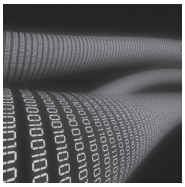

Archives of
Transport System
Telematics

Volume 5

Issue 4

November 2012

Selected shortest path in the graph algorithms with a use of trapezoidal grid in navigation in restricted area

M. DRAMSKI[a], M. MĄKA[a]
[a] MARITIME UNIVERSITY OF SZCZECIN,Faculty of Navigation, Wały Chrobrego 1-2, 70-001 Szczecin, Poland
EMAIL: m.dramski@am.szczecin.pl

ABSTRACT
This paper presents the possiblities of the use of the shortest path in the graph algorithms in ship's safe route choice process in a restricted area. To create a graph, a trapezoidal mesh based on the S-57 digital map data was used. Numerical experiments were carried out and their results are discussed

KEYWORDS: shortest path, safe route, restricted area, navigation, trapezoidal grid

1. Introduction

The main task in navigation is to conduct safely an object from departure point to destination. Besides there is a need to consider all the limitations (e.g. coast line, other objects, existing regulations etc.). This problem is widely investigated by research and commercial units.

Navigation in the restricted area can be described as searching the shortest path in graph created by the grid of points representing this area. In this paper trapezoidal grid and the use of few shortest path algorithms are described. The results conduct to interesting conclusions present in the summary.

2. Trapezoidal grid

2.1. Grid generation algorithm

Trapezoidal grid created on the basis of data obtained from navigational electronic chart S-57 [5] can be used as a tool for designing a graph of possible paths in given area.

Input data is a set of ordered points representing subsequent objects present in the fragment of map. These points will be the nodes of generated trapezoidal grid. The first step is to choose the first object and subjecting it to transform process. In the next steps further object are added and the grid is modified locally.

Grid elements are created by conducting two vertical segments from each node (which is a beginning / end of vectors marking the boundaries of objects). This node is the bottom end for the first segment and a upper end for the second one. The second end of each segment consists of (Fig. 3) [6]:

1. the closest crossing point of the segment with any object's boundary on map or
2. grid coverage area boundary

Each mesh is defined by:

1. the indices of two nodes, which are output vertical segments forming the sides of the mesh,
2. indices of the two lines belonging to the object (objects) on the map restricting elements of the grid at the top and bottom.

If the element is located at the border of a grid coverage area, the edges of the area can be the sides of the mesh.

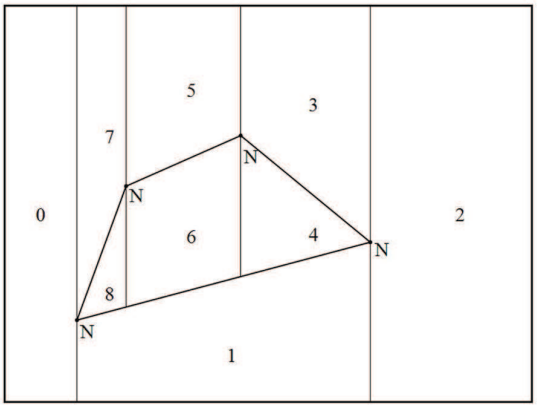

Fig.1. Trapezoidal mesh, (N – nodes, 0 – 8 –numbers of mesh elements)

As a result of the algorithm a graphical representation of a grid and (Fig. 1) a sorted array containing the data for each trapezoid are obtained. The array includes among others: λcoordinates of vertical sides, indices of lines restricting a trapezoid from bottom and top, corners coordinates, classification information – allowed / not allowed, indices of neighbour elements.

The prohibited areas are classified as elements of a grid located on the land, excluded from the inland areas, small depth regions, vessel wrecks areas etc.

Trapezoids are marked as neighoburs only if they are in contact with the vertical side formed by setting segments from the grid node. Each one can have up to 4 neighbours, two from the left and right, defined as (Fig. 2) [4,6]:

- top left;
- bottom left;
- top right;
- bottom right.

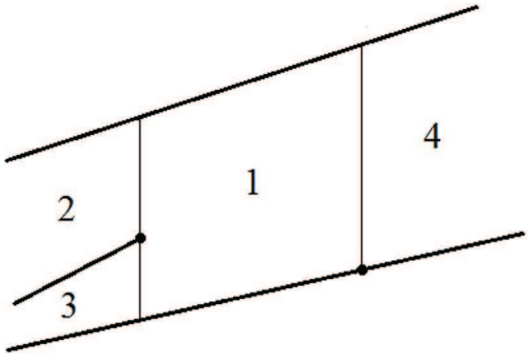

Fig.2. Neighbouring trapezoids, (relatively to trapezoid 1): 2 – top left neighbor, 3 – bottom left, 4 – top right

Determine the relative positions of the grid elements and prohibited elements is the basis for the construction of the graph of possible paths used in route selection and optimization.

2.2. Graph creation algorithm

In described method two ways of positioning of the vertices of the graph were applied:

A In each one grid element and corresponding vertical segments, 3 to 5 points are placed, depending on the number of trapezoids neighbouring:
- one point on the line passing through the middles of segments being bottom and top edges of the element in the middle of the distance between them,
- one point in the middle of each vertical side, if the element has a neighbour,
- two points on each vertical side, if the element has two neighbours – each of the points midway between the element and it's neighbour,

B. In each one grid element and corresponding vertical segments, 9 to 15 points are placed, depending on the number of trapezoids neighbouring:
- three points on the line passing through the middles of segments being bottom and top edge of the element: one midway between them and two in Δϕfrom bottom and top edge of the element,
- three points on each vertical side, if the element has one neighbour – one point in the middle of vertical side and two points in Δj from bottom and top edge of the element,
- six points on each vertical side, if the element has two neighbours – one point in the middle of each segment connecting the element with it's neighbour, two points in Δj from bottom and top edge of the element.

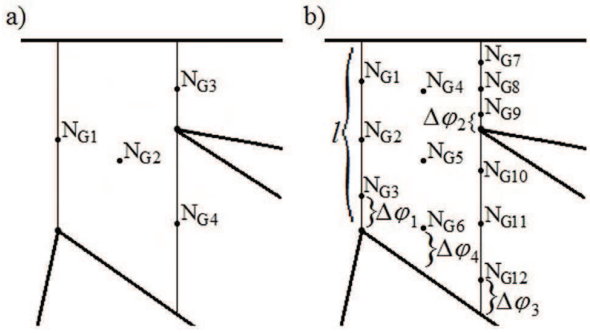

Fig.3. Graph nodes: a) methodA – maximum 5 nodes, b) methodB – maximum 15 nodes

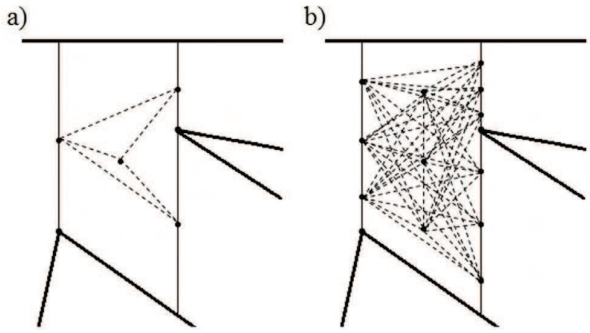

Fig.4.Vertex connections: a) method A – maximum 5 nodes, b) method B – maximum 15 nodes

In first case 8 connections between nodes can appear in one trapezoid. In the second one if the maximum number of nodes (15) is reached – 75 possible connections appear.

The ratio of the distance $\Delta\phi$to the total length of the vertical side of trapezoidlis determined experimentally and ranges from 0.1 to 0.2.

The output data are: the array of nodes with their numbers and coordinates and the adjacent matrix representing the graph of possible paths.

3. Experiment

In this paper three examples of trapezoid grid are given. The test area is present at Fig. 5. Restricted area is represented by gray polygons.

Trapezoid grid for the test area was created. It contained 43 elements (trapezoids). After the elimination of restricted elements 26 elements were obtained (Fig. 5)

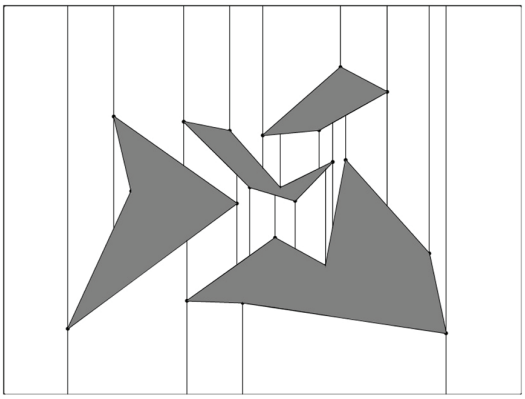

Fig.5. Test area

Three graphs of possible paths were generated:

- First, created using method A described in 2.2 – from 5 to 8 possible connections in one trapezoid (Fig. 6) ,
- Second , created using method B described in 2.2 – from 15 to 75 possible connections in one trapezoid (Fig. 7),
- Third, which is the combinations of methods A and B in narrow passages between two restricted areas. If the height *l* of the vertical side of the trapezoid was less then some assumed value, method A was used. Method B otherwise (Fig. 8).

In the example I and II method B was used for creating a graph with assumed ratio

$$\frac{\Delta\varphi}{l} = 0.1 \qquad (1)$$

In the example I the created graph has 55 nodes and 176 possible connections between them. The average time of the grid generation was about 28 [ms] .

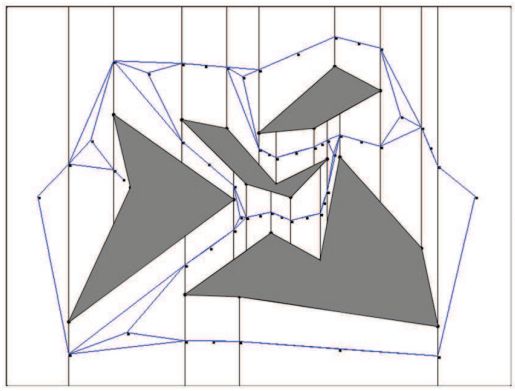

Fig.6. Example I – created with the use of method A–nodes and possible connections

Example II shows a graph containing 163 nodes nad 1722 possible connections between them. The average time of graph generation was about 44 [ms].

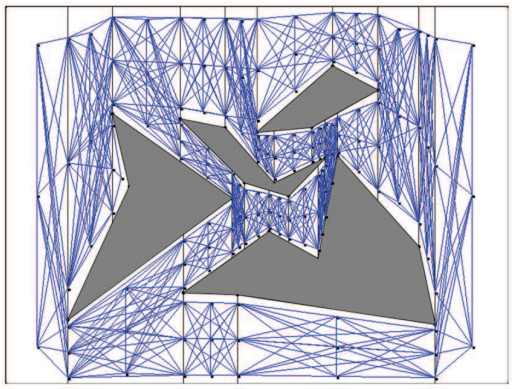

Fig.7. Example II – created with the use of method B–nodes and possible connections

The example III is the graph with 101 nodes and 774 possible connections between them. The average time of graph generation was about 34 [ms].

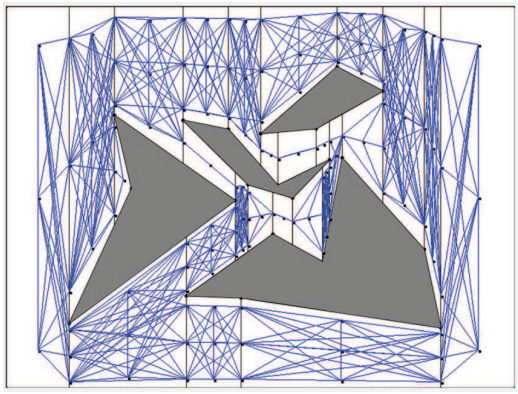

Fig.8. Example III – created with using the combination of two methods: A and B

The use of method A results the shorter time of alogirthms' execution, but obtained paths but the change of direction has a large angular values.

Example II results the longer execution time but offers more possibilities to choose the path. If the $\frac{\Delta\varphi}{l}$ ratio is small, especially if the length l is short, the paths generated are to close to the edges of restricted areas. Besides, despite of increasing the number of nodes, there was no possibility to limit the number of returns by the angle of high value.

The combination of method A and B in the last example resulted the reduction of the number of nodes by 38% and the execution time by 23%.

Besides the arrange of graph nodes in narrow passages using the method A, resulted withdrawal of the paths generated from the edges of prohibited areas

4. Shortest path search

The problem of the shortest path between the nodes of the graph is solved using different methods. The most pupular are: Dijkstra [1], Bellman-Ford, Floyd or A* algorithm. There is also an alternative way – e.g. methods based on artificial intelligence [2]. In [7] Śmierzchalski proposed an approach using evolutionary algorithm. In this paper one of these methods was chosen – simplified ant colony optimization (SACO [2]). The advantages and disadvantages of each approach are described in [3].

As it can be seen at all the figures 9-11, there is no problem wich finding the shortest path between two nodes basing on the graph created by the method described above in section 2 of this paper. Besides the difference between methods A, B and their combination can be observed. There is no figure illustrating the result of SACO algorithm because of the fact that this method doesn't guaranty any optimality, so the route obtained will be always different. Of course this is due to the probabilistic character of this method.

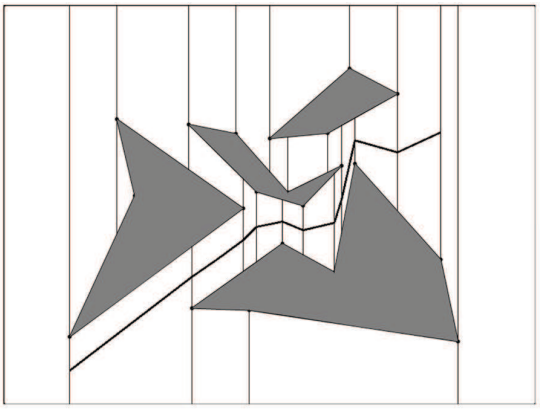

Fig. 9 – Example I – the route

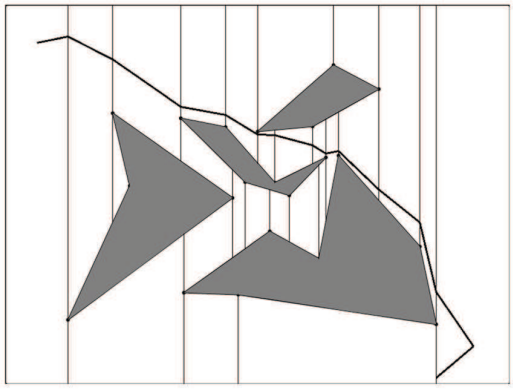

Fig.10. Example II – the route

It can be found e.g. in [5] that each proposed algorithm has different computational complexity which results other execution times. This is very important due to the fact that the proposed methodology is to be used in the future in DSS system supporting the decision process of the navigator at the ship.

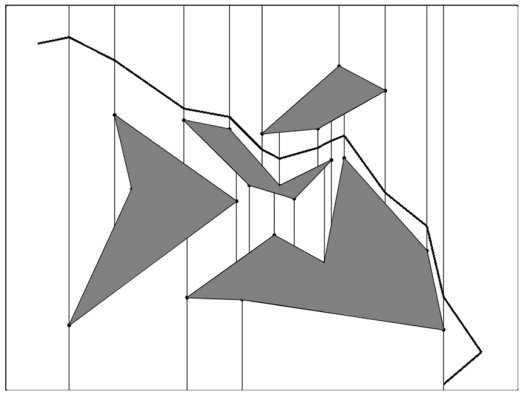

Fig.11.Example III – the route

Table 1 illustrates the average times of shortest path searching in each experiment.

Table 1. Average time for each search path algorithm

Method	Average time [ms]
A*	23
Bellman-Ford	44
Dijkstra	30
Floyd	41
SACO	78

These results above were expexted as it has been proven in [3]. The natural conclusion is that the average time of the execution depends on the computational complexity of the algorithm and the guarantee of optimality. The longer time in the case of SACO algorithm should be explained separateky. Although the computational complexity of SACO and Floyd algorithm is the

same at $O(n^3)$ – the first one has probabilistic character and is not always able to find a solution [2].

The best (shortest) time is obtained when Dijkstra or A* alogrithm are used. In the second case there is a need to know the coordinates of each point in the grid (node). This fact is requires due to the definition of this algorithm. So if we don't know precisely the coordinates of destination point, it would be better to use Dijkstra algorithm, even if the computational complexity requires more calculations.

Anyway, there are two important factors. First of all there is no place for methods that can give no result. This facts excludes SACO algorithm (of course artificial intelligence gives a lot of different interesting methods). The second factor is execution time.

Each navigator has to make right decision in real time operations on the sea. Long data processing can make the decision support system completely useless. So there is a need to find not only a good solution (with as low as possible error), but also a fast one (corresponding to navigation problems).

5. Conclusion

Although there are some proposals of solving the problem of navigation in restricted area, there is still a need to do further research. The safety on the sea requires a lot of work.

Every area has some specific restrictions which cause the navigation problems.

The aim of this paper was to present the problem of navigation in restricted area. A proposal of the methodology is described. First of all there is a need to generate a grid using navigational electronic chart. Authors proposed the use of trapezoidal grids which leads to create the graph of all possible paths. Three approaches (A, B and their combination) were investigated. After obtaining the graph, some methods of the shortest path search were used. The investigations lead to the following conclusions:

- the shortest execution time will be using the method A and A* algorithm,
- the best quality of the solution is the combination of methods A and B and A* algorithm (according to navigation),
- the use of A* algorithm is possible only in the situation when all the coordinates of each point of the grid is given [3],
- if there is a lack of coordinates, it would be better to use Dijkstra algorithm [3],
- in A* algorithm if the situation changes dynamically there is a need to repeat all the calculations. Data processing can't be stoped without any consequences [3].

All the steps described in the article can be ordered as follows:

- data extracting from digital S-57 standard chart [5],
- creating a grid using trapezoidal meshes with the use of method A or B depending the considered restricted area,
- creating an adjacent matrix which is the representation of the graph containing all the possible paths in the route,
- creating a route using one of the shortest path search algorithms.

Four steps above seem to be a natural way of processing to solve the problem of navigation in restricted area.

Obtained results seem to be very promising in further research. This fact justifies the goal of investigations which is to conduct safely a ship from the point of departure to destination in restricted area.

This article treats only about the static situations, so there is a need to further research, having regard to dynamic ones (e.g. moving objects, weather conditions etc.).

Bibliography

[1] DIJKSTRA E. W.: (1959) "A note on two problems in connexion with graphs". *Numerische Mathematik*1: 269–271

[2] DORIGO M., STUTZLE T.: Ant Colony Optimi-zation, The MIT Press 2004

[3] DRAMSKI M.: Shortest path problem in static navigational situations, 18th International Multi-Conference ACS 2012, Międzyzdroje, Poland

[4] van KREVELD M., de BERGH M., OVERMARS M., SCHWARZKOPF O.: Computational geometry – algorithms and applications, Warsaw 2007, WNT (in Polish)

[5] MĄKA M., MAGAJ J.: Data extraction from an electronic S-57 standard chart for navigational decision systems, VII International Conference ExploShip 2012 (in English)

[6] MĄKA M.: The recurent algorithm for area discretization using the trapezoidal mesh method, VII International Conference ExploShip 2012 (in English)

[7] ŚMIERZCHALSKI R., MICHALEWICZ Z.: Modeling of ship trajectory in colision situations by an evolutionary algorithm, IEEE Transactions on Evolutionary Computation, Vol. 4, pp. 227-241