Path Planning for Autonomous Inland Vessels Using A*BG

Linying Chen(✉), Rudy R. Negenborn, and Gabriel Lodewijks

Department of Maritime and Transport Technology,
Delft University of Technology, Delft, The Netherlands
L.chen-2@tudelft.nl

Abstract. To meet the transportation demand and maintain sustainable development, many countries are aiming to promote the competitive position of inland waterway shipping in the transport system. Autonomous transport is seen as a possibility for maritime transport to meet today's and tomorrow's challenges. In realizing autonomous navigation, path planning plays an important role. Being the most widely used path planning algorithm for robotics and land-based vehicles, in this paper we analyze A* and its extensions for waterborne applications. We hereby exploit the fact that for vessels optimal paths typically have heading changes only at the corners of obstacles to propose a more efficient modified A* algorithm, A*BG, for autonomous inland vessels. Two locations where ship accidents frequently occur are considered in simulation experiments, in which the performance of A*, A*PS, Theta* and A*BG are compared.

1 Introduction

Currently, economic development is putting enormous pressure on transport systems. Freight transport is likely to grow over the next decades [7]. If roads and railways are the major means of transport for handling the growth, they will face frequent congestion. Inland waterway shipping still have the capability of transporting large additional volumes. It offers an environment-friendly alternative to road and rail transport in terms of both energy consumption and gas emissions [6]. To meet the transportation demand and maintain sustainable development, many countries are aiming to promote and strengthen the competitive position of inland waterway shipping in the transport system.

Research has proposed many measures to improve the position of inland shipping, such as optimizing ship dimensions [10], removing bottlenecks [5], improving utilization of ports [8] and locks [25]. Among these measures, employing autonomous vessels has recently drawn much attention [13,27]. Autonomous vehicles are already state-of-the-art in the land-based transport domain. There exist several examples of self-driving and automated guided vehicles in modern container terminals [26]. Consequently, applying autonomous vessels is seen as a way to improve the safety and efficiency of inland shipping.

© Springer International Publishing Switzerland 2016
A. Paias et al. (Eds.): ICCL 2016, LNCS 9855, pp. 65–79, 2016.
DOI: 10.1007/978-3-319-44896-1_5

Safety can be improved as human error is one of the main causes of ship accidents. Figure 1 shows the mains causes of ship accidents between 2005 and 2014 in Dutch inland waterways [14]. The category operation error includes alcohol/drug use, wrong estimation, fatigue, etc.; the category communication error indicates not maintaining watch on correct VHF channel, unclear explanation, etc.; the category environmental error includes disturbances caused by wind, wave and current, poor visibility, etc.; the category equipment error indicates the failure of engine, rudder or other navigation equipments. For autonomous vessels, detection of obstacles, estimation of the risk, communication between vessels and infrastructure can be done without humans. Thus, applying autonomous vessels could be an efficient measure to reduce the number of accidents.

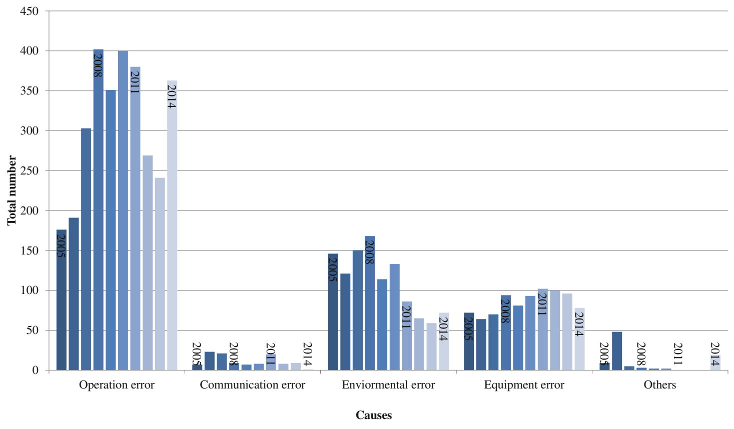


Fig. 1. The causes of the shipping accidents (based on [17]).

Efficiency can be improved by autonomous vessels due to the intelligent path planning and better control of vessel motion. Communication and coordination with infrastructures also make it possible for autonomous vessels to minimize the waiting time at ports, locks, etc.

The overall architecture of an autonomous vessel is shown in Fig. 2. To realize autonomous navigation, a vessel controller uses sensors to get self-state information (e.g., position, speed and heading), environmental information (e.g., wind speed, current velocity) and information of obstacles. Based on the obtained information, optimal paths to follow and desired speed and heading with specified objectives and constraints can be determined. The commands are sent to actuators for autonomous navigation.

In Fig. 2, the module ‘Path planning’ plays an important role in autonomous navigation. It describes how the autonomous vessel make decisions regarding its course to sail. The path planning problem can be subdivided into a global and a local planning task: an approximate global planner computes paths ignoring the kinematic and dynamic constraints; an accurate local planner accounts for the

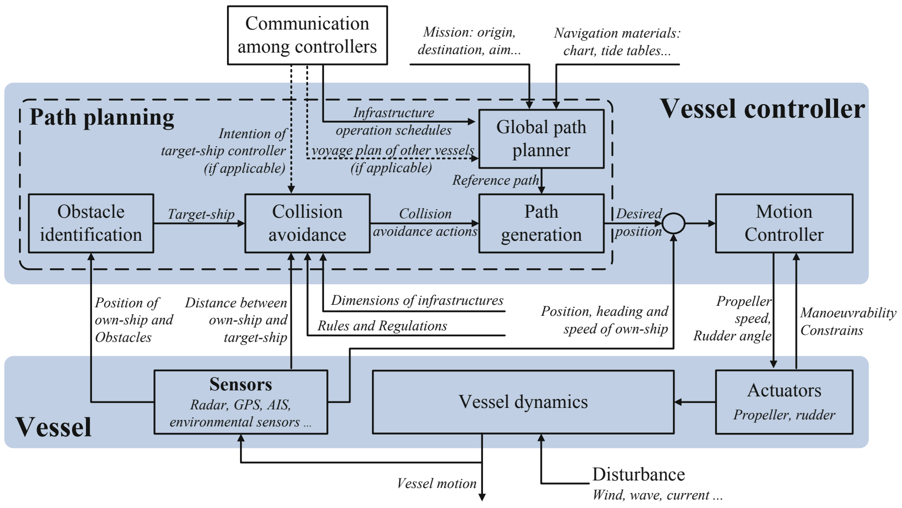

Communication among controllers
Mission: origin, destination, aim...
Navigation materials: chart, tide tables...
Vessel controller
Path planning
Infrastructure operation schedules
Intention of target-ship controller (if applicable)
voyage plan of other vessels (if applicable)
Global path planner
Reference path
Obstacle identification
Target-ship
Collision avoidance
Collision avoidance actions
Path generation
Desired position
Motion Controller
Position of own-ship and Obstacles
Distance between own-ship and target-ship
Dimensions of infrastructures
Rules and Regulations
Position, heading and speed of own-ship
Propeller speed, Rudder angle
Manoeuvrability Constrains
Vessel
Sensors
Radar, GPS, AIS, environmental sensors ...
Vessel dynamics
Actuators
Propeller, rudder
Vessel motion
Disturbance
Wind, wave, current ...

Fig. 2. The overall architecture of autonomous vessels.

constraints and generates feasible local trajectories [20]. The final path are determined on the basis of reference path provided by the global planner according to the transport mission, known stationary obstacles (e.g., islands, shallow waters) and infrastructure operation schedules, and the collision avoidance actions taking into account the regulations and the limitation of infrastructures (e.g., width and depth of waterways). Communication between vessel controllers will help the controller make better path planning decision. As a starting point, this paper focuses on the global path planning problem.

Many path planning algorithms have been developed for the navigation of unmanned surface vehicles as well as robots, such as Artificial Potential Field methods [23], Evolutionary Algorithms [12], and Heuristic Search Algorithms [4,19]. For a detailed review of path planning and collision avoidance technologies and techniques, see [2,21]. Among these methods, the group of heuristic search algorithms, especially A* and its extensions, are commonly used to determine the path from an origin to a destination for land-based vehicles [18,22].

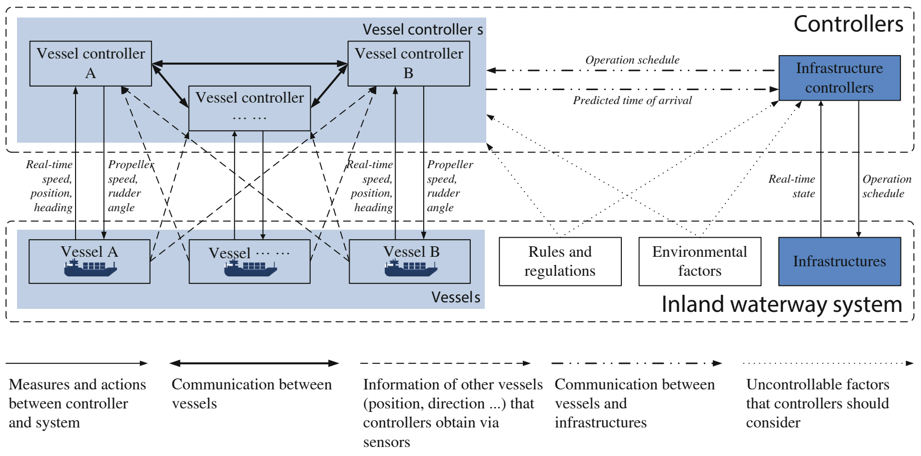
Compared with mobile robotics path planning, the static obstacles in inland waterway networks are usually larger and continuous. Clear passages (waterways) can be found in the map. Moreover, when autonomous vessels are in a hybrid environment where exist vessels operated by humans. In order to ensure safety it is necessary that autonomous vessels comply with navigation rules throughout their missions [2]. Several recent efforts have been made to integrate rules into path planning algorithms [11,19].

In order to find a suitable global path planning algorithm for inland autonomous vessels, in this paper we carry out a comparison among A* and its extensions. We moreover propose a new algorithm called A*BG for autonomous inland vessels. Based on the existing algorithms, A*BG takes advantage of grid search and visibility check, which improves the searching and computational properties.

The remainder of this paper is organized as follows. In Sect. 2, a brief introduction of the inland waterway transport system is provided. A* and its extensions are elaborated on in Sect. 3. Based on this, the new algorithm A*BG is proposed in Sect. 4. Simulation experiments are carried out to assess the performance of the algorithms in Sect. 5. Conclusions and future research are presented in Section 6.

2 Inland Waterway System

The main function of an inland waterway system is to fulfill the transport demand, i.e., to transport goods or people from one place to another. As shown in Fig. 3, two main components in waterway systems are vessels and infrastructures. Vessels are the means of transport. Infrastructures are necessary to guarantee a sound navigation: waterways provide navigable waters; locks create stepped navigational pools with reliable depths; bridges balance the road traffic and the waterborne traffic.



Controllers
Vessel controller s
Vessel controller A
Vessel controller B
Vessel controller … …
Operation schedule
Predicted time of arrival
Infrastructure controllers
Real-time speed, position, heading
Propeller speed, rudder angle
Real-time state
Operation schedule
Vessel A
Vessel … …
Vessel B
Rules and regulations
Environmental factors
Infrastructures
Vessels
Inland waterway system
Measures and actions between controller and system
Communication between vessels
Information of other vessels (position, direction ...) that controllers obtain via sensors
Communication between vessels and infrastructures
Uncontrollable factors that controllers should consider

Fig. 3. Inland waterway system.

Rules and regulations provide suggestions to the skippers. These "rules of the road" specify the types of maneuvers that should be taken in situations where there is a risk of collision. Vessels navigating in waterways are also influenced by the external environment (e.g., wind, current and waves).

The architecture of an autonomous vessel in Fig. 2 can be regarded as the detail explanation of the relation of a vessel controller and a vessel in Fig. 3. When vessels navigating between the origins and destinations, controllers control the propeller and rudder to let the vessel move to desired position. The sensors measure the practical speed and headings of the vessel and provide them to the controllers as feedbacks. Vessel controller can obtain the position and direction of other vessels via sensors. When there is a risk of collision, actions that should be taken to avoid the collision are decided by the controllers. The communication between vessel controllers can help controllers to cooperate with each other.

Infrastructure controllers making schedules with the predicted time of arrival reported by vessel controllers and also keep an eye on the state of the infrastructures (e.g., availability, waiting time and length of the line). In return, the operation schedules also have impacts on vessel controllers decision making on the route, departure time and speed choices.

3 Existing Path Planning Algorithms

In this section, A* and its improved extensions and their characteristics are introduced. The method to apply the algorithms to inland autonomous vessels considering rules and regulations is explained as well.

3.1 A*

A* is the most widely used path planning algorithm, which can be applied on metric or topological map [4]. This algorithm uses a combination of heuristic searching and searching based on the shortest path. A* is defined as best-first algorithm, because each node in the map is evaluated by the function:

$$f(s_{\text{start}}, s, s_{\text{goal}}) = g(s_{\text{start}}, s) + h(s, s_{\text{goal}}) \tag{1}$$

where $g(s_{\text{start}}, s)$ provides the length of the shortest path from a start node s_{start} to node s found so far, $h(s, s_{\text{goal}})$ provides an estimate of the distance from node s to goal node s_{goal}, $f(s_{\text{start}}, s, s_{\text{goal}})$ provides an estimate of the length of a shortest path from the start node s_{start} via node s to the goal node s_{goal}.

A* uses a priority queue *Open* to perform the repeated selection of minimum $f(s_{\text{start}}, s, s_{\text{goal}})$ nodes to expand (expanding a node means this node is a candidate in the shortest path). At each step, the node s with the minimum $f(s_{\text{start}}, s, s_{\text{goal}})$ is removed from *Open*. The unblocked neighbor nodes which are in the line-of-sight of node s are recorded in the set $nbr_{los}(s)$. For each s' in $nbr_{los}(s)$, its related values are updated: $parent(s') = s$, $g(s_{\text{start}}, s') = g(s_{\text{start}}, s) + distance(s, s')$. If s' is already included in *Open*, A* compares the two $g(s_{\text{start}}, s')$ in $nbr_{los}(s)$ and *Open*, and updates the s' with lower $g(s_{\text{start}}, s')$. If not, s' is added to *Open*. The algorithm then repeats this procedure until s is s_{goal}. The length of the path that A* finds is then $f(s_{\text{start}}, s_{\text{goal}}, s_{\text{goal}})$.

The basic A* is restricted to a so-called 8-connectivity grid. This means that the path it finds is based on the connection between the closest possible nodes. The turning angle of each movement is restricted to multiples of 45°, which makes the path linked in a zigzag style. Consequently, the path A* finds is not guaranteed to be the optimal path.

3.2 A* with Larger Neighborhood

The length and smoothness of the paths A* finds are influenced by the connectivity of possible nodes which is determined by so-called '*neighborhood*'. The term '*neighborhood*' indicates the area that A* algorithm explores in a single step, which determines the successor nodes that can be reached from a source node.

One method to improve A* is to enlarge the neighborhood. As shown in Fig. 4, when *neighborhood* = 1 is considered, the algorithm can search 8 successor nodes. This is the most frequently used A*. 8 directions are possible to move into in a single step. When the *neighborhood* increases to 2, 16 more grids and 8 more directions can be searched in each step. Thus, the larger the neighborhood is, the more successor nodes the algorithm can reach, and the more directions are possible to be explored in a single step.

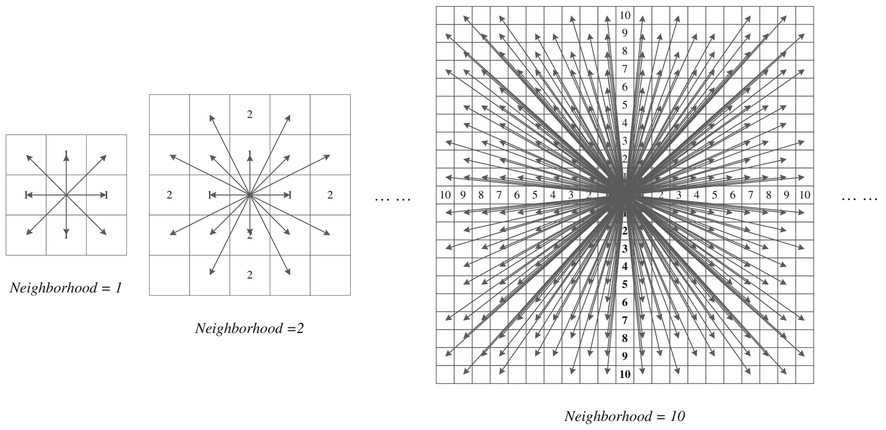


Fig. 4. Neighborhood in A* algorithm.

It is considered that a larger neighborhood results in the discovery of a shorter path due to the increased fineness of possible directions. However, the computation time will also increase since more nodes need to be explored at each step. The trade-off must be made between the optimality of the path and the computation time in terms of requirements during for implementation.

3.3 A* with Post-smoothing

A* with Post-smoothing (A*PS) runs A* on grids and then smooths the resulting path, which often shortens it at the cost of an increase in computation time. Denote by $[s_0, s_1, ..., s_n]$ the path that A* finds on grids, with $s_0 = s_{\text{start}}$ and $s_n = s_{\text{goal}}$. A*PS firstly uses s_0 as the current node. It then find out the farthest node s_i that is in line-of-sight with s_0 on the path from s_n to s_1. Then, A*PS removes the intermediate nodes s_1 to s_{i-1} from the path, thus shortening it. Then s_i becomes the current node and A*PS repeats this procedure until it reaches the end of the path.

A*PS typically finds shorter paths than A* on grids, but is not guaranteed to find the optimal path [3,22]. The reason for this is that it only considers resulting paths and thus cannot make informed decisions regarding other paths during the A* search, which motivates the idea of interleaving smoothing [3].

3.4 Theta*

Theta* is an extension of the A*, which resides in the visibility test between successor nodes and the parent nodes. The main difference between A* and Theta* is that Theta* considers the path from the $parent(s)$ to node s'. In each step, when s (the node with the lowest $f(s_{\text{start}}, s, s_{\text{goal}})$ in *Open*) expanding its successors s' in $nbr_{los}(s)$, the visibility between s' and $parent(s)$ is checked. If $parent(s)$ is visible to s', $parent(s')$ becomes $parent(s)$, and $g(s', s_{\text{start}})$, $h(s', s_{\text{goal}})$ and $f(s_{\text{start}}, s', s_{\text{goal}})$ are updated correspondingly. Thus, $parent(s)$ and s' are directly connected. A detailed description of Theta* can be found in [3].

Theta* reduces some unnecessary heading changes taking advantage of the visibility test. Since Theta* carry out line-of-sight checks between a source node and its neighbor nodes, the computation time of Theta* is longer than A* and A*PS. However, Theta* is not guaranteed to find optimal paths. The parent of a node should be a visible neighbor of the node or a parent of a visible neighbor, which lead to a limitation of expanding nodes [3,22].

3.5 A* Adaptation Considering Navigation Regulations

As mentioned, it is necessary to take the navigation rules and regulations into account when planning paths for inland autonomous vessels. Thus, adaption should be made when applying A* and its extensions to inland vessels.

The main regulations in Dutch inland waterways are the RPR (Rijnvaartpolitiereglement, Rhine Navigation Police Regulations) and the BPR (Binnenvaartpolitiereglement, Inland Waterways Police Regulations). One important item related to global path planning in these two regulations is: 'if two vessels encounter each other with the risk of collision, the vessel not following the starboard side of the waterway must give way to the ship following the starboard side' [15]. Accordingly, vessel controllers generally choose the path on the starboard side of the waterway as preferred path. To reflect this circumstance, the middle line of a waterway is applied to separate the vessel traffic from different directions when implementing the path planning algorithms for autonomous inland vessels.

The paths that the planning algorithms compute are usually close to the border of obstacles. Because of ship-bank interaction, sailing closer to the obstacles will increase the risk of collision [20,24]. For the sake of safety, vessels usually keep a certain distance from the obstacles. Therefore, buffer areas are set around the obstacles. When planning the path, the paths via the buffer areas are still available to vessels with a penalty in path length. In this way, when implementing A* and its extensions, $f(s_{\text{start}}, s, s_{\text{goal}})$ of a grid in a buffer area is larger than its original value when there is no buffer areas.

4 A* on Border Grids

The algorithms presented in Sect. 3 are not guaranteed to find the optimal paths. An algorithm named A* on Visibility Graphs (A*VG) has been proven to be able to find the optimal paths on a map with disjoint polygonal obstacles [1,3]. In A*VG, visibility graphs are constructed before the A* search. If two locations do not pass through any obstacle, an edge is drawn between them to represent the visibility connection. The paths A*VG finds are along the edge and have heading changes only at the border of obstacles. However, A*VG can be slow. Visibility checks need to be performed for every pair of blocked nodes to determine whether or not there should be a visibility edge between them.

The above mentioned algorithms have different advantages and disadvantages. The characteristics of each algorithm are concluded in Table 1. A* on grid maps are simple and with relatively low computation time. However, the path it calculates is usually the longest. Theta* and A*VG take the advantage of the visibility check, and the paths these two algorithm find are relatively shorter. At the same time, their computation times are longer. Based on the comparison, a new algorithm for inland autonomous vessels is proposed next.

Table 1. Summary of the characteristics of the algorithms.

Algorithm	Description	Typical path length	Computation time	Advantage	Disadvantage
A*	A*	Longest	Shortest	Simple; Modifiable	Not any angle; Zigzag style path
A*PS	A* + Post process visibility test	Shorter than A*	Longer than A*	Any angle	Rely on the path found by A*
Theta*	A* + Resides in visibility test	Shorter than A*PS	Longer than A*PS	Any angle	Long computation time
A*VG	A*+Visibility graph	Optimal (with polygonal obstacles)	Longest	Any angle	Long computation time

Inspired by A*VG, in the new algorithm, the border of the obstacles are decomposed into grids. The grids in the line-of-sight of a source node are its successor nodes. This algorithm is represented as A* on Border Grids (A*BG).

Algorithm 1 shows the pseudo code of A*BG. s is the node with the lowest $f(s_{\text{start}}, s, s_{\text{goal}})$ in *Open*. Line-of-sight checks are carried out between the source node s and all border grids. The nodes visible to s are included in the set *Candidates* as the candidate successors to be expanded. For each node s' in *Candidates*, if it is visible to $parent(s)$, its $parent(s')$ and other related values will be updated. Then, the node with lowest $f(s_{\text{start}}, s, s_{\text{goal}})$ in *Open* is assigned to s again. This procedure is repeated until s is s_{goal}.

Algorithm 1. A*BG

```
while $s \neq s_{\text{goal}}$ do
    $s \leftarrow$ node with the smallest $f(s_{\text{start}}, s, s_{\text{goal}})$ in $Open$;
    $Candidates = \emptyset$;
    foreach $n \in BorderGrids$ do
        if $lineofsight(n, s)$ then
            $parent(n) = s$;
            $g(n) = g(s) + distance(n, s)$;    $h(n) = distance(n, s_{\text{goal}})$;
            $Candidates.Insert(n, parent(n), g(s_{\text{start}}, n), h(n, s_{\text{goal}}), f(s_{\text{start}}, n, s_{\text{goal}}))$
    foreach $s' \in Candidates$ do
        if $lineofsight(s', parent(s))$ then
            if $g(s_{\text{start}}, parent(s)) + distance(parent(s), s') < g(s_{\text{start}}, s')$ then
                $parent(s') = parent(s)$;
                $g(s_{\text{start}}, s') = g(s_{\text{start}}, parent(s)) + distance(parent(s), s')$;
        if $s' \in Open$ then
            if $g(s_{\text{start}}, s')$ in $Candidates < g(s_{\text{start}}, s')$ in $Open$ then
                Remove the item $s'$ from $Open$
            else continue;                                    // Do not execute line 18
        $Open.Insert(s', parent(s'), g(s_{\text{start}}, s'), h(s', s_{\text{goal}}), f(s_{\text{start}}, s', s_{\text{goal}}))$
```

Inspired by Theta* and A*VG, A*BG considers the connection of $parent(s)$ and s', and the paths A*BG calculated only have heading changes at where line-of-sight is blocked, which reduces unnecessary heading changes and the path length. Applying border grids instead of visibility graph can greatly reduce the number of visibility test. Using border grids instead of transferring the whole map into grids reduces the amount of nodes the algorithm need to search, which makes the algorithm faster. Moreover, regarding all visible border grids as candidates when expanding successors, A*BG does not influenced by the size of neighborhood and it is able to search in every direction.

5 Simulation Experiments

To test the performance of the algorithms, in this section, we compare A*, A*PS, Theta* and A*BG with respect to their path length and computation time.

5.1 Case Study Areas

Safety is one of the main factors that should be kept in mind when planning for autonomous vessels. Consequently, we choose for our experimental areas inland waterway regions where relatively many accidents have taken place in the past.

The locations of ship accidents occurred in Dutch inland waterways during 2008–2015 are shown in Fig. 5. The places where accidents frequently occur are ports and intersections. Accordingly, we choose an intersection and a port area for carrying out the experiments. Case Study 1 is the area of the Oude Maas.

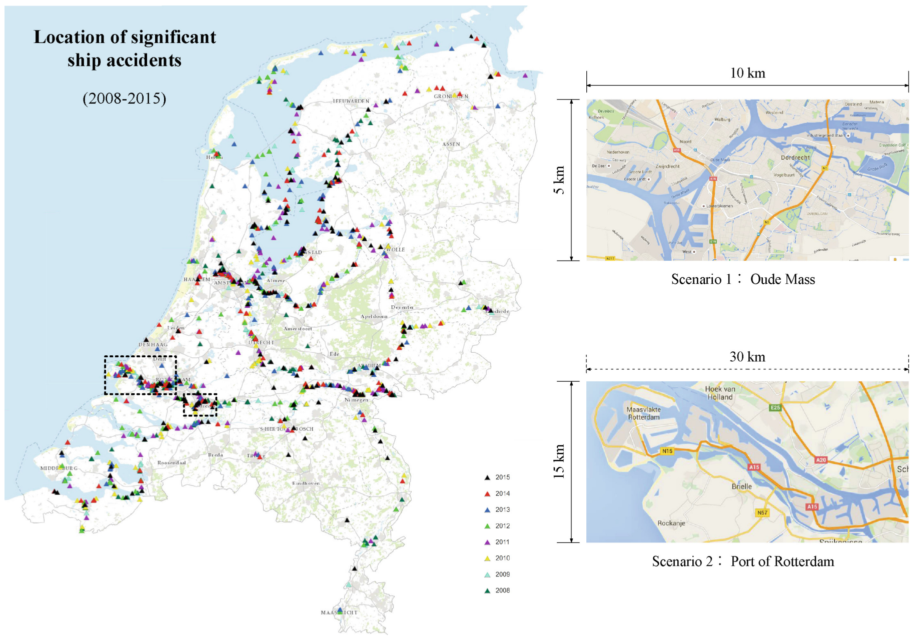


Fig. 5. Location of ship accidents [16] and case study areas (maps taken from [9]).

Table 2. Experimental results.

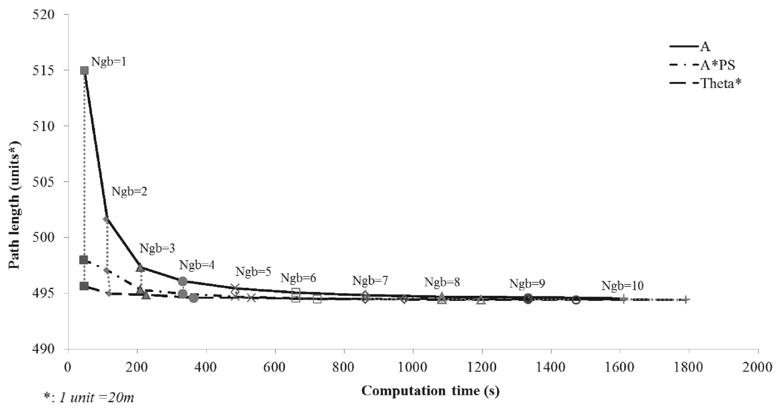
Neighbor-hood	A*		A*PS		Theta*		A*BG	
	Computation time (s)	Path length (unit)	Computation time (s)	Path length (unit)	Computation time (s)	Path length (unit)	Computation time (s)	Path length (unit)
Case Study 1								
1	45.86	514.99	45.93	497.98	46.09	495.63	93.23	494.39
2	112.48	501.63	112.54	497.04	121.57	494.95		
3	210.09	497.32	210.13	495.29	226.37	494.85		
4	333.05	496.06	333.09	494.94	364.32	494.58		
5	485.30	495.45	485.33	494.68	532.33	494.57		
6	660.84	495.07	660.86	494.56	722.57	494.49		
7	862.00	494.84	862.02	494.49	975.32	494.47		
8	1083.59	494.70	1083.61	494.48	1198.15	494.42		
9	1333.76	494.61	1333.79	494.45	1473.00	494.42		
10	1611.61	494.56	1611.63	494.44	1791.48	494.41		
Case Study 2								
1	87.67	540.78	87.77	514.14	94.50	514.24	502.15	510.32
2	241.38	517.98	241.45	512.23	262.70	511.52		
3	468.63	512.68	468.67	511.57	510.03	511.12		
4	771.08	511.49	771.12	511.22	833.87	510.97		
5	1143.20	511.22	1143.25	511.07	1236.45	510.58		
6	1591.53	510.85	1591.57	510.68	1713.21	510.58		
7	2118.28	510.76	2118.32	510.67	2262.04	510.49		
8	2721.35	510.73	2721.39	510.66	2903.01	510.34		
9	3390.31	510.71	3390.35	510.66	3615.47	510.33		
10	4143.80	510.71	4143.84	510.66	4380.30	510.33		

It is an intersection near the Port of Rotterdam, where is the convergent place of river Noord, Benede-Merwede, Dordsche Kil and Oude Maas. Case Study 2 is Port of Rotterdam. It is the largest port in Europe and the place accidents most frequently occurred.

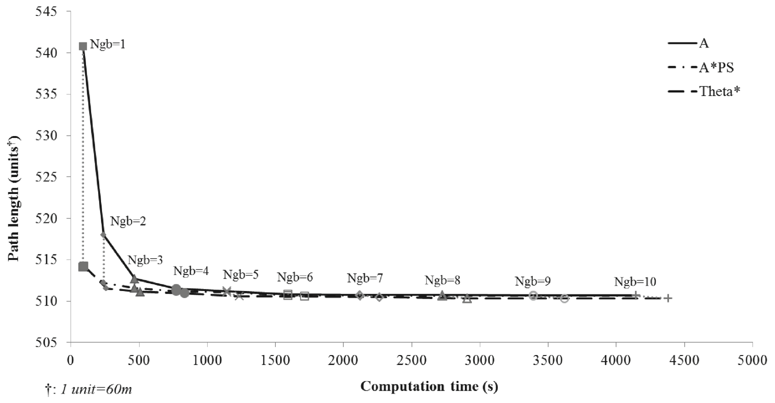
5.2 Setup

All algorithms tested in our experiments are grid-based. Thus, the maps of our case study areas are transfered into 500×250 grids. The length of 1 grid is 1 unit. The buffer area in Case study 1 is two grids near the obstacles and in Case study 2 is 1 grid. Vessels can sail in the buffer area, but with a penalty length. We use middle lines to take the regulations into consideration. To study the influence of the size of neighborhood, the algorithms are carried out with increasing neighborhood (from $neighborhood = 1$ to $neighborhood = 10$).

The algorithms tested in our experiments maintain three values for every node: $g(s_{\text{start}}, s)$ is the length of the path from s_{start} to s; $h(s, s_{\text{goal}})$ is the



(a) Case Study 1.



(b) Case Study 2.

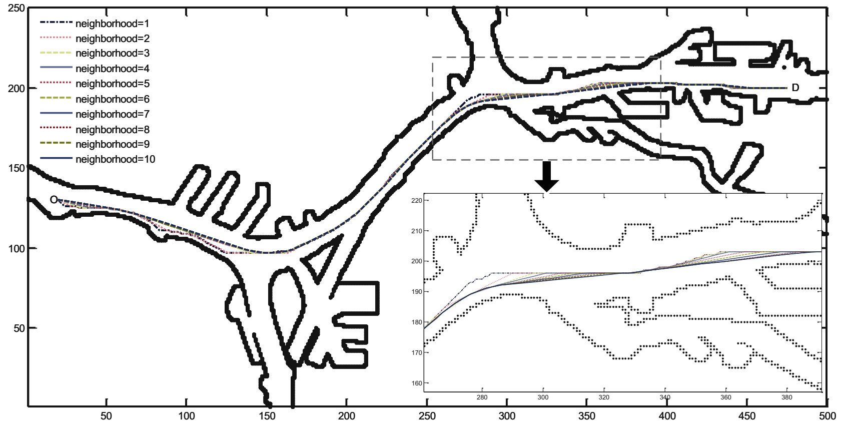
Fig. 6. Experiment results (Ngb: Neighborhood of the algorithms).

straight-line distance of s and s_{goal}; $f(s_{\text{start}}, s, s_{\text{goal}}$ is the sum of $g(s_{\text{start}}, s)$ and $h(s, s_{\text{goal}})$. We use the Euclidean distance in the experiments. The distance between two nodes $N(x, y)$ and $N'(x', y')$ is $\sqrt{(x-x')^2 + (y-y')^2}$. That is, the distance from one grid to an adjacent left\right\up\down node is 1 unit, and to an adjacent diagonal node is $\sqrt{2}$ units.

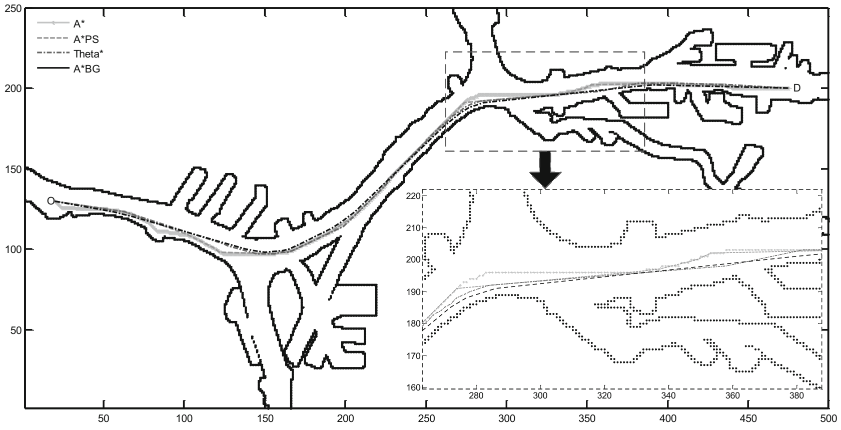
The experiments are run on a PC with a dual-core 3.2GHz Intel(R) Core(TM) i5-3470U CPU and 8GB of RAM. Each case has been repeated for 5 times.

5.3 Experiment Results

The results of the simulation experiments are shown in Table 2 and Fig. 6. The path length and average computation time over 5 repetitions are provided.



(a) Paths calculated by A* with different size of neighborhood.



(b) Paths found by different algorithms ($neighborhood = 1$).

Fig. 7. Paths found in Case study 1.

As shown in Table 2, two case studies show similar relations between the path length, computation time and the size of neighborhood. For A*, A*PS and Theta*, with the increase of neighborhood, the length of the paths becomes shorter, but the computation time increase dramatically as well. When the neighborhood is small, the length of the paths that the three algorithms found differs greatly. The difference decreases when the neighborhood is enlarged (Fig. 6). The path length of the three algorithms then approaches to a certain value.

With respect to A*BG, the size of neighborhood does not influence the results of A*BG. In the two case studies, A*BG shows the best performance. The path it computed is shorter than the shortest path the other three algorithms find, and the computation time is much shorter. Similar to other three algorithms, when the planning area becomes larger, the computation time of A*BG increases.

Figure 7 shows the path found in Case study 1. The paths calculated by A* with different size of neighborhood are shown in Fig. 7(a) as an example to show the impacts of the neighborhood size. Because the length of paths that the tested algorithms find differs greatly when $neighborhood = 1$, this situation is chosen as an example to present the difference of the paths found by different algorithms in Fig. 7(b). The main difference among the paths lies in the bend segments. The algorithms which find the shorter paths, A* when $neighborhood = 10$ and A*BG, find smoother paths at the bend segments.

6 Conclusions and Future Research

Autonomy is seen as a possibility for maritime transport to meet today's and tomorrow's challenges. In realizing autonomous navigation, path planning plays an important role. As a starting point of path planning for inland autonomous vessels, a modified A* algorithm (A*GB) is proposed to solve the global path planning problem. In this paper, we carry out experiments to compare the performance of A*, A*PS, Theta* and A*GB. Two places where ship accidents frequently occurred in the past are chosen as case study areas. The path length and computation time of each algorithms is analyzed. Trading off the path length and computation time, the performance of A*GB is more satisfying for inland autonomous vessels'path planning.

There are several directions in which this research will be extended. Firstly, when the planning area is larger, the computation time increases and the fineness of the grids also decreases, which affect the performance of the algorithm. The principle of Model Predictive Control can then be used to solve this problem. Long voyages are divided into smaller segments, after which a vessel updates its path at subsequence decision steps. Secondly, in this paper, the impact of infrastructures is not included. As important components in inland waterway system, infrastructures such as locks and bridges have great impact on inland shipping. Most delays are caused by operation of locks and bridges. Global path planning should also consider these influences. Finally, real-time information should also be taken into account. If preplanned paths are blocked due to accidents, or if there is a long waiting time at a certain lock, it is important that the vessel can replan its path according to real-time information.

Moreover, the global path planner considered here only provides reference paths considering static obstacles for an autonomous vessel. Algorithms for local path planning, i.e., collision avoidance, are needed to deal with the moving obstacles. These moving obstacles not only include other autonomous vessels, but also vessels operated by humans. With different obstacles, the information available is different. Besides, the actions controllers take and the resulting trajectories of the vessels operated by humans are uncertain. How the autonomous vessel communicates and coordinates with others using different sources of information and deals with uncertainties are future research problems.

Acknowledgment. This research is supported by the China Scholarship Council under Grant 201426950041.

References

1. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications, pp. 323–333. Springer, New York (2008)
2. Campbell, S., Naeem, W., Irwin, G.: A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres. Ann. Rev. Control **36**(2), 267–283 (2012)
3. Daniel, K., Nash, A., Koenig, S., Felner, A.: Theta*: any-angle path planning on grids. J. Artif. Intell. Res. **39**(2010), 533–579 (2010)
4. Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., Jurišica, L.: Path planning with modified a star algorithm for a mobile robot. Procedia Eng. **96**, 59–69 (2014)
5. Economic commission for Europe: inventory of most important bottlenecks and missing links in the E waterway network. Technical report. ECE/TRANS/SC.3/159/Rev.1, Economic Commission for Europe, Inland Transport Committee, United Nations (2013)
6. European Commission: Naiades II: Towards quality inland waterway transport. Technical report COM 623, European Commission (2013)
7. European Commission: The European Union explained: Transport. Technical report European Commission (2014)
8. Froese, J.: Safe and efficient port approach by vessel traffic management in waterways. In: Ocampo-Martinez, C., Negenborn, R.R. (eds.) Transport of Water versus Transport over Water. Operations Research/Computer Science Interfaces Series, vol. 58, pp. 281–296. Springer, New York (2015)
9. Google Maps: Port of Rotterdam, Street map (2016). https://www.google.nl/maps/@51.8820487,4.4343202,10.75z
10. Hekkenberg, R.: Technological challenges and developments in European inland waterway transport. In: Ocampo-Martinez, C., Negenborn, R.R. (eds.) Transport of Water versus Transport over Water. Operations Research/Computer Science Interfaces Series, vol. 58, pp. 297–313. Springer, New York (2015)
11. Kuwata, Y., Wolf, M., Zarzhitsky, D., Huntsberger, T.: Safe maritime autonomous navigation with COLREGS using velocity obstacles. IEEE J. Oceanic Eng. **39**(1), 110–119 (2014)
12. Lazarowska, A.: Ship's trajectory planning for collision avoidance at sea based on ant colony optimisation. J. Navig. **68**(2), 291–307 (2015)

13. Li, S., Negenborn, R.R., Lodewijks, G.: Distributed constraint optimization for addressing vessel rotation planning problems. Eng. Appl. Artif. Intell. **48**(2016), 159–172 (2016)
14. Movares Projectteam MNV'13: Monitoring nautische veiligheid 2013. Technical report, Rijkswaterstaat Water, Verkeer en Leefomgeving, Afdeling Veiligheidsmanagement en Verkeersveiligheid (2013)
15. Rijkswaterstaat: Binnenvaartpolitiereglement (1983). http://wetten.overheid.nl/BWBR0003628/2016-01-01#DeelI_Hoofdstuk6_AfdelingI_Artikel6.01
16. Rijkswaterstaat: Scheepsongevallen significant (2016). https://geoweb.rijkswaterstaat.nl/westnederlandnoord/GeoWeb41/?Viewer=WNN_Scheepsongevallen
17. Rijkswaterstaat: Scheepsongevallendatabase (2016). https://www.rijkswaterstaat.nl/zakelijk/verkeersmanagement/scheepvaart/scheepsongevallenregistratie/index.aspx
18. Sariff, N., Buniyamin, N.: An overview of autonomous mobile robot path planning algorithms. In: Proceedings of the 4th Student Conference on Research and Development, pp. 183–188. Selangor, Malaysia (2006)
19. Shah, B.C., Švec, P., Bertaska, I.R., Sinisterra, A.J., Klinger, W., Ellenrieder, K., Dhanak, M., Gupta, S.K.: Resolution-adaptive risk-aware trajectory planning for surface vehicles operating in congested civilian traffic. Autonomous Robots, first Online (2015)
20. Siegwart, R., Nourbakhsh, I.R., Scaramuzza, D.: Introduction to Autonomous Mobile Robots, 2nd edn. MIT Press, Cambridge (2011)
21. Statheros, T., Howells, G., Maier, K.M.: Autonomous ship collision avoidance navigation concepts, technologies and techniques. J. Navig. **61**(1), 129–142 (2008)
22. Uras, T., Koenig, S.: An empirical comparison of any-angle path-planning algorithms. In: Proceedings of the 8th Annual Symposium on Combinatorial Search, pp. 206–210. Ein Gedi, Israel (2015)
23. Vaneck, T.W.: Fuzzy guidance controller for an autonomous boat. IEEE Control Syst. Mag. **17**(2), 43–51 (1997)
24. Vantorre, M., Delefortrie, G., Eloot, K., Laforce, E.: Experimental investigation of ship-bank interaction forces. In: Proceedings of International Conference on Marine Simulation and Ship Maneuverability, pp. 1–9. Kanazawa, Japan (2003)
25. Verstichel, J., Causmaecker, P.D., Spieksma, F., Berghe, G.V.: The generalized lock scheduling problem: an exact approach. Trans. Res. Part E: Logistics Transp. Rev. **65**, 16–34 (2014)
26. Xin, J., Negenborn, R.R., Corman, F., Lodewijks, G.: Control of interacting machines in automated container terminals using a sequential planning approach for collision avoidance. Transp. Res. Part C: Emerg. Technol. **60**(2015), 377–396 (2015)
27. Zheng, H., Negenborn, R.R., Lodewijks, G.: Predictive path following with arrival time awareness for waterborne AGVs. Transportation Research Part C: Emerging Technologies (2015). http://dx.doi.org/10.1016/j.trc.2015.11.004