Scientific Journals
Maritime University of Szczecin

Zeszyty Naukowe
Akademia Morska w Szczecinie

2014, 39(111) pp. 57–62
ISSN 1733-8670

2014, 39(111) s. 57–62

Bi-directional search in route planning in navigation

Mariusz Dramski

Maritime University of Szczecin
70-500 Szczecin, ul. Wały Chrobrego 1–2, e-mail: m.dramski@am.szczecin.pl

Key words: shortest path, safe route, restricted area, bi-directional search, Dijkstra algorithm

Abstract
The shortest path problem is one of the most significant ones in the field of maritime navigation. One of the most efficient algorithms was proposed by E. Dijkstra in 1959. Taking into account the development of computer technology was offered another interesting approach to the issue. The main idea is to execute the shortest path algorithm simultaneously forward from the source and backward from the target. The results are presented and discussed.

Introduction

The use of shortest path algorithms in navigation comes down to design a graph of possible paths and place it on the map. After graph's obtaining there is a possibility to find the shortest path between all nodes. The nodes represent the point or area on the map. In this paper we will discuss only the classical approach to find a solution for this problem.

The quality of solution is assessed on the basis of two main factors:

- ability to find an optimal path;
- computational complexity.

We consider undirected graph $G = (V, E)$ with n nodes and m edges each, with nonnegative edge $w(u, v)$. The problem is to find the shortest path between the source s and target t. In this case it will be the sum of weights of all edges which create the shortest path and is given by $d(s, t)$.

In navigation the weights of graph's edges are constant, because they represent the geometric distances between the nodes placed on the map. So, it is possible to precompute and store shortest paths between all the nodes.

There are several classical approaches to solve the problem of the shortest path in the graph:

- Dijkstra's algorithm;
- priority queues;
- bi-directional search;
- geometric goal directed search (A^*).

Due to the fact that this article describes a bi-directional search using Dijkstra's algorithm, we will focus only on those methods. The other ones are described e.g. in [1] or [2].

Dijkstra algorithm

In 1959 [3] E. Dijkstra proposed an algorithm, which is currently one of the most popular solutions of the shortest path in the graph of problem. Dijkstra's algorithm finds the path with the lowest cost between a start vertex and every other vertex in the graph. This algorithm is often used in network routing protocols and many other problems where graphs can be applied.

The main steps of Dijkstra's algorithm:

- Let s be the start node, $w(i,j)$ is the weight of edge i, j;
- Create a distance matrix d for all the vertices of graph, assuming $d(s) = 0$ and $d(v) = \infty$ if there is no edge;
- Create a priority queue Q where the priority is a distance from the start node s;
- Repeat until Q is not empty:
  – remove from the queue the vertex u with the lowest priority;
  – for each neighbour v of the vertex u, $d(v) = \min(d(u) + w(u,v), d(v))$;

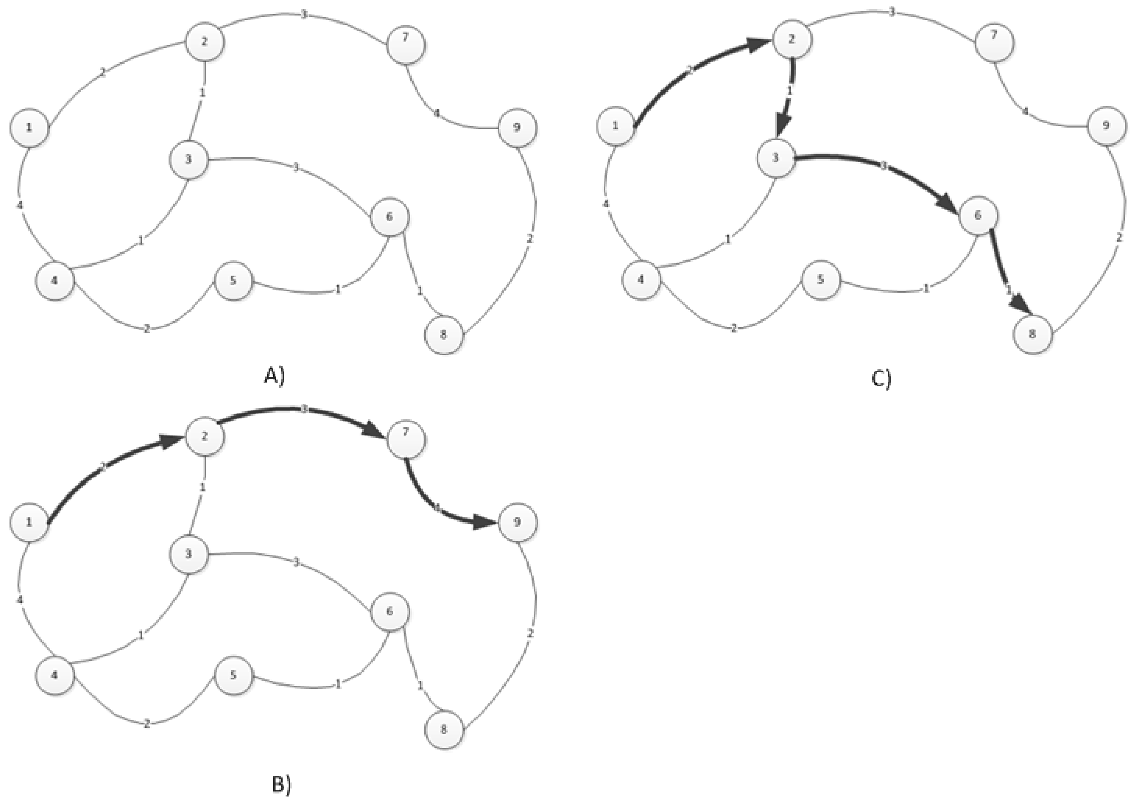


Fig. 1. Illustration of algorithm Dijkstra, A) Graph, B) Path 1–8, C) Path 1–9

- The last row of matrix *d* is a vector containing the shortest distance values from *s* to all the vertices of graph.

The computational complexity of this algorithm is $O(n^2)$, but can be reduced to $O(n\log n)$ using priority queues [1].

Of course, there are other solutions of the shortest path problem such Floyd algorithm, Bellman-Ford algorithm etc. The comparison between them is described in [2].

Figure 1 illustrates the Dijkstra's algorithm results for an example graph consisting of 9 nodes and 11 edges between them. On each edge there is a number telling the weight of the edge. As it can be observed in these two simple experiments, the shortest path search is done forward from the source (node 1) to the target point (8 and 9 respectively). This is a normal behavior of this method according to it's foundation.

Bi-directional search

New processors consist of multiple central processor units which read and execute program instructions. They are called multi-core processors and were developed in early 2000s by Intel, AMD and others. The invention of this technology lets to execute programs in parallel. Of course, earlier there was a possibility to parallel processing but it required some special machines which not always were available for researchers.

Dewelopment of hardware is very fast. New systems are available every year, are more reliable, faster and cheaper.

Due to the fact above (and other not mentioned ones) conducts to the possibility of acceleration of the data processing.

The bi-directional search was proposed initially in [4]. In our case it lies in the fact that Dijkstra's algorithm will be executed simultaneously forward from the source and backward from the target. In this way the shortest path can be derived from the information already gathered. This technique lets to accelerate the calculations.

Figure 2 illustrates the same graph as in figure 1. This time bi-directional search applied using Dijkstra's algorithm. As it can be observed the search was started independently from the start point and the target one. In both cases the search stopped in the middle of path. Due to the fact that, Dijkstra's algorithm has the optimality guarantee and the graph is static, the search will always be stopped in the same point. So, there is no difference in final effect. However, by launching two independent threads can do the same task more quickly.

Figure 3 illustrates the basic diagram of the bi-directional search algorithm.

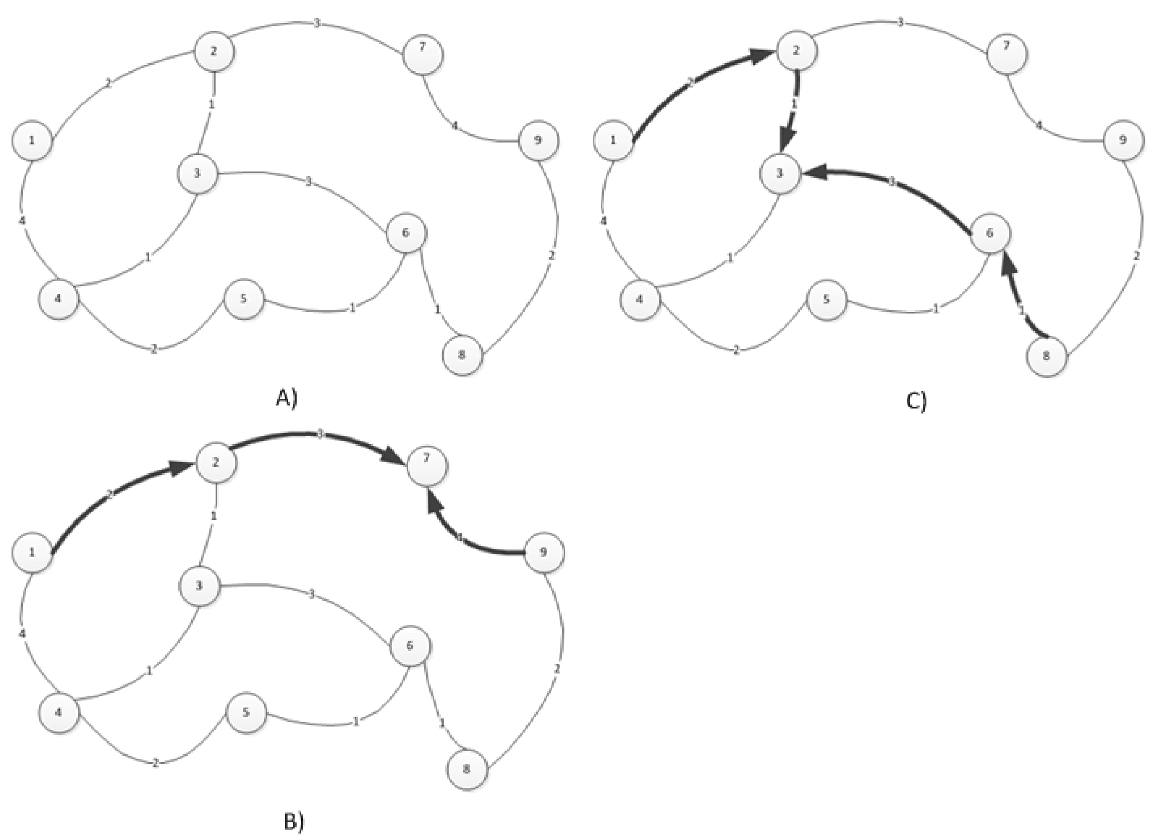


Fig. 2. Illustration of algorithm Dijkstra – bi-directional search, A) Graph, B) Path 1–8, C) Path 1–9

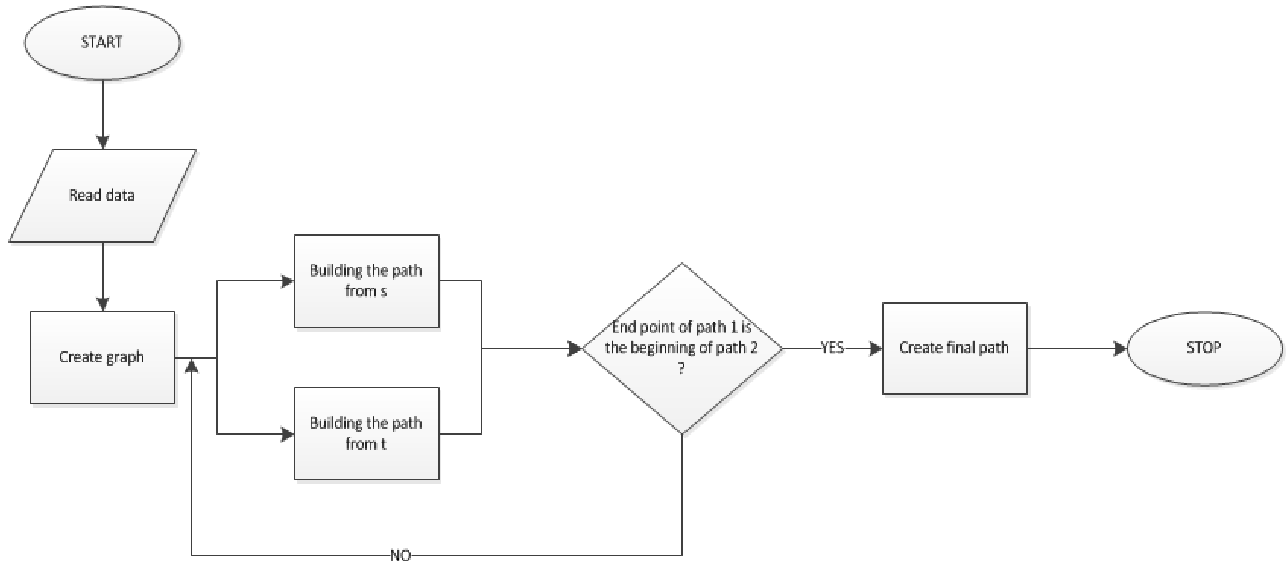


Fig. 3. The bi-directional search algorithm

Restricted area

One of the basic tasks of navigators is to conduct the ship safely from the point of departure to destination. The performance of such task, among other issues, calls for taking into account movements of other objects (ships) in the area (Collision Regulations), restrictions of the area itself, (shape of the shoreline, available depths, other obstructions) or the dynamics of own ship. In [5], it is estimated that about 80% of marine collisions are results from human error.

Navigation in the restricted area is very difficult. An increasing number of devices supporting navigator's work leads to such excess of information that it hampers taking the right decisions aimed at the ship's safety.

An example of restricted area illustrated in figure 4.

Fig. 4. An example of restricted area

Creating a graph of possible paths

Figure 5 illustrates suggestion of graph of possible paths. Each node represents a point in the area which is coordinates. Each edge is a possible way between the nodes. The graph can be obtained in many different ways such Delaunay's triangulation, Voronoi polygons etc. Anyway, in this paper, we do not discuss about creating graph which can be e.g. [6]. Figures 6 and 7 present the shortes path found by bi-directional search from node 1 to 38 and 1 to 32. There is no need to show the results of

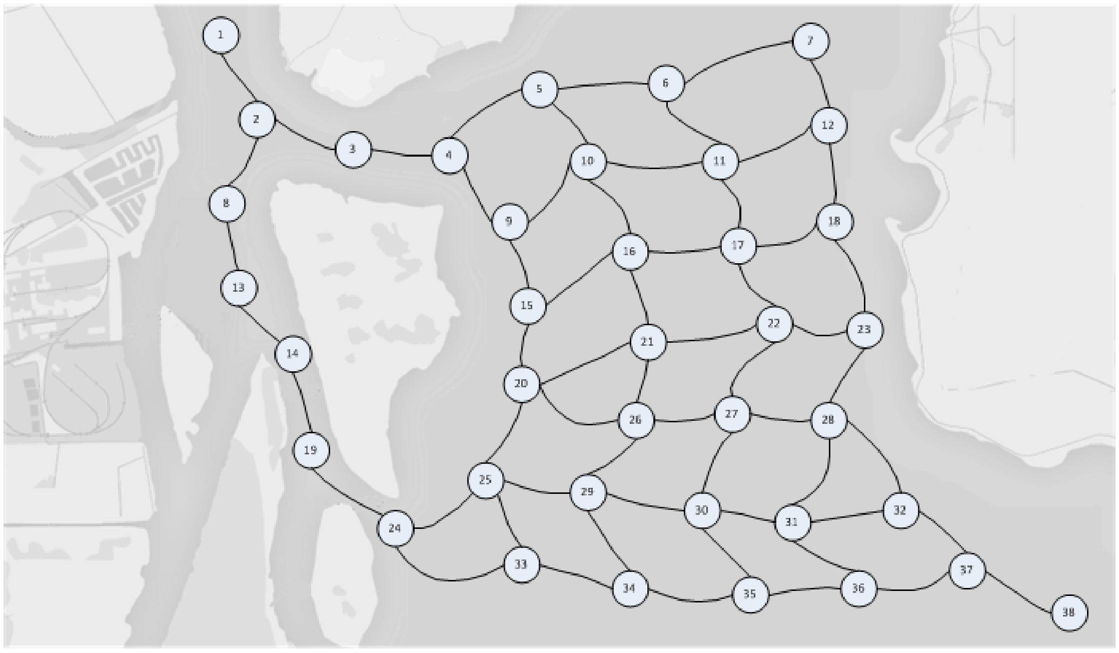


Fig. 5. Restricted area with a graph of possible paths

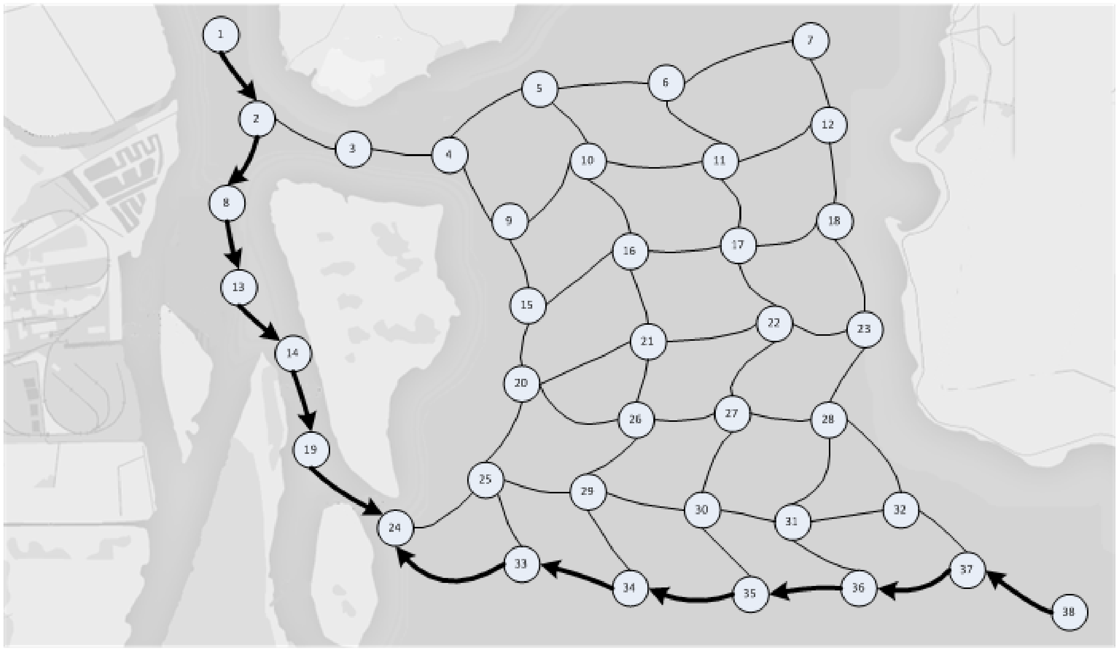


Fig. 6. Restricted area with path 1–38 found by bi-directional search

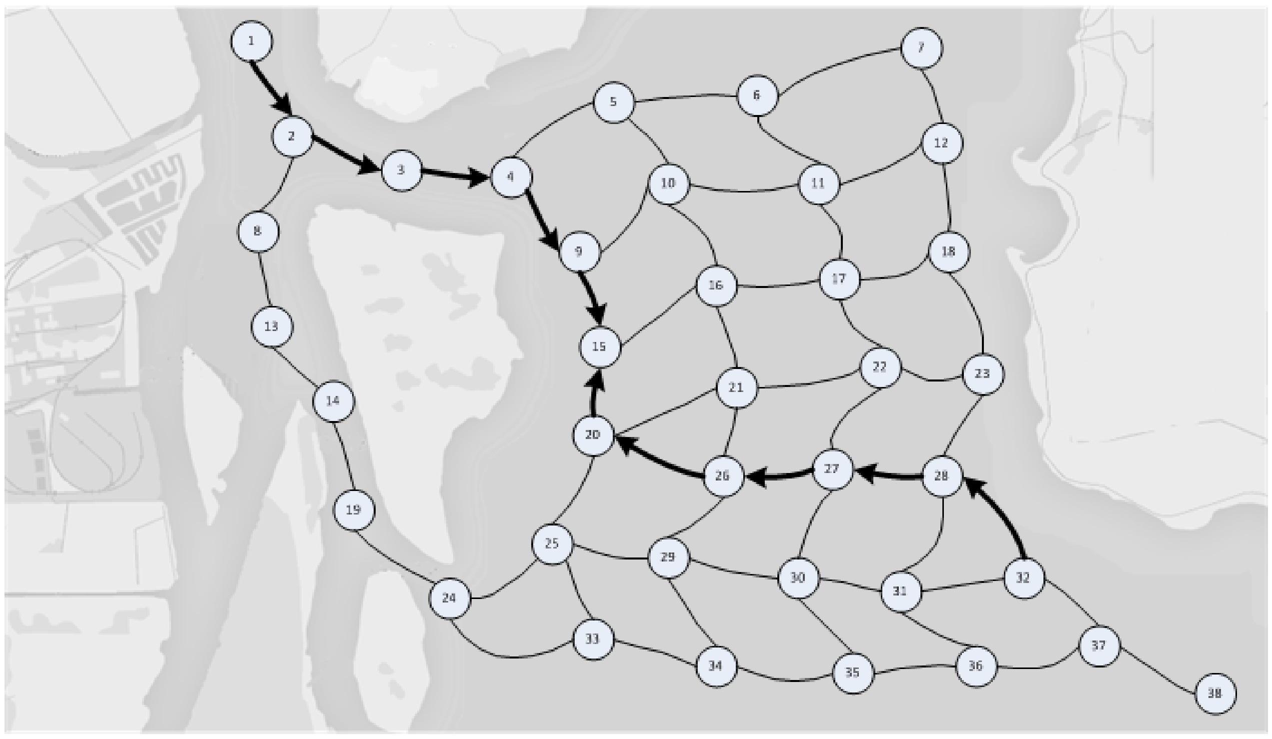


Fig. 7. Restricted area with path 1–32 found by bi-directional search

single use of Dijkstra's algorithm, because the path will be the same.

The arrows in the above illustrations show, how to implement the shortest path search which was done forward from the node 1 and backward from the nodes 32 and 38. As it can be seen, both algorithms created paths which met in the middle of possible route. At the figure 6 it was node 24 and at the figure 7 node 20.

Due to the fact, that in both cases the found path was the same as in single use of Dijkstra algorithm, there is a need to ask a question about the benefits of bi-directional search. In real life the graphs designed have a very big number of nodes, so the time of the search would be significantly long. Although maritime navigation there is a spare of time (in most cases), the computational complecity is still very important. Decision support systems must provide the information in the shortest time possible. Bi-directional search lets to use multi-threading or even multi-core processing. This is an obvious form of make this time shorter.

Unfinite graph experiment

For the purpose of experimental verification of bi-directional search few fragments of infinite graphs were created. The kind of this graph is shown at figure 8. Every vertex of these fragments has 8 edges (sides of the squares and it's diagonals – except external vertices). The bottom left node (number 1) of every fragment is a start node for shortest path routing algorithm and the top right one (number *n*) is the stop node. So, it's easy to observe that the shortest path will always be the diagonal of a square.

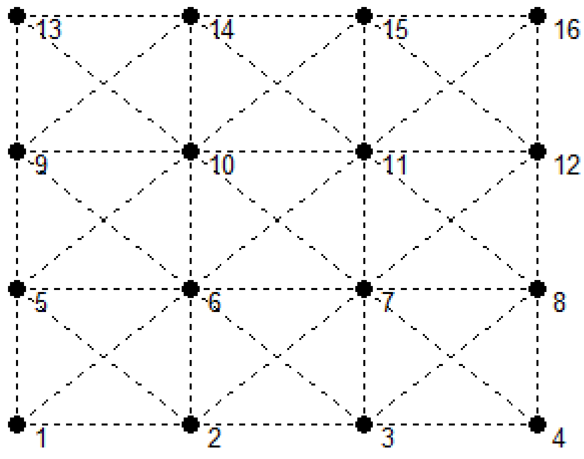
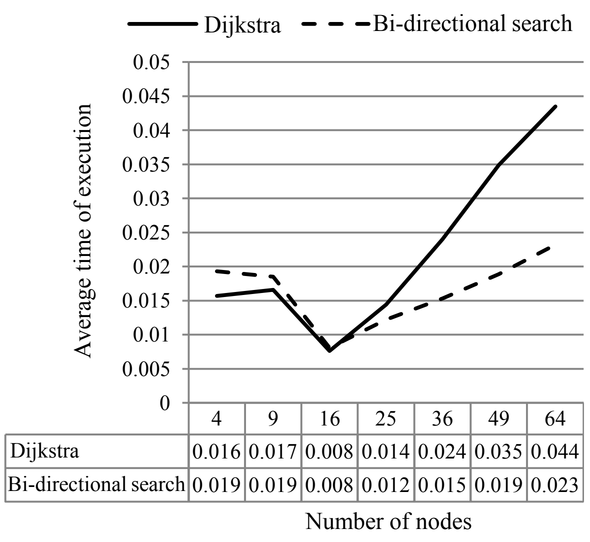


Fig. 8. An example of infinite graph

Figure 9 illustrates a comparison between times of the execution of each approach. When the small number of nodes was considered, the average time was shorter for the classical Dijkstra algorithm. Increasing the number of nodes, the difference in favor of bi-directional search was observed. This the normal situation, caused in the case of parallel processing. If the number of calculations is low, there is no need to do it in parallel. Only the creation of threads requires some time.



	4	9	16	25	36	49	64
Dijkstra	0.016	0.017	0.008	0.014	0.024	0.035	0.044
Bi-directional search	0.019	0.019	0.008	0.012	0.015	0.019	0.023

Fig. 9. Comparison of Dijkstra algorithm and bi-directional search

Conclusions

It has been proven that the bi-directional search has some advantages compared to the traditional approach of such a Dijkstra algorithm. Experiments carried out shown that the difference in average time of execution decreases, depending on the complexity of graph considered. It is reasonable to conduct further research to find a more optimal solution. The response of the decision support system (which may include the proposed methodology) is very important in the determined time period. It is necessary, of course, to carry out more experiments, especially basing on real maritime maps.

References

1. DELLING D., SANDERS P., SCHULTES D., WAGNER D.: Engineering Route Planning Algorithms. Algorithmics of Large and Complex Networks, Springer, 2009.
2. DRAMSKI M.: Shortest path problem in static navigation situations. Metody Informatyki Stosowanej 5, 2011.
3. DIJKSTRA E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 1959, 269–271.
4. POHL I.: Bi-directional Search. In Meltzer Bernard; Michie Donald, Machine Intelligence 6, Edinburgh University Press, 1971, 127–140.
5. LI L.N., YANG S.H., CAO B.G., LI Z.F.: A summary of studies on the automation of ship collision avoidance intelligence. Journal of Jimei University, China, Vol. 11, No. 2, 2006, 188–192.
6. DRAMSKI M., MĄKA M.: Algorithm of Solving Collision Problem of Two Objects in Restricted Area. Communications in Computer and Information Science 395, Springer 2013, 251–257.

Other

7. LIPSKI W.: Kombinatoryka dla programistów. WNT, 2007.