






## Article

# Near-Optimal Weather Routing by Using Improved A\* Algorithm

Yong Woo Shin <sup>1</sup>, Misganaw Abebe <sup>1</sup>, Yoojeong Noh <sup>2,\*</sup>, Sangbong Lee <sup>3</sup>, Inwon Lee <sup>4</sup>, Donghyun Kim <sup>5</sup>, Jungchul Bae <sup>5</sup> and Kyung Chun Kim <sup>2</sup>

<sup>1</sup> Research Institute of Mechanical Technology, Pusan National University, Busan 46241, Korea; archeex@pusan.ac.kr (Y.W.S.); misge98@gmail.com (M.A.)

<sup>2</sup> School of Mechanical Engineering, Pusan Nat'l University, Busan 46241, Korea; kckim@pusan.ac.kr

<sup>3</sup> LabO21, Busan 48508, Korea; sblee@lab021.co.kr

<sup>4</sup> Department of Naval Architecture & Ocean Engineering, Pusan National University, Busan 46241, Korea; inwon@pusan.ac.kr

<sup>5</sup> Korea Marine Equipment Research Institute, Busan 49111, Korea; kdh9942@komeri.re.kr (D.K.); justin@komeri.re.kr (J.B.)

\* Correspondence: yoonoh@pusan.ac.kr; Tel.: +82-51-510-2308

Received: 24 July 2020; Accepted: 26 August 2020; Published: 30 August 2020



**Abstract:** With soaring oil prices worldwide, determining the most optimal routes for economical ship operation has become an important issue. Optimizing ship routes is economically important for ship operation, but it is also essential to meet the standards of environmental regulations recently imposed by the International Maritime Organization. For this purpose, various algorithms for determining ship routes have been developed to ensure the economical operation of ships via utilization of marine climate data and Automatic Identification System (AIS) data. However, such algorithms require a large amount of computational time and do not provide optimal routes because they do not consider practical operating conditions, such as weather and ocean conditions. In this study, an improved A\* algorithm using AIS and weather data is proposed to overcome the limitation of the original A\* algorithm, one of the most widely used path-finding algorithms. The improved A\* algorithm uses an adaptive grid system that efficiently explores nodes according to map grid deformation by latitude. It finds economical routes by minimizing the estimated time of arrival generated by machine learning through 16-way node exploration. For verification of the proposed method, the original A\* algorithm and improved A\* algorithm were compared through a case study.

**Keywords:** A-star algorithm; weather routing; ship speed; machine learning; economical ship operation

## 1. Introduction

International oil prices have been increasing, and environmental regulations pertaining to greenhouse gas emissions have been strengthened. The International Maritime Organization (IMO) 72nd Marine Environment Protection Committee (MEPC) proposed a greenhouse gas reduction strategy aimed at reducing ship greenhouse gas emissions by more than 40% by 2030. The 74th MEPC proposed an agenda for measures to improve the energy efficiency of ships using the Energy Efficiency Existing Ship Index. This increased international oil prices, and stricter international regulations for greenhouse emissions have caused shipping companies to look for ways to reduce the fuel consumption of ships, and the simplest method to reduce fuel consumption is to reduce the travel distance and time by optimizing shipping routes.

There have been many studies on developing optimal algorithms for finding shipping routes. The main goals of route finding can be divided into two objectives: safe operation and economical operation. For safe operation, the dynamic stability of the vessel and the possibility of collision between vessels are commonly considered. For example, Krata et al. [1] proposed a multiobjective evolutionary weather routing algorithm to find routes that can simultaneously meet various IMO safety standards while considering parametric roll risk. However, the various dangerous situations that ships can encounter during operation are not only the instability of the ship itself, but also the collisions between ships. Accordingly, Zhang et al. [2] focused only on safety and studied the methodology that considered collision between ships by utilizing vessel density based on ship operating route information as well as the stability of the ship itself. However, the route optimization, which considers various objective functions, has some limitations because the decision made by the user must be involved. To address these problems, Fabri et al. [3] constructed an intuitive indexing system to select the optimal solution from among the various candidates through the multicriterion weather routing framework based on hyper-radial visualization.

However, if the goal of route finding is economical operation, vessel Fuel Oil Consumption (FOC) is set as an objective function to be minimized in general cases. However, it is not easy to present FOC as an analytical formula because it is affected by various environmental factors. In some cases, there may be significant errors in the predicted results. Chu et al. [4] proposed an ensemble smart voyage planning model to reduce errors in the FOC prediction model based on the US Navy Meteorological and Oceanographic forecast system. However, the methodology is limited in that it is impossible to generate a direct FOC prediction model if the FOC factor is missing from the data used. It also has a high computational demand. Zhou et al. [5] and Takashima et al. [6] also performed Estimated Time of Arrival (ETA)-based route optimization by modeling the speed of the ship. Nevertheless, sufficient accuracy is not guaranteed because the speed of the ship is affected by numerous environmental factors. Thus, the empirical-based prediction model entails unavoidable errors.

To solve these problems, some studies have attempted to improve accuracy by utilizing a learning-based predictive model. Gkerekos et al. [7] produced a variety of learning-model-based regression models and analyzed the predicted accuracy of the FOC, revealing that machine-learning models perform better than traditional regression models, such as conventional linear regression. Further studies suggested a framework for high-accuracy FOC optimal weather routing using an artificial neural network [8]. Once the predictive model for fuel consumption has been defined, i.e., the objective function for ship route optimization, the optimization algorithm should be employed to find the optimal route of the ship. Among the route optimization methods, a routing algorithm based on the evolutionary algorithm is widely used. Tsou et al. [9] conducted routing to ensure the stability of ships based on the ant colony algorithm and proposed an optimal route to satisfy various conditions. Wang [10] and Veneti [11] showed that route optimization using the genetic algorithm yielded highly accurate routing through various simulations. However, because these optimizations require a large amount of calculation and are difficult to use during actual navigation, routing algorithms suitable for grids or graphs are commonly used at sea to compensate for these shortcomings.

The Bellman–Ford algorithm [12–14] and Dijkstra algorithm [15] are representative graph-based routing algorithms. In particular, the Dijkstra algorithm has been used in many studies because of its simplicity and flexibility. For example, Silveira et al. [16] proposed a routing methodology based on traffic density estimations using the Dijkstra algorithm. Wang et al. [17] enabled consideration of various objective functions by applying the Dijkstra algorithm to a 3D graph in which the time domain was added. Despite these efforts for improvement, the Dijkstra algorithm has the disadvantage of requiring more search space than necessary in the graph. To solve this problem, an A\* algorithm with a heuristic function applied to the Dijkstra algorithm was proposed, and time and space complexity were significantly improved when compared with the case of the Dijkstra algorithm. [18]. A number of improved algorithms based on the A\* algorithm have been proposed. In particular, the D\* algorithm enables the A\* one to consider dynamic obstacles by reusing back-pointer information [19], and

the Theta\* algorithm aims to smooth the solution path of the A\* algorithm by using line-of-sight inspection [20].

Although many routing algorithms have been proposed and many studies have improved the algorithm into a form suitable for weather routing, development in terms of the grid map used for weather routing has been insufficient. Thus, in this work, an adaptive grid system is proposed that can facilitate the efficient navigation of the rectangular map commonly used for weather routing. A learning-based regression model for ETA and a 16-way search method were applied to verify the more accurate and realistic alternative route-finding algorithm through simulation.

The remainder of this study is as follows. The canonical A\* algorithm, including the Bellman-Ford and Dijkstra algorithms, is briefly described in Section 2. In Section 3, properties of the improved A\* algorithm, such as the adaptive grid and low-cost heuristic, are explained. In Section 4, a case study using AIS and marine weather data is discussed. Finally, in Section 5, the conclusions of this study and the direction of future research are presented.

## 2. Routing Algorithms

### 2.1. Bellman–Ford Algorithm

The Bellman-Ford algorithm finds the shortest path between a single source node and all other nodes in a given graph of paths through nodes [12–14]. The Bellman-Ford algorithm estimates the lengths of the paths from the source node (start node) to all other nodes and then iteratively relaxes the edges between two vertices by finding new paths that are shorter than the previously estimated paths.

For example, if  $s$  is a source (start) node and  $u$  and  $v$  are arbitrary nodes explored, as shown in Figure 1, the wave arrows  $d(s, u)$  and  $d(s, v)$  indicate the optimal path costs passing through multiple nodes between  $s$  and  $u$  or  $v$ , respectively. The dashed arrow  $c(u, v)$  indicates the path cost (edge weight) of two neighbor nodes  $u$  and  $v$ . When the edge linking  $u$  to  $v$  with path cost of  $c(u, v)$  is found, as shown in Figure 1, and if the inequality  $d(s, u) + c(u, v) < d(s, v)$  is satisfied, the path with a cost of  $d(s, v)$  is no longer the optimal path from  $s$  to  $v$ . Thus, the path through  $u$  is included as a subset of the optimal solution path instead of the existing path with the cost of  $d(s, v)$ . The optimal path between  $s$  and  $v$  is updated by the path via  $u$ , which is called edge relaxation. If  $d(s, u)$  is properly relaxed, i.e.,  $d(s, u)$  yields the suboptimal path of the optimal solution path, the triangular inequality  $d(s, u) + c(u, v) \geq d(s, v)$  is satisfied. This process is repeated until edge relaxation does not occur.

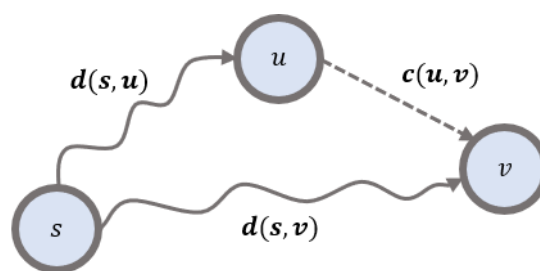


Figure 1. Principle of edge relaxation.

The edge weights can be either negative or positive values according to the sign of the path cost. For example, one can pay the cost from one node to another node (positive weight), but one can obtain some advantage (negative weight) reversely. If the sum of the edges is a negative value, the graph contains a negative cycle. The negative cycle can lead to a routing process with an infinite loop in terms of an algorithm that tries to minimize path cost; thus, the Bellman-Ford algorithm returns an error when any negative cycles are detected.

Let  $N$  be the set of defined nodes,  $G$  denote the finite graph consisting of nodes and edges, and  $d(node)$  denote the cost of the path connecting source  $s$  to  $node$ . The path cost between all nodes is initialized to infinity, and the path cost is reset to zero for the starting node,  $s$ . The triangular inequality

condition shown in Figure 1 is used to find the optimal path that minimizes the path cost of passing through nodes in the graph  $G$ . If there is a negative cycle in  $G$ , the entire process is halted. The basic processes of the Bellman–Ford algorithm are as follows, and Algorithm 1 shows the pseudocode.

- i. For all  $nodes \in N$  in graph  $G$  provided, initialize with  $d(node) = \infty$ .
- ii. For source  $s$ , initialize  $d(s) = 0$ .
- iii. Execute edge relaxation in accordance with triangular inequality.
- iv. Repeat iii. until there is no node left to update  $d$ .
- v. If there are any negative cycles in  $G$ , stop the process, even if the condition iv. is unfulfilled.

---

**Algorithm 1.** Bellman-Ford Algorithm

---

```

1: function BELLMAN-FORD( $G(V, E), W, s, t$ )
2:   for all  $u \in V - \{s\}$  do
3:      $d[v] \leftarrow \infty, pred[v] \leftarrow nil$ 
4:    $d[s] \leftarrow 0$ 
5:   for all  $i \leftarrow 1$  to  $|V| - 1$  do
6:     for all  $e \in \{e_{u,v} \in E \mid u \in V, v \in adjacent[u]\}$  do           : Edge relaxation
7:       if  $d[v] > d[u] + W(e)$  then
8:          $d[v] \xleftarrow{update} d[u] + W(e)$ 
9:          $pred[v] \xleftarrow{update} u$ 
10:    for all  $e \in \{e_{u,v} \in E \mid u \in V, v \in adjacent[u]\}$  do
11:      if  $d[v] > d[u] + W(e)$  then
12:        return  $\emptyset$                                            : Negative cycles in  $G$  detected
13:    return  $d, MAKEPATH(t, pred)$ 

```

---

In Algorithm 1,  $V, E$ , and  $W$  are the sets of nodes, edges, and weights in the graph  $G$ , respectively;  $pred(n)$  is the parent node of an arbitrary node  $n$ ; and  $e_{u,v}$  is the weight of edge linking two adjacent nodes,  $u$  and  $v$ .

## 2.2. Dijkstra Algorithm

The Dijkstra algorithm is similar to the Bellman–Ford algorithm in that it explores an optimal path by performing edge relaxation; the only difference between the two is that the latter can handle negative weights, whereas the former cannot [15]. Unlike the Bellman–Ford algorithm, the Dijkstra algorithm selects the unvisited node with the lowest distance and calculates the distance from it to each unvisited neighbor; then, it updates the distance of the neighbor if it is smaller. This algorithm improves the efficiency of finding the optimal path by reducing the number of candidate nodes to be explored using a queue (or heap), which is a tree-based data structure designed to accelerate the operation of finding the maximum and minimum values.

The Dijkstra algorithm utilizes a queue to save path costs from the start node to neighbor nodes. It sequentially selects the optimal node with the lowest path cost at each step in a queue and attempts to find the overall optimal path in the given graph. Therefore, the method of selecting the optimal path from the entire graph in succession by selecting the nodes considered optimal at each step is called a “greedy algorithm.” The Dijkstra algorithm can be regarded as one of the greedy algorithms. Consequently, because this greedy property of the Dijkstra algorithm limits the number of nodes that need to be explored to find the optimal path, it presents a remarkable advantage in terms of time complexity when compared with the Bellman–Ford algorithm. As described in Section 2.1, although the Dijkstra algorithm is not applicable when a negative edge weight exists, this is not a critical drawback because the optimal path finding for ship routes does not include negative weights. Because the shipping time is related directly to the path cost, the shipping cost does not include negative weights. Additionally, because the Dijkstra algorithm is more efficient than the Bellman–Ford algorithm, it is better to use the Dijkstra algorithm for the purpose of this study.

Similar to the Bellman-Ford algorithm, the Dijkstra algorithm initializes the path cost of the start node to zero, while the other nodes have infinite values. The starting node is first placed in *Queue* as the key of the  $d$  value, and then the neighbor node  $u$  with minimum cost is selected as the priority queue (*Queue*) by calculating the path costs of the adjacent nodes from the starting node, and the path cost is updated. This process is repeated until the path reaches the target node.

The overall processes of the Dijkstra algorithm are as follows.

- i. For all  $nodes \in N$  in graph  $G$  provided, initialize with  $d(node) = \infty$ .
- ii. For source  $s$ , initialize  $d(s) = 0$  and put  $s$  into *Queue* as key with value of  $d$ .
- iii. Extract minimum valued node  $u$  from *Queue*.
- iv. Execute edge relaxation at  $u$ , putting newly visited nodes into *Queue*.
- v. Repeat iii. and iv. until the path reaches the goal node  $t$ .

Here, *Queue* is a priority queue of visited nodes,  $Queue \subseteq V$ , and  $S$  is a set of assessed nodes,  $S \subseteq V$ . Algorithm 2 shows the pseudocode of the Dijkstra algorithm.

---

**Algorithm 2.** Dijkstra's Algorithm

---

```

1: function DIJKSTRA( $G(V, E), W, s, t$ )
2: for all  $u \in V - \{s\}$  do
3:  $d[v] \leftarrow \infty, pred[v] \leftarrow nil$ 
4:  $d[s] \leftarrow 0$ 
5:  $Queue \leftarrow \{s\}, S \leftarrow \emptyset$ 
6: while  $Queue \neq \emptyset$  do
7:  $u \leftarrow Extract\_min_d(Queue)$ 
8:  $S \xleftarrow{update} S \cup \{u\}$ 
9: if  $u = t$  then
10: return  $d, MAKEPATH(t, pred)$  : Tracking backward
11: for all  $e \in \{e_{u,v} \in E \mid u \in V, v \in adjacent[u]\}$  do : Edge relaxation
12: if  $d[v] > d[u] + W(e)$  then
13:  $d[v] \xleftarrow{update} d[u] + W(e)$ 
14:  $pred[v] \xleftarrow{update} u$ 
15:  $Queue \leftarrow Queue \cup \{u\}$ 
16: return  $\emptyset$ 

```

---

### 2.3. A\* Algorithm

The Bellman Ford algorithm calculates all pairs of paths leading to multiple sources, it can be advantageous in the case of an internet network that delivers information from a single source to many users, but it is inefficient for ship route optimization with single source-single destination. Similarly, as described above, the Dijkstra algorithm is not quite efficient for dense distance graphs because it requires a large number of nodes to be assessed to find the optimal path. As a result, although the Dijkstra algorithm can find the optimal path, it requires an excessive amount of computation, and thus, a route-finding algorithm that is more efficient than the Bellman Ford and Dijkstra algorithm is necessary.

The A\* algorithm was proposed by Hart et al. in 1968 [18]. This algorithm is based on the Dijkstra algorithm, but it increases the efficiency of the overall process by applying a new cost assessment methodology called a “heuristic.” A\* has become the widely used routing algorithm because of its simplicity as well as its flexibility enabling utilization in various areas. In particular, the A\* algorithm is specialized in single source–single destination, so that it has a high advantage in terms of computational efficiency in the case of ship operations where the departure and destination are clearly defined compared to the Bellman Ford algorithm specialized in single source-all pairs.

The A\* algorithm determines a path from the given start node to the given target node that has the smallest cost, such as least distance traveled and shortest time, as given in Equation (1).

$$\text{Minimize } f(n) = g(n) + h(n) \quad (1)$$

where  $g(n)$  is the actual path cost from the start node  $s \in N$  to the current node  $n \in N$ , and  $h(n)$  is the heuristic, which is the expected cost of traveling from current node  $n \in N$  to the target node  $t \in N$ . In ordinary shortest-path problems, the  $L^p$ -norms, such as the Manhattan distance ( $L_1$ ) and Euclidean distance ( $L_2$ ), are widely used as heuristic functions. The main loop of the A\* algorithm iteratively determines which of its paths to extend from the start node to the target node.

The heuristic function has the largest value at the start node because the actual path cost is zero. As the current node gets close to the target node  $t$ , the heuristic function value decreases and finally becomes zero for the target node. If the heuristic has a constant value of 0 regardless of the position of the node, the A\* algorithm is equivalent to the Dijkstra algorithm. The Dijkstra algorithm searches all directions around the current node, whereas the A\* algorithm continues to re-evaluate  $g(n)$  and  $h(n)$  values during the path exploration process, looking for a path with the minimum path cost to the target. The A\* algorithm only searches in the direction toward the target, which is more efficient than the Dijkstra algorithm.

If  $h(n)$  is never larger than the actual lowest cost from  $n$  to the goal,  $h(n)$  is considered admissible. If the heuristic function satisfies the admissibility condition, then A\*, which minimizes  $f(n)$ , is guaranteed to find an optimal solution from start to target. Conversely, if A\* is admissible, i.e., there is an optimal path from start to target, the heuristic function is admissible, and A\* terminates by finding the optimal path. If the heuristic  $h$  satisfies the triangular inequality, such as  $h(x) \leq d(x, y) + h(y)$  for every edge  $(x, y)$  of the graph,  $h$  is considered monotonic or consistent. When the heuristic is consistent, A\* using graph-search is guaranteed to find an optimal path. Additionally, A\* is optimally efficient because there is no other algorithm that is guaranteed to expand fewer nodes than A\*. If any path-finding algorithm expands fewer nodes than A\*, the obtained optimal path may not be the real optimal one [21]. As described above, because the A\* algorithm guarantees the admissibility, consistency, and efficiency for optimal path finding, it was used to find the optimal ship routes in this study.

Unlike the Dijkstra algorithm, the A\* algorithm puts the start node into the *Queue* with a key of  $f$  value with a heuristic function value instead of path cost  $d$ . Then, it selects the adjacent node  $u$  with the minimum path cost. The next step is to update the path cost through edge relaxation in the same way as the Dijkstra algorithm. This process is repeated until the target node is reached.

The overall processes of the A\* algorithm are as follows.

- i. For all *nodes*  $\in N$  in graph  $G$  provided, initialize with  $d(\text{node}) = \infty$ .
- ii. For source  $s$ , initialize  $d(s) = 0$  and  $f(s) = d(s) + \text{heuristic}(s)$ , then put into *Queue* as key with value of  $f$ .
- iii. Extract minimum-valued node  $u$  from *Queue*.
- iv. Execute edge relaxation based on  $d$  at  $u$ , putting newly visited nodes into *Queue*.
- v. Repeat iii. and iv. until the path reaches the target node  $t$ .

where  $d$  corresponds to the path cost  $d$  of the Dijkstra algorithm and is the same as the  $g$  function of the A\* algorithm in Equation (1).

Algorithm 3 shows the pseudocode of the A\* algorithm.

Here, HEURISTIC is the cost-evaluation function over a path defined with at least two consecutive nodes.



**Algorithm 3.** A\* Algorithm

---

```

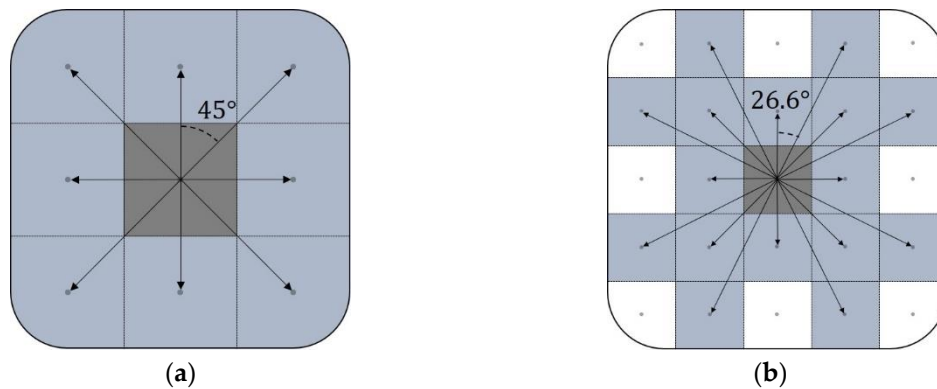
1: function A*(G(V, E), W, s, t)
2: for all  $u \in V - \{s\}$  do
3:  $d[v] \leftarrow \infty$ ,  $pred[v] \leftarrow nil$ 
4:  $d[s] \leftarrow HEURISTIC(s, t)$  : Heuristic function
5:  $Queue \leftarrow \{s\}$ ,  $S \leftarrow \emptyset$ 
6: while  $Queue \neq \emptyset$  do
7:  $u \leftarrow Extract\_min_d(Queue)$ 
8:  $S \xleftarrow{update} S \cup \{u\}$ 
9: if  $u = t$  then
10: return  $d$ ,  $MAKEPATH(t, pred)$  : Tracking backward
11: for all  $e \in \{e_{u,v} \in E \mid u \in V, v \in adjacent[u]\}$  do : Edge relaxation
12: if  $d[v] > d[u] + W(e)$  then
13:  $d[v] \xleftarrow{update} d[u] + W(e) + HEURISTIC(v, t)$ 
14:  $pred[v] \xleftarrow{update} u$ 
15:  $Queue \leftarrow Queue \cup \{u\}$ 
16: return  $\emptyset$ 

```

---

**3. Improved A\* Algorithm****3.1. 16-Way Search**

Ships usually follow a great circle of a spherical Earth to minimize distance and save time and money. The Great Circle Route (GCR) is the shortest course between two points on the surface of a sphere. The GCR appears as curved lines on a map, so that it is difficult to search for curved paths with slope angles using 8-way search, defined at intervals of  $45^\circ$ , in the original A\* algorithm. Hence, in this study, 16-way search was utilized to navigate the optimal route in more directions. Figure 2 shows a schematic of the 8-way and 16-way search methods, where the gray-shaded central square indicates a current node, and the surrounding blue-shaded squares represent the candidate nodes that can be chosen as the subsequent moving node. The 8-way search method can explore the surrounding nodes located in eight directions with a  $45^\circ$  equidistant from the current node. The 16-way search method explores a total of 16 surrounding nodes, including 8-way search nodes, and 8 additional nodes extended left/right and up/down, respectively, from four vertex nodes as shown in Figure 2. This 16-way search method is expected to render the shape of the optimal route more realistic and smoother by allowing movements to neighbor nodes with various directions, as well as by minimizing the total cost to find the optimal route.



**Figure 2.** Configurations of (a) 8-way search method; (b) 16-way search method.

In this study, the ETA calculated using Speed Over Ground (SOG) values was used as the heuristic function, where the SOG was modeled via machine learning. However, because the machine-learning model requires a large computational time, the process of calculating the ETA for every iteration within the algorithm can be extremely inefficient. Therefore, the heuristic costs for ETA were calculated in advance to enable the algorithm to calculate the cost quickly.

Before the ETA is used as the heuristic function, the Course Over Ground (COG) must be defined because the ETA is calculated from the COG. The COGs are defined for the directions in which the ships are heading, so that 16 COG values are obtained for the 16-way search directions used in this study. Thus, the COG values are first defined for eight directions with an equal interval of  $45^\circ$  within the range of  $[0^\circ, 360^\circ]$ . Then, the remaining eight COG values are further defined for an angle of  $\pm \arctan(0.5) \sim 26.6^\circ$  around each direction. Finally, the total COG values are defined for 16 directions.

### 3.2. Heuristic Function Using Haversine Method

#### 3.2.1. Optimality Condition of Haversine Function

Vincenty's formula and the haversine function are used for calculating the distance between two points on a spherical surface. Vincenty's formula calculates the distance between two points on a surface of a spheroid using two related iterative methods used in geodesy, and the haversine function determines the distance between two points on a sphere given their longitudes and latitudes. Vincenty's formula offers higher accuracy than the haversine function, but it is more computationally intensive and slower [22,23]. In this study, because the ship route optimization required many route cost calculations and did not require extremely accurate distance calculations, the haversine function was used as a heuristic function.

However, before the haversine function is used as a heuristic function, it is necessary to determine whether it satisfies the optimality condition, including admissibility and consistency, for optimal path finding. In spherical geometry, even though the shortest distance between two points is an arc of a great circle, it is known that triangular inequality holds [24,25] if it is the length of the minor spherical line segment connecting other endpoints. This kind of heuristic that obeys triangular inequality is called a "consistent" or "monotonic" heuristic.

For example, if one arbitrary node  $u \in N$  and the target node  $t \in N$  on the sphere are considered, one can obtain the following relation by the triangular inequality, which satisfies the consistency condition. The haversine function value for the target node is 0, so that the one for any arbitrary node,  $h(u)$ , is always less than the distance from any arbitrary node  $u$  and the target node  $t$ , indicating that the haversine function is admissible.

$$\begin{aligned} h(u) &\leq d^*(u, t) + h(t) : \text{Consistency} \\ &\leq d^*(u, t) + 0 \\ &\leq d^*(u, t) : \text{Admissibility} \end{aligned} \quad (2)$$

Consequently, the heuristic function is admissible if it is consistent. Furthermore, if it is supposed that there exists node  $v \in N$ , which is the successor of the node  $u \in N$ , the optimality of the A\* algorithm can be derived under the condition of the admissible heuristic:

$$\begin{aligned} f(v) &= g(v) + h(v) \\ &\geq g(u) + d^*(u, v) + h(v) \\ &\geq g(u) + h(u) = f(u) : \text{Admissibility} \end{aligned} \quad (3)$$

Thus, nodes expand in the nondecreasing sense of  $f(n)$ , which results in the optimal solution of the A\* algorithm. Therefore, because the haversine function satisfies the consistency and admissibility conditions, they were used as the heuristic function to find the optimal ship routes in this study.



### 3.2.2. Grid Mapping

Before the route-finding algorithm is run, a base map, which is a graph including nodes and edges to be used as base terrain information for ship operation, must be defined at a constant latitude and longitude interval. However, the node and edge, which are arc shapes on the surface of a sphere, require complex calculations to find the optimal route, so the base map must be transformed into a two-dimensional(2-D) grid map through the equirectangular projection. This 2-D grid map enables the base map to be represented in rectangular shapes, which can help users to visually understand the real sphere map. However, if any place on the sphere is projected into a plane, distortion of its area, angle, direction, and distance occurs. All projections cause distortion in varying degrees and cannot be performed with preserving the above properties [26]. Therefore, projection needs to be chosen to best suit the user's purpose. In the case of route optimization, preserving the angle of the heading direction is very important because the optimal route is found based on the grid. However, AIS and weather data have the problem of using an equidistant latitude and longitude and therefore cannot preserve angles. Therefore, in this study, a method of converting the base map into a conformal map is proposed.

To convert the base map into a conformal map, the base map should locally maintain an equivalent scale in the  $x$ - and  $y$ -directions (east and north) based on the prime meridian. In other words, as the latitude changes from  $0^\circ$  to  $\varphi$ , the radius of the parallel of latitude  $\varphi$  also should be changed from  $R$  to  $R\cos\varphi$ , as shown in Figure 3. Thus, each distance along the parallel is increased by  $\sec\varphi$ , which is a scale factor for the parallel of latitude  $\varphi$ . Because of the conformality, the circle of the parallel is expanded by the factor  $\sec\varphi$  in all directions.

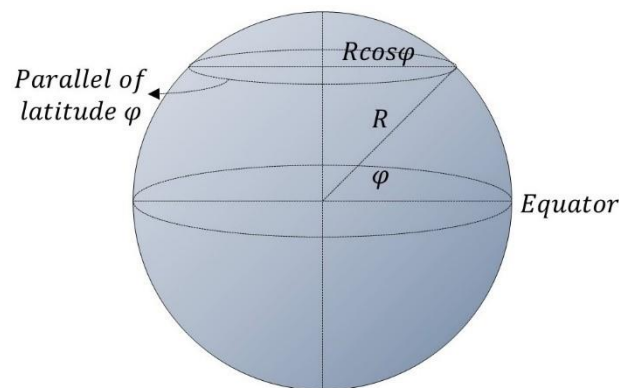


Figure 3. Spherical-shaped earth.

When the geographic coordinates of latitude  $\varphi$  are expressed as Cartesian coordinates of  $y(\varphi)$  on the base map with the  $x$ -axis along the equator and origin on the equator, by integrating  $R\sec\varphi$ , the corresponding  $y$ -coordinate is calculated as follows.

$$y(\varphi) = R \int_0^{\varphi} \sec\varphi d\varphi = R \ln \left| \tan \left( \frac{\varphi}{2} + \frac{\pi}{4} \right) \right| \quad (4)$$

The  $y$ -coordinate calculated using Equation (4) is used to calculate a ship's angle of navigation, COG, on the actual spherical surface of the earth.

Figure 4 shows how COG is defined through the conformal mapping. For this, the distance between two points in the  $y$ -direction is increased by the following scale factor  $S$  for conversion to the

conformal map during the movement of the base grid map from point  $p$  with latitude  $u$  to point  $q$  with latitude  $v$ , as shown in the figure.

$$S = \int_u^v \sec \varphi d\varphi \quad (5)$$

In other words, the spacing between the latitudinal lines of the two points must be increased by  $S$  to maintain the actual distance ratio from the earth. Consequently, the COG can be calculated from any arbitrary points  $p$  to  $q$  and is defined as follows.

$$\text{COG} = \begin{cases} \text{atan2}(n_{\text{col}}, S n_{\text{row}}), & \text{if } \text{atan2}(n_{\text{col}}, S n_{\text{row}}) \geq 0 \\ 360 + \frac{180}{\pi} \text{atan2}(n_{\text{col}}, S n_{\text{row}}), & \text{otherwise} \end{cases} \quad (6)$$

where  $n_{\text{row}}$  and  $n_{\text{col}}$  are the numbers of rows and columns that change when moving to the next node, respectively. The length of each grid is determined by the resolution of the map data.

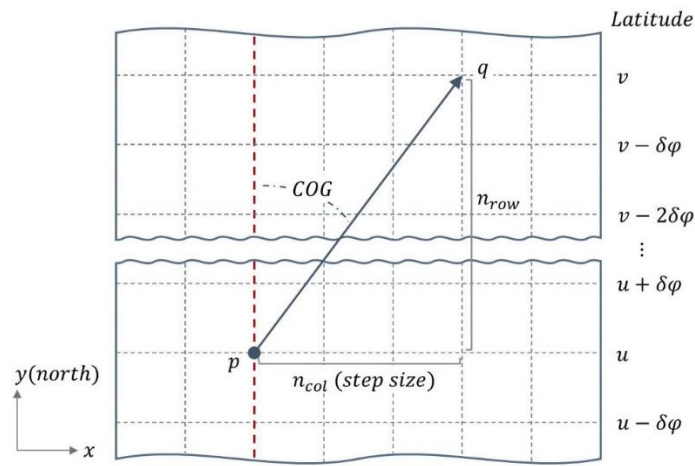


Figure 4. Calculation of COG.

A conformal map with this 2D array format was used to calculate the ETA-based path cost (heuristic cost of the A\* algorithm) without additional computation through simple array indexing, as described in the next section.

### 3.2.3. Cheap Heuristic

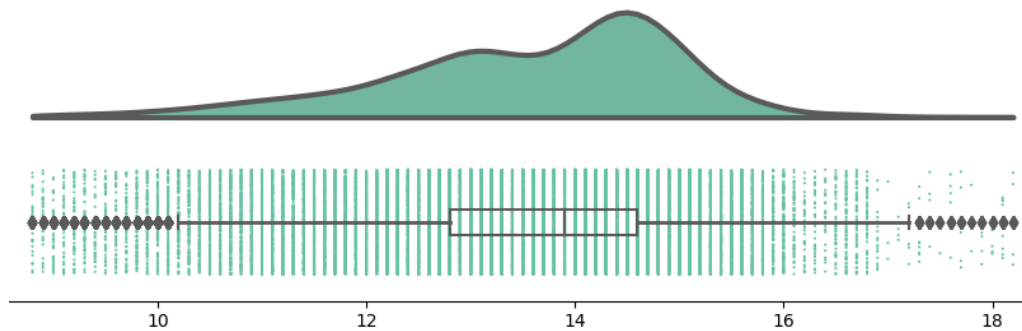
In this study, the edge weight used for the A\* algorithm was set to the ETA values calculated using machine-learning models, which are introduced in Section 4.1.3. The edge weight (actual cost,  $g$ ) is the cost to move from one node to another node, and the edge costs between adjacent nodes were precalculated to improve the efficiency of the algorithm. However, the heuristic cost is the estimated path cost from the current node to the target node; thus, its value cannot be precalculated, unlike the actual cost. As a result, to estimate the path cost accurately, a large number of COG values should be precalculated; however, this is undesirable in terms of computational efficiency. Therefore, the heuristic cost must be calculated in a simple formula without using COG data.

To preserve the admissibility of the A\* algorithm, the heuristic cost should always remain lower than the actual cost. In this study, SOG data contained in AIS data were used to calculate the heuristic cost (ETA) between arbitrary nodes and the target node. Because the ETA is distance/speed, theoretically using the maximum SOG value of a ship recorded in the AIS always results in a smaller value than the actual value, thus satisfying the availability condition. However, as shown in Table 1 and Figure 5, the SOG data distribution is sparse at high SOG values, so using the maximum SOG value results in more-conservative values than necessary. Therefore, as opposed to the perfect admissibility condition

being met, the  $h$  cost was calculated using 99 percentile values for the SOG, as shown in Equation (7). Thus, the optimal ship route obtained using proposed A\* algorithm can be considered near-optimal.

**Table 1.** Statistical characteristics of SOG.

Count	Mean	Std (knots)	Min (knots)	25% (knots)	50% (knots)	75% (knots)	99% (knots)	Max (knots)
44,405	13.5815	1.3941	8.8	12.8	13.9	14.6	16.1	18.2



**Figure 5.** Histogram and boxplot of SOG data.

The empirical maximum value of the SOG for the target vessel can be extracted from the AIS data, and Table 1 summarizes the statistical characteristics of the SOG data of an actual vessel for which the preprocessing has been completed. The methodology of data preprocessing is described in Section 4.1.3.

$$h = \frac{\text{dist}(u)}{\text{SOG}_{99\text{th percentile}}} \quad (7)$$

where  $\text{dist}(u)$  means the distance between an arbitrary node  $u$  and the target node  $t$  and is calculated using the haversine method. This approach has the advantage of enabling the A\* algorithm to converge efficiently to the solution; however, the optimality of the solution cannot be fully guaranteed because the admissibility of the heuristic has been violated.

### 3.3. Adaptive Grid System

In general, map data is collected in the form of a square grid. Thus, it is easy for typical rectangular projection methods such as equirectangular projection method to perform map projection because they can utilize the given form of map data as it is. The equirectangular projection is a very simple map projection used for the base map as meridional intervals of constant spacing and constant intervals of parallels. However, this projection is neither equal area nor conformal and it is distorted along the latitude. This distortion causes the problem of formation of disparate grids when the base map is decomposed in grid format. Even though the grid is defined at equal angle spacing, because the area of interest is close to the polar region, the distance between meridians becomes narrower, which means that the resolution is coarse near the equator and fine in the polar region. However, these irregular resolutions result in an unnecessarily large number of search nodes.

Therefore, in this study, an adaptive grid system was used for efficient space search of route finding, facilitating multiple steps of grids in the transverse direction according to the latitude values, which enables the grid map to be conformal. Although the shape of the grid map cannot accurately reflect the continuously changing latitudes because of its discrete nature, the adaptive grid system can be applied on the basis of the critical points at which the distance size of the grid changes in an integer ratio.

As explained in Section 3.2.2, the closer the latitude is to the polar region, the more distortion of the grid geometry, so the grid close to an equator with little distortion needs to be explored for

each grid in the transverse direction. On the other hand, the closer the latitude is to the polar regions, the closer the distance between the grids on the sphere is by the ratio of  $\cos(\varphi)$ , as shown in Figure 3; the map grids need to be grouped together to avoid unnecessary routing process that explores all grids.

Therefore, it is necessary to increase the efficiency of the routing process by efficiently grouping grids by reflecting the degree of distortion of grids as latitude increases. The grid search needs to determine a step size considering the distortion ratio of the grid shape. As shown in the right side of Figure 6 if the step size increases by 1 from 1 to 2 and 3 according to the latitude increases, an imbalance between the sectors classified according to the step size occurs, and thus, inefficient grid search is performed at high latitudes with severe grid distortion. On the other hand, as shown in the left side of Figure 6, if the step size increases more smoothly from 1 to 1.5 and 2.5, it can mitigate the imbalance between the sectors, enabling more efficient grid search. As a result, this adaptive grid system facilitates the generation of continuous shape paths by smoothly reflecting the distortion of the grid in accordance with continuous latitude changes. Thus, the route can be navigated efficiently, especially when performing route optimization at high latitudes; this thereby improves the ease of route optimization calculation in conjunction with the COG, which calibrates the direction of ship's navigation on a real spherical surface as explained in Section 3.1.

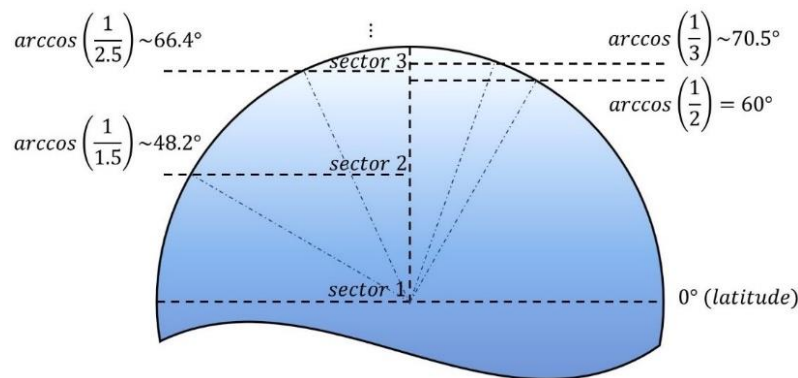


Figure 6. Northern hemisphere of the earth.

## 4. Case Study

### 4.1. Prediction of Ship's Estimated Time of Arrival

To find the optimal ship route, FOC should be used to evaluate the economics of ship routes. However, FOC data were not available in this research; thus, the SOG from AIS and maritime weather data was used to derive the ETA that will be used as economical index instead of FOC, since ETA and FOC parameters are linearly correlated [6]. Nevertheless, to ensure that the optimal route based on the ETA is equivalent to that based on the FOC, the following assumptions are needed [6].

- The revolutions per minute of the propeller should be maintained with a constant value throughout the operation of the vessel,
- At the same time, with condition (a), the main engine should have a constant torque value,

which means that engine power remains constant as follows.

$$\text{Engine Power} = \text{Engine RPM} \times \text{Engine Torque} \quad (8)$$

However, because these assumptions cannot be satisfied under real operational conditions, using an ETA optimal route as the most economical route would include prediction errors. Nevertheless, in general, shipping companies do not share engine data of ships, such as FOC, so ship route optimization with only with AIS and weather data may be necessary. In addition, even if engine data are used to optimize ship routes, because it is necessary to consider more parameters to predict the

FOC, the routing process requires large calculation time. Thus, the route optimization only using AIS and weather data can provide the optimal ship route more efficiently than the one using engine data.

#### 4.1.1. Dataset 1: AIS Data

AIS is a system designed to be capable of providing information about a ship to other ships and to coastal authorities. The information includes the ship location, speed, and other navigational information in real time. The IMO has recommended that vessels use the AIS to enhance the safety and security of the marine environment. Many countries and institutes carry out effective rescue activities and manage the safety of ships through continuous monitoring of AIS data, which also facilitates vessel traffic service. The AIS data are originally divided into static, dynamic, navigational, and text communication information; however, dividing them into static and dynamic information is common. The detailed composition of AIS data generally conforms with international standards [27].

The AIS data used in this study were collected from 76 ships during 2018. The static and dynamic data were collected daily for 1 year. These two types of data tables are defined by three common factors: the Maritime Mobile Service Identity (MMSI); message type and date time stamp that identify which vessel the data were collected from; and what kind of attributes the data have. The MMSI is an identifier of vessels and it enables categorization of ships on the basis of the country to which they belong. The message type is used only for information management in the database and is not directly related to ships. The date time stamp associated with information pertaining to the time at which the data were collected was Korean Central Standard Time (KST).

The static AIS data consist of 11 parameters excluding these three common factors (MMSI, message type and date time stamp). The IMO number is used to distinguish ships along with the call sign, the vessel name, and the type of ship, where the call sign is used as an identifier especially for communication between ships. The dimensions A to D indicate the locations of the electronic position fixing devices from the front, back, left, and right ends of a ship. The ETA in the static AIS data only includes the information about the expected date on which the voyage is finished and not the total time required for the entire voyage. The draft (or “draught”) is the vertical distance from the bottom of the hull to the sea surface, indicating how far the vessel extends below the surface. The static AIS information is shown in Table 2. The data depicted in Table 2 follow the order of the raw static AIS dataset and type 5 indicates the AIS message type ID related to the static and voyage data.

**Table 2.** Static AIS data configurations.

No.	Parameter	Unit	Remark
1	MMSI	-	Type 5 (Static information)
2	IMO number	-	
3	Call sign	-	
4	Name	-	
5	Type of ship	-	
6~9	Dimension A~D	M	
10	Electronic fixing device	-	
11	ETA	YYYYMMDD	
12	Max draught	M	
13	Msg type	-	
14	Date time stamp	KST	

The dynamic AIS data have different factors from the static data, as shown in Table 3. Type 1, 2, and 3 indicate the AIS message IDs related to the position of the ship. First, the position of the vessel is described using the latitude and longitude coordinates. The SOG and COG are the ship’s final speed and direction considering environmental disturbance such as wind and wave. The ROT describes the rotation angle per minute, which can be optionally included in the AIS dataset. The true heading indicates the direction in which the vessel is facing, not considering any external interferences. Thus,

there is always a slight difference between the heading and COG. If the vessel is under calm and still sea conditions with no drift, the heading and COG values are identical; if not, they are somewhat different. The navigation status depicts the operational status of a ship and is a categorical factor that appears as “mooring,” “anchoring,” or “underway using engine” when the vessel is in actual voyage.

**Table 3.** Dynamic AIS data configurations.

No.	Parameter	Unit	Remark
1	MMSI	-	
2	Latitude	DDMM.mmmm	
3	Longitude	DDDMM.mmmm	
4	SOG	knot	
5	ROT	deg/min	Type 1,2,3
6	COG	deg	(Dynamic information)
7	True heading	deg	
8	Navigational status	-	
9	Msg type	-	
10	Date time stamp	KST	

#### 4.1.2. Dataset 2: Marine Weather Data

The marine weather data used in this study were collected in 2018 and acquired at intervals of 1 day at midnight. The datasets are defined in 0.5° resolution for each latitudinal and longitudinal direction, as shown in Table 4. The factors from the first to the ninth describe height, direction, and period of the total wave, wind wave, and swell wave, respectively. It is common to use the total wave information because the total wave is the combination of the wind and swell wave [28]. The wind UV and VV are the velocities along the longitudinal and latitudinal directions in forward order, and the pressure Mean Sea Level (MSL) and pressure surface represent the atmospheric pressure measured at MSL and surface of the earth, respectively. The sea surface temperature and salinity are measured at the sea surface. Finally, the current UV and VV are the velocity components of the ocean current in the same sequence as the format of the wind UV and VV factors. The composition of the marine weather data is shown in Table 4 where the marine weather data were defined in a resolution of 0.5° for latitude and longitude.

**Table 4.** Marine weather data configurations.

No.	Parameter	Unit
1	Total wave height	m
2	Total wave direction	deg
3	Total wave period	sec
4	Wind wave height	m
5	Wind wave direction	deg
6	Wind wave period	sec
7	Swell wave height	m
8	Swell wave direction	deg
9	Swell wave period	sec
10	Wind UV	m/s
11	Wind VV	m/s
12	Pressure MSL	hPa
13	Pressure surface	hPa
14	Ambient temperature	°C
15	Sea surface salinity	psu
16	Sea surface temperature	°C
17	Current UV	m/s
18	Current VV	m/s



#### 4.1.3. Learning-Based Regression Model

As shown in Section 4.1.2, AIS and weather data include many parameters, and their number is large, so that a classical analytical regression model cannot be used [29]. Therefore, this study used a machine-learning-based regression model to address effectively the large number of AIS and marine weather data. To generate the machine-learning-based regression models, SOG is used as a target variable that is used to calculate the values of the ETA. In the given grid map  $G$ , the expected ETA value for traversing between the two nodes  $u, v \in N$  is calculated as follows:

$$ETA = \frac{2distance}{SOG(u) + SOG(v)} \quad (9)$$

Using Equation (9), the ETA is analytically calculated by dividing the distance into the arithmetic mean of the expected SOG values at each node  $u$  and  $v$ . Moreover, the distance can be calculated by using the haversine method discussed in Section 3.2.1. Equation (9) can produce inaccurate results depending on ship and weather conditions at the two points, but the data used in this study are defined in map grid, so Equation (9) is the most economical and realistic way of deriving ETA values in terms of calculation. The ETA corresponds to the objective function to be minimized during the ship route-finding process; therefore, the optimal route derived from the routing process is the ETA-based optimal route.

#### Data Preprocessing

Prior to data preprocessing, candidate features, which are used to generate the machine-learning model for the SOG, should be chosen. The route optimization is performed under the condition that the ship is always in operation, so only the data where the navigational status was “underway using engine” was extracted. Additionally, because the progress direction of the ship within the grid map corresponds to COG, the true heading was not selected as a feature for SOG prediction.

Once the unnecessary features are removed, the raw data need to be refined. The AIS data include anomalies because of sensor malfunction or some other reasons, such as unexpected external disturbance; thus, they should be identified and removed before analysis. The anomalies in the AIS data are either so large or so low that a simple comparison with the mean value makes it easy to identify them. In addition, there are not many outliers in AIS datasets, and complicated statistical analysis may result in reducing the efficiency of the routing process. Thus, in this study, numerical thresholds were used to screen out the outliers, which is also known as “rule-based preprocessing.” Afterward, categorical data in string format excluding operational status information, such as MMSI or IMO number, were removed because they could not be used as input parameters for the regression model. For numerical data, such as latitude, longitude, and COG, data outside the upper and lower limits determined based on physical limits were removed as outliers. Unlike the AIS data, marine weather data do not require additional data preprocessing because they are received in a state of being refined from a specialized company. Consequently, data preprocessing was carried out only for AIS data. Table 5 summarizes the bound values of each numeric AIS factor.

**Table 5.** Bound values for data filtering.

AIS Data Type	Lower Bound	Upper Bound
Latitude	−90	90°
Longitude	−180°	180°
COG	0°	360°
Navigational status	“Underway using Engine”	

#### Feature Extraction and Feature Selection

The wind and ocean current biaxial velocities were converted into speed and angle components through vector operations to provide an intuitive understanding of the contribution of wind and

ocean current to the composition of the regression model. Here, the angle is defined as the clockwise direction in relation to zero degree in the north direction. Thus, UV and VV factors mean the velocities in the directions of 90° and 0° corresponding to the  $x$ -axis and  $y$ -axis, respectively. The velocity and the direction are defined using Equations (10) and (11) as follows.

$$v = \sqrt{UV^2 + VV^2} \quad (10)$$

$$\theta = \begin{cases} \text{atan2}(UV, VV), & \text{if } \text{atan2}(UV, VV) \geq 0 \\ 360 + \frac{180}{\pi} \text{atan2}(UV, VV), & \text{otherwise} \end{cases} \quad (11)$$

In this study, the latitude and longitude were not considered as candidate features because they only serve to filter abnormal data in data preprocessing and are unnecessary factors for predicting the SOG. In other words, the positional information can be used to obtain the weather information at a specific location, but it does not directly affect the SOG. However, because the weather-related features have a direct effect on the SOG, they must be considered as candidate features. In general, when a ship arrives or departs from a port, it often changes its heading direction significantly from its low-speed operation, thus having a high ROT value. However, in this study, route optimization is performed at high speed operating conditions, so ROT is often near zero and therefore is excluded from the candidate features for SOG prediction, and significant ROT values were already filtered by limiting SOG values to more than 5 knots during preprocessing.

Next, gross tonnage data were added to the candidate features and could be obtained by referring to the specifications of each vessel. The gross tonnage represents the volume of a vessel, and 1 volume tonnage corresponds to a volume of 1.133 m<sup>3</sup>. Because all vessels have their unique gross tonnage values, they can be used as important factors to categorize each vessel in the learning model and can represent vessel dimensional information, such as length and width. Because the gross tonnage already includes the vessel dimensional information, dimension A–D information in the static AIS dataset was excluded from the candidate features. Table 6 shows the list of candidate features.

**Table 6.** List of candidate explanatory variables.

Variable Type	No.	Parameter	Unit	Remark
Explanatory	1	COG	deg	AIS
	2	Draught	m	
	3	Total wave height	m	Weather
	4	Total wave direction	deg	
	5	Total wave period	sec	
	6	Wind wave height	m	
	7	Wind wave direction	deg	
	8	Wind wave period	sec	
	9	Swell wave height	m	
	10	Swell wave direction	deg	
	11	Swell wave period	sec	
	12	Pressure MSL	m/s	Extracted
	13	Pressure surface	m/s	
	14	Ambient temperature	hPa	
	15	Sea surface salinity	hPa	
	16	Sea surface temperature	°C	
	17	Wind speed	m/s	
	18	Wind angle	deg	
	19	Current speed	m/s	
	20	Current angle	deg	
	21	Gross tonnage	tons	Additional
Response	1	SOG	knots	AIS

Before generating the learning model, it is important to ensure that multicollinearity exists within the explanatory variables. Multicollinearity indicates the presence of linear dependencies among the explanatory variables, which causes the variance of the regression coefficient to increase, resulting in a decrease of the accuracy of the regression model. To resolve this problem, Variance Inflation Factor (VIF) analysis is widely used, which is an indicator of how well a particular explanatory variable can be described through ordinary least squares regression analysis with the remaining explanatory variables. The formula to determine VIF is as follows.

$$VIF_k = \frac{1}{1 - R_k^2} \quad (12)$$

where  $VIF_k$  is the VIF value of the  $k$ th explanatory variable and  $1 \leq k \leq n$  when  $n$  explanatory variables are given. Thus, when an explanatory variable highly depends on other variables, it has a high VIF value. Otherwise, it has a low VIF value. The dependent variables with high VIF values are dropped before the regression model is generated. Generally, a VIF value of 5 or higher for a descriptive variable is likely to cause an over-fitting problem; therefore, in this study, a variable of 5 or higher was eliminated, and a learning model was created. Table 7 summarizes the feature selection process through VIF analysis.

Table 7. VIF analysis results.

No.	Parameter	Phase 1		Phase 2		Selected
		VIF	Decision	VIF	Decision	VIF
1	COG	1.2962	-	1.2944	-	1.2931
2	Draught	2.3784	-	2.3739	-	2.3738
3	Total wave height	24.4058	-	4.0210	-	3.9379
4	Total wave direction	3.0208	-	1.4624	-	1.4596
5	Total wave period	6.4452	-	3.3574	-	3.3115
6	Wind wave height	9.2041	Dropped	-	-	-
7	Wind wave direction	3.2017	Dropped	-	-	-
8	Wind wave period	3.1675	Dropped	-	-	-
9	Swell wave height	14.0707	Dropped	-	-	-
10	Swell wave direction	1.5663	Dropped	-	-	-
11	Swell wave period	3.1215	Dropped	-	-	-
12	Pressure MSL	243.8376	-	243.4247	Dropped	-
13	Pressure surface	244.5687	-	244.1839	-	1.5887
14	Ambient temperature	17.6496	-	17.45677	Dropped	-
15	Sea surface salinity	1.5039	-	1.4976	-	1.4838
16	Sea surface temperature	15.7028	-	15.5211	-	1.7525
17	Wind speed	4.2820	-	2.4308	-	2.4033
18	Wind angle	1.8113	-	1.3921	-	1.3850
19	Current speed	1.1089	-	1.0993	-	1.0959
20	Current angle	1.1038	-	1.0995	-	1.0991
21	Gross tonnage	2.1512	-	2.1498	-	2.1474

After the number of candidate features was reduced through VIF analysis, the features associated with the wind wave and the swell wave were deleted in first phase. Even though the low VIF values for the direction and period of total wave, wind wave, and swell wave are observed, the VIF analysis, which quantifies the extent of linear correlation based on the least-square method, does not accurately estimates their nonlinear correlation [28]. As mentioned in Section 4.1.2., the total waves are highly correlated with wind waves and swell waves because total waves correspond to the combination of the wind waves and swell waves. As a result, three parameters (direction, height, and period) related to the total wave remain to maintain independence between input variables, and others related to wind and swell are excluded from the candidate group for the simplicity of the learning model.

Because the pressure MSL, pressure surface, the ambient temperature and sea surface temperature are highly correlated, one of them should be removed from the candidate features to avoid multicollinearity of those features. The sea surface temperature and pressure surface were directly measured at sea level and therefore had a direct effect on SOG; however, the pressure MSL and ambient temperature were not. Thus, the pressure MSL and ambient temperature were excluded from the candidate features.

Consequently, 13 out of 21 features were selected as final input variables, and the results are presented in Table 8.

**Table 8.** List of selected explanatory variables.

Variable Type	No.	Parameter	Unit	Remark
Explanatory	1	COG	deg	AIS
	2	Draught	M	
	3	Total wave height	m	Weather
	4	Total wave direction	deg	
	5	Total wave period	sec	
	6	Pressure surface	m/s	
	7	Sea surface salinity	hPa	
	8	Sea surface temperature	°C	
	9	Wind speed	m/s	Extracted
	10	Wind angle	deg	
	11	Current speed	m/s	
	12	Current angle	deg	
	13	Gross tonnage	tons	Additional
Response	1	SOG	knots	AIS

### Learning Model

In this study, the eXtreme Gradient Boosting (XGB) regression model was used to predict the SOG. The XGB is an ensemble model consisting of multiple decision trees and has the advantage of mitigating the possibility of over-fitting by regularizing input variables [30]. The excellent performance of XGB compared with other learning models was verified by Abebe et al. [29].

Hyperparameters of the XGB regressor were optimized using Bayesian optimization methods with 10-fold cross validation. Table 9 shows the search space and optimized values of the hyperparameters, and Table 10 shows the accuracy result of the XGB regression model.

**Table 9.** Search space and optimal value of hyperparameters.

Parameters	Search Space	Optimal Value
colsample_bytree	[0.01, 1]	0.42
learning_rate	[0.01, 1]	0.2
gamma	[0, 30]	0.6
max_depth	[1, 100]	30
subsample	[0.1, 1]	0.76

**Table 10.** Accuracy of XGB regression model.

Data Split Ratio (Train: Test)	R <sup>2</sup> for Train Set	R <sup>2</sup> for Test Set
7:3	99.079	97.888

#### 4.2. Result and Discussion

To carry out the routing simulation, one ship with real AIS data was chosen as the target vessel, and the target route was set to the seaway from the port of Gwang-Yang, Korea, to Westshore Terminal, Canada. The name of the ship has been withheld owing to confidentiality reasons. Because the marine weather data used were defined in a resolution of  $0.5^\circ$  for latitude and longitude, 2D linear interpolation was used to enable its application to the  $0.1^\circ$  resolution of the base map. The target ship traveled continuously from Jan. 11 to Jan. 29, 2018, without any stopovers. Therefore, the heuristic array was precalculated on the basis of marine weather data from 11 January to 30 January 2018, and the daily heuristic cost was calculated according to the change of path cost. In addition, the simulation was performed with a draft value of 18 m and a gross tonnage value of 92,353, referring to the historical operation data of the target ship.

Figure 7 is a schematic diagram of the total wave height from the climate data of 20 January 2018, and Figure 8 represents the SOG map of the target vessel predicted through the obtained learning model. The SOG map shown in Figure 8 was calculated for 16 directions for ships going east as of 20 January 2018, and route optimization was performed using Equation (9) to minimize the ETA values derived from the SOG map for the 16 directions. As shown in Figures 7 and 8, SOGs are generally predicted to be low in areas where high total wave height occurs, which again confirms that weather factors such as wave height are essential considerations in predicting the ship speeds.

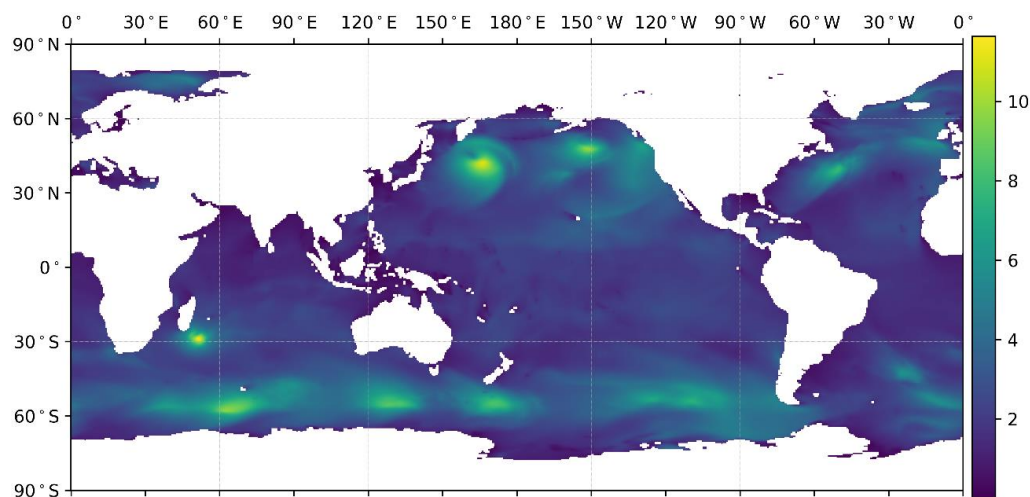


Figure 7. Total wave height map on 20 January 2018 (m).

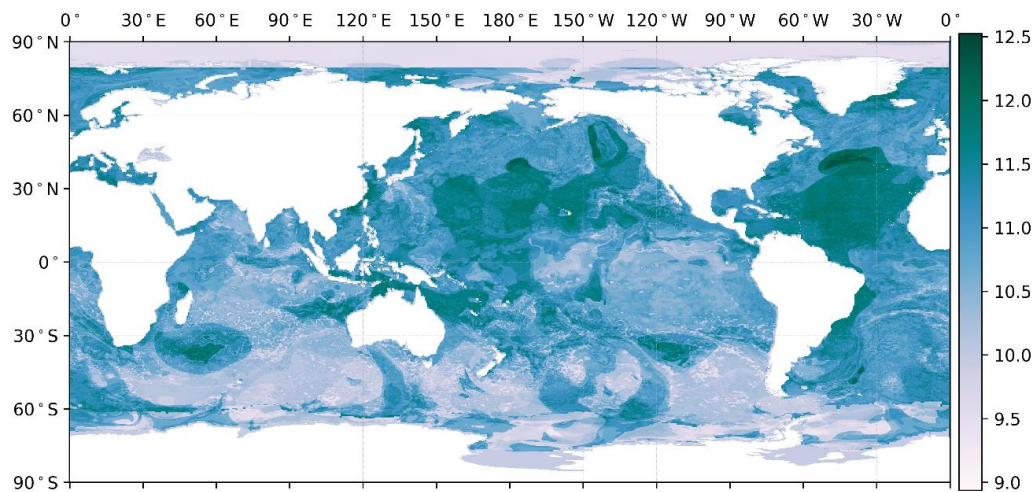


Figure 8. Predicted SOG map on 20 January 2018, East (knot).



The simulation was conducted for four cases, 8-way or 16-way search combined with adaptive or non-adaptive grids. The simulation results using four methods were compared with the sailing time and traveled distance calculated from actual operational data of the target ship, as given in Table 11. Figure 9 shows the real AIS trajectory and simulation results using the 16-way search method with adaptive grids. It shows a map of already finished route optimization, so only the “closed” region is shown, and the red border indicates the area used as “open”. The A\* algorithm contains adjacent nodes of nodes adopted as current nodes in the open set first, and the nodes with the lowest cost in the open set are adopted as current nodes, repeatedly storing them in the closed set, a set of nodes that have been already explored. Thus, while the closed set is around the optimal route, the node in the open set that is not selected as the current node appears to surround the boundary of the closed set as shown in Figure 9.

Table 11. Simulation results.

Method	8-Way Search		16-Way Search		AIS Trajectory
Grid type	Non-Adaptive	Adaptive	Non-Adaptive	Adaptive	
ETA [hour]	445.68	438.75	436.57	429.33	455.6
Distance [km]	9066.38	8900.79	8893.00	8777.16	9899.26
Searched nodes	324,729	311,496	325,412	309,816	-
Path nodes	1097	980	873	634	-

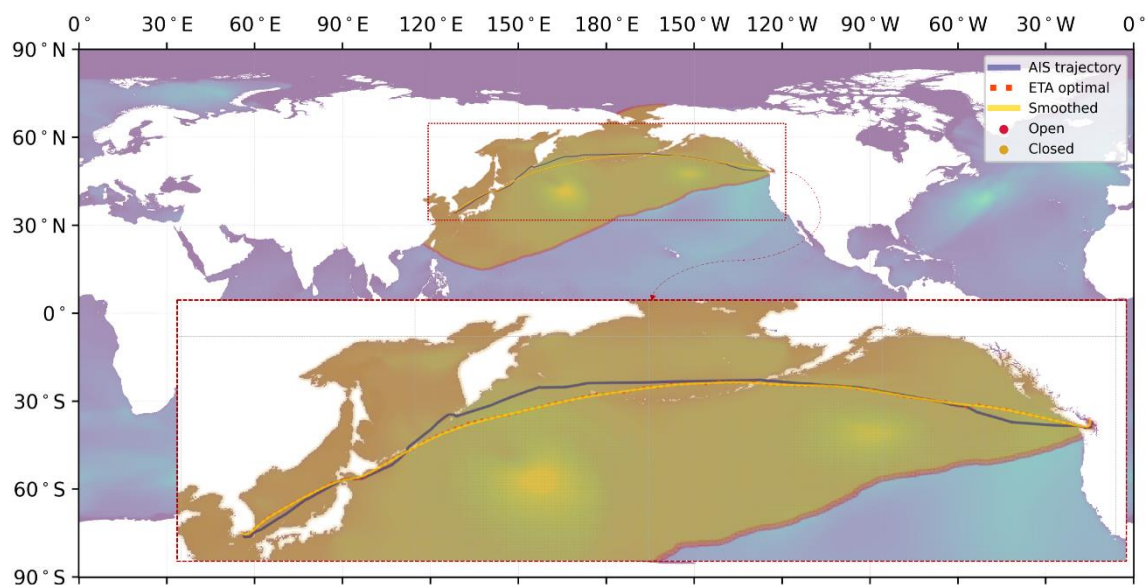


Figure 9. Result of 16-way search with adaptive grid.

However, the optimized route through the grid has a sharp edge, so it is necessary to smooth through smoothing curves such as Beizer curve, b-spline, etc. In this study, Beige curve, commonly used for path smoothing, was used as it is easy to generate and control. As the Figure 9 shows, the predicted traveled distance is much shorter when operating on a more optimized route than the actual AIS trajectory, and the smoothed route can better represent the real ship route with a smooth curve shape.

Likewise, as shown in Table 11, 16-way search has lower values for all ETAs, distances, searched nodes, and path nodes than 8-way search because it selects routes smoother than those of 8-way search. Similarly, adaptive grids do not allow unnecessary grid search, so either 8- or 16-way searches combined with adaptive grids have more efficiency in the ship route optimization process than the ones without



adaptive grids. Thus, the 16-way search method combined with adaptive grids yields more-economical paths than other methods in terms of ETA, distance, and the number of searched nodes.

The 16-way search method combined with adaptive grids reduces the ETA by approximately 1.65% and the number of searched nodes by 4.79%. From the viewpoint of economical operations, although the estimated ETA is not much reduced, the target ship can theoretically save as much as \$30,000 under the assumption that the target ship consumes 1500 gallons of diesel per hour and the cost of diesel is \$2.8 per gallon. In addition, it is observed that the adaptive grids reduce the number of search nodes to approximately 4.79%, facilitating a more effective ship-routing process. The number of search nodes is also decreased by approximately 27.38%, which means that the optimal path with a smoother shape has been derived as opposed to the one without the adaptive grid, enabling reduction of the difference between the original optimal path and the smoothed optimal path.

When the obtained optimal routes using the four methods are compared with the AIS trajectory, they appear to be more continuous than the real AIS trajectory. Accordingly, this continuous ship route reduces the travel distance by approximately 1000 km for all cases when compared with the AIS trajectory, which also results in decreasing the ETA cost. Thus, the 16-way search combined with adaptive grids ensures economical ship operation along with the efficiency of the ship route optimization. When it is used in real operation, the navigator can easily obtain guidelines for economical ship operation, which reduces ship operating costs.

Even though the improved A\* algorithm utilized data operated in the past to compare with actual routes operated, it can be still used for real-time data. In the algorithm, the SOG used for ETA prediction can periodically update the SOG prediction results based on weather data collected every 6, 12 or 24 h independently of ship operation. The results of this calculated SOG prediction can be uploaded in an indexing manner with low memory requirements from the improved A\* algorithm, so the optimum route finding can be updated periodically within about two seconds of the actual voyage.

## 5. Conclusions

An improved A\* algorithm based on adaptive grids was proposed. It constructs learning models using marine weather data and AIS data to evaluate the ETA and economic factor of ship operation, and it enables economical routing of ships on the basis of that factor. The adaptive grid system improves the computational complexity itself through the improvement of map retention to solve the high time complexity of the existing A\* algorithm. It was combined with the 16-way search method to create a more visually natural and economical optimal solution by smoothing the optimal path. The low-cost heuristic of the A\* algorithm was used to reduce the computational time for optimal route finding, and it was confirmed that the admissibility and optimality conditions of the obtained optimal route were satisfied. To validate the proposed method, the routing simulations from Gwang-yang, Korea, to Westshore Terminal, Canada, confirmed that the proposed method provides an economical route compared with the actual AIS trajectory. Future research is planned to determine who chooses the appropriate heuristic and to enable more efficient routing. In addition, the obtained route is still the near-optimal route; therefore, methods to improve the performance of the learning models will be explored.

**Author Contributions:** Conceptualization, M.A., Y.N. and Y.W.S.; data acquisition, Y.W.S. and Y.N.; methodology, M.A.; data curation: S.L.; coding, M.A. and Y.W.S.; validation, M.A.; formal analysis, M.A.; investigation, M.A.; resources, Y.N. and I.L.; writing-original draft preparation, M.A.; writing-review and editing, Y.N. project administration: K.C.K., J.B., D.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIP) through GCRC-SOP (No. 2011-0030013), Korea government(MSIT) (No. 2018R1D1A1A02086093), National Innovation Cluster Program (P0006887, Build on Cloud Intelligence Platform based Marine Data) and Human Resources Development program (No. 20164030201230) funded by the Ministry of Trade, Industry & Energy (MOTIE) and Korea Institute for Advancement of Technology (KIAT).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Krata, P.; Szlapczynska, J. Ship weather routing optimization with dynamic constraints based on reliable synchronous roll prediction. *Ocean Eng.* **2018**, *150*, 124–137. [\[CrossRef\]](#)
2. Zhang, L.; Meng, Q.; Fwa, T.F. Big AIS data based spatial-temporal analyses of ship traffic in Singapore port waters. *Transp. Res. Part E Logist. Transp. Rev.* **2019**, *129*, 287–304. [\[CrossRef\]](#)
3. Fabbri, T.; Vicen-Bueno, R. Weather-Routing System Based on METOC Navigation Risk Assessment. *J. Mar. Sci. Eng.* **2019**, *7*, 127. [\[CrossRef\]](#)
4. Chu, P.C.; Miller, S.E.; Hansen, J.A. Fuel-saving ship route using the Navy's ensemble meteorological and oceanic forecasts. *J. Def. Model. Simulation Appl. Methodol. Technol.* **2013**, *12*, 41–56. [\[CrossRef\]](#)
5. Zhou, P.; Wang, H.; Guan, Z. Ship weather routing based on grid system and modified genetic algorithm. In Proceedings of the 2019 IEEE 28th International Symposium on Industrial Electronics, Vancouver, BC, Canada, 12–14 June 2019; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2019; pp. 647–652.
6. Mezaoui, B.; Takashima, K.; Shoji, R. On the fuel saving operation for coastal merchant ships using weather routing. *Int. J. Mar. Navig. Saf. Sea Transp.* **2009**, *3*, 401–406.
7. Gkerekos, C.; Lazakis, I.; Theotokatos, G. Machine learning models for predicting ship main engine Fuel Oil Consumption: A comparative study. *Ocean Eng.* **2019**, *188*, 106282. [\[CrossRef\]](#)
8. Gkerekos, C.; Lazakis, I. A novel, data-driven heuristic framework for vessel weather routing. *Ocean Eng.* **2020**, *197*, 106887. [\[CrossRef\]](#)
9. Tsou, M.-C.; Cheng, H.-C. An Ant Colony Algorithm for efficient ship routing. *Pol. Marit. Res.* **2013**, *20*, 28–38. [\[CrossRef\]](#)
10. Wang, H.-B.; Li, X.; Li, P.; Veremey, E.I.; Sotnikova, M.V. Application of Real-Coded Genetic Algorithm in Ship Weather Routing. *J. Navig.* **2018**, *71*, 989–1010. [\[CrossRef\]](#)
11. Veneti, A.; Konstantopoulos, C.; Pantziou, G. Evolutionary Computation for the Ship Routing Problem. In *Modeling, Computing and Data Handling Methodologies for Maritime Transportation*; Springer: Cham, Switzerland, 2018; pp. 95–115.
12. Bellman, R. On a routing problem. *Q. Appl. Math.* **1958**, *16*, 87–90. [\[CrossRef\]](#)
13. Ford, L.R., Jr. *Network Flow Theory*; RAND Corp: Santa Monica, CA, USA, 1956.
14. Moore, E.F. The Shortest Path through a Maze. In *Proceedings of an International Symposium on the Theory of Switching*; Harvard University Press: Cambridge, MA, USA, 1959; pp. 285–292.
15. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [\[CrossRef\]](#)
16. Silveira, P.; Teixeira, A.P.; Soares, C.G. AIS Based Shipping Routes Using the Dijkstra Algorithm. *TransNav Int. J. Mar. Navig. Saf. Sea Transp.* **2019**, *13*, 565–571. [\[CrossRef\]](#)
17. Wang, H.; Mao, W.; Eriksson, L. A Three-Dimensional Dijkstra's algorithm for multi-objective ship voyage optimization. *Ocean Eng.* **2019**, *186*, 106131. [\[CrossRef\]](#)
18. Hart, P.; Nilsson, N.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [\[CrossRef\]](#)
19. Stentz, A. Optimal and efficient path planning for partially known environments. In *Intelligent Unmanned Ground Vehicles*; Springer: Boston, MA, USA, 1997; pp. 203–220.
20. Daniel, K.; Nash, A.; Koenig, S.; Felner, A. Theta: Any-Angle Path Planning on Grids. *J. Artif. Intell. Res.* **2010**, *39*, 533–579. [\[CrossRef\]](#)
21. Dechter, R.; Pearl, J. Generalized best-first search strategies and the optimality of A. *J. ACM* **1985**, *32*, 505–536. [\[CrossRef\]](#)
22. Vincenty, T. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Surv. Rev.* **1975**, *23*, 88–93. [\[CrossRef\]](#)
23. Mahmoud, H.; Akkari, N. Shortest path calculation: A comparative study for location-based recommender system. In Proceedings of the (WSCAR 2016): 2016 World Symposium on Computer Applications & Research, Cairo, Egypt, 12–14 March 2016; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2016; pp. 1–5.
24. Ramsay, A.; Robert, R.D. *Introduction to Hyperbolic Geometry*, 1st ed.; Springer: New York, NY, USA, 1995; p. 17.

25. Oliver, B.; Jeff, T.; Fabio, R. Simplex-Tree Based Kinematics of Foldable Objects as Multi-body Systems Involving Loops. In *Robotics: Science and Systems IV*; MIT Press: Cambridge, MA, USA, 2009; pp. 191–198.
26. Pressley, A. *Elementary Differential Geometry*, 2nd ed.; Springer: London, UK, 2010; pp. 247–268.
27. IMO. Available online: <http://www.imo.org/en/OurWork/Safety/Navigation/Documents/227.pdf> (accessed on 3 April 2020).
28. NOAA. Available online: <https://www.ndbc.noaa.gov/waveobs.shtml> (accessed on 9 August 2020).
29. Abebe, M.; Shin, Y.W.; Noh, Y.; Lee, S.; Lee, I. Machine Learning Approaches for Ship Speed Prediction towards Energy Efficient Shipping. *Appl. Sci.* **2020**, *10*, 2325. [[CrossRef](#)]
30. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 13–17.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).