45}=s_{46}$=2;
- replace values d_{51}=100, d_{53}=100и d_{54}=100 with sums of values respectively $d_{52}+d_{21}$=10, $d_{52}+d_{23}$=17, $d_{52}+d_{24}$ =18 and set the values of the elements $s_{51}=s_{53}=s_{54}$=2;
- replace values d_{61}=100, d_{64}=100и d_{67}=100 with sums of values respectively $d_{62}+d_{21}$=16, $d_{62}+d_{24}$=24, $d_{62}+d_{27}$ =25 and set the values of the elements $s_{61}=s_{64}=s_{67}$=2;
- replace values d_{71}=100, d_{73}=100и d_{76}=100 with sums of values respectively $d_{72}+d_{21}$=17, $d_{72}+d_{23}$=24, $d_{72}+d_{26}$ =25 and set the values of the elements $s_{71}=s_{73}=s_{76}$=2.

Overall, matrices D_2 and S_2 acquire the form given in Tables 7 and 8.

Table 7. Matrix D_2.

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

Table 8. Matrix S_2.

	1	2	3	4	5	6	7	8	9	10	11
1	0	2	3	4	2	2	2	8	9	10	11
2	1	0	1	1	5	6	7	8	9	10	11
3	1	1	0	1	2	6	2	8	9	10	11
4	1	1	1	0	2	2	7	8	9	10	11
5	2	2	2	2	0	6	7	8	9	10	11
6	2	2	3	2	5	0	2	8	9	10	11
7	2	2	2	4	5	6	0	8	9	10	11
8	1	2	3	4	5	2	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 3. In this step k=3. This means that the leading row and column will be the third row and the third column, highlighted with a grey hue in the D_2 matrix. The frame and the darker color highlight the elements whose values can be improved using the triangular operator:

- in the first row – elements d_{16}, and d_{19}, as $d_{16}>d_{13}+d_{36}$ and $d_{19}>d_{13}+d_{39}$;
- in the second row – element d_{29}, as $d_{29}>d_{23}+d_{39}$;
- in the fourth row – elements d_{46} and d_{49}, as $d_{46}>d_{43}+d_{36}$ and $d_{49}>d_{43}+d_{39}$;

- in the fifth row – elements d_{59}, as $d_{59}>d_{53}+d_{39}$;
- in the sixth row – elements d_{61}, d_{64}, as $d_{61}>d_{63}+d_{31}$, $d_{64}>d_{63}+d_{34}$;
- in the seventh row – elements d_{79}, as $d_{79}>d_{73}+d_{39}$.

At the third step of calculations in the cycle we form the values of the matrices D_3 and S_3 based on the matrices D_2 and S_2 in the following order:

- replace values $d_{16}=16$, $d_{19}=100$ with the sum of values respectively $d_{13}+d_{36}=14$, $d_{13}+d_{39}=17$, and set values of the elements $s_{16}=s_{19}=3$;
- replace value $d_{29}=100$ with the sum of values $d_{23}+d_{39}=21$ and set the value of the element $s_{29}=3$;
- replace values $d_{46}=24$, $d_{49}=100$ with the sums of values respectively $d_{43}+d_{36}=22$, $d_{43}+d_{39}=25$ and set values of the elements $s_{46}=s_{49}=3$;
- replace value $d_{59}=100$ with the sum of values $d_{53}+d_{39}=27$ and set the value of the element $s_{59}=3$;
- replace values $d_{61}=16$, $d_{64}=24$ with the sums of values respectively $d_{63}+d_{31}=14$, $d_{63}+d_{34}=22$ and set the values of the elements $s_{61}=s_{64}=3$;
- replace values $d_{79}=100$ with the sum of values $d_{73}+d_{39}=34$ and set the value of the element $s_{79}=3$.

Thus, matrices D_3 and S_3 acquire the form shown in Tables 9 and 10.

Table 9. Matrix D_3.

	1	2	3	4	5	6	7	8	9	10	11
1	0	4	7	8	10	14	17	100	17	100	100
2	4	0	11	12	6	12	13	100	21	100	100
3	7	11	0	15	17	7	24	100	10	100	100
4	8	12	15	0	18	22	9	100	25	10	100
5	10	6	17	18	0	3	3	5	27	100	100
6	14	12	7	22	3	0	25	15	12	100	100
7	17	13	24	9	3	25	0	14	34	11	100
8	100	100	100	100	5	15	14	0	13	14	20
9	100	100	100	100	100	12	100	13	0	100	18
10	100	100	100	100	100	100	11	14	100	0	17
11	100	100	100	100	100	100	100	20	100	100	0

Table 10. Matrix S_3.

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

A fragment of the program for using the triangular operator compiled in MATLAB codes is given below:

```
% Iterations using the triangular operator
While k<=n
for I=1:n
for J=1:n
if J~=I && D(I,J)>D(I,k)+D(k,J)
```

```
        D(I,J)=D(I,k)+D(k,J);
        S(I,J)=k;
    end
    end
    end
    end
```

Similarly, calculations are performed in the fourth and further steps up to the 10th step (k=10), at which the cycle of calculations ends (Zhilenkov, 2020). At the step k=11, we do not perform any actions; the calculations are completed.

5. Results

The final matrices D_{10} and S_{10} (see Tables 11 and 12) contain all the information required to evaluate the shortest paths between any two network nodes. For example, the shortest distance between nodes 1 and 11 is 35.

Table 11. Matrix D_{10}.

	1	2	3	4	5	6	7	8	9	10	11
1	0	4	7	8	10	13	13	15	17	18	35
2	4	0	11	12	6	9	9	11	21	20	31
3	7	11	0	15	10	7	13	15	10	24	28
4	8	12	15	0	12	15	9	17	25	10	27
5	10	6	10	12	0	3	3	5	15	14	25
6	13	9	7	15	3	0	6	8	12	17	28
7	13	9	13	9	3	6	0	8	18	11	28
8	15	11	15	17	5	8	8	0	13	14	20
9	25	21	19	27	15	12	18	13	0	27	18
10	24	20	24	20	14	17	11	14	27	0	17
11	35	31	35	37	25	28	28	20	33	34	0

Table 12. Matrix S_{10}.

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

In order to restore the sequence of nodes on the relevant routes, recall that a segment of a route (i, j) consists of an edge (i, j) only if s_{ij}=j. Otherwise, nodes i and j are connected through at least one intermediate node. Let us demonstrate this by example.

We find the sequence of nodes for the shortest path from the first point to the tenth point, i.e., route $1\rightarrow10$ with a distance equal to $d_{1,10}$=18 (Table 11) As $s_{1,10}$=4 ($s_{1,10}\neq10$), then nodes 1 and 10 are not connected by one edge. We are looking for one of the intermediate nodes. As it can be seen from Table 12, three nodes can serve as the first intermediate node - the fourth, seventh and eighth nodes, since for them $s_{4,10}$=10, $s_{7,10}$=10 and $s_{8,10}$=10. Thus, the shortest routes between nodes 1 and 10 first will be as follows:

$$1\rightarrow4\rightarrow10,\ 1\rightarrow7\rightarrow10,\ 1\rightarrow8\rightarrow10. \quad (2)$$

Check which of these routes will be final. Nodes 1 and 4 are connected by one edge, since $s_{1,4}=4$. Therefore, the first of the shortest paths with a length of $d_{1,10}=18$, corresponding to the route $1\rightarrow4\rightarrow10$, will be final. Next, we consider the other two routes.

In the second route $s_{1,7}\neq7$. Consequently, nodes 1 and 7 in the path being evaluated are not connected by one edge. Similarly, in the third route $s_{1,8}\neq8$, therefore, the nodes 1 and 8 in the evaluated path are also not connected by one edge. In the future, we will not consider these last two routes, since they consist of more than one intermediate node and, as the comparative analysis shows, are not the shortest. Thus, the only and final shortest path from node 1 to node 10 of the network graph is route $1\rightarrow4\rightarrow10$ with a distance of $d_{1,10}=18$.

6. Discussion

As it can be seen from the step-by-step implementation of the algorithm, in order to make the correct calculation of the lengths of all shortest paths in the graph, only n–1 iterations are required. If the graph has cycles with negative weight, then formally the Floyd-Warshall algorithm is not applicable to such a graph. However, in fact, the algorithm will work correctly in this case too, since after processing such a graph in the diagonal of the matrix of shortest paths, negative numbers will appear, indicating that the shortest distance from the point in this cycle to it on its own will be less than zero. This means that a negative cycle has been completed. Therefore, the Floyd-Warshall algorithm can be used to evaluate the presence of negative cycles in a graph.

7. Conclusions

- Based on the Floyd-Warshall algorithm, a universal computer model has been developed. It allows automating the operations of determining and constructing the shortest routes between any pair of transit nodes in transport and telecommunication networks and, thereby, providing such a choice of routes for ships, land and air transport, as well as any other mobile objects, which is characterized by efficiency and economical operation.
- A recursive procedure is proposed for determining the sequence of edges that make up the shortest path using the functions of the Optimization Toolbox of MATLAB.
- The correctness and efficiency of the computer model of the modified Floyd-Warshall algorithm and its software implementation is confirmed by a specific example of automation of calculating the shortest distances between any pairs of points on a graph of a transport network with a complex configuration.
- The advantages of the proposed procedure for automating the calculation of shortest paths based on the Floyd-Warshell algorithm are the simplicity of the software implementation and the capability to use the universal tools provided by the MATLAB computing medium.
- The field of widespread use of the Floyd-Warshall algorithm is dynamic optimization, from searching for transitive closure of a graph to genetics and project management, as well as optimization in transport, telecommunication and other networks.

References

Bellman, R., 1958. On a Routing Problem. Quarterly of Applied Mathematics 16, 87-90.

Chertkov, A., 2017. Automation selection shortcuts routes of ships on the basis of modified BellmanFord Algorithm. Vestnik Gosudarstvennogo universiteta morskogo i rechnogo flota imeni admirala S.O. Makarova 9.5, 1113–1122. DOI: 10.21821/2309-5180-2017-9-5-1113-1122.

Castillo, O., Trujillo, A., Melin, P., 2007. Multiple Objective Genetic Algorithms for Path-planning Optimization in Autonomous Mobile Robots. Soft Computing 3.11, 269-279.

Dedkov, V., Kirsanov, M., 2013. The problem of control for multirobot systems. Innovacionnye informacionnye tehnologii 2, 206-213.

Fedorenko, K., Olovyannikov, A., 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, 714-723. DOI: 10.21821/2309-5180-2017-9-4-714-723.

Fonseca, C., Fleming, C., 1993. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. 5th international conference genetic algorithms, 416-423.

Kirsanov, M., 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, 183-190. DOI: 10.21821/2309-5180-2016-8-2-183-190.

Kirsanov, A., Anavatti, S., Ray, T., 2013. Path planning for the autonomous underwater vehicle. Lecture Notes in Computer Science 2, 476-486.

Saharov, V., Chertkov, A., Tormashev, D., 2014. Algoritm optimal'nogo planirovanija gruppovogo vzaimodejstvija robotov. Morskoj Vestnik 4.52, 119-122.

Saharov, V., Sikarev, I., Chertkov, 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, 647-657. DOI: 10.21821/23095180-2018-10-3-647-657.

Sazonov, A., Derjabin, 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, 6-13.

Sokolov, S., et al., 2020. Hybrid neural networks in cyber physical system interface control systems. Bull. Electr. Eng. Inf. 9, 1268-1275. DOI: 10.11591/eei.v9i3.1293.

Zhilenkov, A., et al., 2020. Intelligent autonomous navigation system for UAV in randomly changing environmental conditions. J. Intell. Fuzzy Syst., 1-7. DOI: 10.3233/jifs-179741.