Journal of *Marine Science and Engineering*

MDPI

Article

Research on Real-Time Optimal Path Planning Model and Algorithm for Ship Block Transportation in Shipyard

Chong Wang [1,2], Kang Wang [2], Jiabin Tao [3] and Yongqing Zhou [1,2,*]

[1] Key Laboratory of High Performance Ship Technology, Ministry of Education, Wuhan University of Technology, Wuhan 430063, China; chriswang@whut.edu.cn
[2] School of Transportation, Wuhan University of Technology, Wuhan 430063, China; 259106@whut.edu.cn
[3] School of Energy and Power Engineering, Wuhan University of Technology, Wuhan 430063, China; taojiabin@whut.edu.cn
* Correspondence: yong_qing@whut.edu.cn

Received: 14 October 2020; Accepted: 2 December 2020; Published: 5 December 2020

Abstract: Special vehicles called transporters are used to deliver heavy blocks in the shipyard. With the development and application of information and communication technology in shipyards, the real-time positioning and ship blocks online scheduling system for transporters are being developed. The real-time path planning of transporters is important for maintaining the overall production schedule of ship blocks. Because of the large volume and heavy weight of ship blocks, there may be some problems, such as high energy consumption, block deformation and other security issues, when transporters loading a block make a turn. So, fewer turns of the transporters are also important to make a block transportation schedule. The minimum driving distance and fewer turns are considered simultaneously for transporter real-time path planning in this paper. A hybrid model considering the number of turns and the optimal path of the transporter is constructed. Moreover, the optimal scheduling model, considering path missing, is also discussed. Several shortest path algorithms are analyzed, which show that the Dijkstra algorithm is the best way to solve this model. From the attained simulation results, we demonstrate that the proposed model and algorithm have the ability to effectively solve real-time path planning for the ship block transportation in shipyards.

Keywords: shipyard; block transportation; transporter; real-time path planning; Dijkstra algorithm

1. Introduction

The shipbuilding industry is facing the multifaceted problem of less demand and oversupply with worldwide economic crisis and industrial stagnation. Especially for China shipbuilding, the advantage of "low labor cost" has disappeared, raw material and equipment prices have risen sharply, and the pressure of the cost of the shipyard has increased without price competitiveness. The only way out of this dilemma is to enhance the core competitiveness of Chinese shipyards through improving the quality and efficiency of ship construction, reducing the cost and energy consumption.

The shipbuilding industry is a highly complicated industry that normally operates under a make-to-order strategy to produce large structures characterized by complex configurations, long make-span, and advanced manufacturing technology. A ship is divided into several units called blocks, with various shapes and weights, to increase productivity and reduce the time for final assembly process in a dry dock, which is a process that causes a major bottleneck in the shipbuilding process. This approach has the benefit of enabling the construction of many ships simultaneously [1].

The blocks undergo a series of operations, which include pre-outfitting, painting, assembly, and pre-erection, as work in process before the ship is erected in the dock. The processing of the blocks

is performed at several shops scattered in a shipyard, and special vehicles, called transporters, are used to deliver the heavy blocks from one shop to another. The scheduling of transporters is very important, because delays in moving blocks among processing plants can, in turn, cause delays in the overall production schedule [2].

Many studies have focused on the transporter scheduling problem for shipyards, because it is possible to decrease the production cost and increase the productivity of a shipyard of a ship through efficient transporter operation.

Joo, Lee, Koo, and Lee [3] first studied the scheduling problem of single-type transporters for block transportation in shipyards. They proposed a heuristic algorithm to minimize the total weighted logistic times of blocks delivered by transporters. Park and Seo [4,5] also studied the scheduling of single-type transporters for block transportation in shipyards. A greedy randomized adaptive search procedure (GRASP) algorithm was proposed to maximize the workload balance among transporters under the time constraint that all assembly blocks should be transported within a predetermined time. Roh and Cha [6] expanded the block transportation scheduling problem of Joo et al. [3] to include multi-type transporters having different deadweight. Their objective was to minimize the travel distance without loading of, and interference between, the transporters, while satisfying the constraints on the allowable transportation weights of the transporters. Kim and Joo [7] considered a similar block transportation scheduling problem to minimize the total weighted logistics time, including delay times, tardy times, and empty transporter travel times, for heterogeneous transporters having different deadweight. Joo and Kim [2] expanded the block transportation scheduling problem of Roh and Cha [6] and Kim and Joo [7] by generalizing the block delivery restriction on the multi-type transporter. Each block can, or cannot, be delivered by a transporter, according to the various characteristics of the block. The available set of transporters for each block is assumed to be predetermined. They derive a mixed integer programming (MIP) model and proposed effective meta-heuristic approaches for the block transportation scheduling problem. Most of the present studies focus on the scheduling of multi-transporters to shorten working time [8–15]. What is proposed in this paper focuses on reducing energy consumption by obtaining the optimal path with the least number of turns in the process of block transportation, under the premise of maximum safety.

The above research results have a positive effect on decision-supporting systems available for block transportation scheduling. In order to better manage block transportation in real time, a real-time positioning system based on GPS/Beidou and RFID (Radio Frequency Identification) is developed, which overcomes the problem of inaccurate positioning of transporters in complex metal environments in the shipyard [16–18]. Through GNSS (Global Navigation Satellite System)(GPS/Beidou), a rough global location of the transporter can be obtained. There are various RFID positioning methods, whose positioning accuracy mainly depends on the selected RFID tag type, the density of tag distribution, and the environment of the pending location area. This system adopts the GPS/RFID combined positioning method based on the RFID signal strength for ranging, and is then combined with GPS observation value.

In this study, real-time path planning models and algorithms for ship block transportation are proposed, based on block transportation decision-supporting systems and real-time transporter positioning systems, assuming that the place of departure and destination are known.

The remainder of this paper is organized as follows. Section 2 gives the description of the problem and modeling analysis, and how to solve it. Section 3 describes the numerical experiment and analysis. Finally, a summary and remarks on further research are provided in Section 4.

2. Description of the Problem and Modeling Analysis

2.1. Description of the Problem

Due to the particularity of ship block transportation, when using transporters to load ship blocks, which are always huge with different shapes and very heavy, as shown in Figure 1, the turning process

consumes a lot of energy, and there are problems such as insecurity and deformation of ship blocks. Therefore, when planning the optimal real-time path of the transporter, not only must the shortest ship block transportation path be considered, but also the number of turns of the transporter should be minimized when loading the blocks. This paper mainly studies how to combine the driving distance and number of turns of transport to establish an optimized model of the real-time optimal logistics path of transport. All studies as follows are based on the fact that the starting and target position are known first.

Figure 1. A heavy and huge ship block is being transported (tireworld.com.cn).

According to the Geographic Information System (GIS) map constructed by the shipyard, the connection relationship between the main road of the shipyard and each workshop and stockyard is simplified into nodes, as shown in Figure 2.

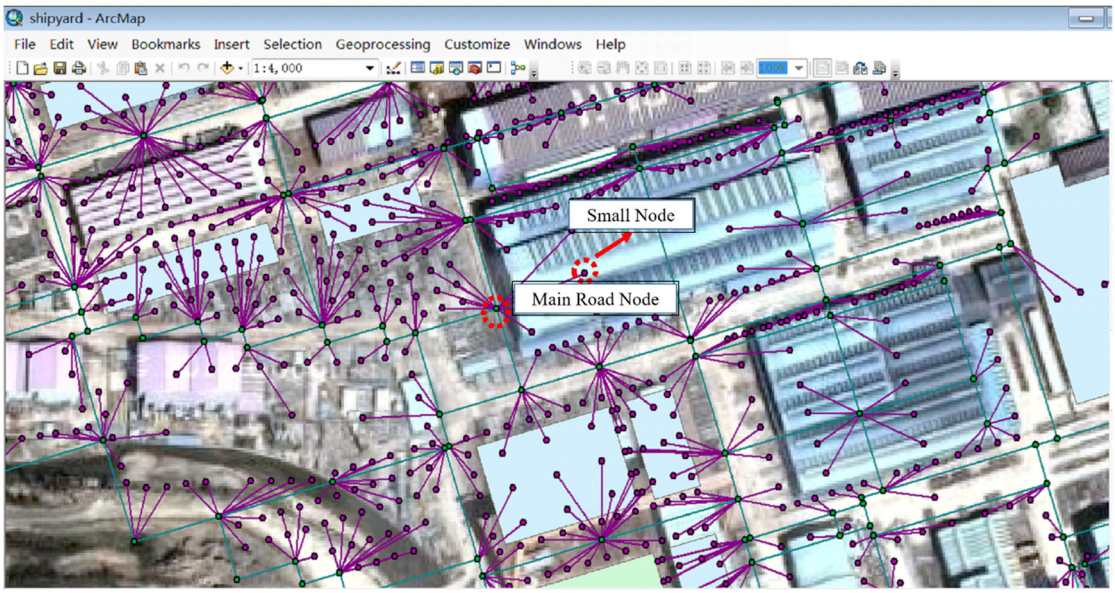


Figure 2. Schematic diagram of main road nodes of shipyard based on Geographic Information System (GIS).

2.2. Modeling Analysis

2.2.1. A Hybrid Graph Model of Node Distance and Number of Turns

From the perspective of energy, the transport operation cost of a transport is considered. Setting the energy consumption of a transport for a unit distance when going straight as E_S, and the energy consumption of a turn as E_T, then the objective function of the optimization model is Equation (1):

$$\min\ (E_S \cdot D + E_T \cdot T) \tag{1}$$

where D represents the distance traveled by the transport and T represents the number of turns of the transport.

The values of E_S and E_T can generally be determined according to the actual situation of the shipyard. However, considering that the specific measurement of the two may be difficult, the proportional relationship between the two is given: $\lambda = E_T/E_S$. Therefore, the objective function of the optimization model can be rewritten as Equation (2):

$$\min(D + \lambda T) \tag{2}$$

where $\lambda \geq 0$ represents how many more times energy it takes to make a turn as it takes to go straight a unit distance.

To solve this problem, it is necessary to find a method that can simultaneously present the distance relationship between nodes and the turning relationship between roads in the same graph to form a classic model in graph theory, and then use the classic shortest path algorithm to solve it.

First, the modeling process of graph theory, with simple examples, is analyzed as follows. As shown in Figure 3, suppose one wants to find an optimal path from a to e. If the shortest distance is considered only, obviously a→b→d→e is optimal, and the path length is $2 + \sqrt{2}$. However, the number of turns on this path is 2. If the path a→b→c→e is chosen, the path length is 4, and the number of turns is 1. And if $\lambda = 1$, the route a→b→c→e is optimal.

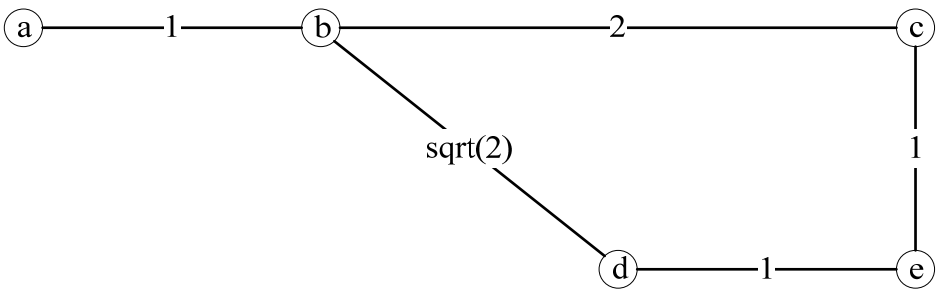
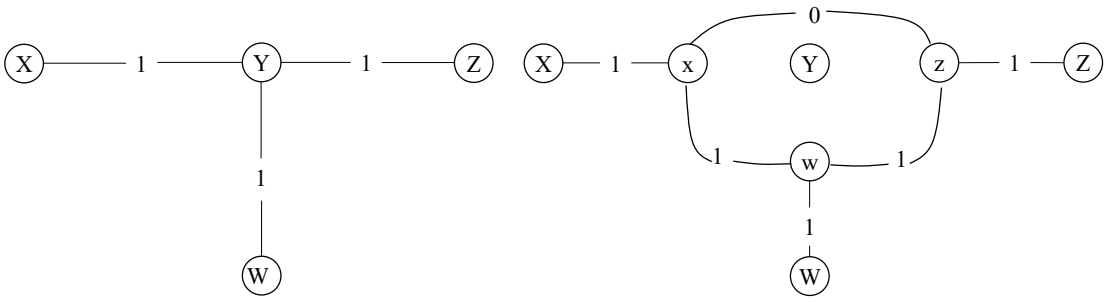


Figure 3. Schematic diagram of turning relationship transformation method.

The graph shown in Figure 3 not only reflects the distance between nodes in terms of weights, but also describes the actual location of nodes in terms of location distribution. Combining Figure 3 as an example, the analysis describing how to combine the distance relationship and turning relationship in Figure 3 on a new graph (Figure 4) is as follows:

The distance between nodes is the relationship between nodes, and whether it turns or not indicates the relationship between edges connected to nodes. In order to convert the relationship between edges to the relationship between nodes, this section introduces two virtual nodes on each edge to represent the turning relationship between edges.



(**a**) A simple three-way intersection (**b**) A virtual node is added to all three edges

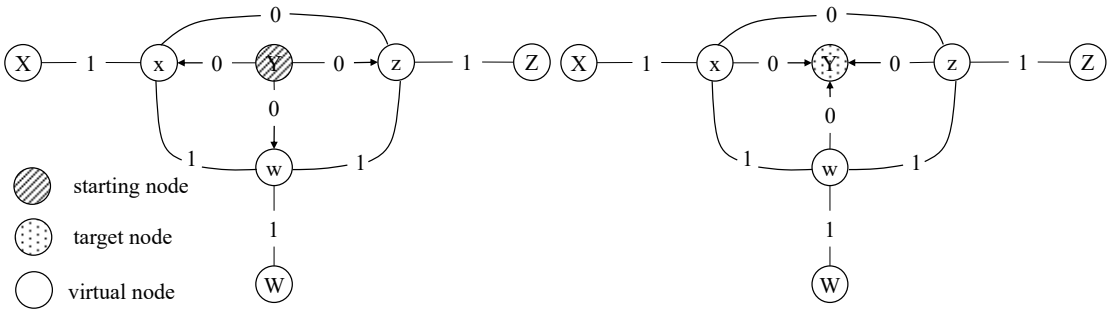
Figure 4. Schematic diagram of turning relationship transformation method.

Figure 4a shows a simple three-way intersection. Among them, the four nodes of XYZW constitute a three-way intersection, and the distance between the nodes is all 1.

In order to be able to describe the turning relationship between the edges connected to the node Y and preserve the distance relationship between the nodes at the same time, as shown in Figure 4b, a virtual node is added to all three edges, denoted as x, w, and z. Then the weights between the virtual nodes are used to record the turning relationship between the edges.

For example, there is no turn from XY to YZ, so the weight between xz is recorded as 0; while from XY to YW, the weight is recorded as 1. In this case, if you want to reach other paths through the Y node, you can only pass the virtual node corresponding to Y. Obviously, this completes the accumulation of the number of turns. For example, the path X→Y→Z in the original image becomes X→x→z→Z, and the sum of weights is 2+0; while in the original image, X→Y→W becomes X→x→w→W, and the sum of weights is 2+1. Therefore, such a transformation can add turning information.

At the same time, considering that node Y can still be used as the starting point and end point, the method shown in Figure 5 is adopted:



(**a**) The starting node (**b**) The target node

Figure 5. Starting node and target node.

① In order for node Y to be the starting point, introduce three unidirectional edges connecting from Y to the virtual node, and set the weight to zero, as shown in Figure 5a.

② In order to make node Y as the end point, it is necessary to introduce three unidirectional edges connecting from the virtual node to node Y, as shown in Figure 5b, and the weight is zero.

③ In order to make node Y both the starting point and the end point, the two kinds of nodes in Figure 5 are introduced into the new graph, that is, a real node Y is split into two virtual nodes, as shown in Figure 5. The one shown in 5a is called the starting node, and the one shown in Figure 5b is called the target node.

At the same time, whether it is the starting node or the target node, when the path needs to pass through the actual node Y, it can only be completed through the virtual path corresponding to Y. This shows that the introduction of these two kinds of nodes does not affect the characteristics of the newly created graph model that can count turns.

Combining the above two strategies, the original map shown in Figure 3 can be converted into the map structure shown in Figure 6. The line segments in the figure all indicate two-way edges, and the arrows indicate one-way edges. The weights of edges without label weights are all 0.

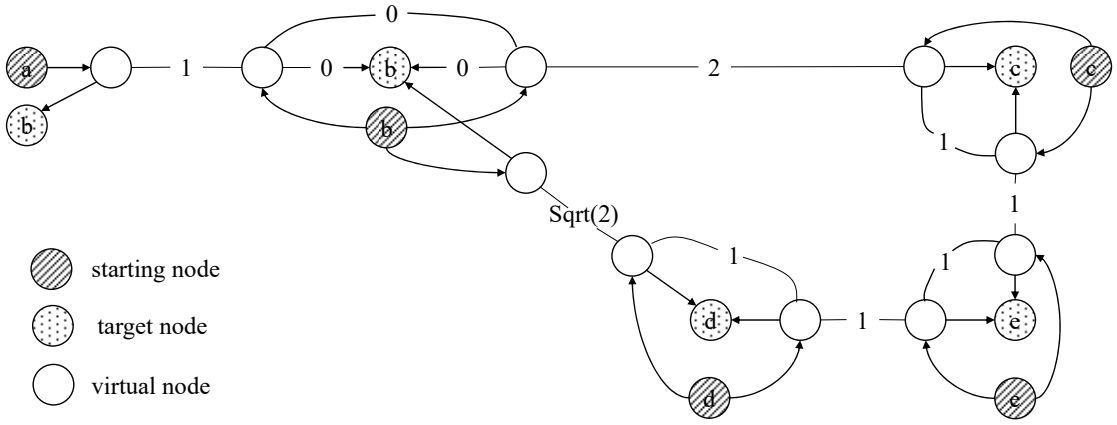


Figure 6. New graph with distance and turn information.

In Figure 6, taking point b as an example, the actual node b is split into a departure node b and a target node b. The starting node has only the edge connected to the virtual node, and the ending node has only the edge entering from the virtual node. At the same time, there are three white virtual nodes on the three edges connected to b. This method is used to record the turning relationship between the paths.

By performing the conversion according to the above rules, there are two virtual nodes on each edge, corresponding to the two endpoints of an edge. The weight relationship between virtual nodes is the turning relationship between edges. Each actual node is divided into a starting node and a destination node. Finding a path from a to e in the actual problem can be transformed into the shortest path from the starting node of a to the target node of e.

In Figure 4, the path between any two actual nodes can be transformed into a path from the starting node to the target node and passing through the virtual node. In addition, the path on the new graph has accumulated distance and weight.

For example, to find the optimal path from a to e, at this time, the weight on a→b→d→e is $4 + \sqrt{2}$, and the weight on a→b→c→e is 5, so considering the weight of the number of turns $\lambda = 1$, one should choose a→b→c→e as the optimal path. When the weight is not 1, just multiply all the weights on the edges representing the number of turns with λ to calculate.

2.2.2. Energy Optimal Scheduling Model Based on Hybrid Graph

In the previous section, a simple problem was used as an example to describe the conversion method of simultaneously representing the distance and the number of turns on a same graph. However, the above graphic model can be solved by algorithm only if it is transformed into a mathematical model. In this section, mathematical language is used to describe a general model symbolically, and to discuss how to choose the appropriate algorithm for optimization.

Given a road network $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents a collection of nodes, and $E = \{e_1, e_2, , e_m\}$ represents a collection of roads:

The weight of the edge is the distance between the nodes, which is represented by the distance matrix $D = \left[d_{ij}\right]_{n \times n}$. If $d_{ij} = \infty$, it means that there is no edge connection between v_i and v_j.

In addition, different from the general graph, the turning relationship between the edges needs to be considered, denoted as $T = \left[t_{ij}\right]_{m\times m}$; if $t_{ij} = 1$, it means that a turn is needed from e_i to e_j; if $t_{ij} = 0$, it means that no turn is needed from e_i to e_j; in addition, if $t_{ij} = \infty$, then it means that e_i and e_j do not have a common vertex.

In order to simultaneously show the distance between nodes and the turning relationship between roads, a new graph, $H = (V', E')$, is constructed.

Where $V' = V_S \cup V_T \cup V_E$, V_S represents the set of starting nodes corresponding to the real node V; V_S represents the set of target nodes corresponding to the real node V; and V_E represents the set of virtual nodes on the edge. Obviously, the nodes in V_S, V_T correspond to the nodes in V one-to-one, and the number is n. There will be two virtual nodes on an edge, so there are $2m$ elements in V_E. Therefore, there are $2m + 2n$ nodes in the graph H.

For the set of edges $E' = E_S \cup E_T \cup E_V$; where E_S represents the set of edges connected to the departure node in V_S; E_T represents the set of edges connected to the departure node in V_s; and E_V represents the set of edges between virtual nodes V_E.

In order to calculate the optimal path on the graph H, the weight of the graph is represented by a matrix W, which is a $(2m + 2n) \times (2m + 2n)$ matrix. In order to represent the matrix, each node in V' needs to be numbered according to certain rules.

The nodes in V_s in this section are numbered from 1 to n in the order of V, while the nodes in V_T are numbered from $n + 1$ to $2n$ rows, that is:

$$\begin{aligned} V_S(v_i) &= i \\ V_T(v_i) &= n + i \end{aligned} \tag{3}$$

The nodes in V_E in this section are numbered in the order of the boundary in E, let $e_k = \left(v_i, v_j\right)$, where $i < j$, then:

$$\begin{aligned} V_E(e_k, v_i) &= 2n + k \\ V_E\left(e_k, v_j\right) &= 2n + m + k \end{aligned} \tag{4}$$

where $V_E(e_k, v_i)$ represents the virtual node of v_i; on edge e_k.$V_E\left(e_k, v_j\right)$ represents the virtual node of v_j on edge e_k.

If graph G numbers the nodes and edges according to the situation in Figure 7, then the number of nodes in graph H is shown in Figure 8. In Figure 8, the number of the starting node is from 1 to 5, and the number of the target node is from 6 to 10. The number of the virtual nodes connected to the node with the smaller number on the edge is 11–15, and the number of the virtual nodes connected to the node with the larger number is 16–20.

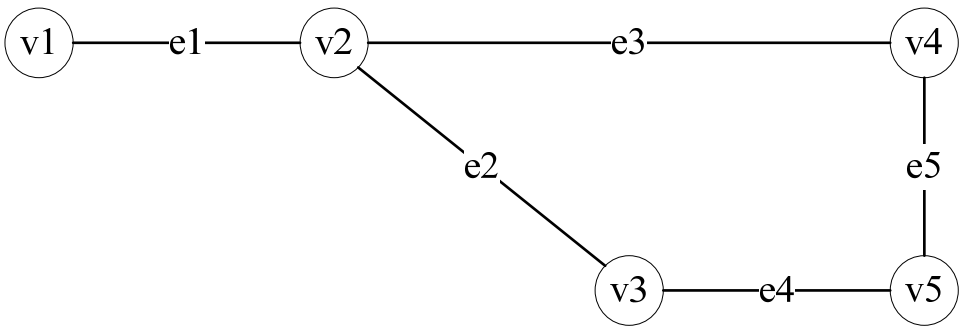


Figure 7. Number of graph G.

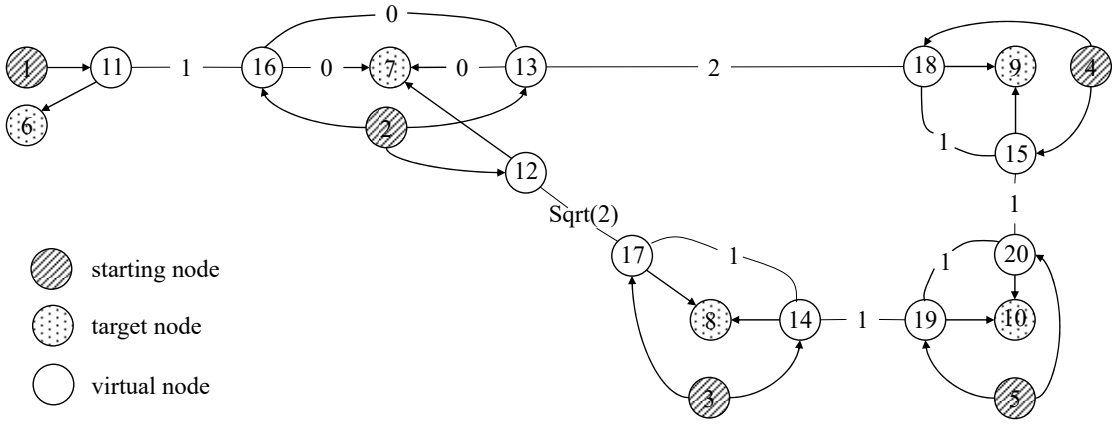


Figure 8. Number of graph H.

After numbering the nodes in the graph H, we can follow the rules described in Section 2.2.1 to give the weight matrix W of the graph H satisfying Equation (5):

$$W(i,j) = \begin{cases} 0 & i \in V_S, j \in V_E \\ 0 & i \in V_E, j \in V_T \\ D(x,y) & i = V_E(e_k, v_x), j = V_E\left(e_k, v_y\right) \\ \lambda T(x,y) & i = V_E(e_x, v_k), j = V_E\left(e_y, v_k\right) \\ \infty & otherwise \end{cases} \tag{5}$$

where,

① The first case represents $w(V_S \to V_E) = 0$, that is, the weight from the starting node to the virtual node is 0;

② The second case represents $w(V_E \to V_T) = 0$, that is, the weight from the virtual node to the target node is 0;

③ The third case means that the weight between two virtual nodes on the same edge is the distance $D(x,y)$ between the corresponding real nodes v_x, v_y;

④ The fourth case indicates that the weight of virtual nodes $V_E(e_x, v_k), V_E\left(e_y, v_k\right)$ of the same node v_k on different edges is the number of turns $T(x,y)$ between edges e_x, e_y multiplied by the weight λ;

⑤ In all cases except these four cases, there is no edge connection between nodes, so the weight is ∞.

From the analysis in Section 2.2.1, we can see that to find the optimal path of $v_i \to v_j$ in the graph G, just find the optimal path of $V_S(v_i) \to V_T\left(v_j\right)$ in the graph H. The optimal path problem on the graph H is a classic shortest path problem, which can be solved by the algorithm for solving the shortest path problem.

Five common shortest path algorithms are shown in Table 1. Among them, the BFS and acyclic algorithms have the lowest complexity, but BFS is an algorithm on an unweighted graph, which is obviously not suitable for solving the problem in this section. Acyclic requires the graph to be a directed acyclic graph, that is, starting from a node, it cannot return to the original node through any path. Obviously, because there are two-way paths on the graph in this section, it is not a directed acyclic graph. The shortest path algorithms that meet the conditions are the Bellman–Ford algorithm and the Dijkstra algorithm. Among them, the complexity of the Dijkstra algorithm is low, and the weights on the map of the shipyard transportation problem are all greater than, or equal to, zero, so the Dijkstra algorithm can be used.

Table 1. Shortest path algorithm.

Algorithm	Feature	Complexity
Bellman–Ford	None	$O(N \cdot E)$
BFS	Unweighted graph	$O(N + E)$
Acyclic	Directed acyclic graph	$O(N + E)$
Dijkstra	Weights greater than or equal to zero	$O(\log(N) \cdot E)$
Floyd	The shortest distance between all nodes, the specific path cannot be obtained at the same time	$O\left(N^3\right)$

In addition, the reasons for not using the Floyd algorithm that can obtain the shortest distance between all nodes are: (1) in the case of missing paths, the graph has a certain degree of dynamics. If the structure of the graph changes, it needs to be recalculated; (2) the Floyd algorithm cannot obtain the specific shortest path while obtaining the shortest distance.

In summary, this section describes how to generate a graph containing distance and number of turns from a road network. At the same time, the Dijkstra algorithm is chosen to solve the optimal path between two nodes to obtain the global optimal solution of the problem.

2.2.3. Algorithm Solution

The Dijkstra algorithm is a classic algorithm to solve the shortest path problem in graph theory. It is based on an abstract network model, in which the road is abstracted as an edge. The weight of the edge is used to represent the parameters of the road. The algorithm determines the path with the least weight from a certain point to all other nodes in the weighted network. The meaning of weight is broad, which can express distance, quantity, cost, etc. Usually, the smallest weight between two points is called the distance between two points, and the corresponding problem is summarized as the shortest path problem expression.

First of all, the Dijkstra algorithm can be described with the following pseudocode:

```
Dijkstra(G, w, s)
    for v ∈ V[G] do
            d[v] = ∞
            π[v] = NULL
        end
        d[s] = 0
        Q = V[G]
        while Q ≠ ∅ do
            u = MIN(Q)
            for v ∈ Adj[u] do
                if d[v] > d[u] + w(u, v) then
                    d[v] = d[u] + w(u, v)
                    π[v] = u
                end
            end
        end
    end
```

where G represents the graph object, w is the weight matrix, and s is the starting node. During initialization, record the shortest distance $d[v] = \infty$ from the marked node v to the source node s, and the predecessor node of each node which means that the previous node in the shortest path is marked as $\pi[v] = NULL$. Then, mark the shortest distance $d[s] = 0$ of the source node itself, and then establish a minimum queue Q according to d.

When the algorithm starts to iterate, the vertex u of the smallest d value in the queue Q is extracted first each step, and all reachable nodes $v \in Adj[u]$ of u is updated. If the distance from node u to node v plus $d[u]$ is less than the original minimum distance $d[v]$ of node v, $d[v] = d[u] + w(u, v)$ is updated, and make u the precursor node of v.

At the end of the algorithm, d records the shortest distance from all nodes to the source node s, and π records the information of the precursor node. If you want to query the shortest path from node s to node t, you only need to start from t and find the previous node according to $\pi[t]$, and so on until the node s is found. Then the shortest path can be outputted reversely.

2.2.4. Optimal Scheduling Model Considering Path Missing

In the actual dispatch process of the shipyard, a certain route in the road network may be temporarily unavailable. Therefore, in this section, an optimal scheduling model considering path missing is established.

In the road network $G = (V, E)$, suppose the transporter is located at the node S_0, and the transportation task is to move the ship block from the node S_1 to the node S_2. When the transporter starts the transportation task, suppose there are some roads temporarily missing in the road network, and the set of these missing roads is recorded as E_B.

The transporter needs to first arrive at node S_1 from S_0, and then transport ship block from node S_1 to S_2. Considering that the energy consumption ratio of turning and going straight is different when the transport is empty and full, it is necessary to solve the $S_0 \rightarrow S_1$ and $S_1 \rightarrow S_2$ in sections. Suppose the weight of $S_0 \rightarrow S_1$ is λ_1 and the weight of $S_1 \rightarrow S_2$ is λ_2.

For $S_0 \rightarrow S_1$, take the weight as λ_1. After constructing the graph H_1 according to the algorithm in Section 2.2.2, the weight matrix W_1 is processed as follows, and the missing path is marked as:

$$W_2(i,j) = W_2(j,i) = \infty \ \ where \ \ e_k = \left(v_x, v_y\right) \in E_B, i = V_E(e_k, v_i), j = V_E\left(e_k, v_j\right)$$

which means the weight between the two virtual nodes on this edge is marked as positive infinity. Then, the shortest path on W_1 from the starting node $V_S(S_0)$ of S_0 to the target node $V_T(S_1)$ of S_1 can be calculated using the Dijkstra algorithm.

Similarly, for $S_1 \rightarrow S_2$, after constructing the graph H_2 according to the algorithm in Section 2.2.2, the weight matrix W_2 is processed as follows, and the missing path is marked as:

$$W_2(i,j) = W_2(j,i) = \infty \ \ where \ \ e_k = \left(v_x, v_y\right) \in E_B, i = V_E(e_k, v_i), j = V_E\left(e_k, v_j\right)$$

which means the weight between the two virtual nodes on this edge is marked as positive infinity. Then the shortest path on W_2 from the starting node $V_S(S_1)$ of S_1 to the target node $V_T(S_2)$ of S_2 can be calculated using the Dijkstra algorithm.

Figure 9 shows a schematic diagram of optimal scheduling. The red road in the picture is the temporarily missing road. The yellow path is the optimal path from the current position of the transporter to the starting point of the mission. At this time, the transporter is empty, and the turning energy is considered negligible, so $\lambda_1 = 0$; the blue path in the picture is the optimal path from the start of the mission to the end of the mission. At this time, the transporter is loaded, and the turning energy consumption is large. Taking $\lambda_2 = 1$ can get the optimal path in the figure.

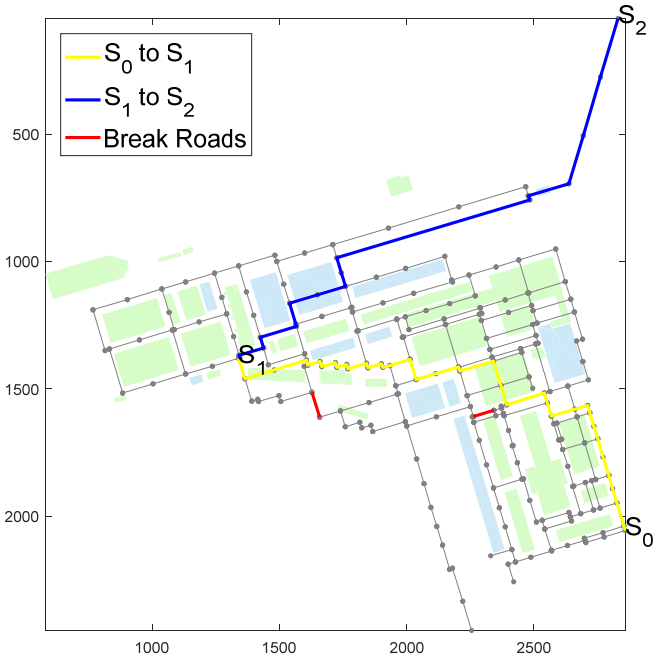


Figure 9. Schematic diagram of scheduling instance.

3. Numerical Experiment and Analysis

3.1. Influence of Parameters λ

From the description in the previous section, it can be seen that the results of the ship block optimal logistics path model with the least energy consumption of the transport are directly affected by the parameters λ. Therefore, the influence of parameters λ on the model results is analyzed in this section.

Figure 10 shows the different effects on the optimal path when the starting node and the target node are the same. The blue path in the figure is gotten when $\lambda = 0$, which means that only minimizing the travel distance is considered, without considering the number of turns. It can be found that when $\lambda = 0$, the length of the path is the shortest, and there are more turns (24 times). The orange path in the figure is the result of $\lambda = 1000$. At this time, only the number of turns is minimized, and the travel distance is not considered. It can be found that when $\lambda = 1000$, the number of turns of the transport is the least (only 11 times), and the transporter basically drives along the straight path along the edge of the road network. The orange-red path is gotten when $\lambda = 1$, which reflects the situation that both distance and the number of turns are considered. At this time, the distance is slightly larger than the result when a = 0, and the number of turns is closer to $\lambda = 1000$. Compared to the case of $\lambda = 0$, the travel distance is only increased by 6 m, and the number of turns is reduced by 12, which is a wonderful result.

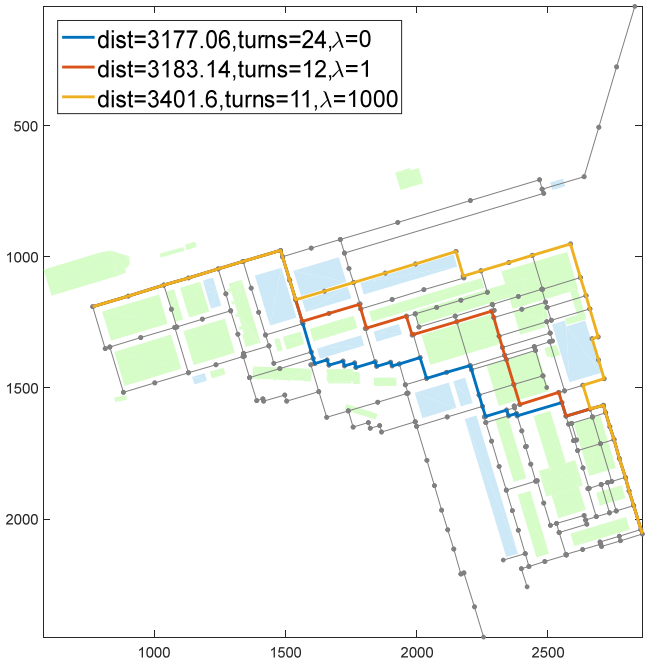


Figure 10. Influence of λ on the optimal path.

3.2. Run Time Analysis

Assuming that there are N nodes in the road network and E roads in total, the transformed graph has $2N + 2E$ nodes and $O(E^2)$ edges. According to the complexity of the Dijkstra algorithm, the complexity of the algorithm in this paper is $O(\log(N + E)E^2)$.

We ran the algorithm on Windows 7 operation system in an Intel Core i7-7500U laptop computer with 8 GB RAM. On the same road network as Figure 10, 1000 random trials were carried out, and the distribution of running time was obtained as shown in Figure 11. It was found that the running time is between 1 to 6 millis, and the speed is very fast.

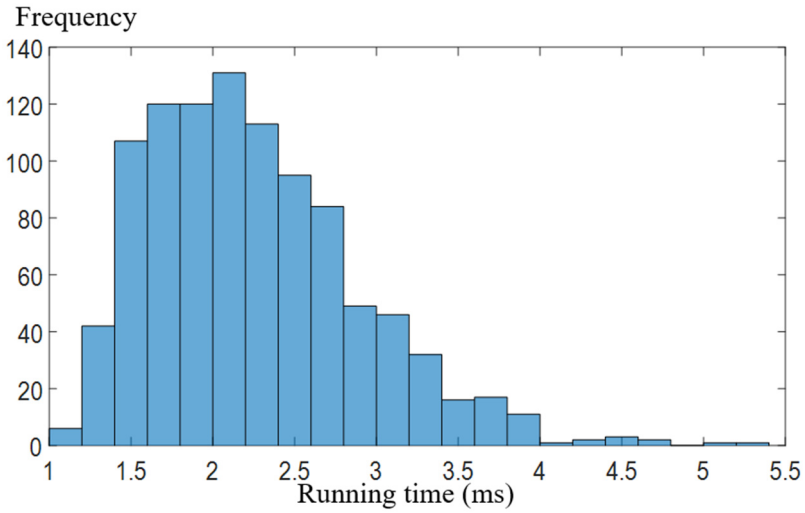


Figure 11. 1000 times trials when $N = 238, E = 296$.

In addition, the complexity of the algorithm proposed in this section is $O(\log(N + E)E^2)$, so the algorithm can be applied to large-scale problems. The results of experiments on problems of different scales are shown in Table 2. The initialization time in the table refers to the process of transforming

the actual road network into the new graph described in Section 2.2.2. This process only needs to be executed once during the entire program execution. It can be seen that, even if the problem size reaches the order of 100,000, the initialization can still be completed in about 1.3 s, and each solution only takes 0.5 s. It fully meets the requirements of real-time optimal logistics path planning for ship block logistics, and has strong real-time performance.

Table 2. Running time under different problem scales.

Number of Nodes	Number of Roads	Initialization (ms)	The Optimal Path (ms)
100	108	3.0	0.7
1000	1099	19.4	2.8
10,000	11,000	134.4	36.0
100,000	109,998	1237.1	493.8

4. Conclusions

In this paper, we have considered how to combine the travel distance and number of turns of transport to establish an optimized model of the real-time optimal path for a transporter. Several common optimal path algorithms were analyzed. The Dijkstra algorithm was selected to solve the proposed model. In addition, sometimes a certain route in the road network may be temporarily unavailable in the shipyard. So, an optimal scheduling model considering path missing was established. Finally, the numerical simulation experiments and analysis with the algorithm chosen in this paper by using GIS map of a shipyard were presented. It can be seen that, even if the problem size reaches the order of 100,000, the initialization can still be completed in about 1.3 s on standard hardware and software conditions, and each solution only takes 0.5 s. The results show that the proposed model and algorithm are efficient to solve the optimal real-time path planning with comprehensive consideration of the travel distance and number of turns of transporter. In this paper, the computational real-time performance is high, and what we have done can fully meet the requirements of an online real-time planning path. In further research, we will test the proposed model and algorithm based on the real-time transporters positioning system in a real shipyard.

Author Contributions: C.W.: software, writing—original draft preparation, methodology. K.W.: writing and editing, supervision. J.T.: investigation. Y.Z.: conceptualization. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of Hubei Province of China under grant 2020CFB196.

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

References

1. Kim, B.; Jeong, Y.; Shin, J.G. Spatial arrangement using deep reinforcement learning to minimise rearrangement in ship block stockyards. *Int. J. Prod. Res.* **2020**, *58*, 5062–5076. [CrossRef]
2. Joo, C.M.; Kim, B.S. Block transportation scheduling under delivery restriction in shipyard using meta-heuristic algorithms. *Expert Syst. Appl.* **2014**, *41*, 2851–2858. [CrossRef]
3. Joo, C.M.; Lee, W.; Koo, P.H.; Lee, K.B. Transporter scheduling for block transportation in shipbuilding. *J. Korea Manag. Eng. Soc.* **2006**, *11*, 169–179.
4. Park, C.; Seo, J. A GRASP approach to transporter scheduling for ship assembly block operations management. *Eur. J. Ind. Eng.* **2013**, *7*, 312. [CrossRef]
5. Park, C.; Seo, J. A GRASP approach to transporter scheduling and routing at a shipyard. *Comput. Ind. Eng.* **2012**, *63*, 390–399. [CrossRef]
6. Roh, M.-I.; Cha, J.-H. A block transportation scheduling system considering a minimisation of travel distance without loading of and interference between multiple transporters. *Int. J. Prod. Res.* **2011**, *49*, 3231–3250. [CrossRef]

7. Joo, C.M.; Kim, B.S. Parallel machine scheduling problem with ready times, due times and sequence-dependent setup times using meta-heuristic algorithms. *Eng. Optim.* **2012**, *44*, 1021–1034. [CrossRef]
8. Yim, S.-B.; Roh, M.-I.; Cha, J.-H.; Lee, K.-Y. Optimal block transportation scheduling considering the minimization of the travel distance without overload of a transporter. *J. Soc. Nav. Arch. Korea* **2008**, *45*, 646–655. [CrossRef]
9. Wang, C.; Mao, Y.-S.; Hu, B.-Q.; Deng, Z.-J.; Shin, J.G. Ship block transportation scheduling problem based on greedy algorithm. *J. Eng. Sci. Technol. Rev.* **2016**, *9*, 93–98. [CrossRef]
10. Yu, H.K. A Study on the Transportation Routing Optimization of Shipbuilding Blocks. Master's Thesis, University of Ulsan, Ulsan, Korea, 2005.
11. Tsai, C.-Y.; Liou, J.J.H.; Huang, T.-M. Using a multiple-GA method to solve the batch picking problem: Considering travel distance and order due time. *Int. J. Prod. Res.* **2008**, *46*, 6533–6555. [CrossRef]
12. Chen, Y.; Jiang, Z.; Li, B. A Hybrid Heuristic optimization approach for green flatcar transportation scheduling in shipbuilding. In *Proceedings of the Advances in Transdisciplinary Engineering*; IOS Press: Amsterdam, The Netherlands, 2020; Volume 12, p. 301.
13. Kim, B.S.; Joo, C.M. Ant colony optimisation with random selection for block transportation scheduling with heterogeneous transporters in a shipyard. *Int. J. Prod. Res.* **2012**, *50*, 7229–7241. [CrossRef]
14. Wang, C.; Mao, Y. Ship block transportation scheduling approach based on genetic algorithm. *J. Shanghai Jiaotong Univ.* **2017**, *51*, 338–343.
15. Guo, Z.; Zhang, D.; Liu, H.; He, Z.; Shi, L. Green transportation scheduling with pickup time and transport mode selections using a novel multi-objective memetic optimization approach. *Transp. Res. Part D Transp. Environ.* **2018**, *60*, 137–152. [CrossRef]
16. Heng, Z.; Zuquan, X.; Chong, W. Real-time integrated monitoring system of ship block outfield logistics. *Shipbuild. Technol.* **2019**, *92*, 83–88.
17. Lee, K.J.; Lee, Y.H.; Lee, K.C.; Son, Y.-D. GPS/INS for logistics of ship-block. In Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference, Toronto, ON, Canada, 17–20 September 2006; pp. 777–782.
18. Lee, Y.H.; Lee, K.C.; Lee, K.J.; Son, Y.D. *Study on the Positioning System for Logistics of Ship-Block*; Special Issue of the Society of Naval Architects of Korea: Seoul, Korea, 2008; pp. 68–75.