sustainability

Article

Tugboat Scheduling Method Based on the NRPER-DDPG Algorithm: An Integrated DDPG Algorithm with Prioritized Experience Replay and Noise Reduction

Jiachen Li [1], Xingfeng Duan [1,2,*], Zhennan Xiong [1] and Peng Yao [1]

[1] College of Navigation, Jimei Universiy, Xiamen 361021, China; 202111823003@jmu.edu.cn (J.L.); znxiong@jmu.edu.cn (Z.X.); 202111823005@jmu.edu.cn (P.Y.)
[2] Maritime and Maritime Law Institute, Jimei University, Xiamen 361021, China
* Correspondence: xfduan@jmu.edu.cn

Abstract: The scheduling of harbor tugboats is a crucial task in port operations, aiming to optimize resource allocation and reduce operational costs, including fuel consumption of tugboats and the time cost of vessels waiting for operations. Due to the complexity of the port environment, traditional scheduling methods, often based on experience and practice, lack scientific and systematic decision support, making it difficult to cope with real-time changes in vessel dynamics and environmental factors. This often leads to scheduling delays and resource waste. To address this issue, this study proposes a mathematical model based on fuzzy programming, accounting for the uncertainty of the arrival time of target vessels. Additionally, we introduce the NRPER-DDPG algorithm (DDPG Algorithm with Prioritized Experience Replay and Noise Reduction), which combines a prioritized replay mechanism with a decaying noise strategy based on the DDPG algorithm. This approach optimizes the time for tugboats to reach the task location as a continuous action space, aiming to minimize the total system cost and improve scheduling efficiency. To verify the effectiveness of the mathematical model and algorithm, this study conducted experimental validation. Firstly, the optimal algorithm hyperparameter combinations were adjusted through random examples to ensure the stability and reliability of the algorithm. Subsequently, large-scale examples and actual port cases were used to further verify the performance advantages of the algorithm in practical applications. Experimental results demonstrate that the proposed mathematical model and algorithm significantly reduce system costs and improve scheduling efficiency, providing new insights and methods for the sustainable development of port operations.

Keywords: port tugboat scheduling; port costs; navigation efficiency; fuzzy programming mathematical model; deep reinforcement learning

check for updates

Citation: Li, J.; Duan, X.; Xiong, Z.; Yao, P. Tugboat Scheduling Method Based on the NRPER-DDPG Algorithm: An Integrated DDPG Algorithm with Prioritized Experience Replay and Noise Reduction. *Sustainability* **2024**, *16*, 3379. https://doi.org/10.3390/su16083379

Academic Editors: Maryam Hamidi and Wu Xing

Received: 27 February 2024
Revised: 15 April 2024
Accepted: 16 April 2024
Published: 17 April 2024

1. Introduction

With the booming development of global trade, ports, as key nodes connecting the ocean and land, have seen their scale and operational needs continue to rise. Against this backdrop, the demand for tugboats in port operations has also shown a rapid growth trend. Tugboats play a crucial role in assisting large ships in berthing, shifting, and departing operations, and are an indispensable part of port operations. However, at the same time, the costs and environmental issues brought about by tugboat operations have also become increasingly prominent [1].

Firstly, as an important auxiliary tool for port operations, the high operating costs of tugs have always been a focus of port operators. The purchase, maintenance, and fuel consumption of tugs account for a significant portion of port operating costs. Especially in the context of rising oil energy prices, the fuel cost of tugs has become a major burden for port operations [2]. Therefore, how to reduce the operating costs of tugs and improve resource utilization efficiency has become an urgent issue for port operators.

Secondly, the environmental issues caused by the operation of tugs cannot be ignored. As tugs mainly rely on petroleum energy for driving, the emission of large amounts of carbon dioxide and other greenhouse gases has had a profound adverse impact on the global climate environment [3]. In the context of an increasingly environmentally conscious era, reducing carbon emissions during port operations and promoting the construction of green ports have become an important direction for the development of global ports [4,5]. Therefore, optimizing the dispatching strategy of tugs and reducing their energy consumption and carbon emissions during operation are of crucial importance for promoting the sustainable development of ports.

In response to the above issues, this article conducts an in-depth study of the complex tugboat scheduling problem involving multiple tugboats and multiple tasks to be performed. In this problem, each ship waiting for work has specific requirements for the power and number of tugboats, which increases the complexity of the scheduling. To better reflect the real port operation scenario, this study also comprehensively considers a series of practical constraints and specifically introduces the uncertainty factor of the arrival time of the tugboats at the target location. Based on these considerations, this article constructs a mathematical model with minimizing time cost and fuel cost as the optimization objective, and uses deep reinforcement learning to solve it. In order to verify the effectiveness of the proposed algorithm and model, this article designs simulation examples of different sizes for comparative experiments, aiming to prove the effectiveness of the proposed method through comparison results. Furthermore, in order to verify the practicality of intelligent scheduling in practical port applications, we selected a real case from Huanghua Port for solving and analysis. Through this series of research and practice, we hope to explore the optimal tugboat scheduling strategy in complex and changing port environments, thus providing strong theoretical support and practical guidance for improving the efficiency and cost savings of port operations.

2. Literature Review

2.1. Tug Dispatching Method

This article studies the problem of tugboat scheduling and designs multi-dimensional decisions for each tugboat, including task allocation and departure time, which makes traditional manual scheduling difficult to adapt to the challenges of large-scale and dynamic changes. In recent years, many scholars have conducted research on this topic.

The optimization method for tugboat scheduling based on operations research and heuristic algorithms is currently the focus of research. Ref. [6] describes tugboat scheduling as a parallel machine scheduling problem with special constraints, and proposes a heuristic algorithm based on hybrid evolutionary strategies to solve it. Ref. [7] proposed an improved discrete particle swarm optimization method, and constructed a mixed integer programming model to effectively solve the tugboat scheduling problem. Ref. [8] deeply explored the port resource allocation problem, especially in terms of tugboat allocation, in which it was assumed that each ship required a single tugboat for assistance and only involved berthing and docking operations. To solve this problem, the author innovatively proposed an evolutionary algorithm aimed at generating local optimal solutions. Ref. [9] developed a new chaotic quantum adaptive satin bow bird optimization algorithm to solve the joint scheduling problem of containers, tugboats, and terminal cranes, and solved it based on simulation examples from two ports in North China, obtaining reasonable scheduling results. Ref. [10] developed an adaptive large-neighborhood search algorithm to solve the tugboat scheduling problem in large-scale instances and obtain good scheduling rules. Ref. [11] constructed a dual-objective mixed integer linear programming model aimed at solving the tugboat scheduling problem. By optimizing both objectives of minimizing maximum completion time and total fuel consumption simultaneously, the model effectively improved the overall efficiency and quality of tugboat services. Ref. [12] used a mixed integer linear programming model to develop tugboat scheduling rules and solved it using a branch and cut algorithm. Experiments showed that it could solve different scale instances within a reasonable time. Ref. [13] introduced a mixed integer program to simulate ship scheduling under channel restrictions and different tugboat allocation strategies, and proved the feasibility of the method through real case analysis. Ref. [14] established a mixed integer programming model, and designed a temporary algorithm to generate towing chains for solving it. Through experimental verification, it can solve large-scale practical problems. Ref. [15] constructed a state-time network from the perspective of tugboats, and used a variable neighborhood search algorithm to obtain reasonable results. Ref. [16] applied Lagrangian relaxation and Benders decomposition combined with iterative methods to optimize tugboat scheduling for ship berth planning, various mobile ships' tugboat points, and their horsepower requirements, aiming to minimize the weighted sum of berthing and departure delays, tugboat operating costs, and the number of unserviced ships.

2.2. Reinforcement Learning Scheduling Method

Heuristic algorithms are based on experience and rules, and seek approximate optimal solutions by continuously adjusting parameters and heuristic functions. They have high efficiency, but due to their experience and rules, they are easily affected by the complexity of the problem, and it is difficult to guarantee the global optimal solution, which is prone to local optimal solutions. Compared to heuristic algorithms, reinforcement learning algorithms have more advantages in solving large-scale and highly complex data in dynamic scenarios [17]. In dynamic environments, feedback data are obtained through interaction with the environment and repeated experiments, and then behavior strategies are continuously optimized to maximize reward feedback [18,19]. The decision-making process is adjusted based on the feedback, thus gradually approaching the optimal solution [20]. Currently, there are many studies applying reinforcement learning to scheduling. Ref. [21] proposes a deep multi-agent reinforcement learning method to address the complex and dynamic job-shop scheduling problem. Experimental results demonstrate its efficient learning and superior performance. Ref. [22] studies the link scheduling problem in the Internet of Things network based on reinforcement learning, overcoming the challenge of exponential growth in state space dimensions, and verifies the feasibility of the algorithm through simulation experiments. Ref. [23] solves the parallel processor scheduling problem with identical acceleration functions by introducing a new state variable into the reinforcement learning algorithm and by using a set of virtual boxes to classify jobs according to remaining processing time, generating reasonable scheduling rules. Ref. [24] proposes an anti-fact attention multi-agent reinforcement learning method that considers a decentralized partially observable Markov decision process, effectively solving the joint scheduling problem of multi-level hybrid assembly lines.

In terms of port scheduling, Ref. [25] constructed a mixed integer linear programming mathematical model, and used an innovative reinforcement learning-based adaptive genetic simulated annealing algorithm to efficiently solve the ship scheduling optimization problem. Ref. [26] used deep reinforcement learning to optimize the AGV scheduling of container terminals, aiming to improve port efficiency and achieve a balance between energy consumption and transportation time. Ref. [27] proposed a model based on artificial potential field and twin delayed deep deterministic policy gradient APF-TD3 to optimize the transportation path of AGV in automated container terminals, achieving efficient, safe, and stable operation of the port. Ref. [28] proposed a dynamic truck scheduling system based on Internet of Things technology, which combines Real2Sim simulation and deep reinforcement learning, and uses historical and real-time port data to learn high-quality truck scheduling strategies to optimize the operational efficiency of maritime container terminals.

It is evident that the application of reinforcement learning in the field of port dispatching has been widely studied. However, there is limited research on the application of reinforcement learning in tugboat dispatching. Therefore, this article abstracts the tugboat dispatching problem as a Markov decision process, models the driving and operating process of the tugboat in the port as state and action transitions, and uses deep reinforcement learning to continuously optimize the optimal strategy with the goal of minimizing fuel costs and total working time for operation.

3. Problem Description

3.1. Tugboat Scheduling Process

The tugboat scheduling problem is a complex optimization issue that revolves around the rational allocation and scheduling of tugboats in harbor channels. Its main objective is to ensure that ships within the harbor or channel can smoothly complete inbound and outbound port operations or navigation tasks. The goal of this problem is to maximize the operational efficiency of tugboats while minimizing their operating costs, all while guaranteeing the safety and efficient operation of ships. This problem encompasses multiple tugboats and vessels, varying task requirements, and diverse constraints such as the number of tugboats, power, speed, navigation distance, and fuel consumption. Therefore, addressing the tugboat scheduling problem requires comprehensive consideration of multiple factors to formulate a reasonable scheduling plan that achieves optimal economic benefits and operational efficiency.

The scenario considered in this paper is when multiple vessels requiring assistance enter the port and the tugboat resources are initially berthed at the tugboat base. Our aim is to effectively assist these vessels in shifting, berthing, and unberthing operations, ensuring that they complete these tasks within their designated time windows and are guided to appropriate berths. Our ultimate goal is to minimize fuel consumption costs and tugboat operating time while enhancing the efficiency and safety of port operations.

To facilitate a more intuitive understanding, Figure 1 illustrates the process of two tugboats assisting a vessel in entering the harbor. When the vessel arrives at the offshore anchorage, the tugboats move towards it, and the point where they meet is referred to as the task start point. The illustration indicates that this particular vessel requires the assistance of two tugboats for its entry into the harbor. Once the tugboats reach the task start point, they assist the vessel in reaching its berth and then return to the tugboat base. During this entire process, it is imperative to consider the fuel consumption and time costs incurred by the tugboats while reaching the task point, assisting the vessel, and returning to the tugboat base. In this paper, we define a "task" as the determination of the required number of tugboats, their power ratings, start and end times, and the final berthing location for a single vessel awaiting operation.

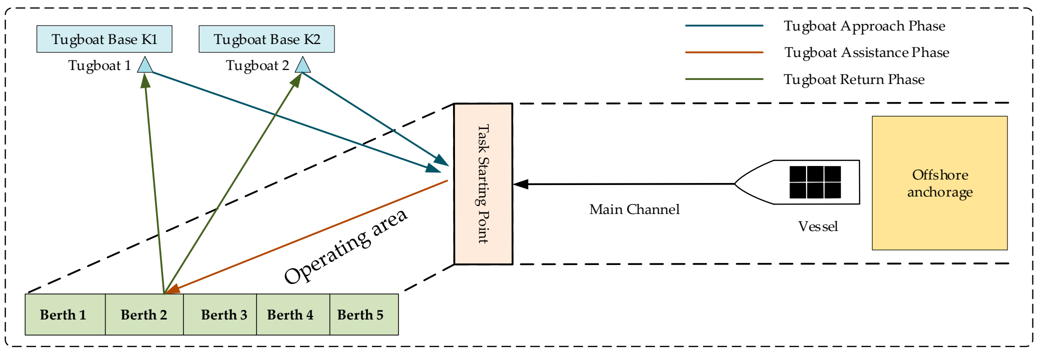


Figure 1. Tugboat scheduling assists in the process of ship entry into the port.

In practical port operations, due to their complexity, the time for tugboats to arrive and return to the tugboat base is often affected by various uncertain factors such as traffic and weather conditions. To more accurately address these uncertainties, this paper chooses to adopt a fuzzy programming mathematical model. It utilizes triangular fuzzy numbers to represent some key parameters, thus making the model more realistic and applicable to actual situations.

3.2. Problem Hypothesis

To facilitate the establishment of a mathematical model, the following assumptions are proposed in this paper:

(1) The start and end times of each task are fixed and not influenced by random factors.
(2) All relevant information about the task, including the required number of tugboats, power ratings, and task duration, is available before scheduling the tugboats.
(3) The location of the tugboat base where the tugboats are docked is known before the start of all tasks.
(4) The time taken by tugboats to dock and depart from the base is negligible compared to their transit time.
(5) Different tugboats have their respective fixed speed (optimal economic speed).
(6) Each tugboat base has an upper limit on the number of tugboats it can accommodate.

3.3. Symbol Definition

The mathematical model for the tugboat scheduling problem is represented by the symbolic parameters, as shown in Table 1, which includes symbolic variables, decision variables, and their respective meanings.

Table 1. Parameters of the Tugboat Scheduling Model.

Parameter Type	**Parameter**	**Description**
Set	i	Tugboat number, $i \in \{1, 2, 3, \cdots, I\}$
	j	Task number, $j \in \{1, 2, 3, \cdots, J\}$
	k	Base number, $k \in \{1, 2, 3, \cdots, K\}$
	$D1_{ij}^{k}$	The distance from tugboat i at base k to the starting point of task j
	$D2_{ij}^{k}$	Distance from tug i at base k to the ending point of task j
	Cn_i	The fuel cost per unit time of tugboat i during navigation assistance
	Ct_i	The fuel cost per unit distance for tugboat i when it is unloaded (or empty)
Input Parameters	P_j	The power required for the tugboats assigned to task j
	O_j	The number of tugboats required for task j
	P_i	The power of tugboat i
	H_j^k	Tugboat i is at base k after completing the previous task
	st_j	Time of arrival of vessel for task j at the starting point
	et_j	Time of arrival of vessel for task j at the ending point
	V_i	The speed of tugboat i when it is unloaded (or empty)
	$[t_s, t_e]$	The time window for the tugboats to arrive at the starting point of the task
	$[t_w, t_k]$	The return time window for tugboats upon task completion.
Output Parameters	Z_1	Total fuel cost incurred by the tugboats
	Z_2	Total waiting time cost incurred by the vessels awaiting operation
Decision variables	A_{ij}	If tugboat i is assigned to task j, the value is 1; otherwise 0
	B_{ij}^{k}	If tugboat i is at base k after task j, the value is 1; otherwise 0

3.4. Mathematical Model

To improve decision-making in port operations, this paper establishes a mathematical model that comprehensively considers the fuel cost incurred by tugboats and the waiting time of vessels awaiting operation. The aim is to minimize fuel costs and operation time within the port by making reasonable tugboat scheduling decisions to complete all tasks.

This paper considers the effective control of tugboat fuel costs, recognizing that the primary expenses during tugboat-assisted berthing and unberthing operations depend on the tugboat's effort and the execution time of the task. Therefore, this paper evaluates the cost of the assistance phase by measuring the costs associated with different tugboat assistance scenarios. The remaining phase, which involves unloaded travel, can be assessed by measuring the cost per unit distance traveled by the tugboat. The cost of the tugboat during the assistance phase is influenced by two key factors: on the one hand, it depends on the tugboat's power, where higher power results in higher costs; on the other hand, it depends on the duration of the work, which is determined by the operation schedule of the target vessel.

Additionally, considering the time cost of tugboat operations, the tugboats should arrive at the task starting location within the required time window and return to the tugboat base within the designated time window after task completion, waiting for the next dispatch. This helps minimize the time cost as much as possible.

In summary, a mathematical model based on fuzzy programming has been established, considering the minimization of tugboat fuel costs (including unloaded travel and assistance phase) and total operation time costs. Equations (1)–(12) represent the constraints and objective function.

$$Z_1 = Ct_i D1_{ij}^k A_{ij} + Cn_i A_{ij}(et_j - st_j) + Ct_i D2_{ij}^k A_{ij} B_{ij}^k \tag{1}$$

$$Z_2 = (D1_{ij}^k + D2_{ij}^k)/V + et_j - st_j, \forall i \in I, \forall k \in K \tag{2}$$

$$\sum_i A_{ij} = O_j, \forall j \in J \tag{3}$$

$$B_{0i}^k = H_i^k, \forall i \in I, k \in K \tag{4}$$

$$P_i \geq O_j A_{ij}, \forall i \in I, \forall j \in J \tag{5}$$

$$\sum_k D1_{ij}^k y_{0,i}^k A_{i1} / V_i \leq st_1, \forall i \in I \tag{6}$$

$$Cr\left\{ t_s \leq \left| st_j - et_{j-1} - \sum_k D1_{i,j-1}^k B_{j-1,i}^k A_{i,j-1} / V_i \right| \leq t_e, \forall i \in I, j \in \{2, 3, \cdots, J\} \right\} \geq \alpha \tag{7}$$

$$Cr\left\{ t_w \leq \sum_k D2_{i,j-1}^k B_{j-1,i}^k A_{i,j-1} / V_i \leq t_k \right\} \geq \alpha \tag{8}$$

$$B_{j-1,i}^k \geq A_{ij} \forall i \in I, \forall j \in J, \exists k \in K \tag{9}$$

$$\sum_k B_{ji}^k \geq A_{ij}, \forall i \in I, \forall j \in J \tag{10}$$

$$\sum_k B_{ji}^k = 1, \forall i \in I, \forall j \in J, k \in K \tag{11}$$

$$B_{j-1,i}^k = B_{ji}^k, \forall i \in I, \forall j \in J, k \in K \tag{12}$$

Equation (1) represents the fuel cost incurred by the tugboat at time t. Equation (2) represents the operational time cost of the tugboats within the port at time t. Equation (3) indicates the number of tugboats assigned to each task. Equation (4) represents the initial position of the tugboats before the start of the first task. Equation (5) ensures that the tugboats assigned to a task must meet the power requirements of that task. Equation (6) states that the tugboats assigned to the first task should arrive at the starting point of the task before it begins. Equation (7) indicates that the feasibility of the tugboats arriving within the designated time window for task commencement should be no less than α. Equation (8) states that the feasibility of the tugboats returning to the tugboat base within the designated time window after task completion should be no less than α. Equation (9) ensures that the tugboats should meet the requirement of returning to a specific tugboat base before the start of the next task. Equation (10) represents that the number of tugboats docked at the base should meet the required number of tugboats for the tasks. Equation (11) states that a tugboat can only be docked at one tugboat base at a given time. Equation (12) indicates that the tugboats remain at their previous tugboat base when not performing tasks.

3.5. Model Processing

By consulting the Ref. [29], this paper adopts a fuzzy ranking method based on credibility measures to address the time window constraints within the model. In this context, $\widetilde{a}$ and $\widetilde{b}$ are designated as two potential parameters represented by triangular possibility distributions, also referred to as fuzzy numbers.

$$\widetilde{a} = TFN(a_1, a_2, a_3), \widetilde{b} = TFN(b_1, b_2, b_3) \tag{13}$$

The degree of possibility for $\widetilde{a} \leq \widetilde{b}$ is calculated as follows:

$$Cr(\widetilde{a} \leq \widetilde{b}) = \begin{cases} 0 & a_1 \geq b_3 \\ \frac{b_3 - a_1}{b_3 - b_2 + a_2 - a_1} & a_2 \geq b_2, a_1 \leq b_3 \\ 1 & a_2 \leq b_2 \end{cases} \tag{14}$$

When the confidence level is α, $\widetilde{a} \leq \widetilde{b}$ can be re-expressed as:

$$Cr(\widetilde{a} \leq \widetilde{b}) \geq \alpha \equiv \begin{cases} \frac{b_3 - a_1}{2(b_3 - b_2 + a_2 - a_1)} \geq \alpha & 0 \leq \alpha \leq 0.5 \\ \frac{a_3 - b_1 + 2b_2 - 2a_2}{2(b_2 - b_1 + a_3 - a_2)} \geq \alpha & 0.5 \leq \alpha \leq 1 \end{cases} \tag{15}$$

Therefore, for symmetric triangular fuzzy numbers, the definite equivalent of the fuzzy equality $Cr(\widetilde{a} \leq \widetilde{b}) \geq \alpha$ can be simplified to:

$$a^c + (2\alpha - 1)\omega_a \leq b^c + (1 - 2\alpha)\omega_b \tag{16}$$

In this context, a^c and ω_a represent the center and spread of a symmetric fuzzy number $\widetilde{a}$, respectively. Similarly, b^c and ω_b represent the center and spread of a symmetric fuzzy number $\widetilde{b}$, respectively.

If all fuzzy parameters in the model are considered as symmetric triangular fuzzy numbers with a 10% distribution on both sides, i.e., $\widetilde{a} = \langle a^c, 0.1a^c \rangle \equiv TFN(0.9a^c, a^c, 1.1a^c)$, the following can be derived:

$$(0.9 + 0.2\alpha)a^c \leq (1.1 - 0.2\alpha)b^c, \forall \alpha \in [0, 1] \tag{17}$$

Therefore, Equations (7) and (8) related to the time window in the model can be rewritten as:

$$(0.9 + 0.1\alpha)t_s \leq \left| st_j - et_{j-1} - \sum_k D1^k_{i,j-1} B^k_{j-1,i} A_{i,j-1} / V_i \right| \leq (1.1 - 0.2\alpha)t_e, \forall i \in I, j \in \{2, 3, \cdots, M\} \tag{18}$$

$$(0.9 + 0.1\alpha)t_w \leq \sum_k D2^k_{i,j-1} B^k_{j-1,i} A_{i,j-1} / V_i \leq (1.1 - 0.1\alpha)t_k \tag{19}$$

The above is the fuzzy programming mathematical model for tugboat scheduling established in this paper.

4. Improved DDPG Algorithm

4.1. Markov Decision Process

Within the framework of the defined tugboat scheduling model, the tugboat scheduling problem can naturally be represented as a problem of state space and state transition. At any given point in time, the state of the system can be used to describe the locations, demands, and availability of tugboats and vessels. The process of transitioning a tugboat from one state to another is defined as a state transition, which can be represented by the model. Since the state of the tugboat is not influenced by its historical states, i.e., the future state only depends on the current state and the action taken, this property possesses the basic characteristics of a Markov Decision Process (MDP). Therefore, the tugboat scheduling problem can be abstracted as an MDP.

A Markov Decision Process (MDP) is a five-tuple: <S, A, P, R, γ>, where S represents the state space of the system, A represents the action space, P represents the state transition function, R represents the immediate reward associated with taking an action, and γ is a parameter ranging from 0 to 1 that determines the degree of discounting for future reward values. When γ is closer to 1, the algorithm pays more attention to future rewards, whereas when γ is closer to 0, the algorithm focuses more on immediate rewards. The objective of an MDP is to determine an optimal policy that maximizes the long-term cumulative reward value for the individual. By solving the MDP, a clear and executable decision-making plan can be obtained to guide reasonable decisions during scheduling [30].

4.2. State Space Definition

In tugboat scheduling, the construction of the state space requires consideration of various factors related to task execution. In this paper, the state variable is defined as s_t, representing the system state at the decision stage t for each task. s_t is a comprehensive information set that incorporates critical information about the tugboat scheduling system when making decisions at stage j. This information can significantly impact the decision-making efficiency of the entire scheduling system, and it is crucial to extract these key pieces of information during the optimization process [31,32]. By utilizing system input data and considering the state changes that occurred after the previous decision, an accurate assessment of the current system state information can be made. This paper characterizes the system state information from two perspectives: task sequence information, and tugboat status information. A tugboat status vector is established, and at time t, the system state is represented as:

$$s_t = \{O(t), P(t), x_1(t), x_2(t), \cdots, x_I(t)\} \tag{20}$$

where $O(t)$ represents the current demand for tugboats based on the task requirements, $P(t)$ represents the required tugboat power for the task, and $x_1, x_2, \cdots x_n$ represents the location ID of the tugboat base where the tugboat is currently stationed. If the tugboat is engaged in a task and not at the tugboat base, the value of (z) is set to 0.

4.3. Action Space Definition

Based on the description of the tugboat scheduling problem and the construction of the mathematical model, a more reasonable action space is required for the scheduling system to effectively plan the execution sequence of tugboat tasks and consider the allocation of tugboat resources within the time window of arrival at the target task. Due to the increase in port tasks and the limited number of tugboats, the decision-making system must accurately determine the task execution order within a limited time [33]. Therefore, in the optimization process of tugboat scheduling, this paper considers the strategy of the tugboat's departure time for each task and incorporates the departure time as a key parameter into a continuous action space framework. The aim is to ensure that the tugboat arrives within the task's start time window at the most economical speed possible, optimizing both energy consumption and time costs. Therefore, the action space is defined for each state as whether the tugboat is dispatched to the task destination and the dispatch time parameter. The action space is as follows:

$$A_j = \left\{(a_1, x'_1), (a_2, x'_2), \cdots, (a_i, x'_i), \cdots, (a_N, x'_N); \lambda_1(t), \lambda_2(t), \cdots, \lambda_i(t), \cdots, \lambda_N(t)\right\} \tag{21}$$

In the action space definition, a_i represents whether a tugboat is dispatched to execute the current task. It is a binary variable, $a_i \in \{0,1\}$. x'_i represents the base number where the tugboat returns after completing the task. If the tugboat is not dispatched, the base number remains unchanged. $\lambda_i(t)$ represents the time delay parameter for each tugboat dispatch, where $\lambda_i(t) \in [0,1]$.

4.4. Reward Setting

The setting of reward values in tugboat scheduling aims to guide the agent in reinforcement learning training to find a balance between minimizing the fuel cost of tugboats and the cost of on-time arrival. Through iterative training, the reward values are maximized to achieve improved performance in tugboat scheduling. At each decision-making stage of a task, the scheduling system acquires the current state variable s_t and makes a decision $\pi(s_t)$ [34]. During the decision-making process, the system assigns tugboats to target vessels based on the state variables and returns the tugboats to their corresponding bases after task completion.

An effective metric is needed to measure the effectiveness of task completion. Therefore, two key optimization objectives are considered: minimizing task delay time, and minimizing the fuel consumption cost of tugboats. Based on these objectives, a reward function is designed to calculate feedback reward values, which are used to evaluate actions and optimize strategies. The start time of each target vessel's task is set as the arrival time of the tugboat, and the arrival time is calculated by multiplying the time delay parameter for dispatching the tugboat with the time window. Therefore, the reward value setting at each time step t is redefined as:

$$r(t) = e^{-\eta Z_1} + e^{-(1-\eta)Z_2} \tag{22}$$

where η represent the weights of fuel cost and time cost, respectively. To maximize the expected cumulative long-term reward, the reward function is ultimately expressed as:

$$R_n = \sum_n \gamma^{t-n} r(s_t, a_t) \tag{23}$$

4.5. DDPG Algorithm

As a policy gradient algorithm that searches for an optimal policy in the policy space to maximize expected returns, the Policy Gradient (PG) algorithm estimates the policy gradient through sampled trajectories and uses the gradient to update policy parameters. This is done by increasing or decreasing the probability of taking a certain action in a given state to maximize expected returns. In the PG algorithm, actions are sampled by randomly selecting actions from the policy's probability distribution. Unlike the PG algorithm, the Deterministic Policy Gradient (DPG) algorithm uses a deterministic policy, which is a deterministic mapping from states to actions, to address continuous action space problems. The Deep Deterministic Policy Gradient (DDPG) algorithm is an extension of the DPG algorithm that applies deep neural networks to the learning of both the policy and the value function. This allows the DDPG algorithm to handle high-dimensional state and action spaces. In the context of tugboat scheduling, where time is considered as a continuous action space, the application of the DDPG algorithm offers more flexible and applicable optimization capabilities [35].

The DDPG algorithm combines the DPG algorithm with the DQN algorithm by introducing an experience replay mechanism. Through the training of neural networks, the agent is able to select the optimal action based on the current state and gradually learn how to take the optimal sequence of actions in a given state by learning approximate action functions and approximate deterministic policies using deep neural networks. The algorithm introduces the Actor-Critic network framework, where the Critic network, also known as the value network, is used to evaluate the Q-value of the current state-action pair. After the evaluation, it provides gradient information to the policy network for updating its parameters. The Actor network, also known as the policy network, is used to calculate the deterministic policy corresponding to the current action [36]. The structural framework of the Actor-Critic network is illustrated in Figure 2.

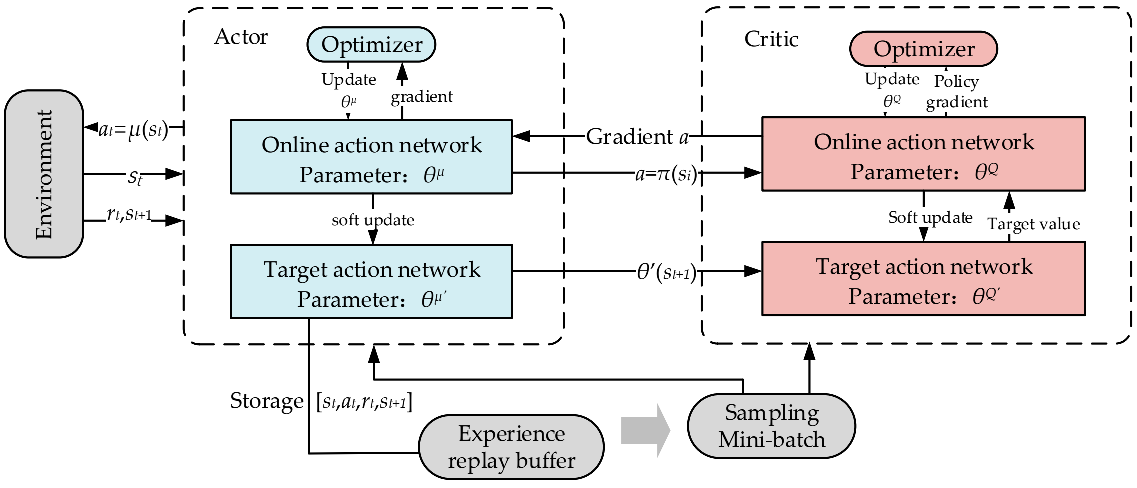


Figure 2. DDPG algorithm framework.

In the DDPG algorithm, to address the instability issues associated with the single-network learning of the DQN algorithm, a dual-network architecture is employed. Both the value network and the policy network consist of two neural networks: a predictor network, and a target network. The introduction of an experience replay mechanism enhances the stability and convergence of the DDPG algorithm. Within this framework, the online policy network generates actions by maximizing the action value $Q(s_t, a_t)$ associated with the state s_t. The goal of the Actor network is to learn actions with higher Q values, leading to better performance. The Critic network consists of an online network and a target network, where the online network estimates the state-action value. The training objective is to optimize the Critic network such that it can generate more accurate Q values, resulting in superior estimation performance. Figure 3 illustrates the framework structure of the DDPG algorithm.

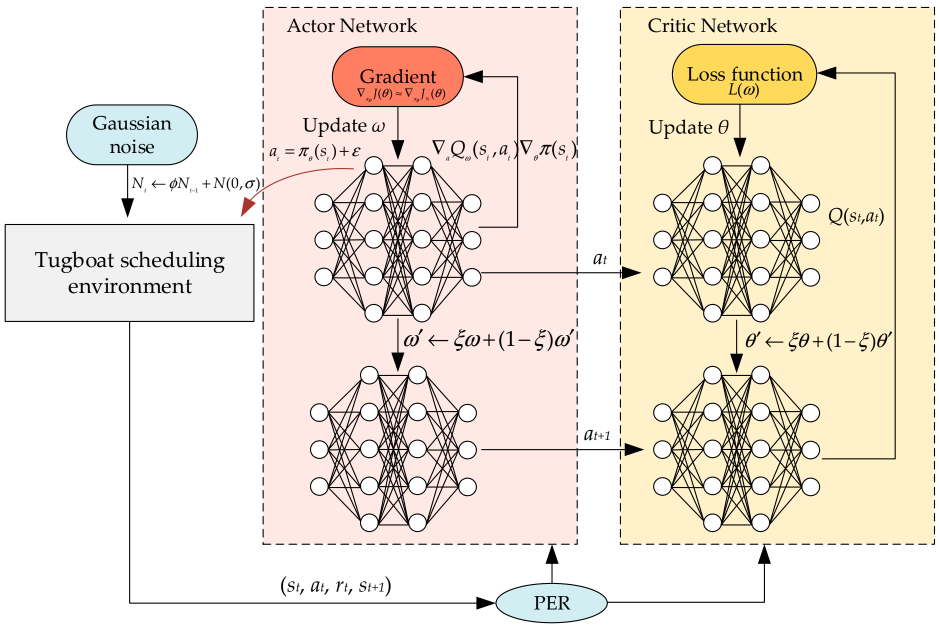


Figure 3. Improved algorithm architecture diagram.

The DDPG algorithm consists of four networks used for approximating the *Q*-value function and the policy. Among them, the Critic target network is used to estimate the *Q*-value function $Q_{\omega'}(s_{t+1}, \pi_{\theta'}(s_{t+1}))$ for the state-action function at the next time step. This *Q*-value function aims to approximately estimate the next action value $\pi_{\theta'}(s_{t+1})$ through the Actor target network. Through this process, the target value of the Q-value function under the current state can be obtained. Hence, the target value of the Q-value function under the current state can be represented as:

$$y_t = r_t + \gamma Q_{\omega'}(s_{t+1}, \pi_{\theta'}(s_{t+1})) \tag{24}$$

The Critic network outputs the Q-value function for the current state-action pair, serving as a critical evaluation of the current policy's performance. To enhance the agent's exploration in the environment, the DDPG algorithm introduces a Gaussian noise function to the action policy, introducing a degree of randomness during training.

The target $y_t - Q_\omega(s_t, a_t)$ for the Critic network is obtained from the Actor target network by estimating the next action values and then calculating the corresponding Q-values for those actions. The parameters of the Critic network are updated by minimizing the loss value, aiming to approximate the target Q-values. The loss function is calculated as follows:

$$L(\omega) = \frac{1}{N}\sum_{t=1}^{N} (y_i - Q_\omega(s_t, a_t))^2 \tag{25}$$

where, $a_t = \pi_\theta(s_t) + \varepsilon$, ε represents the exploration noise added to the action policy. In a deterministic environment, the gradient of the objective function with respect to the parameters θ is equivalent to the expectation of the gradient of the Q-value function with respect to θ. To obtain an estimate of this expectation, a random sample of N mini-batches of data is drawn from the experience replay buffer. Subsequently, mini-batch gradient descent is used to maximize the objective function $J(\theta)$:

$$\nabla_{\pi_\theta} J(\theta) \approx \nabla_{\pi_\theta} J_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} [\nabla_a Q_\omega(s_t, a_t) \nabla_\theta \pi(s_t)] \tag{26}$$

To update the target network parameters ω' and θ', the algorithm employs a soft update mechanism, where updates occur at a frequency determined by the target network update rate ξ, ensuring stable convergence during training. The DDPG algorithm combines value-based and policy-based methods, enabling it to effectively handle continuous action spaces while maintaining a degree of exploration capability.

$$\omega' \leftarrow \xi\omega + (1 - \xi)\omega' \tag{27}$$

$$\theta' \leftarrow \xi\theta + (1 - \xi)\theta' \tag{28}$$

where $\xi \in [0, 1]$.

4.6. Prioritized Experience Replay Mechanism

The PER (Prioritized Experience Replay) mechanism introduces the concept of priority on the basis of experience replay. Its core idea is to assign different priorities to the experiences stored in the experience replay buffer, such that during training, the experiences that are more helpful for learning are selectively sampled with higher probabilities [37]. The prioritization is typically based on the TD error (Temporal Difference Error) of the experiences, which represents the difference between the model's estimation of the action-value function in the current state and the actual reward received. The TD error can be computed using the Bellman equation:

$$T_\tau = r_t + \gamma Q'_\omega(s_{t+1}, a_{t+1}) - Q_\omega(s_t, a_t)) \tag{29}$$

To optimize the experience replay process and improve the learning efficiency of important experiences, the TD error can be converted into a priority score. This allows the training process to focus more on experiences that have a greater impact on updating the network parameters. Initially, the computed TD error is converted into a priority score:

$$p_\tau = |T_\tau| + \mu \tag{30}$$

where μ is a small constant used to avoid zero priority.

Then, the probability of each selected sample is calculated by normalizing the priorities into a probability distribution. An adjustable hyperparameter β is used to control the distribution of sample weights. The probability distribution is defined as:

$$P(t) = \frac{p_t^{\beta}}{\sum_k p_k^{\beta}} \tag{31}$$

where $P(t)$ is the sampling probability of the t-th sample, which is positively correlated with its priority. After sampling, the importance sampling weight of each sample relative to the others is computed and used to adjust the gradients during the update of network parameters. The weight for the t-th sample is given by:

$$\omega_t = \left(\frac{1}{N} \cdot \frac{1}{P(t)}\right)^{\beta} \tag{32}$$

The gradients are computed using the sampled experiences, and the parameters of the DDPG's policy and value networks are updated accordingly. Additionally, to maintain the freshness of the experience replay buffer, the priorities within the buffer are periodically updated by mapping new TD errors to the corresponding experiences. Therefore, the loss function equation is modified as follows:

$$L(\omega) = \frac{1}{N} \sum_{t=1}^{N} \omega_t (Q(s,a) - Q_{\omega}(s,a))^2 \tag{33}$$

4.7. Noise Reduction Method

To address the balance between exploration and exploitation and prevent the agent from getting stuck in local optima, noise is added to the action space of the algorithm to encourage the agent to better understand the environment [38,39]. Adding noise to the action space refers to adding noise to the computed actions, allowing the agent to explore new actions and preventing it from settling for suboptimal solutions:

$$a_{t+1} = \pi_{\theta}(s_t) + N_t \tag{34}$$

The noise source is Gaussian noise with a mean of 0 and a variance of σ. The recursive formula for noise is defined as:

$$N_t \leftarrow N_{t-1} + N(0, \sigma) \tag{35}$$

where s denotes the degree of retention of the noise from the previous step to the current step.

To encourage more exploration during the initial stages of training and to focus more on the learned strategy in the later stages, the method of gradually reducing the noise standard deviation σ over the course of training is adopted. The method used in this paper is exponential decay, which gradually decreases the intensity of exploration. The exponential decay of the standard deviation σ is defined as:

$$\sigma_t = \sigma \cdot \exp(\tau t) \tag{36}$$

where τ is the decay rate that controls the speed of decay, and t is the current training step number. By decaying the exploration parameter exponentially, the agent becomes more inclined to make decisions based on learned experience, which can improve the stability of the algorithm.

Based on the aforementioned improvements to the DDPG algorithm, the enhanced algorithm is named NRPER-DDPG (DDPG Algorithm with Prioritized Experience Replay and Noise Reduction). Drawing upon the preceding description of the enhanced

PER-DDPG algorithm, Figure 4 offers a visual representation of the intricate architecture underlying the algorithm's operation. This figure provides a detailed breakdown of the various components and their interactions, offering a clear picture of how the algorithm functions, such as the prioritized replay buffer, which ensures that important transitions are replayed more frequently, and the dual neural networks, which separately estimate the value function and select actions. The Figure also demonstrates the decaying noise strategy employed to encourage exploration during the initial stages of learning.

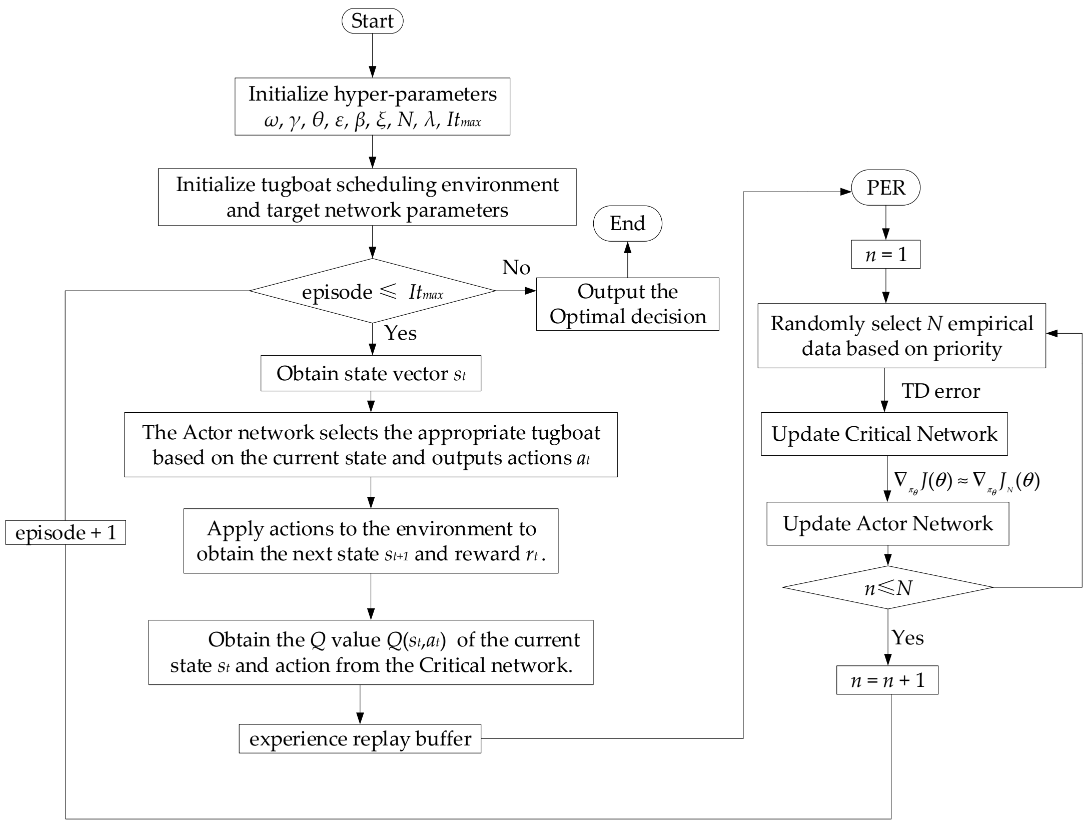

Start
Initialize hyper-parameters
$\omega, \gamma, \theta, \varepsilon, \beta, \xi, N, \lambda, It_{max}$
Initialize tugboat scheduling environment and target network parameters
episode ⩽ It_{max}
No
Yes
Output the Optimal decision
End
Obtain state vector s_t
The Actor network selects the appropriate tugboat based on the current state and outputs actions a_t
Apply actions to the environment to obtain the next state s_{t+1} and reward r_t.
Obtain the Q value $Q(s_t,a_t)$ of the current state s_t and action from the Critical network.
experience replay buffer
episode + 1
PER
$n = 1$
Randomly select N empirical data based on priority
TD error
Update Critical Network
$\nabla_{\pi_\theta} J(\theta) \approx \nabla_{\pi_\theta} J_N(\theta)$
Update Actor Network
$n \leqslant N$
Yes
$n = n + 1$

Figure 4. Solving tugboat scheduling flowchart based on the NRPER-DDPG algorithm.

4.8. Tugboat Scheduling Method Based on the NRPER-DDPG Algorithm

Based on the aforementioned theoretical approach, the tugboat scheduling problem is solved by combining mathematical models and algorithmic improvements. The solution process is divided into the following steps:

(1) Acquisition of Port Task-Related Information: Firstly, information related to port tasks is obtained, including the arrival time of each task, the required number of tugboats, expected working hours, and specific tugboat power requirements. Subsequently, detailed data on the existing tugboats in the port are collected, covering the number of tugboats, power, idle speed, idle cost, and assistance cost, as well as the number and specific geographical location of tugboat bases, to initialize the tugboat scheduling environment.

(2) Initialization of Experience Replay Buffer: Experience samples are constructed from information related to decision-making, such as state, action, and reward, and the experience replay buffer is initialized. Subsequently, these experience samples are added to the buffer by randomly generating initial solutions, ensuring that the agent interacts with the environment to generate new experiences and provide more learning samples.

(3) Actor-Critic Network Model Training: The Actor network receives environmental state information as input and outputs tugboat actions. Noise is introduced into the action

output to improve exploration. Meanwhile, the environmental state information and the action information generated by the Actor network are used as inputs to the Critic network, which estimates the value of the actions. The output of the Critic network is an estimate of the value function for a given state and action, and this output is used to calculate the TD error for prioritized experience replay.

(4) Iterative Optimization of Reinforcement Learning Model: The deep reinforcement learning model based on the Actor-Critic network is trained to learn which actions to take in different states to maximize cumulative rewards. This involves adjusting decisions such as tugboat task allocation and departure times to optimize fuel costs and task completion times. After multiple iterations of training, the model continuously optimizes the tugboat scheduling strategy.

The flowchart of the algorithm is shown in Figure 5.

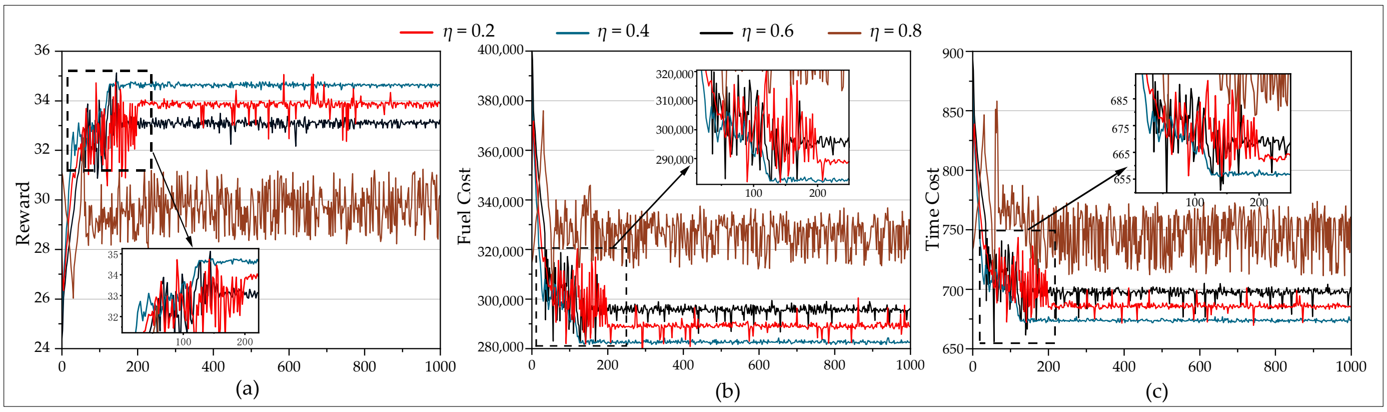


Figure 5. Training process of the algorithm with different η values. (**a**) Cumulative reward value; (**b**) fuel cost; (**c**) time cost.

Algorithm 1 presents the pseudocode for addressing the tugboat scheduling problem through the utilization of the NRPER-DDPG algorithm. This pseudocode outlines the process of implementing the algorithm, providing a detailed breakdown of the steps involved. These steps encompass the initialization of necessary variables, the definition of state and action spaces, and the iterative learning and updating of policies to optimize tugboat scheduling decisions. Through this pseudocode, readers can gain a deeper understanding of how the NRPER-DDPG algorithm operates and how it can be effectively applied to address real-world scheduling challenges in port operations. This enhanced understanding can facilitate the development and implementation of more efficient and sustainable tugboat scheduling strategies in port management.

Algorithm 1. Tugboat Scheduling Method Using the NRPER-DDPG Algorithm

```
Initialize: tugboat scheduling environment, parameters for Actor and Critic networks, experience replay buffer ReplayBuffer, noise parameters
Input: hyper-parameters $\omega$, $\gamma$, $\theta$, $\varepsilon$, $\beta$, $\xi$, $N$, $\lambda$, $It_{max}$
Output: Tugboat scheduling decision
For episode in range(Itmax):
    Initialize state $s$
    For step in range(max_steps):
        The Actor target network approximates the next action value $\pi_{\theta'}(s_{t+1})$ based on the current state $s_t$
        Target $Q$-values = Critic_target($s_{t+1}$, $a$)
        Store state transition sequence [$s_t$, $a_t$, $r_t$, $s_{t+1}$] to the experience replay memory
        Update priority function using the current $Q$-values output by Critic
        If replay_buffer is full:
            Randomly sample a minibatch of transitions [$s_t$, $a_t$, $r_t$, $s_{t+1}$] from replay_buffer
            Compute target $Q$-values y_target using Critic_target
            Calculate the mean square error loss function $L(\omega)$
            Update Critic neural network $\omega' \leftarrow \xi\omega + (1-\xi)\omega'$
                    Using Small Batch Gradient Descent Method to Calculate the Maximum Value of $J(\theta)$
            Update Actor neural network $\theta' \leftarrow \xi\theta + (1-\xi)\theta'$
        If episode % update_freq = 0:
            Soft update target networks Actor_target and Critic_target
            Decay noise to a smaller value
        $s = s_{t+1}$
```

5. Experimental Discussion

To robustly validate the effectiveness of the proposed NRPER-DDPG algorithm in addressing the intricate fuzzy programming mathematical model for tugboat scheduling, comprehensive simulation experiments were conducted. These experiments aimed to demonstrate the algorithm's performance and applicability in real-world scenarios, thereby bolstering the credibility and reliability of the research findings. To ensure accurate and efficient execution, the experimental code was written in Python 3.10, a widely used and reliable programming language. The code was then executed on a system equipped with an Intel i7-13700KF @ 2.70GHz processor, 32GB RAM, and a 64-bit Windows operating system, all manufactured by Intel, located in Santa Clara, CA, USA. Through these rigorous simulation experiments, the study aimed to provide a solid foundation for the application of the proposed stable noise PER-DDPG algorithm in practical tugboat scheduling operations.

5.1. Experimental Parameter Settings

To thoroughly evaluate the performance of the meticulously designed algorithm and mathematical model, this paper utilizes China's port charging standards [40] as a reliable reference point. In order to ensure the experiments' efficacy and practicality, multiple simulation random cases of varying scales are constructed. These cases aim to capture the nuances and complexities of real-world port operations. The parameters of these simulation random cases, along with the neural networks employed, are comprehensively outlined in Table 2. These simulations take into account three crucial aspects of constraints: the demand for tugboats for each individual task, the operational limitations inherent to the tugboats themselves, and the capacity limitations of the tugboat bases. By considering these factors, the simulations aim to reflect the realities of port operations, and thereby provide a robust assessment of the algorithm and mathematical model's performance.

Table 2. Parameters in Simulation Cases and Neural Networks.

Parameters	Value	Parameters	Value
O_j	Uniform [1, 4]	Task generation probability	0.4
$D1_{ij}^k$ (n mile)	Uniform [10, 20]	Learning rate l_r	1×10^{-4}
$D2_{ij}^k$ (n mile)	Uniform [10, 20]	Reward discount factor γ	0.1
Ct_i (USD)	Uniform [8, 12]	Soft update coefficient ξ	0.01
Cn_i (USD)	Uniform [20, 28]	Maximum number of episodes It_{max}	1000
P_i (hp)	Uniform [3000, 6000]	Target network parameter update frequency	100
P_j (hp)	Uniform [3000, 6000]	Experience replay buffer capacity R	3×10^5
V_i (n mile/h)	Uniform [6, 15]	Batch size N	64

5.2. Parameter Experiments

To strike a balance between the fuel cost and time cost of tugboats, this paper introduces weighted objectives into the reward function. The selection of these weights plays a crucial role in the solution accuracy and convergence speed of the algorithm. To investigate the impact of these weights, an experiment was conducted using a case with $J = 50$ objectives and $I = 60$ tugboats, as designed in Table 2. In the experiment, the value of η was set to 0.2, 0.4, 0.6, and 0.8 to observe its effect on the algorithm's performance. The results are presented in Table 3 (-- indicates that the algorithm did not converge).

Table 3. Scheduling Results with Different η Values.

η	Reward	Fuel Cost (USD)	Time Cost (min)
0.2	33.8847	288,830.9563	683.5225
0.4	34.6327	282,682.3915	652.3833
0.6	33.2221	296,113.7766	706.1413
0.8	--	--	--

Figure 5 illustrates the variation curves of cumulative rewards, fuel costs of tugboats, and time costs of the algorithm. By observing the charts, it can be found that the optimization objectives exhibit different trends with changes in the η value. An appropriate η value helps ensure that the optimization objectives move towards minimization. However, excessively high or low η values may result in the algorithm failing to achieve ideal optimization objectives after convergence or even cause the algorithm to diverge. Therefore, after comprehensive consideration, this paper sets the η value to 0.4 to balance the algorithm's accuracy and convergence speed. This aims to ensure that the algorithm maintains high solution accuracy while converging quickly during optimization, thereby facilitating better decision-making.

5.3. Simulation Case Study and Comparison

To further validate the effectiveness of the tugboat scheduling method proposed in this paper, multiple sets of simulation cases were designed, encompassing a range of different case sizes, as detailed in Table 4. These cases were specifically designed to simulate real-world tugboat operations, incorporating various practical scenarios encountered in day-to-day operations. By simulating these diverse scenarios, the aim was to comprehensively assess the performance of the proposed scheduling method across different conditions, providing a robust evaluation of its practical applicability and reliability.

The results of these simulation cases were compared with the standard DDPG algorithm and the PER-DDPG algorithm. In this paper, large-scale case 8 is used as an example to demonstrate the performance comparison between the two algorithms. As shown in Figure 6a–c, the convergence curves of the three methods in terms of reward value, fuel consumption, and time cost are presented, respectively. From the figures, it can be observed that the NRPER-DDPG algorithm significantly outperforms the standard DDPG algorithm and the PER-DDPG algorithm in key performance metrics such as reward value acquisition, fuel consumption, and time cost. Specifically, the NRPER-DDPG algorithm achieves an improvement of 7.25% and 16.67% in cumulative reward value compared to the standard DDPG algorithm and the PER-DDPG algorithm, respectively. It also reduces fuel consumption costs by 13.67% and 20.05%, and reduces time costs by 5.59% and 13.11%, respectively.

Table 4. Scale of the example.

Numerical Example	Vessel	Tugboat	Tugboat Base
1	10	16	6
2	20	32	12
3	30	48	18
4	40	64	24
5	50	80	30
6	65	104	39
7	75	112	45
8	85	120	51

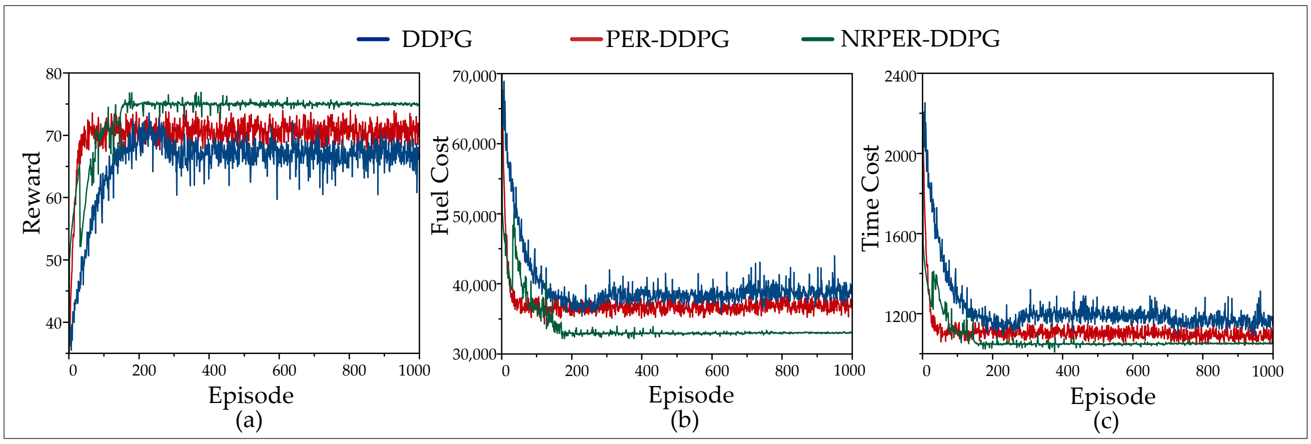


Figure 6. Training process of the NRPER-DDPG algorithm on Case 8. (**a**) Cumulative reward value; (**b**) fuel cost; (**c**) time cost.

To further validate the superiority and stability of the proposed algorithm, three reinforcement learning algorithms were used to conduct 20 repeated experiments on large-scale case 8. In each experiment, data on cumulative reward value, fuel consumption, and time cost were recorded for each algorithm. The upper edge, upper quartile, median, lower quartile, lower edge, and average of the data were calculated separately. To more intuitively demonstrate the distribution and dispersion of the data, a box plot was created, as shown in Figure 7.

Based on the results of the 20 experiments conducted, compared to the standard DDPG algorithm and the PER-DDPG algorithm, the average cumulative reward value of the NRPER-DDPG algorithm increased by 5.13% and 4.21%, respectively. The fuel cost decreased by 17.9% and 14.1%, and the time cost decreased by 19.4% and 10.93%, respectively. Additionally, as can be seen from the box plot, the NRPER-DDPG algorithm exhibits greater stability in terms of reward value, fuel cost, and time cost compared to the other two algorithms. The data distribution of the NRPER-DDPG algorithm is more compact and has lower dispersion, demonstrating its high stability.

The original slow convergence speed of the standard DDPG algorithm was significantly improved by introducing the prioritized replay mechanism. However, the stability of its performance after convergence still appeared insufficient, exhibiting relatively large fluctuations. By further introducing the decaying noise processing, the NRPER-DDPG algorithm not only maintained the advantage of fast convergence, but also significantly improved its stability after convergence, significantly reducing fluctuations. This significantly enhanced the robustness and practicality of the algorithm. After solving eight simulation cases, the convergence stability of the three algorithms is shown in Figure 8, and the results of solving each case using NRPER-DDPG are recorded in the table below the figure. It can be seen from the figure that the standard DDPG algorithm has the lowest cumulative reward value but the highest standard deviation. After introducing the prioritized replay mechanism, the algorithm effectively utilizes the data stored in the experience replay buffer, resulting in an increase in cumulative reward. After using the decaying noise method, the stability of the strategy is improved, further increasing the cumulative reward value and reducing the standard deviation. These improvements collectively enhance the robustness and practicality of the algorithm, enabling it to perform better when solving complex problems.

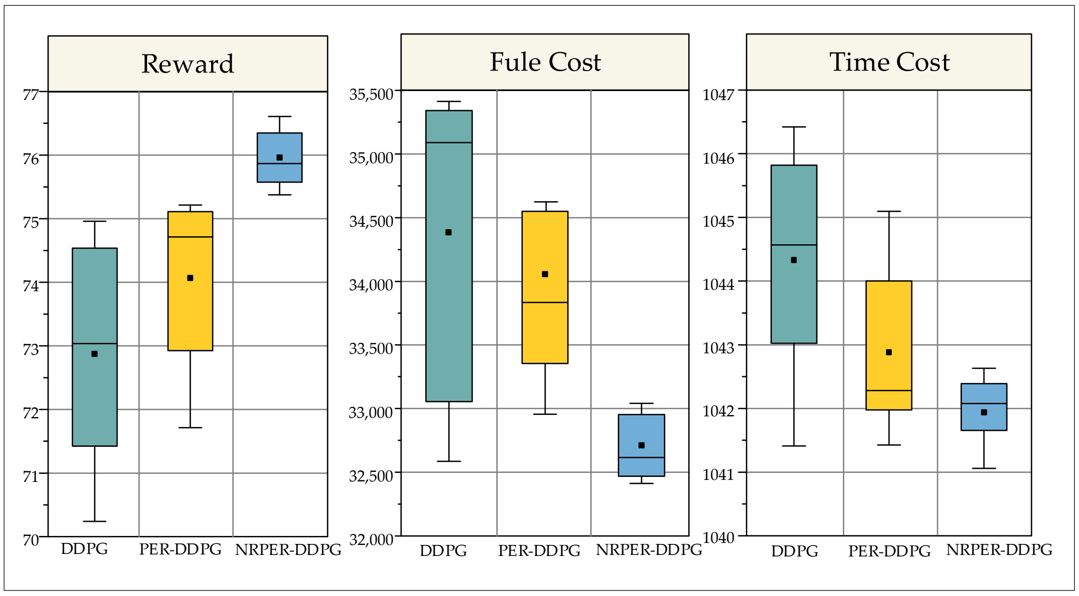
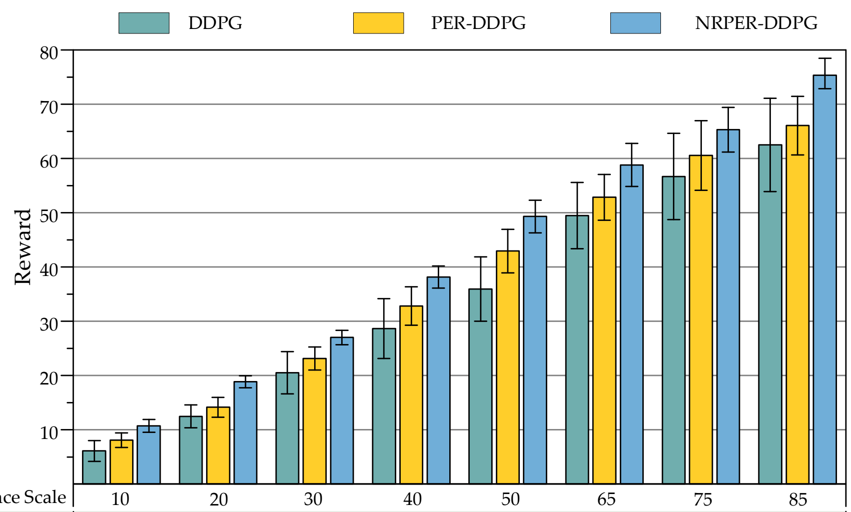


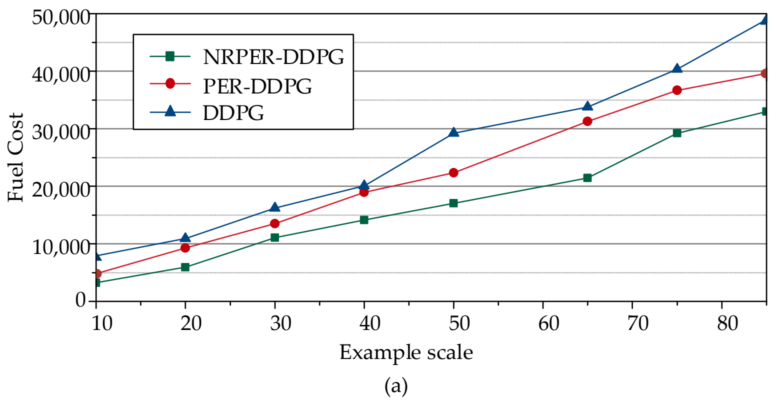
Figure 7. Box plot of solution result distributions for different algorithms.



Instance Scale	10	20	30	40	50	65	75	85
Reward	10.89	18.85	27.14	38.14	49.30	58.79	65.31	75.34
Fuel Cost	1261.76	2115.80	5162.26	8419.93	14,251.52	19,362.21	28,115.35	32,432.359
Time Cost	389.59	402.93	499.87	554.32	674.93	743.58	937.75	1048.43

Figure 8. Performance comparison chart after algorithm convergence.

The fuel cost and time cost results obtained by using the three algorithms to solve the eight simulation cases are shown in Figure 9a,b. It can be seen that the proposed NRPER-DDPG algorithm can achieve the optimal solution for different sizes of simulation cases. When the simulation case size is small, the performance of the NRPER-DDPG algorithm on the tugboat scheduling problem is similar to that of the standard DDPG algorithm and the PER-DDPG algorithm. As the simulation case size gradually increases, the proposed NRPER-DDPG algorithm demonstrates significant advantages, significantly reducing the fuel cost and time cost incurred by tugboat scheduling. This proves its effectiveness in handling large-scale simulation cases.



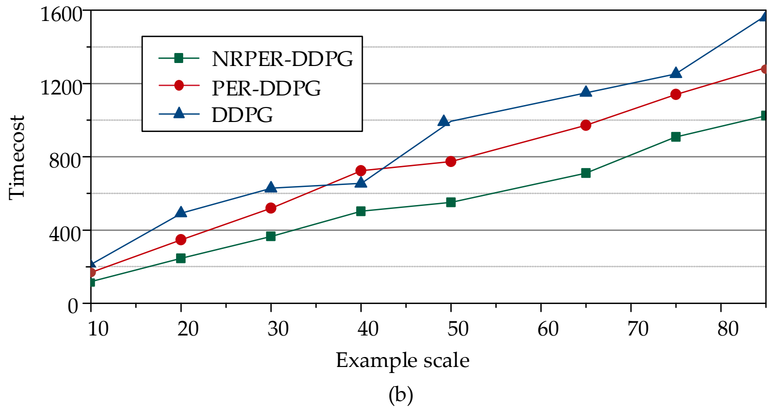


Figure 9. Fuel cost performance of different algorithms under different scales of examples. (**a**) Line chart of fuel cost after solving different instances using three reinforcement learning algorithms; (**b**) Line chart of time cost after solving different instances using three reinforcement learning algorithms.

5.4. Experimental Comparison with CPLEX Solver

To validate the effectiveness of the deep reinforcement learning algorithm and model proposed in this paper for small-scale tugboat scheduling problems, we designed 18 small-scale instances based on Algorithm 1. By conducting comparative experiments between these instances and the CPLEX mathematical solver, we evaluated the performance of our algorithm in solving small-scale tugboat scheduling problems. The results are summarized in Table 5. The first four columns of the table represent the instance number, the number of tugboats I, the number of ships waiting for operation J, and the number of tugboat bases K, respectively. Columns 5 to 7 present the fuel cost, time cost, and solution time of the NRPER-DDPG method. Columns 8 and 9 list the fuel cost, time cost, and solution time of CPLEX, where '--' indicates that the CPLEX mathematical solver did not find a feasible solution within 2000 s.

From Table 5, it becomes apparent that the CPLEX mathematical solver is only capable of addressing small-scale tugboat scheduling problems. As the size of the problem instances increases, the solution space experiences exponential growth. Notably, when the tugboat fleet expands to 13 vessels, the computational time required by the CPLEX solver surpasses 2000 s, leading to a memory overflow and subsequent failure in obtaining a solution. Additionally, as the problem scale continues to escalate, CPLEX encounters difficulties in finding optimal solutions. In contrast, when utilizing the NRPER-DDPG algorithm proposed in this study, all instances can be solved within 4 s. Furthermore, an examination of the objective function reveals that, on average, the NRPER-DDPG algorithm achieves a 3.19% reduction in fuel costs and a 1.76% decrease in time costs compared to the solutions obtained by the CPLEX mathematical solver.

Based on a thorough analysis of the performance of the objective function, Figures 10 and 11 present a detailed comparison of the maximum time costs and fuel costs obtained using the CPLEX solver and the NRPER-DDPG algorithm across 18 case studies, excluding those instances where the CPLEX mathematical solver failed to find solutions. Through comparative analysis, it is observable that the NRPER-DDPG algorithm, while maintaining computational efficiency, is capable of delivering optimization results that are comparable or close to those of the CPLEX solver within a reasonable timeframe. Notably, when the algorithm exhibits superiority in terms of fuel cost, it is often accompanied by an increase in time cost, further validating the difficulty in simultaneously achieving optimal states for both fuel cost and time cost. This finding demonstrates that the NRPER-DDPG algorithm, when addressing small-scale tugboat scheduling optimization problems, not only maintains the quality of solutions, but also significantly reduces the computation time, striving to strike a balance between the two objectives of fuel cost and time cost. Consequently, this validates the effectiveness of the method proposed in this paper for solving tugboat scheduling problems.

Table 5. Comparison of NRPER-DDPG and CPLEX on 18 small-scale instances.

ID	*I*	*J*	*K*	NRPER-DDPG			CPLEX		
				Fuel Cost	Time Cost	CPU Time	Fuel Cost	Time Cost	CPU Time
1	5	5	3	249.62	98.87	2.52	249.62	98.87	0.34
2	5	6	4	298.81	114.24	2.63	298.81	114.24	0.29
3	5	7	5	441.13	168.36	2.68	441.13	168.36	0.36
4	7	7	5	430.40	162.14	2.74	430.40	162.14	0.58
5	7	8	6	527.58	181.89	2.73	527.58	181.89	1.73
6	7	9	7	618.23	212.23	2.77	618.23	212.23	3.69
7	9	9	6	624.85	208.36	2.89	624.85	208.36	5.80
8	9	10	7	693.46	218.07	2.62	693.46	218.07	12.84
9	9	11	8	775.91	243.94	3.04	782.73	221.67	33.47
10	11	11	7	747.15	218.93	3.15	713.26	231.94	58.62
11	11	12	8	825.41	241.34	3.22	825.41	241.34	113.71
12	11	13	9	897.36	245.16	3.35	897.36	245.16	158.79
13	13	13	8	857.94	233.98	3.37	857.94	233.98	367.60
14	13	14	9	1069.70	287.52	3.40	--	--	2000
15	13	15	10	1143.87	307.12	3.64	1143.87	307.12	875.91
16	15	15	9	1125.72	289.23	3.72	1284.62	268.35	1374.25
17	15	16	10	1205.43	308.10	4.86	--	--	2000
18	15	17	11	1267.51	319.79	6.88	--	--	2000

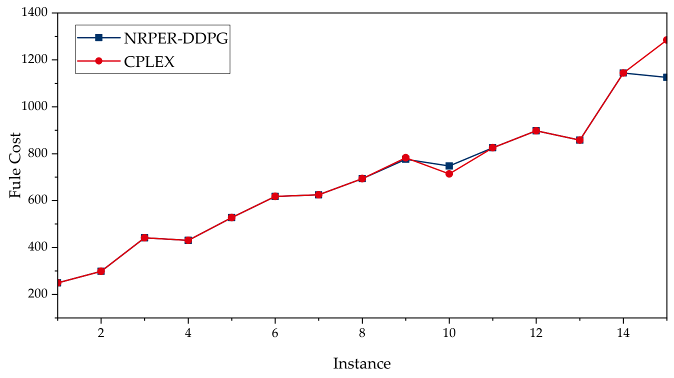


Figure 10. The fuel costs of NRPER-DDPG and CPLEX after solving 15 instances.

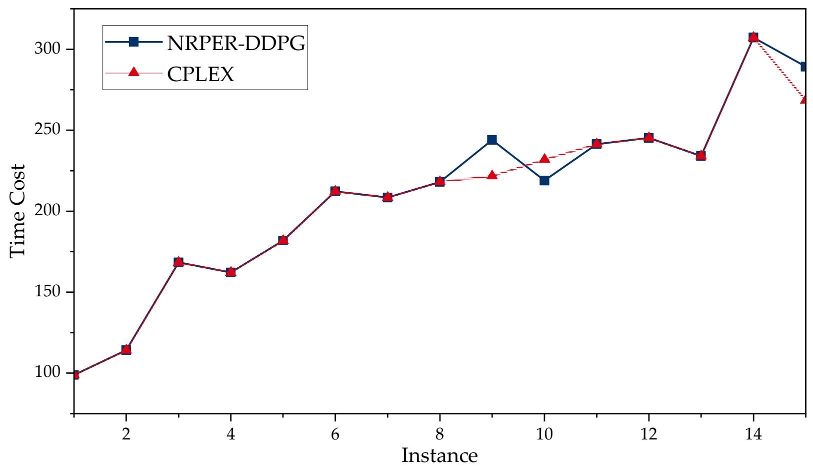


Figure 11. The time costs of NRPER-DDPG and CPLEX after solving 15 instances.

5.5. Conduct Comparative Experiments with the Existing Literature

Currently, the research on intelligent tugboat scheduling primarily relies on heuristic and meta-heuristic algorithms. However, this paper introduces reinforcement learning into the tugboat scheduling problem for the first time. To validate the effectiveness of the proposed reinforcement learning algorithm in addressing large-scale tugboat scheduling problems, we generated 18 sets of large-scale instances based on Algorithm 1. These instances were compared with the ALNS method proposed in Ref. [10], which employed a weighted linear summation of tugboat operational costs in defining the objective function. The experimental results of running both methods 10 times are summarized in Table 6. The first four columns represent the instance number, the number of tugboats (I), the number of vessels requiring service (J), and the number of tugboat bases (K). Columns 5 to 7 present the average values of fuel cost, time cost, and computation time obtained from 10 runs using the NRPER-DDPG method, while Columns 8 and 9 show the corresponding results for ALNS.

Table 6. Comparison of NRPER-DDPG and ALNS on 18 large-scale instances.

ID	I	J	K	NRPER-DDPG (10 runs)			ALNS [10] (10 runs)		
				Fuel Cost	Time Cost	CPU Time	Fuel Cost	Time Cost	CPU Time
1	30	20	12	2044.04	357.18	10.21	2087.13	349.18	7.46
2	40	22	15	2265.39	406.16	10.32	2379.26	413.57	14.51
3	45	28	18	2518.23	462.15	22.97	2562.57	445.30	19.68
4	55	30	22	6111.06	478.71	24.16	6254.71	470.19	33.62
5	65	32	25	8016.29	494.35	35.80	8114.63	484.31	42.59
6	70	36	28	8644.38	513.73	47.89	8982.86	517.22	60.68
7	75	44	31	11,330.87	536.02	61.26	11,426.92	528.61	88.20
8	80	46	34	13,008.06	584.22	84.19	14,295.65	554.37	102.69
9	85	52	37	16,081.48	643.65	116.12	15,973.71	662.11	166.25
10	90	54	40	17,402.96	671.08	252.68	17,681.45	679.48	274.91
11	95	60	43	18,120.26	695.48	314.09	18,355.94	701.38	368.35
12	100	62	46	18,768.37	707.83	435.15	18,406.62	758.97	487.24
13	105	68	49	20,206.46	714.16	497.07	21,564.17	684.34	531.54
14	110	70	52	20,443.59	781.52	625.98	21,648.43	681.16	753.80
15	115	86	55	28,397.06	801.51	726.09	31,751.16	793.50	813.44
16	120	88	58	28,909.40	966.10	810.80	30,149.57	924.28	1348.21
17	125	94	61	33,723.62	1011.44	872.15	35,729.83	997.55	1815.57
18	130	96	64	34,294.32	1167.19	938.56	33,915.36	1354.70	1893.26

Through the analysis of the data presented in Table 6, we observed that both algorithms were able to generate scheduling results within 2000 s for all 18 instances. When the instance size was smaller, the solutions obtained by the two algorithms exhibited a high degree of similarity. However, as the number of tugboats increased beyond 80 and the number of vessels waiting for service exceeded 46, the NRPER-DDPG algorithm began to demonstrate its superiority over ALNS. Specifically, in terms of average fuel consumption, the NRPER-DDPG algorithm achieved an average reduction of 3.51% compared to the ALNS algorithm. Additionally, there was an average decrease of 0.34% in time cost. These data indicate that the NRPER-DDPG algorithm can not only reduce the fuel cost of tugboats during the solution process, but also find optimal solutions in terms of time cost. Furthermore, in terms of solution time, apart from instance 1, the method proposed in this paper was able to find the optimal solution faster than the ALNS algorithm. The average solution time for the 18 instances was reduced by 49.8% compared to the ALNS algorithm. Especially in the case of larger instances, the proposed NRPER-DDPG algorithm reduced the solution time by up to 108.17% compared to ALNS. This proves that the method proposed in this paper can find optimal solutions in a shorter time frame.

To further validate the stability and consistency of the algorithm proposed in this paper for problem-solving, we specifically plotted the average values and standard deviations of the 10 solution results obtained by the two methods under their respective case studies into bar charts with error bars, as shown in Figure 12. This visualization provides an intuitive demonstration of the superiority and reliability of the algorithm's performance.

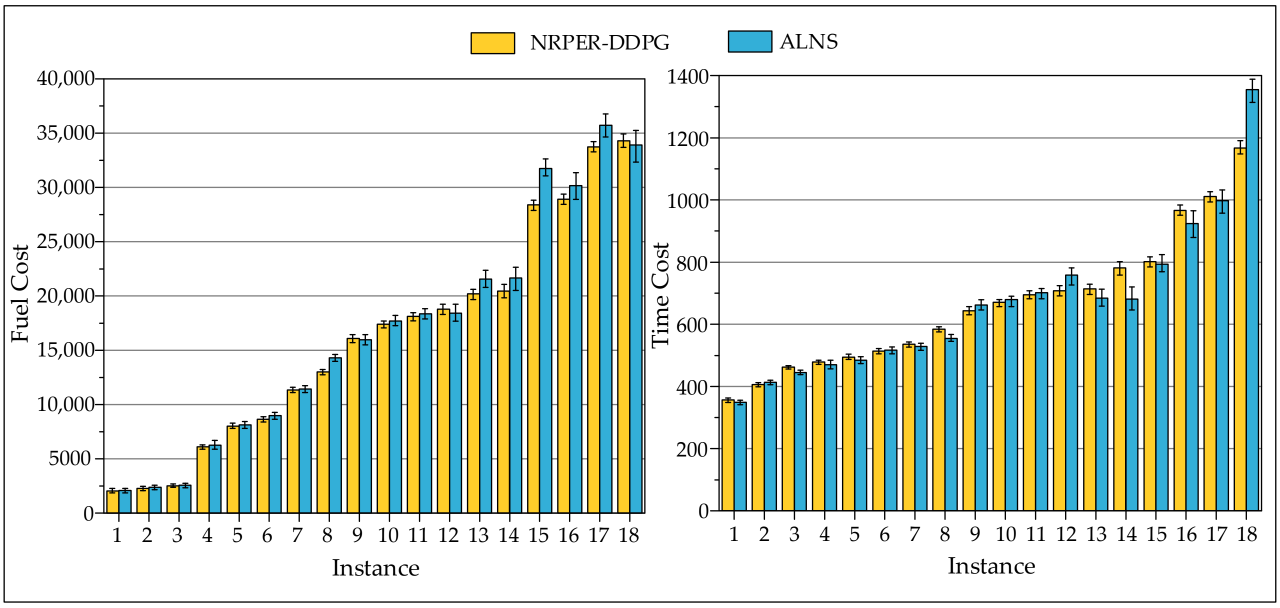


Figure 12. The average and standard deviation of the 10 time solution results of two algorithms.

By observing the solution results presented in the Figure, we discovered that when the size of the case study is relatively small, the standard deviation of the five solutions obtained by both algorithms remains at a low level. However, as the scale of the case study continues to expand, the standard deviation of the NRPER-DDPG algorithm is significantly lower than that of the ALNS algorithm. This trend strongly demonstrates the clear advantages of the stability and robustness of the method proposed in this paper compared to the ALNS algorithm.

5.6. Real-Case Experiment at Huanghua Port

To further validate the effectiveness and practicality of the tugboat scheduling method proposed in this paper, we selected the Huanghua Comprehensive Port Area, one of the important ports in China. Located on the coast of Bohai Sea in Huanghua City, Cangzhou City, Hebei Province, China, with geographic coordinates of 38°22′ N latitude and 117°38′ E longitude, it sits at the junction of Hebei and Shandong provinces and is a core area of the Bohai Rim Economic Circle. Due to the complex scheduling demands and rigorous operational environment of the Huanghua Comprehensive Port Area, its test results are of great significance for evaluating the practicality and reliability of the method. In this section, we use the actual port work plan for 30 May as the test scenario. Through detailed analysis and application of the work plan for that day, a comprehensive performance test of the method was conducted. The accuracy and reliability of the test results are crucial for assessing the practical value and broad applicability of the method. The specific work arrangement for Huanghua Port on 30 May is presented in Table 7. By addressing this specific scenario, we can more intuitively evaluate the performance of the algorithm in practical applications.

Table 7. Work Schedule of Huanghua Port on 30 May.

Task	Required Number of Tugboats	Required Tugboat Power (hp)	Berthing or Departure	Task Start Time
1	2	2000	Departure	0:00
2	2	2000	Berthing	2:30
3	2	3000	Berthing	5:00
4	3	3000	Departure	9:00
5	2	2000	Berthing	10:00
6	2	3000	Departure	12:00
7	3	5000	Departure	12:00
8	2	4000	Berthing	13:30
9	2	2000	Berthing	14:30
10	2	2000	Berthing	16:00
11	3	3000	Departure	18:00
12	2	2000	Departure	21:00
13	5	5000	Berthing	21:00
14	3	4000	Departure	22:00
15	3	3000	Departure	23:00

Using the method proposed in this study, the scheduling problem was solved, generating fuel consumption data for different time periods. To intuitively describe the distribution characteristics of these data, we specifically created a histogram, as shown in Figure 13. Furthermore, to demonstrate the practicality and superiority of the proposed method, a comparative analysis was conducted between this method, the CPLEX mathematical solver, and the ALNS method from reference [10]. The comparison focused on two key indicators: time cost, and fuel cost. Table 8 organizes and presents the relevant data.

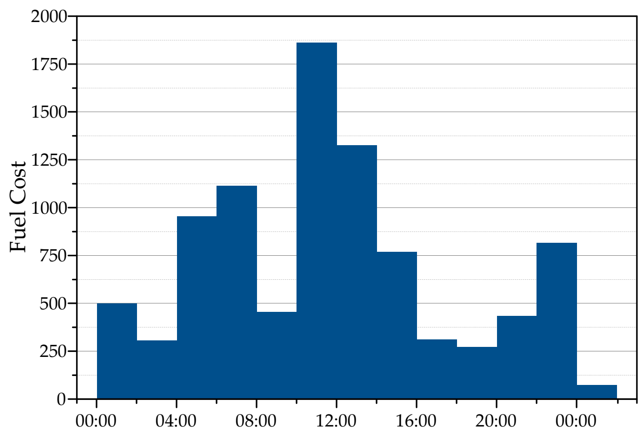


Figure 13. Histogram of fuel costs for different time periods.

Table 8. Solution results of the instance by three methods.

Method	Fuel Cost (USD)	Time Cost (min)
CPLEX	8774.62	1088.32
ALNS	9137.81	974.25
NRPER-DDPG	8774.62	1088.32

Based on the experimental results, it is evident that the proposed method in this paper demonstrates superior performance in solving the Huanghua Comprehensive Port Area instances. Specifically, in terms of fuel cost, our approach achieves a reduction of 18.56% and 9.79% compared to the CPLEX mathematical solver and ALNS, respectively. Furthermore, in terms of time cost, our method outperforms the other two methods by 4.13% and 9.54%, respectively. The comparative experiments conducted in the Huanghua Comprehensive Port Area provide compelling evidence of the practicality of the algorithm presented in this paper.

6. Conclusions and Prospects

This paper delves deeply into the mathematical modeling and algorithm optimization of tugboat scheduling problems, innovatively constructing a tugboat scheduling mathematical model based on fuzzy programming theory. This model meticulously considers the complex state transitions and dynamic action sets of tugboats and vessels waiting for operations in continuous time dimensions. Building upon this foundation, the paper further proposes a novel NRPER-DDPG algorithm that intelligently adjusts and refines scheduling decisions through continuous simulation training and iterative strategy optimization, thereby significantly enhancing decision accuracy and stability. To verify the scientific validity of the constructed mathematical model and the advancement of the algorithm, this paper not only conducts systematic testing in simulation examples, but also comprehensively compares it with current mainstream reinforcement learning algorithms. It is worth mentioning that we specifically compare the proposed NRPER-DDPG algorithm with the widely recognized ALNS algorithm under the same experimental conditions in terms of detailed performance. Experimental results fully demonstrate that the proposed algorithm exhibits significant advantages in both decision efficiency and solution superiority when addressing tugboat scheduling problems. Furthermore, to enhance the practical significance and application value of the research, this paper integrates the algorithm into a real-world tugboat scheduling case at Huanghua Port. Through on-site data collection and model application, the effectiveness and practicality of the algorithm in actual operations are successfully verified, providing solid support for the translation of theoretical achievements into practical applications. In summary, this paper comprehensively explores the mathematical modeling, algorithm design, simulation comparison, and practical application of tugboat scheduling, fully proving the outstanding contributions and practical value of the proposed model and algorithm in this field.

In future research, we can build upon the existing foundation to further refine and enhance the model and algorithm, aiming to strengthen their adaptability and stability in complex scenarios. Specifically, we can integrate diverse factors such as weather conditions, tidal variations, and ship types into the model, thereby improving its realism and practicality. Furthermore, to more accurately simulate the changing conditions in the actual environment, we should also consider employing multi-source data fusion techniques to seamlessly integrate data from various sources, providing richer and more precise input information for the model. Through these improvements, we expect to enhance the model and algorithm's effectiveness in solving practical problems such as tugboat scheduling.

Author Contributions: Conceptualization, J.L.; methodology, X.D. and Z.X.; software J.L.; validation, X.D. and P.Y.; formal analysis, J.L.; investigation, J.L. and Z.X.; resources, J.L. and Z.X.; data curation, X.D.; writing—original draft preparation, J.L.; writing—review and editing, J.L. and X.D.; visualization, Z.X. and P.Y.; supervision, X.D.; project administration, P.Y.; funding acquisition, J.L. and P.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fujian Provincial Natural Science Foundation (Project No. 2019J01325) and the School of Navigation of Jimei University (Project No. HHXY2020004).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

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

References

1. Zhao, S.; Duan, J.; Li, D.; Yang, H. Vessel Scheduling and Bunker Management with Speed Deviations for Liner Shipping in the Presence of Collaborative Agreements. *IEEE Access* **2022**, *10*, 107669–107684. [CrossRef]
2. Notteboom, T.E.; Vernimmen, B. The effect of high fuel costs on liner service configuration in container shipping. *J. Transp. Geogr.* **2008**, *17*, 325–337. [CrossRef]
3. Wang, H.; Zhou, P.; Liang, Y.; Jeong, B.; Mesbahi, A. Optimization of tugboat propulsion system configurations: A holistic life cycle assessment case study. *J. Clean. Prod.* **2020**, *259*, 120903. [CrossRef]
4. Zhu, J.; Chen, L.; Wang, B.; Xia, L. Optimal design of a hybrid electric propulsive system for an anchor handling tug supply vessel. *Appl. Energy* **2018**, *226*, 423–436. [CrossRef]
5. Chen, Z.S.; Lam, J.S.L. Life cycle assessment of diesel and hydrogen power systems in tugboats. *Transp. Res. Part D Transp. Environ.* **2022**, *103*, 103192. [CrossRef]
6. Liu, Z. Hybrid Evolutionary Strategy Optimization for Port Tugboat Operation Scheduling. In Proceedings of the 2009 Third International Symposium on Intelligent Information Technology Application, Nanchang, China, 21–22 November 2009; pp. 511–515. [CrossRef]
7. Wang, S.; Kaku, I.; Chen, G.Y.; Zhu, M. Research on the modeling of tugboat assignment problem in container terminal. *Adv. Mater. Res.* **2012**, *433*, 1957–1961. [CrossRef]
8. Ilati, G.; Sheikholeslami, A.; Hassannayebi, E. A Simulation-based optimization approach for integrated port resource allocation problem. *PROMET-Traffic Transp.* **2014**, *26*, 243–255. [CrossRef]
9. Yang, Z.-Y.; Cao, X.; Xu, R.-Z.; Hong, W.-C.; Sun, S.-L. Applications of chaotic quantum adaptive satin bower bird optimizer algorithm in berth-tugboat-quay crane allocation optimization. *Expert Syst. Appl.* **2024**, *237*, 121471. [CrossRef]
10. Wang, X.; Liang, Y.; Wei, X.; Chew, E.P. An adaptive large neighborhood search algorithm for the tugboat scheduling problem. *Comput. Ind. Eng.* **2023**, *177*, 109039. [CrossRef]
11. Zhong, H.; Zhang, Y.; Gu, Y. A Bi-objective green tugboat scheduling problem with the tidal port time windows. *Transp. Res. Part D Transp. Environ.* **2022**, *110*, 103409. [CrossRef]
12. Wei, X.; Jia, S.; Meng, Q.; Tan, K.C. Tugboat scheduling for container ports. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *142*, 102071. [CrossRef]
13. Kasm, O.A.; Diabat, A.; Ozbay, K. Vessel scheduling under different tugboat allocation policies. *Comput. Ind. Eng.* **2023**, *177*, 108902. [CrossRef]
14. Kang, L.; Meng, Q.; Tan, K.C. Tugboat scheduling under ship arrival and tugging process time uncertainty. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *144*, 102125. [CrossRef]
15. Hao, L.; Jin, J.G.; Zhao, K. Joint scheduling of barges and tugboats for river–sea intermodal transport. *Transp. Res. Part E Logist. Transp. Rev.* **2023**, *173*, 103097. [CrossRef]
16. Jia, S.; Li, S.; Lin, X.; Chen, X. Scheduling tugboats in a seaport. *Transp. Sci.* **2021**, *55*, 1370–1391. [CrossRef]
17. Morariu, C.; Morariu, O.; Răileanu, S.; Borangiu, T. Machine learning for predictive scheduling and resource allocation in large scale manufacturing systems. *Comput. Ind.* **2020**, *120*, 103244. [CrossRef]
18. Liu, Z.; Wang, Y.; Liang, X.; Ma, Y.; Feng, Y.; Cheng, G.; Liu, Z. A graph neural networks-based deep Q-learning approach for job shop scheduling problems in traffic management. *Inf. Sci.* **2022**, *607*, 1211–1223. [CrossRef]
19. Zonta, T.; da Costa, C.A.; Zeiser, F.A.; Ramos, G.d.O.; Kunst, R.; Righi, R.d.R. A predictive maintenance model for optimizing production schedule using deep neural networks. *J. Manuf. Syst.* **2022**, *62*, 450–462. [CrossRef]
20. Wang, L.; Pan, Z.; Wang, J. A review of reinforcement learning based intelligent optimization for manufacturing scheduling. *Complex Syst. Model. Simul.* **2021**, *1*, 257–270. [CrossRef]
21. Liu, R.; Piplani, R.; Toro, C. A deep multi-agent reinforcement learning approach to solve dynamic job shop scheduling problem. *Comput. Oper. Res.* **2023**, *159*, 106294. [CrossRef]
22. Zou, Y.; Yin, H.; Zheng, Y.; Dressler, F. Multi-agent reinforcement learning enabled link scheduling for next generation Internet of Things. *Comput. Commun.* **2023**, *205*, 35–44. [CrossRef]
23. Ziaei, F.; Ranjbar, M. A reinforcement learning algorithm for scheduling parallel processors with identical speedup functions. *Mach. Learn. Appl.* **2023**, *13*, 100485. [CrossRef]
24. Zhang, N.; Shen, Y.; Du, Y.; Chen, L.; Zhang, X. Counterfactual-attention multi-agent reinforcement learning for joint condition-based maintenance and production scheduling. *J. Manuf. Syst.* **2023**, *71*, 70–81. [CrossRef]
25. Li, R.; Zhang, X.; Jiang, L.; Yang, Z.; Guo, W. An adaptive heuristic algorithm based on reinforcement learning for ship scheduling optimization problem. *Ocean Coast. Manag.* **2022**, *230*, 106375. [CrossRef]
26. Drungilas, D.; Kurmis, M.; Senulis, A.; Lukosius, Z.; Andziulis, A.; Januteniene, J.; Bogdevicius, M.; Jankunas, V.; Voznak, M. Deep reinforcement learning based optimization of automated guided vehicle time and energy consumption in a container terminal. *Alex. Eng. J.* **2023**, *67*, 397–407. [CrossRef]

27. Chen, X.; Liu, S.; Zhao, J.; Wu, H.; Xian, J.; Montewka, J. Autonomous port management based AGV path planning and optimization via an ensemble reinforcement learning framework. *Ocean Coast. Manag.* **2024**, *251*, 107087. [CrossRef]
28. Jin, J.; Cui, T.; Bai, R.; Qu, R. Container port truck dispatching optimization using Real2Sim based deep reinforcement learning. *Eur. J. Oper. Res.* **2024**, *315*, 161–175. [CrossRef]
29. Tofighi, S.; Torabi, S.; Mansouri, S. Humanitarian logistics network design under mixed uncertainty. *Eur. J. Oper. Res.* **2016**, *250*, 239–250. [CrossRef]
30. Zheng, X.; Liang, C.; Wang, Y.; Shi, J.; Lim, G. Multi-AGV Dynamic Scheduling in an Automated Container Terminal: A Deep Reinforcement Learning Approach. *Mathematics* **2022**, *10*, 4575. [CrossRef]
31. Bachiri, K.; Yahyaouy, A.; Gualous, H.; Malek, M.; Bennani, Y.; Makany, P.; Rogovschi, N. Multi-Agent DDPG Based Electric Vehicles Charging Station Recommendation. *Energies* **2023**, *16*, 6067. [CrossRef]
32. Jiang, X.; Zhong, M.; Shi, G.; Li, W.; Sui, Y. Vessel scheduling model with resource restriction considerations for restricted channel in ports. *Comput. Ind. Eng.* **2023**, *177*, 109034. [CrossRef]
33. Sha, Z.; Huo, R.; Sun, C.; Wang, S.; Huang, T. A Task-Oriented Hybrid Routing Approach based on Deep Deterministic Policy Gradient. *Comput. Commun.* **2023**, *210*, 183–193. [CrossRef]
34. Liu, Y.; Liang, H.; Xiao, Y.; Zhang, H.; Zhang, J.; Zhang, L.; Wang, L. Logistics-involved service composition in a dynamic cloud manufacturing environment: A DDPG-based approach. *Robot. Comput.-Integr. Manuf.* **2022**, *76*, 102323. [CrossRef]
35. Liu, G.; Chen, G.; Huang, V. Policy ensemble gradient for continuous control problems in deep reinforcement learning. *Neurocomputing* **2023**, *548*, 126381. [CrossRef]
36. Park, H.; Choi, D.G.; Min, D. Adaptive inventory replenishment using structured reinforcement learning by exploiting a policy structure. *Int. J. Prod. Econ.* **2023**, *266*, 109029. [CrossRef]
37. Zhu, M.; Tian, K.; Wen, Y.Q.; Cao, J.N.; Huang, L. Improved PER-DDPG based nonparametric modeling of ship dynamics with uncertainty. *Ocean. Eng.* **2023**, *286 Pt 1*, 115513. [CrossRef]
38. Cai, Z.; Lee, F.; Hu, C.; Kotani, K.; Chen, Q. NAEM: Noisy Attention Exploration Module for Deep Reinforcement Learning. *IEEE Access* **2021**, *9*, 154600–154611. [CrossRef]
39. Han, S.; Zhou, W.B.; Lu, J.Y.; Liu, J.; Lü, S. NROWAN-DQN: A stable noisy network with noise reduction and online weight adjustment for exploration. *Expert Syst. Appl.* **2022**, *203*, 117343. [CrossRef]
40. Ministry of Transportation and Communications. *Circular of the National Development and Reform Commission on the Revision and Issuance of the Measures for the Billing of Port Charges*; Ministry of Transportation and Communications: Beijing, China, 2019.