Journal of *Marine Science and Engineering*

MDPI

Article

Improved D* Lite Algorithm for Ship Route Planning

Yuankui Li [1,*], Fang Yang [1], Xinyu Zhang [1], Dongye Yu [1] and Xuefeng Yang [2]

[1] College of Navigation, Dalian Maritime University, Dalian 116026, China; funfun678@163.com (F.Y.); zhangxy@dlmu.edu.cn (X.Z.); yudongye@dlmu.edu.cn (D.Y.)
[2] School of Shipping and Naval Architecture, Chongqing Jiaotong University, Chongqing 400074, China; yangxuefeng536@126.com
* Correspondence: liyuankui@dlmu.edu.cn; Tel.: +86-18941190273

Abstract: To address the issue of intelligent ship route planning, a ship planning method based on the improved D* Lite algorithm is proposed. Firstly, a navigation environment grid map is constructed using the acquired meteorological and hydrological datasets. The grids are divided into navigable and non-navigable according to navigation requirements, and a route planning model is built. Secondly, the heuristic function and the path function of the D* Lite algorithm are improved. The heuristic function is optimized and weighted, and a risk factor is introduced into the path function to enhance efficiency of path planning while maintaining a safe distance between the planned route and obstacles. Finally, by dynamically adjusting the search step length and the selectable directions of the D* Lite algorithm, the number of waypoints is reduced, and the voyage of the planned route is shortened, resulting in a smooth and collision-free route of ships. The effectiveness of the proposed algorithm is verified through three sets of simulation experiments. The simulation results show that the proposed method in this paper is more suitable for ship route planning and ship maneuvering in practice and can effectively avoid non-navigable grids while optimizing path length, path smoothness, and computation time, making the routes more aligned with actual navigation tasks.

Keywords: intelligent route planning; improved D* lite algorithm; route optimization; curve optimization

Citation: Li, Y.; Yang, F.; Zhang, X.; Yu, D.; Yang, X. Improved D* Lite Algorithm for Ship Route Planning. *J. Mar. Sci. Eng.* **2024**, *12*, 1554. https://doi.org/10.3390/jmse12091554

Academic Editor: Marco Cococcioni

Received: 3 August 2024
Revised: 28 August 2024
Accepted: 4 September 2024
Published: 5 September 2024

1. Introduction

With the rapid development of the shipping industry and the continuous maturity of intelligent technology, autonomous navigation technology for ships has attracted significant attention. The core of intelligent navigation lies in the autonomous navigation capability of ships, where they can safely and efficiently complete navigation tasks without human intervention by adapting to environmental changes and mission requirements. Against this backdrop, intelligent ship route planning has become a crucial research area in the shipping industry. The goal is to use intelligent methods to allow ships to plan safe and efficient routes based on meteorological information, sea conditions, and voyage tasks, thus improving the safety and economic efficiency of ship navigation. Additionally, as the problem of intelligent route planning becomes increasingly complex, the demands on algorithms are also becoming higher.

Currently, the methods used in the field of route planning include the isochrone method, graphical search algorithms, and intelligent optimization algorithms. Graphical search algorithms are path planning methods based on environmental modeling. These methods plan navigation routes by gridding the navigation environment and then applying appropriate algorithms such as Dijkstra's, A*, D*, and D* Lite. Among these, the A* algorithm is widely used to solve path planning problems in static environments due to its heuristic search characteristics. However, in actual maritime navigation, as navigation time increases, the accuracy of meteorological and sea condition forecasts decreases. Furthermore, the validity of the forecast leads to the inability to predict all the environmental information accurately along the whole routes. In these cases, path planning algorithms

designed for static environments often fail to adapt to unknown or partially known navigation environments. Effectively handling dynamically changing meteorological and sea condition information becomes a challenge in route planning.

To address these issues, Wu et al. [1] incorporated ships' loss in speed caused by wind and waves into the path cost estimation to improve the heuristic function of the A* algorithm. They periodically updated maritime meteorological information and repeatedly used the improved A* algorithm to propose a dynamic route planning method for ships considering complex meteorological changes. Li et al. [2] extended the two-dimensional grid data of a single time to a three-dimensional matrix to establish a three-dimensional dynamic maritime model with complex meteorological conditions. Based on actual navigation conditions, they improved the A* algorithm and proposed a collaborative optimization method for ship course and speed over the entire voyage. Walther et al. [3] viewed the ship weather routing problem as either a single-objective or multi-objective optimization problem, modeling it as a constrained graph problem, a constrained nonlinear optimization problem, or a combination of both. Vettor et al. [4] developed a ship route optimization system for ship weather routing that can model any sea condition to enhance ships' response, taking into account wave directional spreading and utilizing the robust capabilities of multi-objective evolutionary algorithms. Pennino et al. [5] proposed a new adaptive weather routing model based on the Dijkstra shortest path algorithm. This model selects the optimal route based on the weather forecasts and sea condition information ships are expected to encounter, thereby enhancing ships' seaworthiness. Zaccone et al. [6] parameterized ship navigation as a multi-stage decision process, thereby formulating a dynamic programming optimization problem. By estimating wave and wind conditions for each route segment using weather forecast maps, while considering ship movements caused by waves and increased resistance, a ship route optimization method based on three-dimensional dynamic programming was proposed. Geng et al. [7] addressed the autonomous navigation problem of Maritime Autonomous Surface Ships (MASSs) by developing a velocity obstacle (VO) model for both dynamic and static obstacles and proposed a greedy interval-based motion planning algorithm, which improved the safety and efficiency of ship navigation. Zaccone et al. [8] proposed a dynamic programming method for collision avoidance in autonomous vessels, formulating the optimal path planning problem as a multi-stage decision process to address dynamic maritime environments. Xiao et al. [9] used the improved D* Lite algorithm in polar sea environments, introducing the Polar Operational Limit Assessment Risk Indexing System (POLARIS) risk model to construct a grid environment map with a navigation risk index, then proposed an improved D* Lite dynamic path planning method combined with POLARIS for polar operating restriction risk assessment. Yao et al. [10] applied a new dynamic obstacle modeling method in complex environments such as ports and coasts, combining the D* Lite algorithm with the constrained artificial potential field method. They proposed a real-time path planning method for unmanned boats in complex environments based on the D* Lite algorithm.

Although the D* Lite algorithm performs well in dynamic environments, traditional D* Lite algorithms still face several issues in practical applications, including excessive node expansion, low planning efficiency in complex environments, poor path smoothness, and paths that are too close to obstacles. To address these problems, Yu et al. [11] improved the D* Lite algorithm by optimizing the path cost function and limiting the expansion direction of nodes. This work increased the computational efficiency of the traditional D* Lite algorithm, shortened the path length using inverse distance weighting interpolation, and introduced the Dubins method to improve smoothness of the local path. Jin et al. [12] proposed a Conflict-Based D* Lite Search (CBS-D*) algorithm, which expands the search range to three neighborhoods, improving path planning efficiency and safety in complex environments. Xie et al. [13] adopted a sub-node selection priority strategy to improve path safety and weighted the heuristic function to enhance path planning efficiency. Lin et al. [14] integrated the D* Lite algorithm with enhanced neural network algorithms, proposing a deep learning fusion algorithm for path planning in complex large maps. Le et al. [15]

introduced a new D* Lite with Reset (DLR), which establishes a threshold based on the ratio of the traversal path length to the total path length. When this threshold is surpassed, the D* Lite algorithm is reset, thereby enhancing performance in complex environments. Przybylski et al. [16] introduced the D* Extra Lite algorithm, which improves computational efficiency by reinitializing the affected search space and pruning branches of the search tree, outperforming the traditional D* Lite algorithm. Reyes et al. [17] proposed the D* Lite with Poisoned Reverse algorithm, which identifies and removes nodes that are double-counted during updates. This approach minimizes node expansion and prevents path retracing in unknown environments. Maurović et al. [18] developed a path planning algorithm for active Simultaneous Localization and Mapping (SLAM) that enhances a robot's localization accuracy continuously while ensuring smooth movement. Chao et al. [19] integrated the principles of Rapidly exploring Random Tree (RRT*) with D* Lite, using D* Lite's grid search strategy to sample search graphs generated from the grid search process, thereby refining existing path lengths. Osmankovic et al. [20] proposed a multi-stage technique based on the fast D* Lite algorithm for global path cost calculation, which addresses constrained optimal control problems using a Model Predictive Control (MPC) planning approach. Wu et al. [21] introduced improved Jump Point Search into the D* Lite algorithm, achieving dual optimization of planning time and path length, and combined the D* Lite algorithm with time elasticity methods to ensure path safety. Huang et al. [22] introduced a method that combines the Lazy Eye algorithm with distance transform to improve the D* Lite algorithm, significantly enhancing path safety and smoothness. Xu [23] proposed a fusion algorithm with layered planning, using D* Lite for global planning and enhanced neural network algorithms for local planning, greatly optimizing path planning time. Zhang et al. [24] divided an environmental map into cells, set core grids within the cells, and constructed search linked lists sequentially to guide the correct search direction, improving the path planning efficiency of the D* Lite algorithm. Hu et al. [25] proposed an algorithm that combines the improved D* Lite algorithm with the artificial potential field method, enhancing planning efficiency while achieving global planning and local obstacle avoidance. Fu et al. [26] introduced the concept of attractive fields from the artificial potential field (APF) and proposed a Continuous D* Lite (CD* Lite) algorithm for continuous dynamic path planning, which improves planning efficiency and smooths the path.

Based on the advantages of the D* algorithm in dynamic environments, these scholars have made many improvements to the application of the D* algorithm in their research fields. However, for ship route planning, it still has several issues, including excessive node expansion, low efficiency in complex environments, a high number of path waypoints, and paths that are too close to obstacles. Therefore, this paper applies the D* Lite algorithm to the problem of intelligent ship route planning and makes the following improvements: the heuristic function of the D* Lite algorithm is optimized and weighted; a risk factor is introduced into the cost function; and the algorithm's search step length and selectable directions are dynamically adjusted. These improvements aim to achieve safe and efficient route planning while ensuring that the planned routes are more aligned with actual navigation tasks and meet ships' navigation requirements.

2. Model Construction

The D* Lite algorithm, developed by Koenig S. and Likhachev M. [27], is an incremental, heuristic-based dynamic path planning algorithm. One of its notable advantages is its high adaptability to dynamic environments, allowing it to quickly find optimal paths as the environment changes. The D* Lite algorithm operates on a grid-based map by simplifying complex environments into smaller cells, where each cell can be defined to be navigable or non-navigable according to the issue. These grid cells enhance the efficiency and adaptability of path planning. To apply the D* Lite algorithm for planning the routes of ships, it is necessary to construct a navigational grid map. Thus, this paper uses a grid-based approach to process the navigation environment, converting it into a simple geometric mode which serves as the environmental model for maritime navigation and then constructs a ship

route planning model by improving the D* Lite algorithm to overcome its limitations and improves overall planning effectiveness.

2.1. Navigation Environment Model

The navigation environment model plays a crucial role in ensuring ship navigation safety and route planning, providing foundational data and decision support for route planning. In addition to being influenced by known environmental factors such as land, reefs, and navigational aids, ships at sea are also affected by unknown factors like water depth, wind, and waves. Considering the effect of the Earth's curvature and the subsequent construction of the grid map, this paper uses an equidistant cylindrical projection for mapping. The environmental modeling is informed by meteorological and hydrological data obtained from the European Centre for Medium-Range Weather Forecasts (ECMWF) [28] and ocean depth data acquired from the General Bathymetric Chart of the Oceans (GEBCO) [29]. These data are grid data with a resolution of $0.5^\circ \times 0.5^\circ$. The environmental models generated using MATLAB 2023b are visualized and shown in Figure 1a,b. The size of the grid corresponds to the resolution of the environmental data, which is $0.5^\circ \times 0.5^\circ$. The mapping relationship between the center coordinates of each grid cell and the latitude and longitude coordinates of the environment map is as follows:

$$\begin{cases} lat = n \times h + lat_{first} \\ lon = m \times h + lon_{first} \end{cases} \tag{1}$$

where lat and lon represent the latitude and longitude of the grid point coordinates, respectively; m and n represent the horizontal and vertical coordinates of the grid map, respectively; h represents the interval of latitude and longitude between two adjacent coordinates, which is set to 0.5° in this paper; and lat_{first} and lon_{first} represent the initial latitude and longitude coordinates of the selected maritime area, respectively.

The distance L between nodes k and l is calculated using the great-circle distance formula as follows:

$$L = R\arccos(\sin lat_k \times \sin lat_l + \cos lat_k \times \cos lat_l \times \cos(lon_k - lon_l)) \tag{2}$$

where R is the Earth's average radius (6371 km); lat_k and lon_k represent the latitude and longitude of node k, respectively; and lat_l and lon_l represent the latitude and longitude of node l, respectively.

Taking into account ships' specifications and performance factors, a pre-processing step is applied to the selected navigational grids of the sea area. A binary classification method is used to divide the navigational sea area into two categories: to ensure route planning safety, grids containing navigation obstacles such as land, islands, and reefs are marked as non-navigable grids (the values are set to 0), while those without obstacles are marked as navigable grids (the values are set to 1). Additionally, taking into account wind speed, wave height, water depth, and a ship's performance et al., further evaluation is performed to determine if the navigable grids meet ships' navigation safety requirements. Navigable grids that do not meet these safety requirements are reset as non-navigable (the values are changed from 1 to 0). This processing method effectively reduces the search space for the algorithm, thus enhancing search efficiency.

The binary processing result considering the water depth requirements in Figure 1a is shown in Figure 1c, while the binary processing result further considering wind speed and wave height requirements is shown in Figure 1d. In these binary grids, white represents navigable grids, and black represents non-navigable grids that need to be avoided during route planning, indicating areas where navigation is obstructed. Different ships have varying navigation safety requirements depending on their performance and specific conditions, and thresholds can be set accordingly. In this paper, the thresholds are set as follows: a water depth threshold of -15 m, a wind speed threshold of 22 knots, and a wave height threshold of 3.5 m. Grids exceeding these thresholds are designated as

non-navigable grids, as heavy wind and wave areas are indicated by the red-bordered area in Figure 1d.

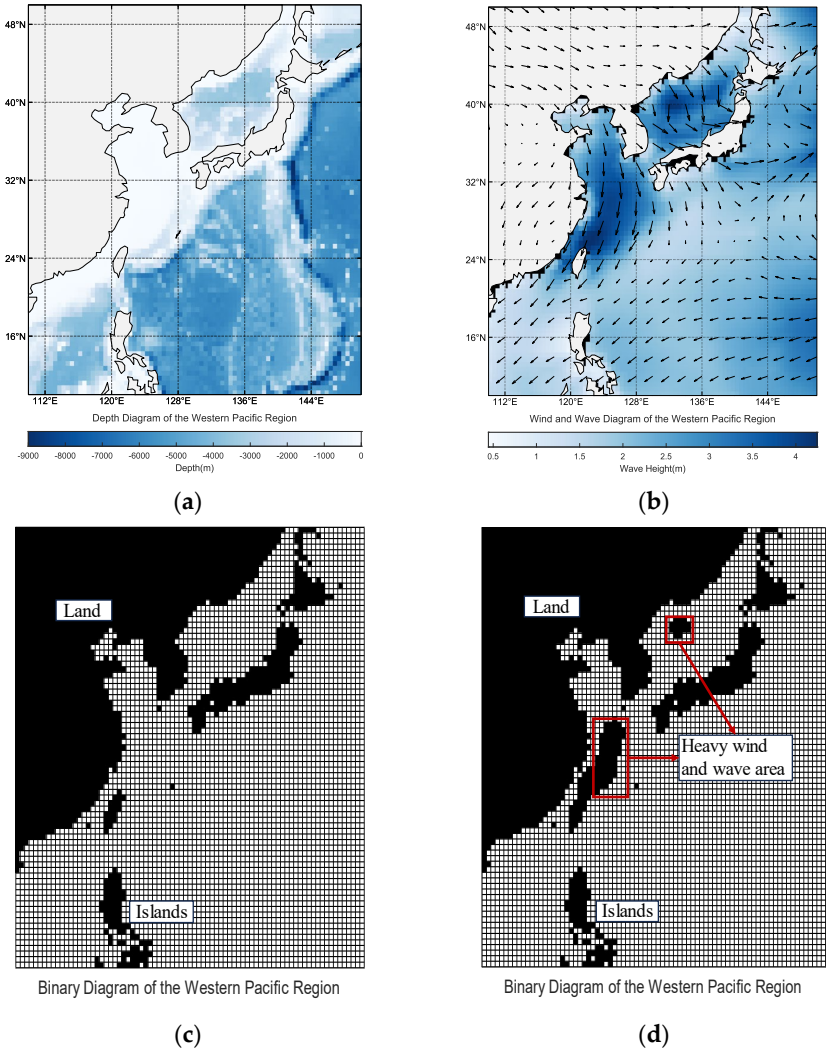


Figure 1. Model of the navigation environment. (**a**) Depth map of the sea area; (**b**) Wind and wave map of the sea area; (**c**) Grid map of the sea area considering water depth requirements; (**d**) Grid map of the sea area considering water depth, wind speed, and wave height requirements.

2.2. Route Planning Model

The navigation environment is complex and variable, making it difficult to predict weather conditions along the entire route. As the voyage progresses, the accuracy of weather forecasts decreases, posing significant challenges to the planning, safety, and stability of ship voyage; this requires a route planning method in order to have the characteristics of dynamically adjusting and rapidly responding to real-time environmental changes.

The D* Lite algorithm, which utilizes a reverse search approach, is well suited to address dynamic and uncertain maritime route planning problems. It can quickly adjust routes in response to environmental changes, thereby ensuring navigational safety. The performance of D* Lite is comparable to that of the original D* algorithm but has greater efficiency. Although D* Lite is a global path planning algorithm, its incremental nature allows it to efficiently handle local environmental changes and re-plan paths in response to unknown or sudden obstacles. The incremental feature means that when the environment changes after a path has already been found, D* Lite can dynamically adapt without recalculating the entire global path. It effectively uses previously acquired search information to update the existing path. Additionally, D* Lite is a heuristic algorithm that utilizes heuristic

information to guide the search process, thus helping to find the optimal path more quickly. This approach improves the efficiency and performance of path planning. Therefore, this paper designs and solves a ship route optimization model based on the D* Lite algorithm.

The D* Lite algorithm's core is assuming unknown areas are free spaces. Based on this, it initially plans an optimal path from the destination to the start by evaluating the shortest path costs to each node and establishing a path field. When ships navigate along a pre-planned route and encounter unknown obstacles, the algorithm designates these map positions as obstacle spaces. It then updates the heuristic values and cost estimates from the destination to the new start, using existing path field information to re-plan an optimal path. This approach reduces redundant computations, improves re-planning efficiency, and ensures the safety of planned routes.

However, when applied to ship route planning, the D* Lite algorithm faces several challenges, including the expansion of too many nodes, inefficiency in complex environments, paths that are too close to obstacles, and an excessive number of waypoints. This paper proposes improvements to the traditional D* Lite algorithm:

(1) The heuristic function is optimized and weighted to address the issue of excessive expanded nodes and low efficiency in complex environments.

(2) A risk factor is introduced into the cost function to ensure that the planned route maintains a safe distance from obstacles, addressing the issue of routes being too close to obstacles.

(3) The algorithm's search step length and selectable directions are dynamically adjusted to optimize path curvature, thereby reducing the number of waypoints in the planned route, as well as achieving smooth and continuous turning of ships at waypoints.

These improvements result in a smooth and collision-free ship route that meets navigational requirements. The route planning model is shown in Figure 2.

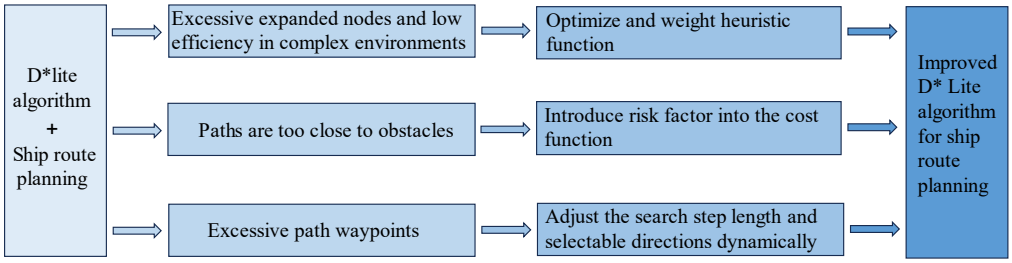


Figure 2. Route planning model.

3. Algorithm Design

In this section, we explore the D* Lite algorithm's core principles and its enhancements for ship route planning, including key function definitions, node state updates, heuristic and path function optimizations for efficiency and safety, and a curve optimization strategy for smoother navigation. We conclude with detailed flowcharts and pseudocode to illustrate the algorithm's practical implementation.

3.1. Principles of the D Lite Algorithm*

In the D* Lite algorithm, the nodes to be expanded are stored in the priority queue U and their priority is determined based on the evaluation function $key(s)$. The node with the smallest $key(s)$ is selected for expansion, where a smaller $key(s)$ indicates a more optimal path through that point. The D* Lite algorithm progressively expands the most optimal nodes by continuously updating $key(s)$ to eventually obtain the global optimal path. The $key(s)$ in the D* Lite algorithm consists of two parts: $k_1(s)$ and $k_2(s)$. This composition helps to avoid issues where nodes with the same priority might affect the selection of the optimal node. When sorting nodes, $k_1(s)$ is compared preferentially; then, only if $k_1(s)$ of the nodes are equal, $k_2(s)$ is compared. The definition of the $key(s)$ [11] is as follows:

$$key(s) = \begin{cases} k_1(s) = \min(g(s), rhs(s)) + h(s, s_{start}) + k_m \\ k_2(s) = \min(g(s), rhs(s)) \end{cases} \quad (3)$$

where $g(s)$ is the path function, representing the cost from the destination node s_{des} to the current node s, representing the "actual distance" already sailed; it helps the algorithm select the optimal path, typically obtained from $rhs(s)$. $rhs(s)$ represents the estimated cost from the destination node s_{des} to the current node s and is a look-ahead value of $g(s)$. The definition of $rhs(s)$ [11] is as follows:

$$rhs(s) = \begin{cases} 0 & , s = s_{des} \\ \min_{s' \in succ(s)}(g(s') + c(s, s')) & , otherwise \end{cases} \quad (4)$$

where s' is node next to s; $succ(s)$ is the set of s'; $g(s')$ is the actual cost from the destination node s_{des} to node s'; and $c(s, s')$ is the cost between node s and s'.

$h(s)$ is the heuristic function, $h(s, s_{start})$ represents the cost from current node s to start node s_{start}. It estimates the cost from the current node to the start node, representing the "remaining distance" that needs to be sailed. Its purpose is to guide the search process toward to the start node, thereby accelerating the search speed.

k_m is the key modifier, representing the actual distance from s_{last} to start node s_{start}, where s_{last} is the previous start node before the path is reconstructed.

When the navigation environment changes, it is necessary to update the node statuses based on the values of $g(s)$ and $rhs(s)$. Nodes can be categorized into three states:

(1) Locally consistent state: $g(s) = rhs(s)$ indicates that the node is stable and does not require recalculation or status updates.

(2) Locally over-consistent state: $g(s) > rhs(s)$ indicates that s can achieve a shorter path to the destination by using a s'. This may occur if an obstacle on the path has been removed or if a shorter path has been found. In this case, $g(s)$ is set to $rhs(s)$, s changes to a locally consistent state, and the impact is propagated to all predecessor nodes.

(3) Locally under-consistent state: $g(s) < rhs(s)$ indicates that a s' has suddenly become an obstacle, causing the path cost from s' to the s_{des} to increase. In this case, $g(s)$ is set to infinity, the node is removed from the priority queue, and the $key(s)$ of this node and its related nodes are recalculated and updated. The status of the node changes to a locally over-consistent state and is reinserted into the priority queue U for further processing, eventually reaching a locally consistent state.

3.2. Improved D Lite Algorithm*

To better adapt to ship route planning issues and meet actual navigation requirements, we have improved the D* Lite algorithm by optimizing two functions (heuristic function optimization and path function optimization) and enhancing curve optimization.

3.2.1. Function Optimization

In the D* Lite algorithm, each node optimization requires the calculation and comparison of the evaluation function $key(s)$ in the priority queue. The accuracy and computation speed of $key(s)$ play a crucial role in the algorithm's path planning efficiency and node expansion. The value of evaluation function $key(s)$ is primarily influenced by the heuristic function $h(s)$ and the path function $g(s)$.

To solve the problems of excessive node expansion and paths being too close to obstacles in the D* Lite algorithm, this paper improves the heuristic and path functions to optimize the algorithm's performance and enhance the safety of the planned paths. It should be noted that in this paper, the distance formula chosen for heuristic and path functions is used to guide the search direction, while the actual distance calculation employs the great-circle distance formula, as shown in Equation (2).

1. Heuristic Function Optimization

In a planar grid graph, common distance evaluation functions between two points (x_1, y_1) and (x_2, y_2), including the following:

$$\text{Manhattan Distance}: \ D = d(|x_1 - x_2| + |y_1 - y_2|) \quad (5)$$

$$\text{Chebyshev Distance}: \ D = d(\max(|x_1 - x_2|, |y_1 - y_2|)) \tag{6}$$

$$\text{Euclidean Distance}: \ D = \sqrt{(d(x_1 - x_2))^2 + (d(y_1 - y_2))^2} \tag{7}$$

$$\text{Diagonal Distance}: \ D = d(|x_1 - x_2|) + d(|y_1 - y_2|) + (d' - d)\min(|x_1 - x_2|, |y_1 - y_2|) \tag{8}$$

where d represents the movement distance between horizontally or vertically adjacent cells, and d' represents the movement distance between diagonally adjacent cells.

The above distance functions have different applicability and efficiency characteristics for different types of grid maps and pathfinding, as shown in Table 1.

Table 1. The heuristic function and their characteristics.

Heuristic Function	Characteristics	Remarks
Manhattan Distance	Appropriate for grid maps where movement is confined to horizontal and vertical directions.	
Chebyshev Distance	Appropriate for grid maps where movement is confined to horizontal and vertical directions.	
Euclidean Distance	Applied to calculate the shortest distance between two points and suitable for distance calculations in any direction. However, it is more complex to compute.	D* Lite
Diagonal Distance	Suitable for diagonal movements, offering simple calculations and fewer node expansions, which can effectively improve computation speed while accurately describing the path cost in grid maps.	This work

The traditional D* Lite algorithm uses Euclidean Distance as the heuristic function, which results in a high number of expanded nodes. Therefore, this paper optimizes the heuristic function based on considerations of computation time and the number of expanded nodes, choosing the Diagonal Distance function as the heuristic function. Although Diagonal Distance is simple with less expanded nodes and can effectively reduce computational time, considering that the incremental nature of the D* Lite algorithm utilizes the path field information searched during the initial planning, the number of expanded nodes cannot be too few. Thus, a weighted adjustment is applied to the Diagonal Distance function.

The effects of different weights under the same conditions obtained from our experiments are shown in Table 2.

Table 2. Comparison of the effects of different weights.

Heuristic Function Weight	Number of Expanded Nodes	Computation Time (s)
0.7	4186	137.36
0.8	3182	104.22
0.9	2343	78.02
1.0	1646	54.99
1.1	1428	48.13
1.2	1342	45.55

As shown in Table 2, the heuristic function weight directly impacts the searching performance and the computation time of the algorithm. The following are shown:

(1) If the weight is set too small, that is, only the path function $g(s)$ is effective, the D* Lite algorithm behaves similarly to a breadth-first search algorithm. In this case, the found path is the shortest, but the algorithm expands more nodes and runs slowly.

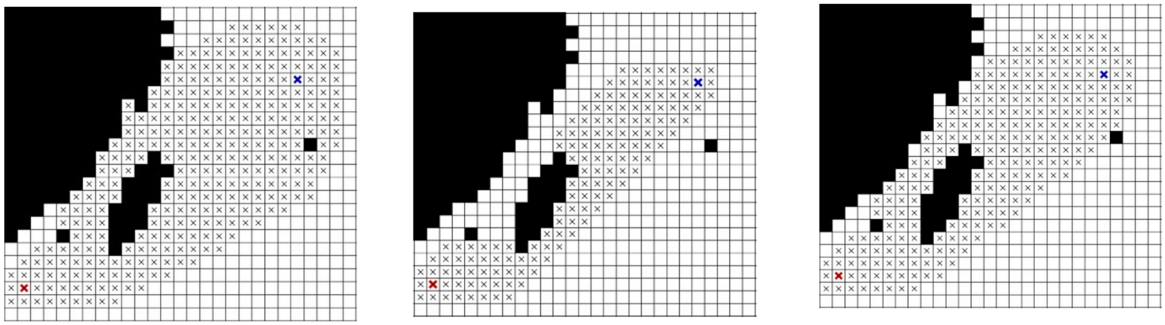
(2) If the weight is set too large, then only the heuristic function $h(s)$ is in effect, the D* Lite algorithm will be approximated as a depth-first search algorithm. In this situation, due

to the expanded nodes being too few, it cannot guarantee finding the shortest path, but it runs quickly.

Therefore, while appropriately increasing the number of expanded nodes and considering the running time of the algorithm, this paper designs a weighted diagonal function with a 0.9 multiplier as the heuristic function, represented as follows:

$$h(s, s_{start}) = 0.9\big(d|x_s - x_{start}| + d|y_s - y_{start}| + (d' - d)\min(|x_s - x_{start}|, |y_s - y_{start}|)\big) \quad (9)$$

The search performance with different heuristic functions is compared in Figure 3. In the figure, using the center of each grid cell as the path search point, the red "×" represents the start point, the blue "×" represents the destination point, and the black "×" markers represent expanded nodes. The number of expanded nodes using the Euclidean Distance heuristic function is 301, that for the Diagonal Distance heuristic function is 143, and that for the heuristic function proposed in this paper is 232.



(**a**) (**b**) (**c**)

Figure 3. Comparison of search performance with different heuristic functions. (**a**) Euclidean Distance; (**b**) Diagonal Distance; (**c**) This work.

2. Path Function Optimization

In actual maritime navigation, due to the large size of vessels, routes that are too close to obstacles may result in collisions and potential damage. To address this issue, this paper introduces a risk factor, *Danger*, into the path function of the D* Lite algorithm. This risk factor penalizes grids near obstacles by increasing the cost from the current grid to those near obstacles, thereby prioritizing the selection of other grids with lower costs, ensuring that the planned route maintains a safe distance from obstacles.

Additionally, this paper will increase the path search step length and selectable directions later in Section 3.2.2. Due to the limitations of the Diagonal Distance in terms of its applicability and direction of movement, curve optimization is no longer suitable for path function $g(s)$. Thus, the Euclidean Distance used in traditional D* Lite is selected for path function $g(s)$.

The path function $g(s)$ represents the actual cost from the destination node to the current node s, typically assigned by $rhs(s)$. Since $rhs(s)$ is updated more frequently, it can record node states in real time. According to Equation (4), it can be derived that $rhs(s)$ is mainly influenced by $c(s, s')$. Therefore, the optimization of the path function primarily focuses on improving the cost function $c(s, s')$. The improved cost function with a risk factor *Danger* can be represented as follows:

$$c(s, s') = \sqrt{(x_s - x_{s'})^2 + (y_s - y_{s'})^2} + Danger \quad (10)$$

where (x_s, y_s) represents the location of the current node s, $(x_{s'}, y_{s'})$ represents the location of s', and the risk factor *Danger* can be adjusted to select an appropriate safety distance based on actual requirements.

3.2.2. Curve Optimization

The traditional D* Lite algorithm uses an eight-direction search strategy with a step length of 1, as shown in Figure 4. This approach considers the eight adjacent grids around the current grid as search targets, which results in paths with excessive path waypoints, suboptimal path lengths, and difficulties in achieving smooth and continuous turns at the waypoints during actual navigation, leading to poor maneuverability. Therefore, this paper optimizes the path using curve optimization techniques to reduce the number of waypoints, shorten the path length, align it more closely with actual ship routes, and improve ships' maneuverability at waypoints.

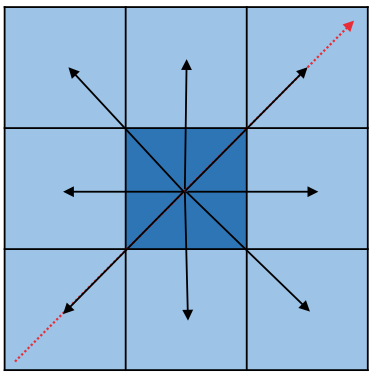


Figure 4. Traditional search strategy.

Curve optimization can be classified into post-processing optimization and in-process optimization based on the timing and method of optimization. Currently, the widely used method is post-process optimization, where adjustments and optimizations are made after the curve generation or calculation is completed. This includes techniques such as B-spline curves, Bézier curves, linear interpolation, spline interpolation, and least-squares fitting curves, which are used to post-process the planned path, making it smoother. However, post-processing optimization alters the path points of the pre-planned path, rendering the original path information invalid and thus potentially affecting the accuracy of re-planning results. Therefore, the path pre-planned by the D* Lite algorithm is not suitable for optimization through post-processing.

In-process optimization, on the other hand, involves adjustments and optimizations during the path search process. This can be achieved by adjusting parameters of the path planning algorithm, such as the path search step length, search direction, and path cost weights, to obtain a smoother path. Therefore, the better optimization method for the D* Lite algorithm is in-process optimization, which ensures path integrity while achieving curve optimization.

In this paper, the curve optimization is achieved by dynamically adjusting the search step length and selectable directions in the D* Lite algorithm, as shown in Figure 5. Firstly, the search step length is increased, allowing the algorithm to expand beyond adjacent grids. Secondly, the number of selectable directions is increased, allowing the algorithm to choose better grids and reduce the path length. The performance of different step lengths and directions from our experiments is shown in Table 3 and Figure 6.

Table 3. Comparison of the performance of different step lengths and directions.

Search Step Length	Selectable Directions	Path Length (n mile)	Number of Waypoints	Execution Time (s)
1	8	1496.10	14	99.95
2	16	1368.57	6	106.39
3	32	1305.45	5	153.32
4	56	1260.42	5	305.63

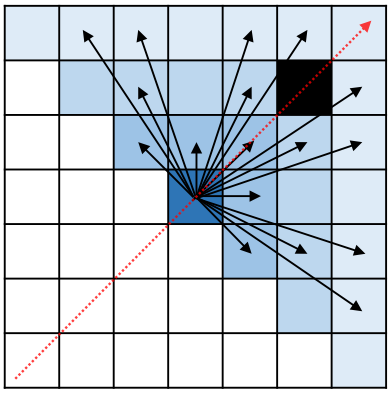


Figure 5. Sector-based search strategy.

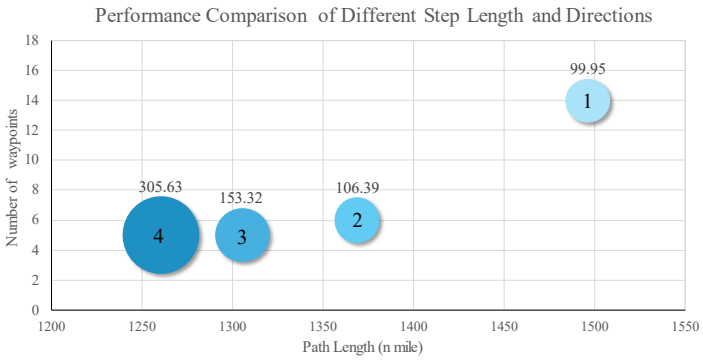


Figure 6. Bubble chart of performance comparison of different step lengths and directions.

From Table 3 and Figure 6, it is visually apparent that, as the search step length and selectable directions increase, both the path length and the number of waypoints decrease. However, this increase in parameters also leads to an increase in algorithm complexity, resulting in reduced calculation efficiency. To address this problem, this paper proposes a sector-based search strategy to mitigate the complexity increase associated with larger search step lengths and more selectable directions.

The sector-based search strategy constrains the algorithm's search angle range, reducing the number of expanded nodes and thus decreasing the algorithm's computational load. Since the search strategy with a step length of four (56 directions) has a long planning time and high computational complexity, a step length of three is selected in this study. The sector-based search strategy is illustrated in Figure 5, where the red arrow indicates the direction towards the destination and the black grids represent non-navigable areas. As shown in Figure 5, increasing the search step length may lead to the path crossing non-navigable areas, which can compromise the safety of ship navigation. To address this issue, this paper employs a strategy of dynamically adjusting the search step length, which uses a mixed search approach with different step lengths to solve the problem of crossing non-navigable grids, thereby enhancing path safety. The specific method is as follows:

(1) Initially, the search step length is set to three for pathfinding.

(2) If obstacles are present within the step length range, the search step length is dynamically adjusted to two for continued searching.

(3) If obstacles are still present within this range, the step length is further adjusted to 1.

This approach enables efficient path planning while avoiding diagonal traversal through obstacles.

3.3. Algorithm Flow

According to the improvements to the D* Lite algorithm for ship route planning and incorporating the principles of the D* Lite algorithm, the specific steps, flowchart (Figure 7) and pseudocode (Algorithm 1) of the designed algorithm are as follows:

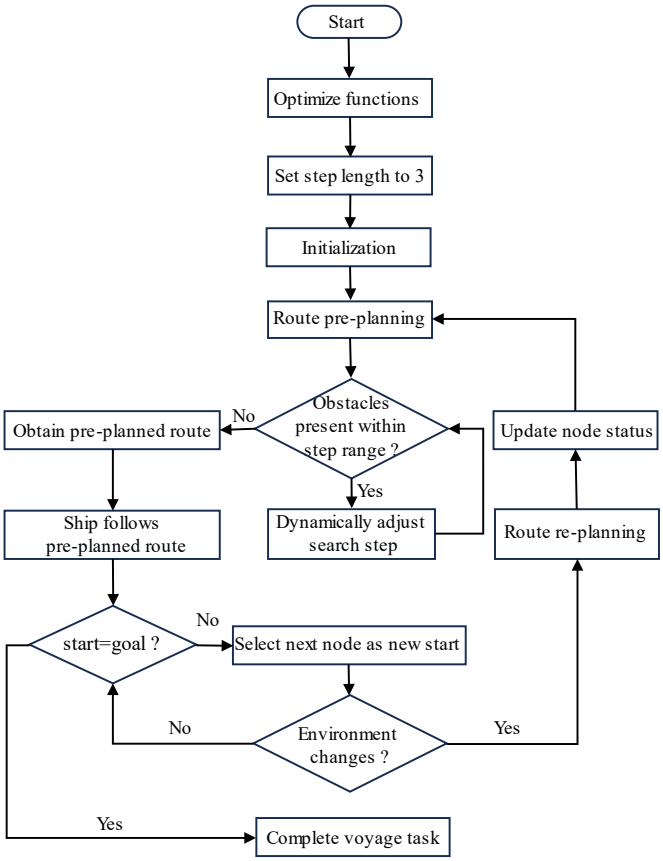


Figure 7. Algorithm flowchart.

(1) Function optimization and parameter readjusting: Optimize the heuristic and path functions of the D* Lite algorithm, and set the initial search step length to three.

(2) Algorithm initialization: Clear the priority queue U. Set the destination node's $rhs(s_{des})$ to 0, $g(s_{des})$ to infinity, and k_m to 0. Set the $g(s)$ and $rhs(s)$ of all other nodes to infinity. At this point, the destination node is in a locally inconsistent state and is placed in priority queue U. Compute the evaluation function $key(s)$ of the destination node based on the start node.

(3) Route pre-planning: For each step length, check if there are any obstacles. If obstacles are present, dynamically adjust the search step length. If no obstacles are found, dequeue the node s with the smallest value of $key(s)$ from queue U. Then, calculate and update the $rhs(s)$ of all neighbors of node s, and compare them with their $g(s)$. Finally, update the nodes and the queue if a locally inconsistent state occurs, and sort the queue based on their evaluation function $key(s)$.

This process continues until the start node is dequeued. The nodes of the current shortest path are obtained through these steps mentioned above. Using Equation (1), the coordinates of these path nodes are converted into latitude and longitude coordinates, and the distances between nodes are calculated using the great-circle distance formula as shown in Equation (2). These result in the pre-planned route.

(4) Ship navigation: Ships navigate along the planed route, while simultaneously monitoring the navigation environment for changes, specifically checking if any nodes are in a state of local inconsistency.

(5) Route re-planning: When a change in the navigation environment is detected, the state information of the affected node and its neighboring nodes is updated based on the previous path field information. The process then returns to the step (3) to re-plan a new route from the destination node to the current node.

Algorithm 1. The improved D* Lite algorithm

CalculateKey(*s*)
return [$min(g(s), rhs(s)) + h(s, s_{start}) + k_m$, $min(g(s), rhs(s))$];
Initialize()
$U = 0$, $k_m = 0$, $search\_radius = 3$;
for all s in S: $rhs(s) = \infty$, $g(s) = \infty$;
$rhs(s_{des}) = 0$;
U.Insert(s_{des}, CalculateKey(s_{des}));
UpdateVertex(*s*)
if $s \neq s_{des}$ **then**
  $rhs(s) = \min_{s' \in succ(s)} (c(s, s') + g(s'))$;
if $s \in U$ **then**
  U.Remove(s);
if $g(s) \neq rhs(s)$ **then**
  U.Insert(s, CalculateKey(s));
ComputeShortestPath()
while U.TopKey() < CalculateKey(s_{start}) or $rhs(s_{start}) \neq g(s_{start})$
  k_{old} = U.TopKey();
  s = U.pop();
  if k_{old} < CalculateKey(s) **then**
    U.Insert(s, CalculateKey(s));
  else if $g(s) > rhs(s)$ **then**
    $g(s) = rhs(s)$;
    for all $s' \in$ succ(s): UpdateVertex(s');
  else
    $g(s) = \infty$;
    for all $s' \in$ succ(s): UpdateVertex(s');
GetPreds()
for all s' within $search\_radius$ of s: Compute angle between s' and s;
if angle is within the relevant range **then**
  if s' is within bounds and is an obstacle **then**
    if $search\_radius = 3$ **then**
      $search\_radius = 2$;
      break loop;
    else if $search\_radius = 2$ **then**
      $search\_radius = 1$;
      break loop;
Main()
$s_{last} = s_{start}$;
Initialize()
GetPreds()
ComputeShortestPath()
while $s_{start} \neq s_{des}$
  if $g(s_{start}) = \infty$ **then**
    return "No known path";
  $s_{start} = \text{argmin}_{s' \in succ(s)} (c(s, s') + g(s'))$;
  Move to s_{start};
  GetPreds()
  for all s' within $search\_radius$ of s;
    if any edge costs changed **then**
      $k_m = k_m + h(s_{last}, s_{start})$;
      $s_{last} = s_{start}$;
      for all directed edges (s, s') with changed edge costs:
        Update the edge cost $c(s, s')$
        UpdateVertex(u)
      ComputeShortestPath()

4. Simulation Results and Analysis

To verify the effectiveness of the proposed intelligent route planning algorithm, three sets of experiments were conducted using MATLAB 2023b software under the same hardware platform conditions, with the complexity of these conditions gradually increasing from Experiment 1 to Experiment 3. Experiments 1 and 2 have the same start and destination coordinates, (21.25° N, 117.75° E) and (45.25° N, 139.25° E), but have different hydrological and meteorological conditions. For Experiment 3, the start and destination coordinates were (13.75° N, 118.25° E) and (42.25° N, 147.25° E), with different hydrological and meteorological conditions compared to Experiments 1 and 2. This is designed to verify the effectiveness of the improved algorithm in scenarios with longer routes and more complex environments.

The evaluation criteria for routes included the following:

(1) Safety: This criterion measures the distance the route maintains from areas deemed non-navigable or hazardous. It specifically refers to the distance between the planned route and any non-navigable grids. A higher safety rating implies that the route is relatively farther from these hazardous areas, which helps avoid having routes that are too close to obstacles that may result in collisions and potential damage. However, it is important to note that this distance is not always better when it is larger, as a greater distance might lead to an increase in the vessel's travel distance.

(2) Distance of the whole voyage: This criterion assesses the total length of the route. It is calculated as the sum of the distances between adjacent waypoints on the route, stretching from the starting point to the destination. A smaller value is preferable in this paper.

(3) Number of waypoints: This criterion counts the total number of waypoints in the route. Waypoints are specific locations where the route changes direction. Frequent turns at the waypoints during actual navigation may lead to poor maneuverability, making a smaller number of waypoints preferable.

(4) Computation time: This criterion measures the time required to compute the route using the algorithm. It represents the execution time of the algorithm used to plan the route. A shorter computation time indicates a more efficient algorithm, which is important for real-time navigation and timely route planning.

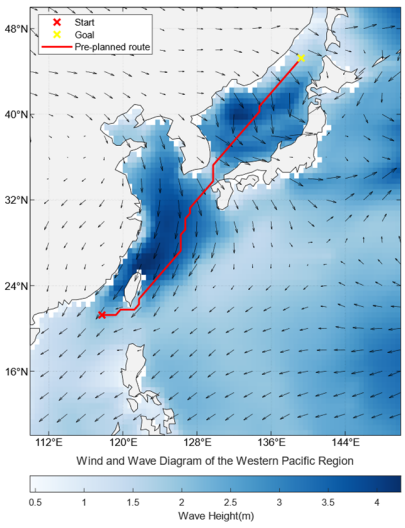
4.1. Pre-Planning Scenario

In this scenario, the pre-planning effectiveness of the algorithm is evaluated based on three simulation experiments. The precise performance comparison for the pre-planning phase is detailed in Figure 8 and Table 4, where the red solid lines represent pre-planned routes.

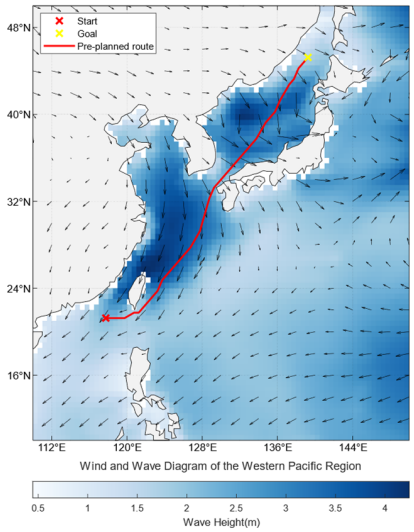
In terms of route safety, as depicted in in Figure 8, both algorithms effectively avoided non-navigable grids. However, compared to the traditional D* Lite algorithm, the path planned by the improved D* Lite algorithm maintained a greater distance from non-navigable areas, resulting in higher safety.

Table 4. Comparison of performance in pre-planning.

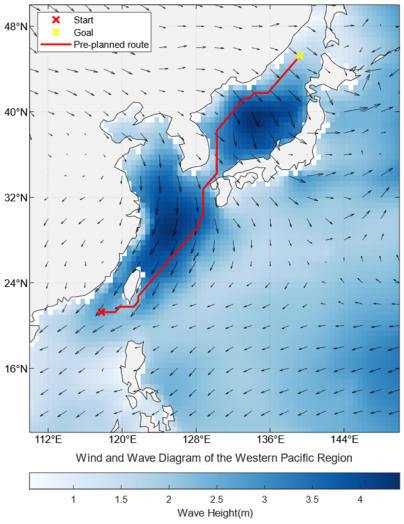
Experiment	Algorithm	Distance (n mile)	Number of Waypoints	Computation Time (s)
1	Traditional D* Lite Algorithm	1936.29	15	77.32
	Improved D* Lite Algorithm	1891.37	7	74.93
2	Traditional D* Lite Algorithm	1994.15	15	84.48
	Improved D* Lite Algorithm	1954.78	10	82.23
3	Traditional D* Lite Algorithm	2535.43	9	171.22
	Improved D* Lite Algorithm	2458.37	6	170.13



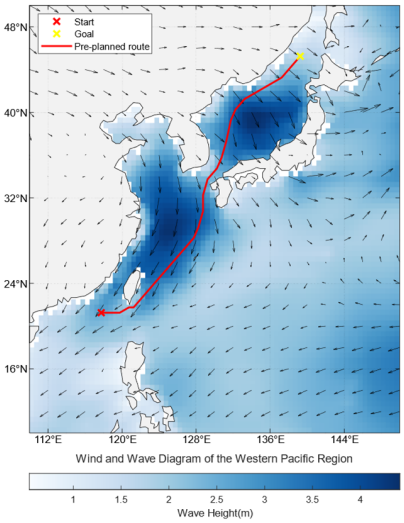
Experiment 1 (a)



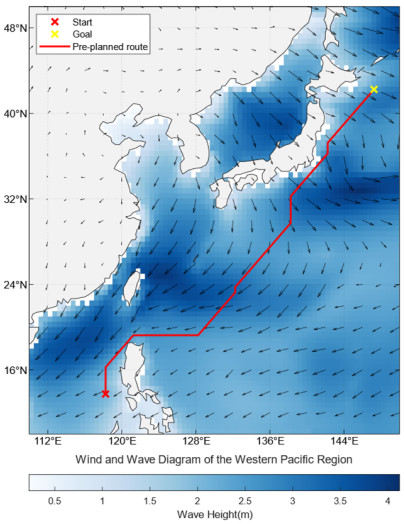
Experiment 1 (b)



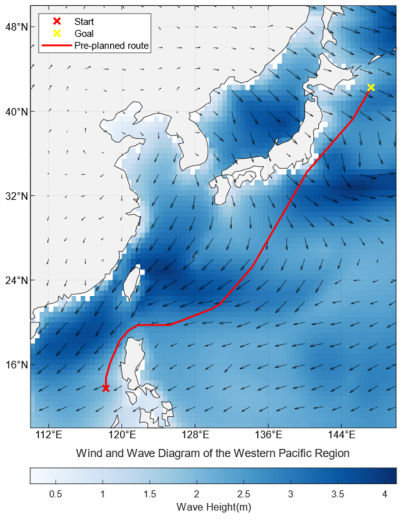
Experiment 2 (a)



Experiment 2 (b)



Experiment 3 (a)



Experiment 3 (b)

Figure 8. Comparison of pre-planned routes. (**a**) Traditional D* lite algorithm; (**b**) Improved D* lite algorithm.

In terms of route distance, the improved D* Lite algorithm in Experiment 1 reduced the route distance from 1936.29 nm to 1891.37 nm, achieving a reduction of 2.32% compared to the traditional D* Lite algorithm. In Experiment 2, the improved algorithm reduced the route distance from 1994.15 nm to 1954.78 nm, resulting in a 1.97% reduction. In Experiment 3, the improved algorithm reduced the route distance from 2535.43 nm to 2458.37 nm, achieving a reduction of 3.03%.

In terms of waypoints, the number of waypoints in Experiment 1 decreased from 15 to 7, in Experiment 2, it decreased from 15 to 10, and in Experiment 3, it decreased from 9 to 6, significantly improving the smoothness of the route and making it more aligned with actual navigation, thereby meeting the maneuvering needs of ships at waypoints.

Regarding planning efficiency, the improved D* Lite algorithm reduced computation time in Experiment 1 from 77.32 s to 74.93 s; in Experiment 2, it reduced from 84.48 s to 82.23 s; and in Experiment 3, it reduced from 171.22 s to 170.13 s. Although the increase in the size of search steps and the number of selectable directions expanded the range of node expansion and increased computational effort per step, the overall execution time decreased due to subsequent optimization processes. Consequently, the improved D* Lite algorithm demonstrates excellent planning efficiency.

Thus, the pre-planned route using the improved D* Lite algorithm demonstrates superior performance in length, smoothness, safety, and computation time compared to the traditional D* Lite algorithm, indicating better overall performance and effectiveness.

4.2. Re-Planning Scenario

In this scenario, the environment changes dynamically, requiring the algorithms to re-plan routes. This tests the adaptability and efficiency of the algorithms in real-time or changing conditions. To test the re-planning performance of the proposed algorithm, the navigation environment was updated during ships' travel along the pre-planned route. When ships detected changes in node states that affected its navigation, it re-planned the route using the current node as the new starting point. Experiment 1 simulated a scenario where ships' travel along the pre-planned route was affected by strong wind and waves after the environment update, while Experiment 2 simulated a scenario where the environment update provided ships with a better route option than before. Due to significant environmental changes, the re-planned route in Experiment 3 has undergone considerable alterations compared to the pre-planned route.

The results of re-planning simulation are shown in Figure 9, where the red solid lines represent the pre-planned routes and the yellow dashed lines indicate the re-planned routes. A comparison of specific performance during the re-planning scenario is illustrated in Table 5.

In the re-planning scenario, in Experiment 1, the improved D* Lite algorithm reduced the route distance from 1632.97 nm with the traditional D* Lite algorithm to 1611.91 nm, achieving a reduction of 1.28%; the number of waypoints decreased from nine to six; and the computation time reduced from 66.37 s to 64.77 s. In Experiment 2, the improved D* Lite algorithm reduced the route distance from 1369.28 nm to 1347.51 nm, achieving a reduction of 1.58%; the number of waypoints also decreased from three to two; and the computation time reduced from 50.98 s to 47.66 s. In Experiment 3, the improved D* Lite algorithm reduced the route distance from 1093.40 nm to 1075.70 nm, achieving a reduction of 1.62%; the number of waypoints also decreased from six to three; and the computation time reduced from 48.29 s to 46.47 s. Additionally, the re-planning scenario also shows that the improved algorithm performs well in the case of shorter routes. Thus, the improved D* Lite algorithm demonstrates enhanced planning efficiency.

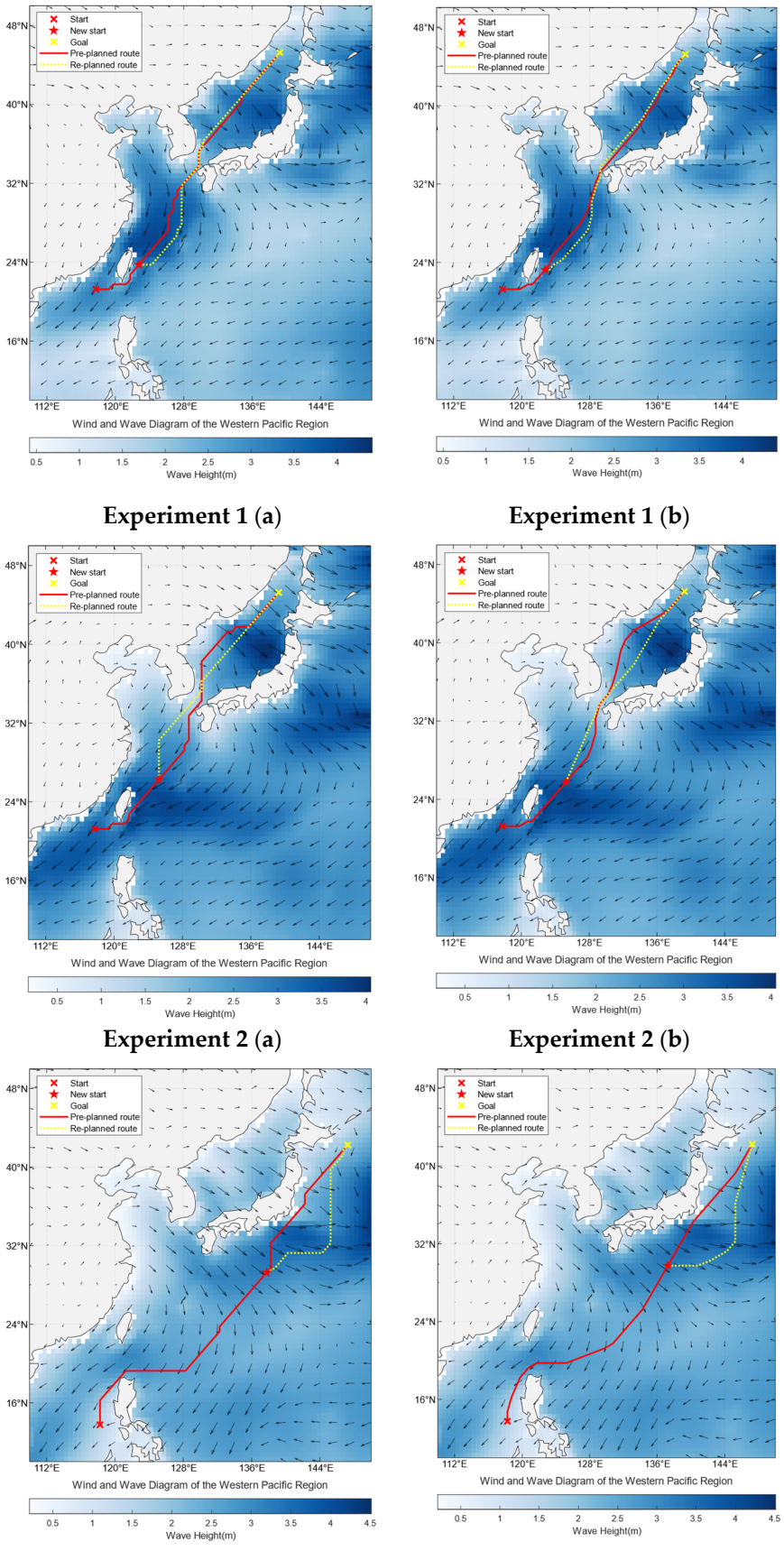


Figure 9. Comparison of re-planned routes. (**a**) Traditional D* lite algorithm; (**b**) Improved D* lite algorithm.

Table 5. Comparison of performance in re-planning.

Experiment	Algorithm	Distance (n mile)	Number of Waypoints	Computation Time (s)
1	Traditional D* Lite Algorithm	1632.97	9	66.37
	Improved D* Lite Algorithm	1611.91	6	64.77
2	Traditional D* Lite Algorithm	1369.28	3	50.98
	Improved D* Lite Algorithm	1347.51	2	47.66
3	Traditional D* Lite Algorithm	1093.40	6	48.29
	Improved D* Lite Algorithm	1075.70	3	46.47

5. Conclusions

Autonomous navigation is emerging as the future of maritime development, garnering substantial attention within the industry. As a pivotal component of autonomous navigation technologies, intelligent ship route planning has become a critical area of research. In intelligent navigation systems for ships, the ability to efficiently find routes in dynamic environments is essential. These systems enable ships to plan safe and efficient routes by considering real-time meteorological data and sea conditions, ultimately enhancing the safety and economic efficiency of maritime navigation.

In this paper, we applied the D* Lite algorithm, recognized for its effectiveness in dynamic environments, to the challenge of ship route planning. However, it has the issue of expanding too many nodes, leading to inefficiency in complex environments, routes that are too close to obstacles, and an excessive number of waypoints, which prompted the research conducted in this paper.

For the expansion of too many nodes, this paper designs a weighted diagonal function with a 0.9 multiplier as the heuristic function instead of the Euclidean Distance in the traditional D* Lite algorithm. This approach balances the number of expanded nodes and the computation time.

To address the issue of routes being too close to obstacles, which may lead to potential damage to large-sized vessels, this paper introduces a risk factor of 15 nm, corresponding to half the size of a grid cell, into the cost function to ensure that the planned route maintains a safe distance from obstacles.

To tackle the issue of an excessive number of waypoints, this paper introduces an in-process optimization method rather than a post-process approach, which can invalidate the original path information and compromise the accuracy of the planning results. In our approach, the initial search step length is set to three, covering 56 directions. The strategy then dynamically adjusts the step length and selectable directions, utilizing a mixed search method with diminishing step lengths to overcome the challenge of crossing non-navigable grids, thereby enhancing route safety. This method not only reduces the number of waypoints but also decreases the computational load of the algorithm.

Through the above effort, the D* Lite algorithm was used in ship route planning for simulation verification. The simulation results demonstrate that the improved D* Lite algorithm presented in this paper can calculate the shortest collision-free and smooth route with fewer waypoints, making it suitable for actual navigation. Moreover, the method can rapidly adjust routes in response to changing navigation conditions, making the planned routes more aligned with actual navigation requirements. This approach is highly suitable for ship route planning and can provide substantial support for autonomous ship navigation.

In next step, we will compare the results of this study with the A* algorithm to analyze their respective advantages and disadvantages. Additionally, if feasible, we will combine their characteristics and strengths to propose an enhanced algorithm for ship route optimization.

Future research directions for this paper primarily focus on multi-objective optimization, which integrates multiple factors such as safety, fuel consumption, and speed to

achieve comprehensive optimization. By combining speed optimization and energy efficiency analysis, the energy-saving effect of route planning can be enhanced, leading to more environmentally friendly shipping solutions.

Author Contributions: Conceptualization, Y.L. and F.Y.; methodology, Y.L. and F.Y.; software, F.Y. and D.Y.; validation, Y.L., F.Y. and X.Z.; formal analysis, X.Z.; data curation, X.Y.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L. and F.Y.; supervision, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Liaoning Provincial Department of Education funding for research projects (NO. JYTMS20230163); Fundamental Research Funds for the Central Universities (NO. 3132024132); and Fund of the National Engineering Laboratory of Transport Safety and Emergency Informatics (NO. YW170301-05).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The hydrological data presented in this study are openly available in ECMWF Climate Data Store at https://doi.org/10.24381/cds.adbb2d47 [28], and ocean depth data in GEBCO at https://www.gebco.net/data_and_products/gridded_bathymetry_data [29]. The data that support the findings of this study are available from the corresponding author, Yuankui Li, upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wu, G.; Wang, L.; Zheng, J.; Wang, C. Intelligent Dynamic Route Planning Method for Ships Considering Complex Meteorological Changes. *J. Shanghai Marit. Univ.* **2021**, *42*, 1–6+12. [CrossRef]
2. Li, Y.; Cui, J.; Zhang, X.; Yang, X. A Ship Route Planning Method under the Sailing Time Constraint. *J. Mar. Sci. Eng.* **2023**, *11*, 1242. [CrossRef]
3. Walther, L.; Rizvanolli, A.; Wendebourg, M.; Jahn, C. Modeling and optimization algorithms in ship weather routing. *Int. J. E-Navig. Marit. Econ.* **2016**, *4*, 31–45. [CrossRef]
4. Vettor, R.; Soares, C.G. Development of a ship weather routing system. *Ocean Eng.* **2016**, *123*, 1–14. [CrossRef]
5. Pennino, S.; Gaglione, S.; Innac, A.; Piscopo, V.; Scamardella, A. Development of a new ship adaptive weather routing model based on seakeeping analysis and optimization. *J. Mar. Sci. Eng.* **2020**, *8*, 270. [CrossRef]
6. Zaccone, R.; Ottaviani, E.; Figari, M.; Altosole, M. Ship voyage optimization for safe and energy-efficient navigation: A dynamic programming approach. *Ocean Eng.* **2018**, *153*, 215–224. [CrossRef]
7. Geng, X.; Wang, Y.; Wang, P.; Zhang, B. Motion plan of maritime autonomous surface ships by dynamic programming for collision avoidance and speed optimization. *Sensors* **2019**, *19*, 434. [CrossRef]
8. Zaccone, R. A Dynamic Programming Approach to the Collision Avoidance of Autonomous Ships. *Mathematics* **2024**, *12*, 1546. [CrossRef]
9. Xiao, J.; Zhang, J.; Wu, D.; Han, B. Improved D*Lite Path Planning Method for Polar Ships under Dynamic Sea Ice Conditions. *J. Dalian Marit. Univ.* **2023**, *49*, 20–27+36. [CrossRef]
10. Long, Y.Y.; Feng, L.X.; Zhi, L.M.; Kai, Y.; Zhe, C.; Ben, N.C.; Yue, T. Path Planning Method Based on D* lite Algorithm for Unmanned Surface Vehicles in Complex Environments. *China Ocean Eng.* **2021**, *35*, 372–383.
11. Yu, J.; Yang, M.; Zhao, Z.; Wang, X.; Bai, Y.; Wu, J.; Xu, J. Path planning of unmanned surface vessel in an unknown environment based on improved D*Lite algorithm. *Ocean Eng.* **2022**, *266*, 112873. [CrossRef]
12. Jin, J.; Zhang, Y.; Zhou, Z.; Jin, M.; Yang, X.; Hu, F. Conflict-based search with D* lite algorithm for robot path planning in unknown dynamic environments. *Comput. Electr. Eng.* **2023**, *105*, 108473. [CrossRef]
13. Xie, K.; Qiang, J.; Yang, H. Research and optimization of d-start lite algorithm in track planning. *IEEE Access* **2020**, *8*, 161920–161928. [CrossRef]
14. Lin, Z.; Lu, L.; Yuan, Y.; Zhao, H. A Novel Robotic Path Planning Method in Grid Map Context Based on D* Lite Algorithm and Deep Learning. *J. Circuits Syst. Comput.* **2023**, *33*, 2450057. [CrossRef]
15. Le, A.T.; Bui, M.Q.; Le, T.D.; Peter, N. D* lite with reset: Improved version of D* lite for complex environment. In Proceedings of the 2017 First IEEE International Conference on Robotic Computing (IRC), Taichung, China, 10–12 April 2017; pp. 160–163.
16. Przybylski, M.; Putz, B. D* Extra Lite: A dynamic A* with search–tree cutting and frontier–gap repairing. *Int. J. Appl. Math. Comput. Sci.* **2017**, *27*, 273–290. [CrossRef]
17. Reyes, N.H.; Barczak, A.L.; Susniak, T. Autonomous navigation in partially known confounding maze-like terrains using D* Lite with poisoned reverse. In Proceedings of the 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA), Kosice, Slovakia, 23–25 August 2018; pp. 67–76.

18. Maurović, I.; Seder, M.; Lenac, K.; Petrović, I. Path planning for active SLAM based on the D* algorithm with negative edge weights. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *48*, 1321–1331. [CrossRef]
19. Chao, N.; Liu, Y.-K.; Xia, H.; Peng, M.-J.; Ayodeji, A. DL-RRT* algorithm for least dose path Re-planning in dynamic radioactive environments. *Nucl. Eng. Technol.* **2019**, *51*, 825–836. [CrossRef]
20. Osmankovic, D.; Tahirovic, A.; Magnani, G. All terrain vehicle path planning based on D* lite and MPC based planning paradigm in discrete space. In Proceedings of the 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Munich, Germany, 3–7 July 2017; pp. 334–339.
21. Wu, T.; Xie, Z.; Chen, K. Path Planning for Mobile Robots Using Improved D*Lite and Time Elastic Band Methods. *J. Sens. Technol.* **2022**, *35*, 486–494.
22. Huang, L.; Zhou, F. Path Planning for Mobile Robots Based on Path Optimization D*Lite Algorithm. *Control Decis.* **2020**, *35*, 877–884. [CrossRef]
23. Xu, K. Research on Path Planning for Mobile Robots Based on D*Lite Algorithm. Master's Thesis, Harbin Institute of Technology, Harbin, China, 2017.
24. Zhang, Y.; Shi, M. Improved D*Lite Path Planning Algorithm Based on Unit Decomposition. *J. Chongqing Univ. Posts Telecommun. (Nat. Sci. Ed.)* **2021**, *33*, 1007–1013.
25. Hu, L.; Zeng, W.; Chen, C.; Zhang, P.; Li, Y.; Li, T. Path Planning of Inspection Robots Based on Improved D*Lite-APF Algorithm. *Mod. Electron. Technol.* **2024**, *47*, 155–159. [CrossRef]
26. Fu, L.; Liu, F.; Liu, S.; Han, X. Continuous Dynamic Planning Algorithm for 2D Paths Based on Improved D *Lite. *Radio Commun. Technol.* **2023**, *49*, 1042–1051.
27. Koenig, S.; Likhachev, M. Improved fast replanning for robot navigation in unknown terrain. In Proceedings of the 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292), Washington, DC, USA, 11–15 May 2002; pp. 968–975.
28. Hersbach, H.; Bell, B.; Berrisford, P.; Biavati, G.; Horányi, A.; Muñoz Sabater, J.; Nicolas, J.; Peubey, C.; Radu, R.; Rozum, I. ERA5 Hourly Data on Single Levels from 1940 to Present. Available online: https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-single-levels?tab=overview (accessed on 21 January 2024).
29. GEBCO. GEBCO Gridded Bathymetry Data. Available online: https://www.gebco.net/data_and_products/gridded_bathymetry_data (accessed on 21 January 2024).