Journal of
Marine Science and Engineering

Article

Scheduling of AGVs in Automated Container Terminal Based on the Deep Deterministic Policy Gradient (DDPG) Using the Convolutional Neural Network (CNN)

Chun Chen, Zhi-Hua Hu * and Lei Wang

Logistics Research Center, Shanghai Maritime University, Shanghai 201306, China; 201930510010@stu.shmtu.edu.cn (C.C.); 202030510279@stu.shmtu.edu.cn (L.W.)
* Correspondence: zhhu@shmtu.edu.cn

Abstract: In order to improve the horizontal transportation efficiency of the terminal Automated Guided Vehicles (AGVs), it is necessary to focus on coordinating the time and space synchronization operation of the loading and unloading of equipment, the transportation of equipment during the operation, and the reduction in the completion time of the task. Traditional scheduling methods limited dynamic response capabilities and were not suitable for handling dynamic terminal operating environments. Therefore, this paper discusses how to use delivery task information and AGVs spatiotemporal information to dynamically schedule AGVs, minimizes the delay time of tasks and AGVs travel time, and proposes a deep reinforcement learning algorithm framework. The framework combines the benefits of real-time response and flexibility of the Convolutional Neural Network (CNN) and the Deep Deterministic Policy Gradient (DDPG) algorithm, and can dynamically adjust AGVs scheduling strategies according to the input spatiotemporal state information. In the framework, firstly, the AGVs scheduling process is defined as a Markov decision process, which analyzes the system's spatiotemporal state information in detail, introduces assignment heuristic rules, and rewards the reshaping mechanism in order to realize the decoupling of the model and the AGVs dynamic scheduling problem. Then, a multi-channel matrix is built to characterize space–time state information, the CNN is used to generalize and approximate the action value functions of different state information, and the DDPG algorithm is used to achieve the best AGV and container matching in the decision stage. The proposed model and algorithm frame are applied to experiments with different cases. The scheduling performance of the adaptive genetic algorithm and rolling horizon approach is compared. The results show that, compared with a single scheduling rule, the proposed algorithm improves the average performance of task completion time, task delay time, AGVs travel time and task delay rate by 15.63%, 56.16%, 16.36% and 30.22%, respectively; compared with AGA and RHPA, it reduces the tasks completion time by approximately 3.10% and 2.40%.

Keywords: automated container terminal; automated guided vehicles; dynamic scheduling; deep reinforcement learning

check for updates

Citation: Chen, C.; Hu, Z.-H.; Wang, L. Scheduling of AGVs in Automated Container Terminal Based on the Deep Deterministic Policy Gradient (DDPG) Using the Convolutional Neural Network (CNN). *J. Mar. Sci. Eng.* **2021**, *9*, 1439. https://doi.org/10.3390/jmse9121439

Academic Editors: Fausto Pedro Garcia Marquez, Mayorkinos Papaelias and Simone Marini

Received: 9 November 2021
Accepted: 1 December 2021
Published: 16 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the development of the international logistics industry, about 70% of the world's total trade volume is borne by ocean shipping. Over the past decade, the rapid growth of ship size and the rapid development of automated terminals maximized the throughput of container terminals in order to reduce the turnover time of container ships [1], which posed many new challenges to the planning of automated container terminals, and new methods are urgently needed to improve the service level and efficiency of automated terminals.

The automated terminal is a multi-level logistics operation process, which mainly includes Quay Cranes (QCs) operations, Yard Cranes (YCs) operations and the horizontal transportation of AGVs, in which AGVs are the key equipment connecting QCs operations and YCs operations, and run through the entire port's import and export container transshipment operations. The layout of the automated container terminal is shown in Figure 1. AGVs scheduling is one of the main parts of the AGVs control system in the automated terminal, which takes charge of the optimal matches between AGVs and each task in the most efficient way, while the work procedures between AGVs and other equipment are coupled and decisions are interdependent, making the process of logistics operations extremely complex [2]. To improve the throughput of automated terminals, the interaction between AGVs and crane equipment is required, which depends to a large extent on the synergy of equipment resources [3–6], it is necessary to determine the time for the container to be transported from one equipment to another. To be more specific, when AGV and QC process the container delivery, the two should cooperate with each other; if AGV fails to reach the designated delivery location within the scheduling time, QC will not work. Naturally, it delays the completion of a certain task.

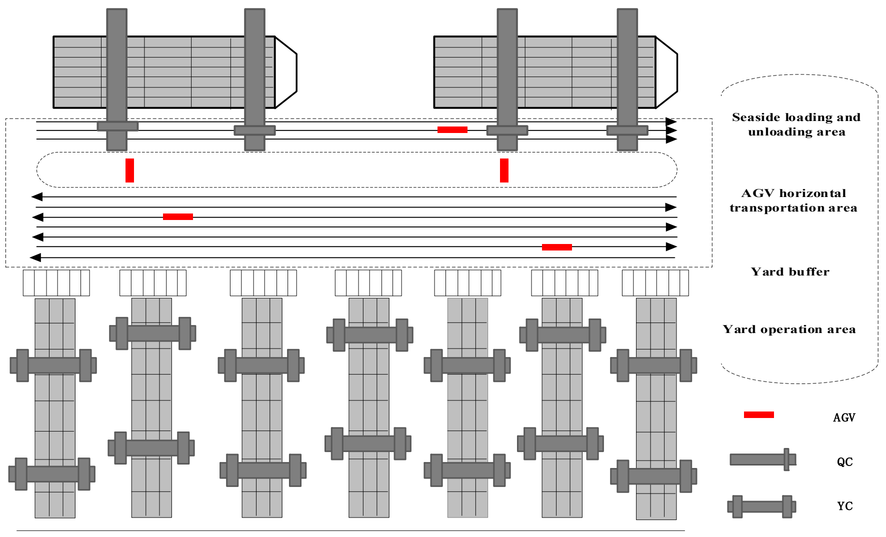
Seaside loading and unloading area
AGV horizontal transportation area
Yard buffer
Yard operation area
AGV
QC
YC

Figure 1. The layout of the automated container terminal is shown in the figure.

At present, the front of the block area of the automated terminal is equipped with a sufficient number of AGV mates as auxiliary equipment for the horizontal transportation of AGVs, helping to reduce the waiting time related to YCs and AGVs, while the seaside QCs are not equipped with auxiliary equipment such as AGV mates. QCs need to consider the delayed delivery time of the task during the loading and unloading of the container task. Since QCs are the bottleneck resource of the automated terminal, it is necessary to fully consider the resource utilization of QCs [7]. In order to realize the autonomous learning and sustainable development capabilities of the AGVs scheduling system in the automated terminal. Considering the task delivery delay time and the AGVs travel time, we combined CNN and DDPG based on the Actor–Critic (AC) framework proposed by Degris T. et al. [8] to build a Deep Convolution Deterministic Policy Gradient AGVs dynamic scheduling (CDA) algorithm framework for AGVs dynamic scheduling problems; experimental cases verify the reliability and effectiveness of the method.

The main contributions of this paper are summarized as follows: (1) the AGVs scheduling problem is defined as a sequence decision-making problem, modeled as a Markov decision-making process, the scheduling system. The scheduling environment dynamically and interactively makes decisions, and then updates the scheduling strategies through the deep reinforcement learning algorithm to realize the autonomous learning of the scheduling system. (2) The complex environment information of the terminal system information is stored in the multi-channel, two-dimensional matrix of the class diagram, and the complex CNN is created by multi-layer convolution, which effectively extracts the key information from the state information, and through this flexible combination ensures that the deep

CNN has a sufficient expression ability and generalization ability. (3) The solution to the problem combines convolutional neural networks and reinforcement learning algorithms, and proposes a Deep Convolution Deterministic Policy Gradient AGVs dynamic scheduling (CDA) algorithm framework, which dynamically selects heuristic rules to determine the highest priority container task, and flexibly assigns the task to a AGV. (4) Due to the time difference between the execution of the action and the observation of the impact of the action on the return, a reward reshaping mechanism is proposed to alleviate the excessive instability of the environment caused by the time difference, and adjust the overall impact of the strategy on the scheduling goal.

Most studies use modeling optimization under a strict problem definition; problem definitions and modeling optimization methods are different for different automated terminal scenarios. As a data-driven intelligent control method, DRL breaks through the strict modeling optimization method, which can be applied to all operation scenarios and improve the generalization ability of the method to solve the AGVs scheduling problem. The system information in automated terminal logistics and transportation systems is uncertain and will change over time, and the AGVs scheduling in such dynamic scenarios usually requires fast and efficient algorithms. The rule-based scheduling approach has a low time complexity and the ability to react to dynamic changes rapidly; therefore, it is very suitable for solving scheduling problems in a dynamic environment with a high complexity, as in real production. The combination of the DRL method and rule-based scheduling method has a strong generalization capability, in addition to a strong optimization capability that has theoretical research value. The artificial intelligence approach empowers the logistics transportation system in the automated terminal, automatically acquiring knowledge and skills through computer or intelligent simulation system simulation learning, continuously improving the system performance to achieve a self-improvement of the system. This effectively alleviates the difficult problem of AGVs dynamic scheduling with the participation of multiple logistics equipment. In our computational experiments, on the one hand, we effectively alleviate the spatio-temporal asynchrony problem of the task handover between QCs and AGVs, and greatly reduce the task delay time and task delay rate. On the other hand, through reasonable tasks allocation and effective integration of resources, we improve the operational efficiency of the horizontal transportation of AGVs, which helps to improve the throughput rate of automated terminal. It has some reference significance for the optimization of AGVs dynamic scheduling in a real automated terminal.

The rest of the paper is organized as follows: Section 2 reviews the relevant literature. Section 3 defines the important elements of the Markov decision process and develops the AGVs scheduling model. Section 4 describes the CDA algorithm in detail, including the network structure of the CDA algorithm and the update process. In Section 5, the effectiveness of the algorithm is verified by numerical simulation experiments, and the results of AGA and RHPA are compared with the CDA algorithm under different case experiments.

2. Literature Review

Over the past few decades, many scholars made great contributions to the research and development of port dispatching, including many studies on the dispatching of terminal AGVs. The core of dispatching lies mainly in the problem model and the solution algorithm, both of which are modeled and optimized based on the assumptions determined by the port operating environment. Rashidi H. et al. [9] established an AGVs scheduling model that minimizes the cost flow of automated terminals, and compared and analyzed the extended network simplicity algorithm (NSA+) and the incomplete greedy vehicle search algorithm (GVS). The results show that these two algorithms complement each other in problem-solving effects, and the NSA algorithm can be used to solve large-scale problems. Grunow M. et al. [10] studied the multi-load AGVs scheduling problem, proposed a complex offline heuristic strategy, and used the extended simulation model to evaluate the performance of the scheduling strategy. The simulation results showed that, compared with the online heuristic algorithm, the offline heuristic algorithm greatly improves the utilization of terminal AGVs. Taking the weighted sum minimization of the QCs delay time and the AGVs no-load travel distance as the joint optimization goal of AGVs scheduling, Kim K H. et al. [5] established a mixed-integer programming model for AGVs optimal task allocation, and proposed a heuristic algorithm that solves the problem, analyzing and comparing the performance of the heuristic algorithm with other scheduling rules. In a deterministic environment and a random environment, the algorithm achieves the joint optimization goal of terminal AGVs scheduling. Pjevčević D. et al. [11] analyzed in depth the influence of the number of AGVs and their dispatching rules on the process of container unloading at the terminal, established an AGVs simulation model and efficiency evaluation system, and proposed an efficient data envelopment analysis (DEA) for processing the strategic decision-making method, by setting a reasonable number of AGVs and selecting appropriate AGVs dispatching rules, which improved the efficiency of the terminal's operation in the unloading mode. Skinner B. et al. [12] considered the automated terminal container transportation scheduling problem, established a modified mathematical model, and proposed a two-part chromosome that improved the genetic algorithm to solve the problem. In simulation experiments of different scheduling scenarios, the improved genetic algorithm and the sequential operation scheduling method are evaluated and compared, and the method is applied to the Brisbane container terminal in Australia. The above problem only considers a single planning problem, which is only a partial optimization of the automated terminal. The automated terminal is a whole in which multiple loading and unloading equipment influences and restricts each other, the operation process of the automated terminal needs to be optimized from a global perspective. Iris, Ç. et al. [6] proposed a new flexible ship loading problem (FSLP), which comprehensively considered the management of loading operations and scheduling of transport vehicles. Various modelling enhancements and a mathematical model to obtain a strong lower bound were designed and a heuristic algorithm was used to solve the problem. The results show that the proposed model and heuristic algorithm can effectively generate high-quality solutions, and can greatly save costs. Luo J. et al. [13] considered the integrated problem of AGVs scheduling and container storage allocation in the automated terminal in the mode of loading and unloading, and established a mixed-integer programming model to minimize ship berth time. The effectiveness of the model and evolutionary genetic algorithm is verified through a large number of numerical experiments, and the experimental results show that the method can effectively solve problems of different scales, and provides a good solution for terminal AGVs scheduling and container storage allocation problems. The results show that the improved genetic algorithm can provide effective solutions and improve the overall operating efficiency of the terminal. There is a strong correlation between modeling optimization and problem definition in this particular environment. Different terminal environments have different problem definitions, and the established models are different. The algorithm lacks generalization ability.

In the actual port operation environment, the AGVs scheduling problem usually faces diversified automation equipment and other unexpected working conditions, such as machine failures, network delays, etc., due to the complexity and dynamics of the port environment and system information over time, these make AGVs scheduling easily affected by random disturbances. Therefore, it is necessary to increase the speed of the algorithm solution to meet the requirements of the complex and changeable port environment for processing timeliness. The general the heuristic search algorithm easily falls into the local optimal and the time cost to solve it is high. In order to adapt to the highly dynamic and complex port environment and improve the flexibility of AGVs scheduling, many scholars on AGVs dynamic scheduling problems carried out various studies, most of them put forward the concept of rescheduling. Angeloudis P. et al. [14] proposed A rolling horizon optimization algorithm based on the concept of cost or benefit. The results show that this method can minimize the maximum completion time and improve the efficiency of the terminal AGVs horizontal transportation operation. Klerides E. et al. [15] established a model to minimize the cost of AGVs scheduling and proposed a rolling time method. Through the analysis of port cases of different scales, the method of rolling time method was verified. The rescheduling strategy of dynamic decision-making according to the different states of the system can make AGVs quickly adapt to a complex and dynamic working environment. Cai B. et al. [16] proposed two rescheduling strategies for the Autonomous Straddle Carriers (ASC) scheduling problem, which take advantage of the cost of autonomous straddle carrier transportation, the cost of waiting time, and the delay of high-priority tasks. The weighted sum of the cost is the optimization goal. The branch-and-bound algorithm with column generation is used to solve the newly arrived task rescheduling strategy and the unexecuted task rescheduling combination strategy; the usage scenario for both strategies is compared through simulation experiments. In addition, some scholars made improvements based on the static AGVs scheduling algorithm and studied a variety of dynamic scheduling algorithms. Xin J. et al. [17] studied the dynamic scheduling problem of the automated collision-free path of terminal AGVs; introduced the concept of hierarchical control architecture to human–computer interaction scheduling and automatic guided vehicle path planning; and proposed a neighborhood variable-search, meta-heuristic algorithm based on hierarchical control structure, which could be applied to simulated static obstacle and dynamic obstacle scheduling scenarios. The results show that the hierarchical control system algorithm ensures the collision-free horizontal transportation of AGV and improves the operation efficiency of the automated terminal. Kim J. et al. [18] proposed a multi-standard AGVs scheduling strategy. At the same time, the QCs delay time minimization and the AGVs empty travel minimization were considered as objective functions. A mixed-integer programming model was established, and solved the above problems step-by-step through the use of a multi-objective evolutionary derivative algorithm, and thus the goal of AGVs dynamic scheduling was achieved. The application of machine learning algorithms to dynamic scheduling problems became a popular topic in the study of combinatorial optimization problems. Machine learning algorithms learn the optimal strategy for scheduling through simulation experiments or real data, so that they can adapt to different job conditions and scheduling environments, and have the ability to resist interference.

In recent years, with the development of Artificial Intelligence, the Internet of Things, Big Data, and other technologies, many scholars have begun to pay attention to how to improve the dynamic response speed of automated terminal scheduling and promote the autonomous learning of the scheduling system to solve the dynamic problems of the automated terminal operating environment. Han B A. et al. [19] pointed out that adaptive scheduling can select the optimal scheduling strategy according to the optimization goal and system status information in the dynamic production environment, and the scheduling strategy can be regarded as a function of system state information to scheduling operation, which reduces the calculation time of the scheduling process and ensures the speed of a dynamic response. In the acquisition of scheduling strategy, some scholars conducted corresponding work. Based on the inventory management model, Briskorn D. et al. [20] converted the AGVs dynamic scheduling problem into a dynamic allocation transportation task problem; the task allocation strategy was obtained with the greedy limited rule heuristic algorithm and the precise algorithm. A large number of comparative experiments shows that the model and algorithm have a robust performance, which can reduce the empty travel time of the AGVs, thereby improving the horizontal transportation efficiency of the automated terminal. Focusing on the dynamic AGVs scheduling problem against the background of the uncertainty of the automated terminal, and to minimize the QCs operating time and the AGVs empty travel distance, Choe R. et al. [21] proposed a paired preference function and an online preference learning algorithm combined with a deep neural network. A large number of simulation experiments verify that the method can dynamically adjust the AGVs scheduling strategy according to the actual operating conditions of the automated terminal. The task of reinforcement learning is to dynamically learn the optimal strategy by observing the rewards of the action feedback after performing a series of actions. To reduce the average waiting time of trucks, Fotuhi F. et al. [22] proposed an agent-based YCs scheduling model and q-learning algorithm, and the optimal YCs scheduling strategy is learned through a large number of simulation experiments. The experiment further verified the applicability and robustness of the model and q-learning algorithm to the YCs scheduling problem. An increasing number of scholars focused on Deep Reinforcement Learning (DRL) to solve practical problems, Hu H. et al. [23] proposed an adaptive, deep reinforcement learning, hybrid-rule, AGVs dynamic scheduling method to address the dynamics and uncertainties in material handling in flexible workshops. Experiments verified the feasibility and effectiveness of the method. For the dynamic and stochastic nature of order dispatching in ride-sharing platforms, Tang X. et al. [24] proposed an order dispatching solution based on deep reinforcement learning, and verified the effectiveness of the algorithm through large-scale online tests. In addition, the application of DRL to network flow control problems [25], financial market intraday trading [26], subway train dispatching [27], etc. proved the superiority and effectiveness of DRL in solving sequence decision-making.

In summary, for the AGVs dynamic scheduling optimization method in the automated terminal, the main focus is on periodic static dispatch or rescheduling, dynamics are not fully considered. Therefore, these scheduling methods have certain limitations for the complex port environment. The research of a few AGVs dynamic scheduling problems is combined with machine learning algorithms, and the performance and learning efficiency of the algorithms used are limited by the scale of the tasks that are to be solved. In the automated terminal, as of yet, no scholars applied a deep reinforcement learning algorithm to solve the dynamic scheduling problem. The DRL algorithm framework is applied to AGVs dynamic scheduling for the first time in this paper.

3. Mathematical Model

3.1. Problem Description

Horizontal transportation is an important part of the terminal operating system, which is the link connecting QCs and YCs. In order to improve terminal operating efficiency and reduce terminal operating costs, this paper adopts the dual-cycle mode of simultaneous loading and unloading [28], in which the AGVs dynamic scheduling process is divided into the following: the scheduling system assigns the container to AGV, AGV travels to the loading and unloading point at the QC, AGV picks up or unloads the container, transports the container, and the container is then unloaded or picked at the front buffer area of the yard. As an example of import container unloading operations, first, the scheduling system obtains the current information of all containers and the AGVs space-time information, and allocates the container to be executed to the AGV. Then the AGV arrives at the designated QC location for container handover. Finally, the container is transported by the AGV to the block of yard, where the empty AGV can accept the container assignment of the dispatch system again. The dynamic scheduling process of a single AGV is shown in Figure 2. AGV's operation time at the QC location should be as close as possible to QC's loading and unloading time, reducing the delay time of container handover. In order to better describe whether there is a delay in the handover process of containers, the earliest possible event time of each container task is set [5]. If the containers fail to be executed by the AGVs before the earliest possible event time, the handover of the container is delayed.

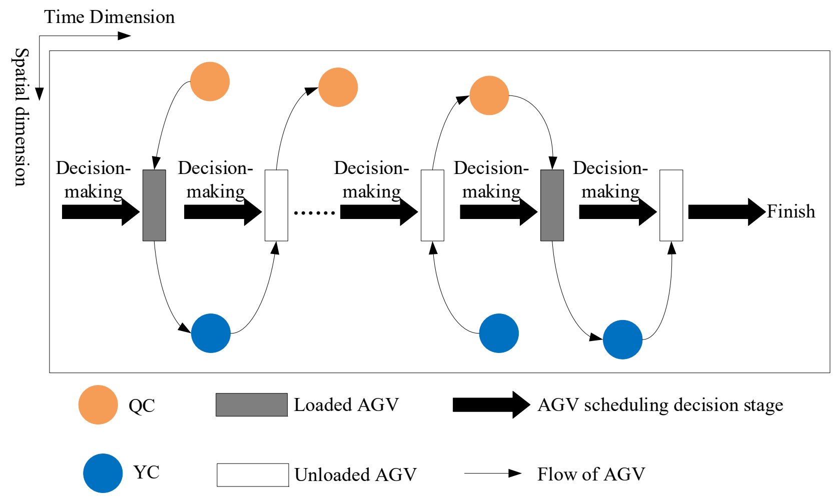

Figure 2. The dynamic scheduling process of a single AGV.

3.2. Notation of the Global Parameters

We introduce the following AGVs scheduling-related model notations as shown in Tables 1 and 2.

Table 1. Parameter notations.

Parameters	Notations
V	Set of AGVs, indexed by $v \in V$.
Q	Set of QCs, indexed by $q \in Q$.
Y	Set of YCs, indexed by b ∈ Y.
N_q	Set of containers for QC q, indexed by $i \in N_q$.
N	Set of all containers, $N = \cup N_q$.
el_i	The earliest possible event time container i.
G	Automated terminal horizontal transportation map of path network.
$Dis(.,.)$	The function of time for each container transported by AGV.
t_i^q	Service time of QC q loading or unloading container i.
t_i^b	Service time of YC b loading or unloading container i.
T	Time interval $T = [0, t_m]$, t_m is the final completion time of the task.
MR	Set of heuristic container allocation rule, $MR = \{mr_1, \ldots, mr_{18}\}$.
Δ	Action space.
S	State space.
$Dim(.)$	Representation of the dimensions of the matrix.
lr	Learning rate of Actor network and Critic network.
M	Capacity of experience replay memory.
BS	Batch size.
it_{max}	Maximum number of episodes.
C_u	Target network parameter update frequency.
l_s	Algorithm training time step.
τ	Target network parameter soft update coefficient.
γ	The discount coefficient of accumulate reward.
β	The scaling factor in the reward function.

Table 2. Variable notations.

Variables	Notations
K	Set of scheduling decision stages, indexed by $k \in K$.
s_k	State variable represents the environment information in decision stage k, $s_k \in S$.
s_k^1	The component of the state variable represents the information of containers.
s_k^2	The component of the state variable represents the AGV information.
t_k	Time of decision stage k.
u_{ik}	The urgency of container i in the decision stage k.
$\pi(\Delta\|s_k)$	The decision strategy to determine the probability of each action based on s_k.
Δ_k	The action in the decision stage k.
$s_{k'}$	The state variable of next decision stage.
$t_{k'}$	Time of decision stage k', which is also the completion time of the individual container task.
t_{vk}^{qi}	The actual time that container i is handed over between AGV v and QC q.
D_{vk}^{qi}	The delay time of container i transported by AGV v.
C_{ivk}	The travel time of container i transported by AGV v.
r_k	The reward value of the state variable transition from s_k to $s_{k'}$.
π^*	The optimal strategy.
D_{av}	Average delay time of all containers.
C_{av}	Average travel time per container for AGVs transportation.
N_r	Number of containers delayed.

3.3. AGVs Dynamic Scheduling Model

The dynamic scheduling of AGVs in the automated terminal can be described as a staged container–AGV matching problem. The entire container–AGV matching process is discretely partitioned in time, thus transforming the scheduling problem into a finite stochastic dynamic decision process, which is then modeled as a Markov decision process (MDP). The scheduling system interacts intermittently with the environment in all decision stages, and then dynamically assigns the highest priority container to the available AGV for execution until all tasks are completed. Such a complete process is called an episode. At each decision stage k, the scheduling system senses the environment state variable s_k, the scheduling system makes a corresponding action Δ_k in response to the interaction state variable s_k based on the policy $\pi(\Delta_k|s_k)$. At this time, the container and the available AGV achieve a successful matching. After the current container transportation is over, the environment generates a numerical reward r_k as effective feedback for the decision. According to the AGVs scheduling process described above, we define the state variables, action rules, reward functions, and policy definitions.

3.3.1. System State Information

The local and global characteristic information of the system is described by defining state variables as s_k in the decision-making stage k. State variables are the key information for the scheduling system to make a decision, and this decision directly affects the efficiency of the entire system. Therefore, extracting system state information is vital in optimizing the entire scheduling process. AGVs scheduling in the automated terminal involves multiple equipment, determining the amount of information involved in the characterization system. Data are obtained through various data collection equipment, and then multiple sources of information are combined to accurately determine the current system state information. This article characterizes the system state information from two perspectives of the sequence information of QCs containers and AGVs information, which are represented by s_k^1 and s_k^2, respectively, the final state variable of the decision stage is $s_k = [s_k^1, s_k^2]$.

For the QCs containers, sequence information s_k^1 is represented by a four-channel matrix, which describes the global features of the environment, storing different categories of feature information in different channels. The dimension of s_k^1 can be expressed as $Dim(s_k^1) = 4 \times |Q| \times max\{|N_q||q \in Q\}$; the height and width of each channel are set to the number of QCs and the maximun number of tasks of single QC, respectively. Learning lessons from Kim J. et al. [18] in solving the AGVs scheduling problem, we select the characteristic attributes used to describe the state component s_k^1. The details of s_k^1 are as follows:

(1) The channel of container types is used to indicate the types of containers. In the dual-cycle mode of loading and unloading, the execution of different types of containers by the AGVs affects the synchronization of the operation of the QCs and the AGVs. For example, when the AGVs transport the importing containers, they directly travel to the QCs to pick up the container, this process only includes the empty travel time of the AGVs. When the AGVs transport and export the container, they first travel to the front of the YCs to pick up container, and then travel to the QCs to unload the container, the process includes two processing stages, so it takes longer for AGVs to reach QCs. The values 0 and 1 are used to denote the import and export boxes, respectively.

(2) The channel for container urgency indicates the urgency of each container. In order to ensure the synchronization of the QCs in loading and unloading each container with the AGVs, and to reduce the delay time of containers handover, the urgency of a container can be expressed as the remaining time until the earliest possible event time. Once the container is transported, the urgency of the container is set to 0; otherwise, the urgency of the unexecuted container u_{ik} can be expressed as:

$$u_{ik} = el_i - t_k,\ i \in N, k \in K \tag{1}$$

(3) The estimated load travel time channel records the estimated load travel time for each container not transported by the AGV. The load travel time for the AGV to execute the container is estimated using an automated terminal horizontal transportation map of path network G, according to the container loading and unloading location. The estimated load travel time is set to 0 if the container is assigned to the AGV for execution.

(4) The completion time channel records the actual time that each container is completed, initialized to an all-0 matrix.

The AGVs scheduling process is driven not only by the containers' information but also the spatio-temporal information of the AGVs themselves. Since the information of the AGVs is constantly changing in time and space, only the local state information of the AGVs can be captured. The AGVs spatio-temporal state information s_k^2 is mapped as a two-dimensional matrix with dimensions $Dim\left(s_k^2\right) = |V| \times (1 + |Q| + |Y|)$, initialized to 0. The first column of the matrix specifies the working status of the AGV, "0" means the AGV is idle, "1" means the AGV is assigned a container; the remaining columns indicate the travel time from the current position of the AGV to each QC q position, and to each YC b position, respectively, and the specific values can be obtained by querying the automated terminal horizontal transportation map of path network G.

3.3.2. Action Space Expression

In the process of AGVS dynamic scheduling, the scheduling system needs to plan a suitable container and assign a suitable AGVS for the container to execute. As ships become larger and the number of containers increases dramatically, it is difficult to accurately determine the containers that need to be transported within the limited decision-making time in dynamic scheduling. The heuristic assignment rule is a scheduling behavior of the system that schedules AGVs to transport containers, speeding up the dynamic response of the scheduling system and determining the order of container assignment in the scheduling process. The principle of validity and sufficiency is followed when determining the assignment rules. Validity is reflected in the influence of assignment rules on the convergence effect of the objective function, where different assignment rules determine the order of containers execution and directly affect the convergence degree of the objective function. Sufficiency refers to the diversity of the design of assignment rules, which can not only overcome the short-sightedness of a single assignment rule, but can also select appropriate rules based on different state information in different decision-making stages. Based on the above two principles, 18 heuristic assignment rules are designed in this paper with reference to the criterion proposed by Choe R. et al. [21], where rules 7–18 are hybrid assignment rules combined by two single assignment rules. The specific details of the 18 heuristic assignment rules are shown in Table 3.

Table 3. Specific details of the heuristic assignment rules.

Symbols	Rules	Description
mr_1	LTT	The container with the longest transport time from origin to destination is assigned to an AGV.
mr_2	STT	The container with the shortest transport time from origin to destination is assigned to an AGV.
mr_3	GUT	The container with the greatest urgency is assigned to an AGV.
mr_4	LUT	The container with the least urgency is assigned to an AGV.
mr_5	LPT	The container with the longest processing time is assigned to an AGV; the processing time includes the time for loading and unloading the container at QCS and YCS, as well as the transportation time from the origin to the destination of the container.
mr_6	SPT	The container with the shortest processing time is assigned to an AGV.
mr_7	LQ-LTT	Selecting the QC q with the most remaining containers, from which the container with the longest transport time is selected.
mr_8	LQ-STT	Selecting the QC q with the most remaining containers, from which the container with the shortest transport time is selected.
mr_9	SQ-LTT	Selecting the QC q with the fewest remaining containers, from which the container with the longest transport time is selected.
mr_{10}	SQ-STT	Selecting the QC q with the fewest remaining containers, from which the container with the shortest transport time is selected.
mr_{11}	LQ-GUT	Selecting the QC q with the most remaining containers, from which the container with the greatest urgency is assigned to an AGV.
mr_{12}	LQ-LUT	Selecting the QC q with the most remaining containers, from which the container with the least urgency is assigned to an AGV.
mr_{13}	SQ-GUT	Selecting the QC q with the fewest remaining containers, from which the container with the greatest urgency is assigned to an AGV.
mr_{14}	SQ-LUT	Selecting the QC q with the fewest remaining containers, from which the container with the least urgency is assigned to an AGV.
mr_{15}	LQ-LPT	Selecting the QC q with the most remaining containers, from which the container with the longest processing time is assigned to the an AGV.
mr_{16}	LQ-SPT	Selecting the QC q with the most remaining containers, from which the container with the shortest processing time is assigned to an AGV.
mr_{17}	SQ-LPT	Selecting the QC q with the fewest remaining containers, from which the container with the longest processing time is assigned to an AGV.
mr_{18}	SQ-SPT	Selecting the QC q with the fewest remaining containers, from which the container with the shortest processing time is assigned to an AGV.

The type of AGV determines the AGVs to be assigned to the selected container, and each AGV has a unique ID. In this paper, the action space consists of heuristic assignment rules and types of AGV. The action space can be represented as $\Delta = \{(mr, v) | mr \in MR, v \in V\}$.

3.3.3. Reward Design and Reshaping

At each decision stage k, the scheduling system obtains the state variable s_k and makes a decision $X^{\pi}(s_k)$, and the corresponding container is assigned to the AGV. After the AGV completes the container task, it needs a measurement standard to measure the task completion effect. Therefore, the two optimization goals of minimizing the delay time of the tasks and the travel time of the AGVs are fully considered, and the reward function to calculate the reward value of the feedback is designed, which is then used for action evaluation and strategy optimization. The end of each container task is transported by AGV, the actual time t_{vk}^{qi} that container i is handed over between AGV v and QC q, and actual completion time of each container $t_{k'}$. The concepts of individual container task delay time cost and AGV travel time cost are introduced based on the optimization objectives of task delay time and single AGV travel time, as shown in Equations (2) and (3):

$$D_{vk}^{qi} = \max\{(t_{vk}^{qi} - el_i), 0\} \tag{2}$$

$$C_{ivk} = t_{k'} - t_k - t_i^b - t_i^q \tag{3}$$

Generally, a simple numerical summation of the above-defined cost can be used as the reward value of the feedback. However, when the AGV transports a container task, there is a time difference between the start of the task, and the reward is observed at the end of the task. Within this time difference, if there are other tasks that match the AGV, then these will change the successor state information $s_{k'}$, causing the entire system environment to become excessively unstable. Therefore, the reward reshaping mechanism proposed in this paper calculates the average delay cost D_{av} for all tasks and the average trip cost C_{av} for all AGVs for the whole process. A difference factor in the terms $D_{vk}^{qi} - D_{av}$ and $C_{ivk} - C_{av}$ between the cost of a single container task is introduced, and the average cost of the entire process, as well as the scaling factor of the difference factor, is proposed to reshape the reward function. The process of reward function reshaping is shown in Formulas (4)–(8):

$$D_{av} = \frac{1}{K}\sum_{k\in K} D_{vk}^{qi} \tag{4}$$

$$C_{av} = \frac{1}{K}\sum_{k\in K} C_{ivk} \tag{5}$$

$$D_{vk}^{qi} = D_{vk}^{qi} + \beta\left(D_{vk}^{qi} - D_{av}\right) \tag{6}$$

$$C_{ivk} = C_{ivk} + \beta(C_{ivk} - C_{av}) \tag{7}$$

$$r_k = e^{-D_{vk}^{qi}} + e^{-C_{ivk}} \tag{8}$$

3.3.4. Optimal Scheduling Strategy

The strategy $\pi(\Delta|s_k)$ is the probability of all actions in the action space under the condition of the state variable s_k in the decision state k. The scheduling system can select the container task to pair with AGV based on known action probabilities. Associated with the policy π is the action value function $Q^\pi(s_k, \Delta_k)$ that represents the expected cumulative discount reward after the execution of action Δ_k in state variable s_k using the policy π, with the formula shown in (9):

$$Q^\pi(S, \Delta) = E_\pi\{\sum_{i=k+1}^{K} \gamma^{i-k-1} r_i | s_k, \Delta_k\} \tag{9}$$

From the action value function of Formula (9), the Bellman equation under the general policy can be written, as shown in Formula (10):

$$Q^\pi(s_k, \Delta_k) = E_\pi\{r_k + \gamma Q^\pi(s_k', \Delta_{k'}) | s_k, \Delta_k\} \tag{10}$$

The basic idea of RL is to iteratively update the following Bellman equation to learn an optimal strategy π^* to maximize the expected cumulative discount reward. The cumulative reward under the optimal strategy is shown in Formula (11).

$$Q^{\pi^*}(s_k, \Delta_k) = maxE_{\pi^*}\{\textstyle\sum_{i=k+1}^{K} \gamma^{i-k-1} r_i | s_k, \Delta_k\} = maxQ^\pi(s_k, \Delta_k) \tag{11}$$

The AGVs dynamic scheduling problem is modeled as MDP. The ultimate goal is to find the optimal scheduling strategy π^* to obtain the maximum expected cumulative discount reward.

4. CDA Scheduling Algorithm

In this paper, the DDPG algorithm is used to achieve AGVs dynamic scheduling. In the previous section, we defined the large-scale discrete action space consisting of a combination of heuristic assignment rules and AGVs types. Based on the original DDPG algorithm, the discrete action space reparameterization trick [29] is introduced and the DDPG algorithm is slightly modified, so that the algorithm can better search for the optimal scheduling strategy π^* in the discrete action space. This method combines the deterministic policy gradient (DPG) and the deep CNN, which can be robustly learned. The DDPG algorithm is based on the network structure of the DQN algorithm and uses the fixed network technique to design the evaluation network structure and the target network structure to mitigate the instability of the target network update. The DDPG algorithm is based on the AC framework, and both of the above network architectures—two neural network approximators, the Actor network and the Critic network. In the estimation network, the Actor estimation network $\mu(s_k|\theta_\mu)$ is the policy function of the state variable mapping action; the Critic estimation network $Q(s_k, \Delta_k|\theta_Q)$ is the parameterized value function that approximates the $Q(s_k, \Delta_k)$ values of the state variable s_k and the action Δ_k given by the Actor estimation network, which is then used to evaluate the Actor estimation network the quality of the action Δ_k and guide the direction of the strategy π update [28], where θ_μ and θ_Q are the parameters of the Actor estimation network and the Critic estimation network, respectively. In the estimation network parameter updating process, the Actor target network $\mu(s_{k'}|\theta'_\mu)$ temporarily fixes the Actor estimation network parameters; and the Critic target network $Q(s_{k'}, \Delta_{k'}|\theta'_Q)$ temporarily fixes the Critic estimation network parameters to improve the stability and convergence of algorithm training, which are structurally consistent with the estimation networks of Actor and Critic, where θ'_μ and θ'_Q are the parameters of the Actor target network and the Critic target network, respectively.

4.1. CDA Algorithm Network Structure

Actor network and Critic network are function approximators, and the design of their network structure is very important for the nonlinear approximate estimation of the value function. Because the state information represents the port scheduling environment from multiple perspectives, the dimensions and distribution of data are different due to different data sources. In this paper, the state variable s_k, which represents the system information, consists of component s_k^1 and component s_k^2, and the data structures are a 4-channel 2D matrix and a single-channel 2D matrix, respectively. There are differences in the dimensions of the 2D matrix for different components, and it is not possible to fuse each component information directly from the channels. The CNN has a wide range of applications in image classification, image recognition, and video processing, etc. Therefore, this paper uses multi-layer CNN to combine simple patterns into complex patterns when designing the network structure. This flexible combination can extract the data correlation and ensure that the deep CNN has a sufficient expressive ability and generalization. A hierarchical processing idea is introduced to handle the different components, including s_k^1 and s_k^2 of state variable s_k. Two deep CNN network structures are used to extract the key feature information of each state component, and then the multi-dimensional key features are made one-dimensional to achieve multi-source information fusion. Deep CNN usually consists of three types of layers: convolution layers, pooling layers and fully connected layers, with the convolution and pooling layers being structurally contiguous. In the convolution layers, the local features of the matrix data are extracted by convolution operations. The main role of the pooling layer is to further reduce the number of parameters by subsampling the unimportant features after the convolution operation.

The convolution layers and the pooling layers are equivalent to feature engineering, while fully connected layers are equivalent to feature weighting, which acts as a "classifier" in the whole neural network. The details of the Actor network and Critic network structure are shown in Tables 4 and 5, respectively.

Table 4. Details of the Actor network structure.

Layers	Number of Neurons	Activation Functions	Descriptions
Input layer 1	$4 \times \lvert Q\rvert \times \max\{\lvert N_q\rvert\}$	None	Input of state component s_k^1.
Input layer 2	$\lvert V\rvert \times (1 + \lvert Q\rvert + \lvert B\rvert)$	None	Input of state component s_k^2.
Convolution layer 1	32	Swish	Extracting s_k^1 information.
Pooling layer 1	None	None	Extracting s_k^1 information.
Convolution layer 2	64	Swish	Further extraction of s_k^1 information.
Pooling layer 2	None	None	Further extraction of s_k^1 information.
Flatten layer 1	None	None	One-dimensionalization of s_k^1.
Convolution layer 3	64	Swish	Extracting s_k^2 information.
Pooling layer 3	None	None	Extracting s_k^2 information.
Flatten layer 2	None	None	One-dimensionalization of s_k^2.
Concatenation layer	None	None	Concatenation of s_k^1 and s_k^2.
Fully connected layer	100	Relu	None
Output layer 1	$\lvert V\rvert$	Softmax	Output AGV information.
Output layer 2	$\lvert MR\rvert$	Softmax	Output assignment rule information.
Gumbel-softmax layer	1	None	Matching of AGV with container.

Table 5. Details of the Critic network structure.

Layers	Number of Neurons	Activation Functions	Descriptions
Input layer 1	$4 \times \lvert Q\rvert \times \max\{\lvert N_q\rvert\}$	None	Input of state component s_k^1.
Input layer 2	$\lvert V\rvert \times (1 + \lvert Q\rvert + \lvert B\rvert)$	None	Input of state component s_k^2.
Input layer 2	2	None	Actor network output of action Δ_k.
Convolution layer 1	32	Swish	Extracting s_k^1 information.
Pooling layer 1	None	None	Extracting s_k^1 information.
Convolution layer 2	64	Swish	Further extraction of s_k^1 information.
Pooling layer 2	None	None	Further extraction of s_k^1 information.
Flatten layer 1	None	None	One-dimensionalization of s_k^1.
Convolution layer 3	64	Swish	Extracting s_k^2 information.
Pooling layer 3	None	None	Extracting s_k^2 information.
Flatten layer 2	None	None	One-dimensionalization of s_k^2.
Concatenation layer 1	None	None	Concatenation of s_k^1 and s_k^2.
Concatenation layer 2	None	None	Concatenation of s_k and Δ_k.
Fully connected layer 1	100	Relu	None
Fully connected layer 2	50	Relu	None
Output layer	1	Relu	$Q(s_k, \Delta_k \lvert \theta_Q)$ corresponding to states and actions.

4.2. Algorithm Update Process

The goal of the CDA algorithm is to optimize the policy $\pi \to \pi^*$ of AGVs scheduling by the non-approximated estimation of action value functions through deep neural networks. The algorithm update process contains two sub-processes, which are the scheduling process of AGVs and the training process of the algorithm. The update process of the algorithm is shown in Algorithm 1. The purpose of the AGVs scheduling process is to obtain a tuple of transfer sequences consisting of dynamic state information, action information, and reward information by interacting with the automated terminal environment to provide data support for the training process of the CDA algorithm. The scheduling process of AGVs can be generally described as follows: the scheduling system obtains the state variable s_k from the scheduling environment of the AGVs. Based on the results of state variable s_k processing by the Actor estimation network, the scheduling system determines the assignment rule and AGV with high priority. According to the above assignment rule, AGV is assigned to high-priority container tasks and transports the container to designated handover points (including QCs andYCs); after the implementation of the scheduling program, the scheduling system receives the feedback reward r_k and obtains the current state variable $s_{k'}$, and the transfer sequence tuple $[s_k, \Delta_k, r_k, s_{k'}]$ of the interaction between the system is stored in the experience replay memory. The above process corresponds to lines 3–6 of Algorithm 1 and the blue solid arrows in Figure 3.

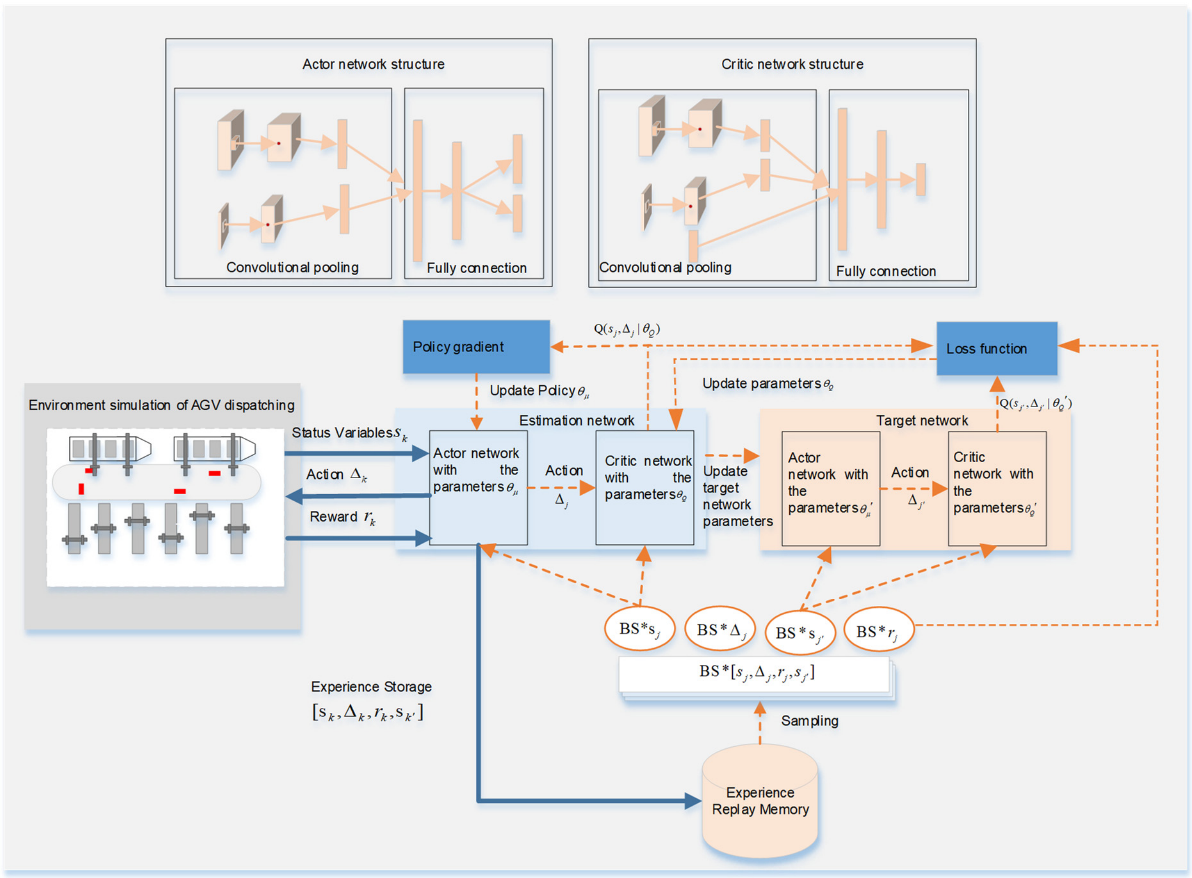
Actor network structure
Convolutional pooling
Fully connection
Critic network structure
Convolutional pooling
Fully connection
Policy gradient
$Q(s_j,\Delta_j|\theta_Q)$
Loss function
Update Policy θ_μ
Update parameters θ_Q
$Q(s_{j'},\Delta_{j'}|\theta_Q')$
Environment simulation of AGV dispatching
Status Variables s_k
Action Δ_k
Reward r_k
Estimation network
Actor network with the parameters θ_μ
Action Δ_j
Critic network with the parameters θ_Q
Update target network parameters
Target network
Actor network with the parameters θ_μ'
Action $\Delta_{j'}$
Critic network with the parameters θ_Q'
$BS*s_j$
$BS*\Delta_j$
$BS*s_{j'}$
$BS*r_j$
$BS*[s_j,\Delta_j,r_j,s_{j'}]$
Experience Storage
$[s_k,\Delta_k,r_k,s_{k'}]$
Sampling
Experience Replay Memory

Figure 3. Overview of the CDA algorithm.

The training process of the algorithm is performed every fixed time step l_s: samples of $BS*[s_j, \Delta_j, r_j, s_{j'}]$ are sampled uniformly and randomly from the experience replay memory, the state variables s_j and $s_{j'}$ are input to the Actor estimation network, and Actor target network, respectively. The actions Δ_j and $\Delta_{j'}$ are given based on the predicted values of the network. The Critic target network takes the state variable $s_{j'}$ and the action $\Delta_{j'}$ given by the Actor target network as an input to predict the state action value $Q(s_{j'}, \Delta_{j'}|\theta'_Q)$ and calculates the state action target value $Q(s_j, \Delta_j|\theta'_Q)$ through Equation (12):

$$Q(s_j, \Delta_j|\theta'_Q) = \begin{cases} r_j, & t_j < t_{max} \\ r_j + \gamma Q(s_{j'}, \Delta_{j'}|\theta'_Q), & t_j \geq t_{max} \end{cases} \quad (12)$$

The loss function $L(\theta_Q)$ is equal to the mean square error of the action value $Q(s_j, \Delta_j|\theta_\mu)$ predicted by the Critic estimation network and the state action target value $Q(s_j, \Delta_j|\theta'_Q)$

calculated by Equation (12), $L(\theta_Q)$ is shown in Equation (13). The Critic estimation network parameters θ_Q are updated by gradient back propagation:

$$L(\theta_Q) = \frac{1}{BS}\sum_{j=1}^{BS}\left(Q(s_j, \Delta_j, \left|\theta'_Q\right.) - Q(s_j, \Delta_j|\theta_Q)\right)^2 \tag{13}$$

The policy gradient $\nabla J(\theta_\mu)$ is the gradient of the state action value $(s_j, \Delta_j|\theta_\mu)$ of the Critic estimation network to action Δ_j, and is used to update the strategy parameters θ_μ of the Actor estimation network, causing the neural network select the action with the highest payoff or higher likelihood. The formula for the strategy gradient $\nabla J(\theta_\mu)$ is shown below. The following equation calculates the strategy gradient $\nabla J(\theta_\mu)$:

$$\nabla J(\theta_\mu) = -\frac{1}{BS}\sum_{j=1}^{BS}\left(\nabla_{\Delta_j} Q(s_j, \Delta_j|\theta_\mu)\nabla_{\theta_\mu}\Delta_j\right) \tag{14}$$

In order to ensure the stable convergence of the training process, under the condition of meeting the target network parameter update frequency C_u, the "soft" target update method is used to update the parameters of the Actor target network and Critic target network. The target network parameters are updated as shown in Equations (15) and (16):

$$\theta'_\mu = \tau\theta_\mu + (1-\tau)\theta'_\mu \tag{15}$$

$$\theta'_Q = \tau\theta_Q + (1-\tau)\theta'_Q \tag{16}$$

Lines 7–16 of Algorithm 1 describe the learning process of the algorithm and the process of updating network parameters in detail; the above processes correspond to the orange dashed arrow in Figure 3. Figure 4 illustrates the entire algorithm process more clearly in the form of a flowchart.

Algorithm 1. Solving AGV scheduling based on CDA algorithm.

```
Input: Hyperparameters $M, BS, \alpha, \beta, \gamma, \tau, lr, it_{max}, C_u, l_s$, the automated terminal horizontal transportation map of path network $G$, transport time function $Dis(.,.)$
Output: The optimal strategy $\pi^*$ is the optimal Actor estimation network parameter $\theta_\mu$
1: Initialize estimated network and target network parameters, including $\theta_\mu, \theta_Q, \theta'_\mu, \theta'_Q$
2: For episode: = 1 to $it_{max}$
3: Scheduling system obtains state information from the AGV scheduling terminal environment
4: The Actor estimation network gives the decision action $\Delta_k = \pi_{\theta_\mu}(s_k)$ based on the state variable $s_k$
5: After the AGV completes the container transportation, the scheduling system gets the new state variable $s_{k'}$ and reward $r_k$, and determines if the task is completed
6: Store the state transfer sequence $[s_j, \Delta_j, s_{j'}, r_j]$ into the experience replay memory
7: If $l_s$ meets the conditions, start training the network
8: For $j$: = 1 to $BS$
9: Sample $[s_j, \Delta_j, s_{j'}, r_j]$ from the experience replay memory, and calculate the state action target value $Q(s_j, \Delta_j|\theta'_Q)$
10: Calculate the mean square error loss function $L(\theta_Q)$ and update the Critic estimation network parameters $\theta_Q$ by the gradient descent algorithm
11: Use the backpropagation of the policy gradient $\nabla J(\theta_\mu)$ to update the Actor target network parameters $\theta_\mu$
12: End for
13: End if
14: If the condition of $C_u$ is met
15: Update the parameters $\theta'_\mu$ and $\theta'_Q$ using the "soft" target update method
16: End if
17: End for
```

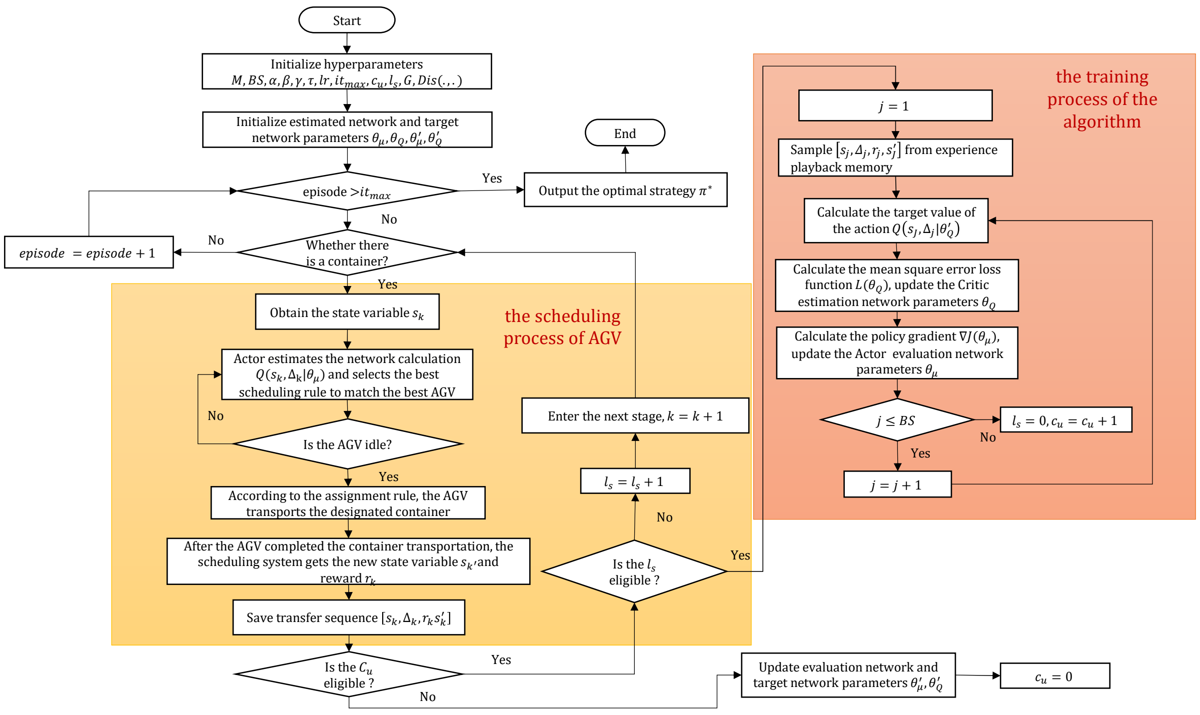
Start
Initialize hyperparameters $M, BS, \alpha, \beta, \gamma, \tau, lr, it_{max}, c_u, l_s, G, Dis(.,.)$
Initialize estimated network and target network parameters $\theta_\mu, \theta_Q, \theta'_\mu, \theta'_Q$
episode $> it_{max}$
Yes
Output the optimal strategy π^*
End
No
Whether there is a container?
No
$episode = episode + 1$
Yes
Obtain the state variable s_k
Actor estimates the network calculation $Q(s_k, \Delta_k|\theta_\mu)$ and selects the best scheduling rule to match the best AGV
No
Is the AGV idle?
Yes
According to the assignment rule, the AGV transports the designated container
After the AGV completed the container transportation, the scheduling system gets the new state variable $s_{k'}$ and reward r_k
Save transfer sequence $[s_k, \Delta_k, r_k s'_k]$
the scheduling process of AGV
Enter the next stage, $k = k + 1$
$l_s = l_s + 1$
No
Is the l_s eligible ?
Yes
Is the C_u eligible ?
Yes
No
Update evaluation network and target network parameters θ'_μ, θ'_Q
$c_u = 0$
the training process of the algorithm
$j = 1$
Sample $[s_j, \Delta_j, r_j, s'_j]$ from experience playback memory
Calculate the target value of the action $Q(s_j, \Delta_j|\theta'_Q)$
Calculate the mean square error loss function $L(\theta_Q)$, update the Critic estimation network parameters θ_Q
Calculate the policy gradient $\nabla J(\theta_\mu)$, update the Actor evaluation network parameters θ_μ
$j \le BS$
No
$l_s = 0, c_u = c_u + 1$
Yes
$j = j + 1$

Figure 4. Dynamic scheduling process of AGV based on CDA algorithm.

4.3. Implementation of AGVs Dynamic Scheduling

The implementation of AGVs dynamic scheduling using CDA algorithm is basically the same as the algorithm update process. However, at this time only the algorithm-trained optimal policy is needed to guide the scheduling system to choose the optimal heuristic assignment rules as well as the best AGV in different states, and the training of the CDA algorithm is no longer needed.

As shown in the Figure 5, a certain decision-making stage in an episode is taken as an example to visually illustrate the decision-making process and state variable transition process. The task situation is set in the current state as shown in the Table 6, and the loading and unloading time of 10 s for a task is assumed on both the sea side and the shore side. First, the current state variable s_k is calculated according to the definition of the state variable (Part 3 Section 3), marked with a light-yellow rectangular box. At this time, AGV 2 is idle, indicated as "0", and marked with a green filled rectangle; AGV 1 and AGV 3 are busy, indicated as "1", marked with a red filled rectangle. Then, the state variable s_k is used as the input to the optimal action estimation network. Under the premise that there are tasks, the idle AGV is specified and the task assignment rule is predicted. In this example, the action is $\Delta_k = (mr_{11}, AGV\ 2)$. According to the definition of the assignment rule mr_{11}, there are three tasks in the task sequence of QC 3 and the task numbered 3-1 is the most urgent; therefore, AGV 2 considers the task numbered 3-1 first. Finally, after AGV 2 completes the task numbered 3-1, the delay time and the travel time of the task numbered 3-1 transported by AGV 2 are calculated by Formulas (2) and (3), which are 15 s and 20 s, respectively. The state variable $s_{k'}$ at the end of the task numbered 3-1 is obtained by a calculation, which is marked by a light-blue rectangular box in the figure. The above task assignment process is repeated until all tasks are executed, and then the reshaping reward r_k is calculated based on Equations (4)–(8), and the transfer sequence tuple $[s_k, \Delta_k, r_k, s_{k'}]$ of each decision stage is stored in the experience replay memory for the training of the algorithm; the algorithm training process is shown in the previous section.

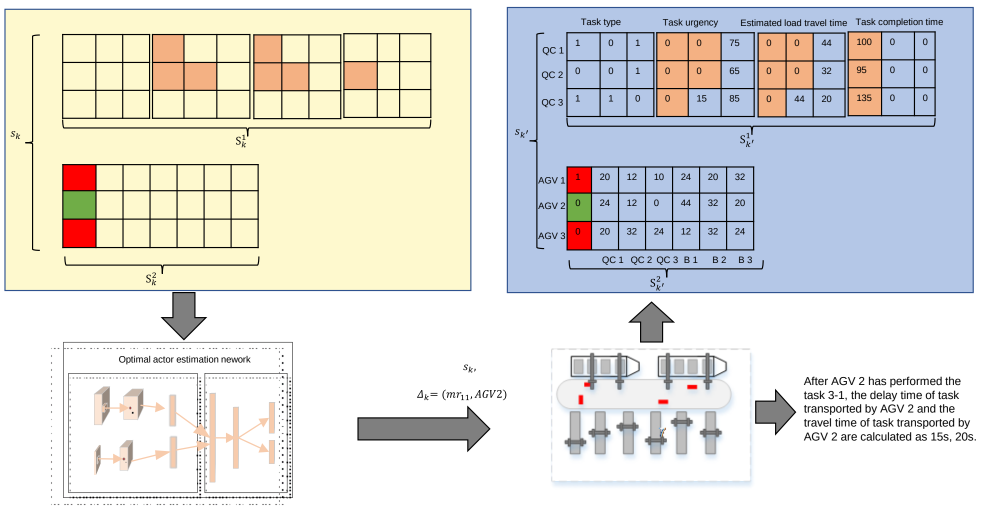
s_k
S_k^1
S_k^2
Task type
Task urgency
Estimated load travel time
Task completion time
QC 1
QC 2
QC 3
AGV 1
AGV 2
AGV 3
QC 1 QC 2 QC 3 B 1 B 2 B 3
$s_{k'}$
$S_{k'}^1$
$S_{k'}^2$
Optimal actor estimation nework
$s_{k'}$
$A_k = (mr_{11}, AGV2)$
After AGV 2 has performed the task 3-1, the delay time of task transported by AGV 2 and the travel time of task transported by AGV 2 are calculated as 15s, 20s.

Figure 5. The decision stage demonstration in an episode.

Table 6. Detailed description of the parameters of the CDA algorithm.

QC ID	Task ID	Type	From Location	To Location	Earliest Event Time	Estimated Load Travel Time	Whether to Be Executed	Task Competition Time
QC 1	1-1	1	B 2	QC 1	70	32	Y	
	1-2	0	QC 1	B 1	130	20	N	
	1-3	1	B 3	QC 1	210	44	N	
QC 2	2-1	0	QC 2	B 3	60	32	Y	95
	2-2	0	QC 2	B 2	140	20	Y	
	2-3	1	B 1	QC 2	200	32	N	
QC 3	3-1	1	B 2	QC 3	80	32	N	
	3-2	0	QC 3	B 1	150	44	N	
	3-3	1	B 3	QC 3	220	20	N	

Note: Y means the task has been assigned or executed; N means the task has not yet been assigned.

5. Numerical Experiments

This section details the factors affecting the scheduling performance of AGVs. First, the optimal value of the parameter β in Equations (6) and (7) is determined empirically to ensure that the CDA algorithm has a reliable convergence as well as scheduling accuracy. Second, the scheduling results of a single assignment rule and other scheduling algorithms are compared and analyzed in different instances to verify the reliability and validity of the model and algorithm proposed in this paper. Finally, the simulation experiment is carried out under the double-cycle operation mode. The implementation of the AGVs dynamic scheduling simulation experiment consists of two parts: to build a simulation terminal environment for AGVs scheduling, and to implement the CDA algorithm architecture based on the deep framework, Tensorflow. To achieve this, all experiments were conducted on Windows 10, Intel(R) Core(TM) i5-10200H CPU @ 2.40 GHz, 16 GB RAM NVIDIA GeForce GTX 1650 Ti, python2019 Professorial. Each test case result is the average of five results.

5.1. Experimental Parameters Setting

There are complexities and uncertainties in the container handling and transportation environments of the automated terminal. In order to reflect the real automated terminal environment as accurately as possible, some important experimental parameters and constraints are set in this paper, as described below.

(1) The number of container tasks considered in each episode $|N| \in [50, 500]$, where 50–100 containers are considered for the small-scale problem and 100–500 containers for the large-scale problem; the number of QCs on the sea side $|Q| \in [2, 8]$ and the number of YCs on the land side $|Y| \in [4, 10]$; and the number of AGVs $|V| \in [5, 15]$ are considerations of this study.
(2) Studying the adaptability of the scheduling algorithm in dynamic situations by introducing uncertainty in the operation time of QCs and YCs, is a consideration of this study. It is assumed that the operation time of QC, loading containers onto or unloading containers from the AGV obeys uniform distribution $t_i^q = U(20 - 30)$s; the time of the AGV loading and unloading containers at YC obeys uniform distribution $t_i^b = U(15 - 25)$s.
(3) This experiment takes the port operation values obtained from Xiamen Yuanhai Container Automation Terminal as a reference. The length and width of the horizontal transport area are set to 240 m and 100 m, respectively, and the speed of AGV is 5 m/s [30].
(4) According to the containers and the task sequence arranged before the ship berthing, the earliest possible event time for each container task is set. The earliest possible event interval of the preceding and following tasks in the tasks sequence obeys a normal distribution $N(60, 80)$, e.g., if container j is the successor of container i in the tasks sequence, then the earliest possible event time of container j is $el_j = el_i + N(60, 80)$.

In general, the hyperparameter in the DRL algorithm plays a significant role in the convergence and training effect of the algorithm. However, due to a large number of hyperparameters and the large search space of parameters in the DRL algorithm, it is difficult to find the optimal value. In this paper, we refer to the hyperparameters given by Liu D. et al. [31] and obtain the final CDA algorithm-related parameters through preliminary experiments, as shown in Table 7.

Table 7. Detailed description of the parameters of the CDA algorithm.

Parameters	Parameter Values	Description of Parameters
lr	0.001	Learning rate of Actor network and Critic network.
M	10,000	Experience replay memory capacity.
BS	32	Batch size.
it_{max}	1500	Maximum number of episodes.
C_u	100	Target network parameter update frequency.
l_s	80	Algorithm training time step.
τ	0.01	Target network parameter soft update coefficient.
γ	0.9	The discount coefficient of accumulative reward.

5.2. Parameter Experiment

In order to alleviate the difficulty of convergence of the CDA algorithm due to the unstable transition of the scheduling environment, the scaling factor of the differential term is introduced in the reward reshaping, and the setting of the scaling factor β has an impact on the accuracy and convergence speed of the algorithm solution. To address the above problem, the experimental arithmetic case is designed with the number of containers $|N| = 100$, the number of QCs $|Q| = 4$, the number of YCs $|Y| = 6$, and the number of AGVs $|V| = 20$, and the experiment is carried out with β taking values of 0.1, 0.3, 0.5, 0.7, and 0.9. The results are shown in Table 8. Figures 6–8 show the cumulative rewards of the algorithm, the delay time of containers tasks, and the total travel time of the AGVs, respectively. It can be seen that β at different values can ensure that the objectives of optimization tend towards the direction of minimization, but values of β that are too high or too low lead to the difficulty of the convergence of the algorithm, and an unsuitable value of β leads to the unideal objectives after the convergence of the algorithm. Considered comprehensively, the scaling factor β in this paper is set to 0.5.

Table 8. Scheduling results of CDA algorithm with different values of β.

β	Reward	Delay Time of Tasks (s)	Total Travel Time of the AGV (s)
0.1	135.57	3194	7722
0.3	135.37	3264	7727
0.5	136.83	2848	7585
0.7	135.91	3103	7680
0.9	135.80	3133	7696

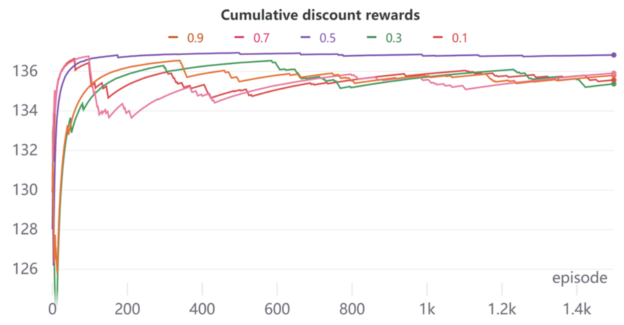

Figure 6. CDA algorithm for cumulative discount rewards for different values of β.

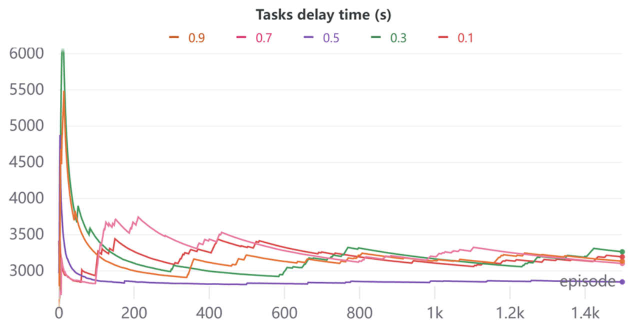

Figure 7. Task delay time of CDA algorithm with different values of β.

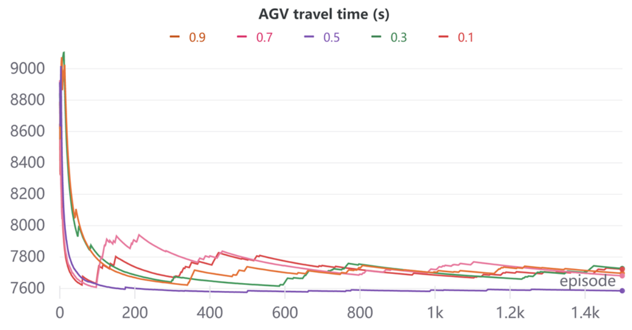

Figure 8. AGV travel time of CDA algorithm with different values of β.

5.3. Comparison of Experimental Results

In implementing the CDA algorithm proposed in this paper for AGVs dynamic scheduling, the scheduling system assigns tasks to the specified AGVs based on the designed heuristic assignment rules, and the optimal policy learned by this algorithm is the task assignment rules in different states. The results obtained in this way are better than the single rule [19]. It is necessary to compare the scheduling results of the CDA algorithm and the single assignment rule to verify the effectiveness of the model algorithm. In addition to using the proposed task delay time and AGVs travel time as evaluation metrics, the container task delay rate and the tasks completion time are also introduced for the comprehensive evaluation of the CDA algorithm and the single assignment rule. The container task delay rate is calculated as follows:

$$D_r = \frac{N_r}{N} \tag{17}$$

Experiments are designed for small-scale container tasks and the number of AGVs in cases 1–10. Tables A1–A4 in Appendix A show the scheduling results for 18 single heuristic rules and the scheduling results of the CDA algorithm on different metrics; the optimal solutions of the CDA algorithm and scheduling rules on different metrics are marked in bold. The experimental results show that the algorithm performs well in all 10 cases. Overall, the CDA algorithm improves the average performance on the metrics of task completion time, the delay time of tasks, total delay time of AGVs, and delay rate of container tasks by 15.63%, 56.16%, 16.36%, and 30.22%, respectively.

The training process of the algorithm in case 8 is shown in Figure 8. As the training proceeds, the cumulative reward quickly converges to a maximum value, as shown in Figure 9a–d, which shows the trends of tasks completion time, tasks delay time, and travel time of the AGVs with the training process, respectively, exactly the opposite of the cumulative reward. Figures A1–A4 in Appendix A clearly show the scheduling results of the CDA algorithm and the single assignment rule in case 8. The CDA algorithm outperforms the single task assignment rule in terms of both tasks completion time and AGVs travel time, while it is slightly inferior to the GUT rule and the LQ-GUT rule in the two metrics of task delay rate and task delay time. Since these two rules always assign the urgent task to AGV for execution first, this will lead to the growth of AGVs travel time, and thus the overall operational efficiency of terminal AGVs scheduling decreases. Comparing Figures A2 and A3 in Appendix A, we can find that CDA is better than the GUT rule in reducing the delay rate of tasks, but slightly inferior to the LQ-GUT rule. In conclusion, improving the overall efficiency of dock-level transportation lies in the fact that the optimal task assignment rules can be selected according to different situations during AGVs scheduling.

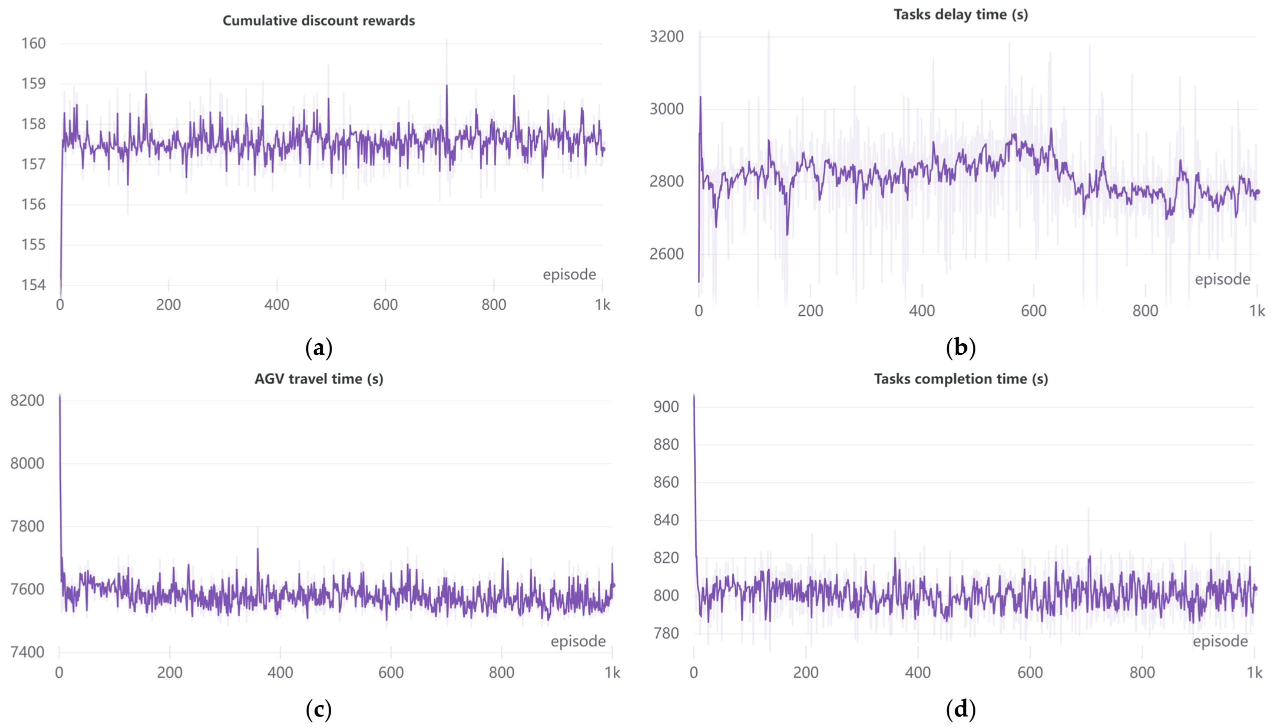

Figure 9. Training process of CDA algorithm on case 8. (**a**) Convergence of cumulative discount rewards; (**b**) Convergence of tasks delay time; (**c**) Convergence effect of AGV travel time; (**d**) Convergence effect of completion time.

To further verify the superiority of the algorithms in solving large-scale problems, this paper sets up a comparison experiment of different algorithms, including the adaptive genetic algorithm (AGA) [1], which is commonly used for dock scheduling, and the rolling time-domain algorithm (RHPA) [32], which is used for dynamic scheduling. To measure the difference in scheduling results between the CDA algorithm and other algorithms, the GPA value of the maximum completion time is used, and the GAP value of the CDA algorithm and AGA are calculated as shown in Equation (18):

$$GPA_{CDA-AGA} = \frac{t_m^{AGA} - t_m^{CDA}}{t_m^{CDA}} \times 100\% \tag{18}$$

where t_m^{CDA}, t_m^{AGA} are the maximum completion times of the CDA algorithm and AGA, respectively. For the evaluation criterion of completion time, if the GAP value is positive, it means that the CDA algorithm is superior; otherwise, AGA is superior. Similar to Equation (18), the GAP values of the CDA algorithm and RHPA are calculated as follows:

$$GPA_{CDA-AGA} = \frac{t_m^{AGA} - t_m^{CDA}}{t_m^{CDA}} \times 100\% \tag{19}$$

In this experiment, three algorithms are used to solve cases 11–30, and the results are shown in Table A5 in Appendix A. From the experimental results, it can be seen that the proposed CDA algorithm can obtain the approximate optimal solutions for different scales of the arithmetic cases. By calculating $GAP_{CDA-AGA}$ and $GAP_{CDA-RHPA}$, and plotting the curves of GAP, as shown in Figure 10, both curves have a slow rising trend, and the performance of the CDA algorithm on scheduling problems is close to that of AGA and RHPA when the size of the arithmetic cases is relatively small. As shown in Table A5 in Appendix A, for example, in Case 13, $GAP_{CDA-AGA}$ is -1.50%, the CDA algorithm has slightly worse scheduling results than AGA in this case; in Case 16 and Case 17, the $GAP_{CDA-RHPA}$ values are -1.92% and -1.34%, respectively, and RHPA has better results than the CDA algorithm in solving these two cases. The scheduling performance of the CDA algorithm becomes significantly better as the size of the cases increases (the number of container tasks ≥ 300). In conclusion, the CDA algorithm is slightly less capable of solving small-scale problems compared to large-scale problems, since the state space of large-scale problems provides a larger optimization space for the algorithmic network to learn and reduce training errors. Analyzing the results of cases 11–30, it can be seen that the CDA algorithm improves by 3.10% and 2.40% in average performance over AGA and RHPA, respectively.

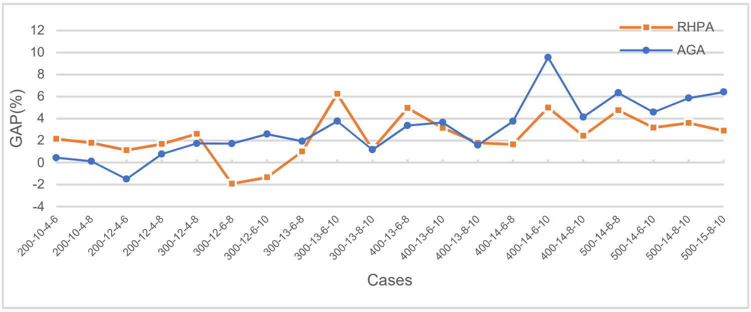

Figure 10. The curve of GAP value under different arithmetic cases.

For different numbers of AGVs, the scheduling results of the CDA algorithm are compared with those of RHPA and AGA. The number of pre-considered tasks is set to 300; the number of QCs and the number of YCs are set to 4 and 8, respectively; and the number of AGVs is assigned to 10, 12, 14, 16, 18, and 20, in that order. The results of the above six groups of experiments are shown in Table 9. Figures 11–13 depict the trends of tasks delay time, AGVs travel time, and tasks completion time, respectively. The following conclusions can be drawn: (1) the travel time of the AGVs maintains almost constant values as the number of AGVs increases; (2) the three algorithms are sensitive to the metric of tasks delay time, and all of them decrease with the number of increasing AGVs; (3) The sensitivity of AGA to this indicator is greater, followed by RHPA, and the sensitivity of the CDA algorithm to this indicator is the least; (4) The sensitivity of the three algorithms to the indicator of the task completion time is similar to the conclusion (2); it reflects the fact that the optimization of task completion times depends to a greater extent on the degree of equipment synergy; (5) In terms of task delay times and completion times, the difference between the CDA algorithm, and AGA and RHPA, is significant when the number of AGVs is small, when the number of AGVs is greater than 16, and the difference between task delay duration and task completion time between the CDA algorithm and other algorithms gradually decreases as the number of AGVs increases, and the overall scheduling performance of the CDA algorithm is significantly better than that of AGA and RHPA.

Table 9. Scheduling results of three algorithms with different number of AGVs.

$\lvert N\rvert \times \lvert V\rvert \times \lvert Q\rvert \times \lvert Y\rvert$	Tasks Completion Time (s)			Tasks Delay Time (s)			AGVs Travel Time (s)		
	CDA	RHPA	AGA	CDA	RHPA	AGA	CDA	RHPA	AGA
300-10-4-8	2273	2325	2332	23,134	36,705	23,505	23,357	23,492	23,016
300-12-4-8	1960	2011	1994	17,319	23,110	13,921	23,070	23,873	23,567
300-14-4-8	1679	1735	1739	10,731	19,670	12,810	23,635	23,902	23,492
300-16-4-8	1527	1552	1574	4620	75,215	7691	23,534	23,606	23,353
300-18-4-8	1331	1352	1371	4946	9494	6631	22,792	22,752	23,684
300-20-4-8	1234	1232	1214	5755	7466	6652	22,701	22,949	23,475

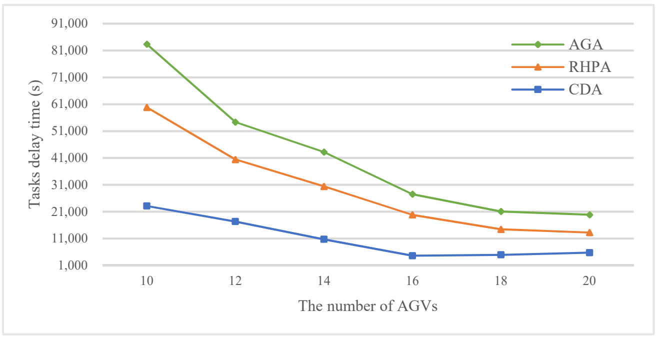

Figure 11. Curve of task delay time with the number of AGVs.

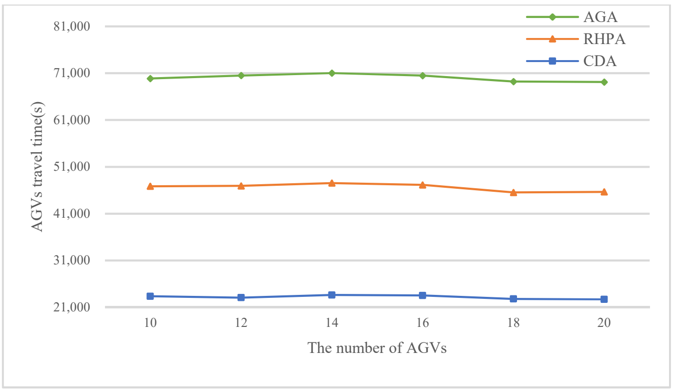

Figure 12. Curve of AGVs travel time with the number of AGVs.

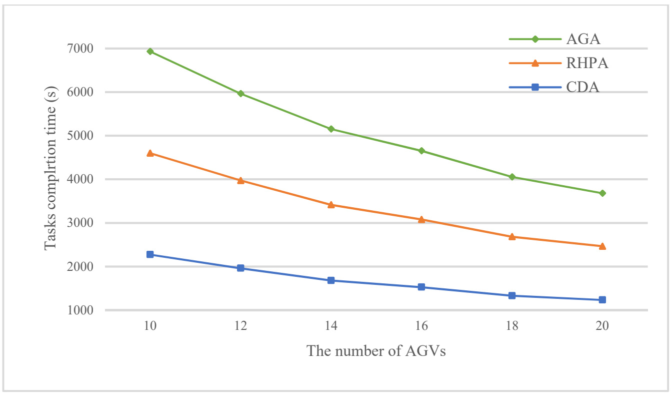

Figure 13. Curve of task completion time with the number of AGVs.

6. Conclusions and Future Research Direction

This article discusses how to choose the AGV and container assignment method to improve the synchronization of handling equipment and transportation equipment in the automated terminal, and transforms the dynamic scheduling problem into a sequential decision problem. The scheduling state, represented as a matrix of multiple channels, heuristic assignment rules, and reward functions, is introduced to simplify the complex AGVs dynamic scheduling process. A reinforcement learning algorithm using deep convolutional networks and hybrid heuristic rules is proposed to optimize the mapping space from the state–action space to the optimal policy. The real AGVs horizontal transportation scenario is simulated, and uncertain task loading and unloading time is considered. In order to obtain the optimal hybrid scheduling rules for AGVs in different states, a large number of experimental cases are designed, and, in this paper, the CDA algorithm is trained for these cases. Comparing the scheduling results of the CDA algorithm with each single scheduling rule defined in this paper, and other solution algorithms including AGA and RHPA, the effectiveness and superiority of the proposed algorithm are verified; the scheduling performance of the CDA algorithm improves by 29.59% on average over a single scheduling rule, and this algorithm can reduce the task operation time by about 3.10% and 2.40% in AGA and RHPA, respectively. A sensitivity test on the number of AGVs further demonstrates the performance of the CDA algorithm. The results show that as the number of containers and AGVs increases, the advantages of CDA become more apparent.

In the future research, in addition to the dynamic scheduling of AGVs, the dynamic path planning problem of AGVs can also be studied. Multiple AGVs share a road network in the automated terminal, not only to ensure the shortest path for the AGVs transportation of containers, but also to consider whether the AGV driving trajectory cross or overlap results in AGVs collision, congestion and other conflict issues. If the AGVs path conflict is not handled properly, it will not only prolong the travel time of the AGVs, but also increase the waiting time of QCs or YCs, resulting in a decrease in operational efficiency and a significant increase in the operation cost. Therefore, AGVs dynamic scheduling combined with AGVs dynamic path planning is one of the main directions for future research. At present, the application of the DRL algorithm to AGVs dynamic scheduling and AGVs dynamic path planning requires further research and exploration in the actual automated terminal. In addition to the synchronous operation of QCs and AGVs, the task delivery efficiency between AGVs and YCs also urgently must be resolved. In the future, the coupling constraints between AGVs and YCs can be considered, such as the shortage of AGV mates in the yard area that delays the tasks. In the new U-shape trafficked automated terminal [33], Both the internal AGVs and the external trucks are delivery tasks directly with the YCs. There is no buffering function in the multiple material handling equipment, and the synchronized operation of AGVs and YCs needs to be considered. Therefore, DRL-based algorithms need to be improved to adapt to the more complex operating environment of the automated terminal.

Author Contributions: Conceptualization, resources, review, editing, supervision and funding acquisition, Z.-H.H.; methodology, software, validation, formal analysis, writing and project administration, C.C.; resources, L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This study is funded by the National Natural Science Foundation of China (NSFC) under the project of Panoramic Modeling and Experiments on the Spatial Pattern Evolution of Maritime Corridors in the South China Sea in the Context of Belt and Road and Big Data (71871136).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are included within the article.

Conflicts of Interest: The authors declare that there is no conflict of interest regarding the publication of this paper.

Appendix A

Table A1. Tasks completion time of CDA algorithm and assignment rules under different cases.

Cases	Tasks Completion Time (s)																		
	LTT	STT	LPT	SPT	GUT	LUT	LQ-LUT	LQ-GUT	SQ-LUT	SQ-GUT	SQ-LPT	SQ-SPT	LQ-LPT	LQ-SPT	SQ-LTT	SQ-STT	LQ-LTT	LQ-STT	CDA
1	933	917	985	912	956	904	945	958	983	963	937	1034	991	962	959	1031	964	1008	**826**
2	609	575	659	627	625	601	592	581	632	638	625	648	565	654	619	651	578	610	**513**
3	938	973	1020	959	933	912	939	907	1014	999	979	1030	951	941	996	1031	946	1004	**760**
4	909	919	1015	932	913	884	926	959	978	985	954	984	954	975	959	990	912	953	**809**
5	723	780	782	749	747	737	734	767	773	755	719	761	748	753	733	764	724	733	**691**
6	759	787	844	755	765	776	811	805	772	806	760	773	806	772	774	769	785	754	**662**
7	918	907	1009	904	909	936	911	943	950	989	906	957	897	949	910	942	905	946	**799**
8	928	993	1064	960	982	935	938	957	984	989	965	994	940	996	974	989	953	978	**806**
9	756	809	868	761	781	774	776	768	760	808	748	792	766	797	750	761	761	783	**637**
10	781	804	884	792	800	785	786	815	823	806	818	822	776	837	818	826	809	775	**680**

Note: The size $|N| \times |V| \times |Q| \times |B|$ of the cases 1–10 are $50 \times 5 \times 2 \times 4$, $50 \times 8 \times 2 \times 4$, $80 \times 8 \times 2 \times 4$, $80 \times 8 \times 4 \times 4$, $80 \times 10 \times 4 \times 4$, $80 \times 10 \times 4 \times 6$, $100 \times 10 \times 4 \times 4$, $100 \times 10 \times 4 \times 6$, $100 \times 12 \times 4 \times 4$, $100 \times 12 \times 4 \times 6$.

Table A2. Tasks delay time of CDA algorithm and assignment rules under different cases.

Cases	Tasks Delay Time (s)																		
	LTT	STT	LPT	SPT	GUT	LUT	LQ-LUT	LQ-GUT	SQ-LUT	SQ-GUT	SQ-LPT	SQ-SPT	LQ-LPT	LQ-SPT	SQ-LTT	SQ-STT	LQ-LTT	LQ-STT	CDA
1	4228	1924	4809	3713	**367**	3585	6239	1423	4333	2451	3417	3234	4089	1861	3175	3107	2917	3209	1328
2	902	1435	1472	188	355	1450	2915	**205**	1774	1526	1420	1372	938	1583	1109	1228	297	1717	491
3	3841	2839	4898	3890	384	4122	7071	**145**	3559	3664	3177	2963	3269	2768	2693	2801	1203	2506	1360
4	7760	5092	9244	6425	**1002**	9376	12,254	1669	7252	5368	6334	6270	7098	6170	6147	6145	5886	6921	2731
5	3732	4167	4623	4019	522	5998	7732	**412**	5222	2940	4211	4579	3821	4260	4136	4519	2391	4459	2648
6	4779	5266	5147	5954	**725**	5622	8823	975	4983	4413	4301	4275	4469	5124	4242	4225	3543	6313	1633
7	6269	6158	8639	7111	**829**	9513	12,695	918	7410	6083	6338	6375	6523	5643	5990	6398	4580	6739	3345
8	6447	6771	8114	6861	**1017**	10,016	12,952	2810	5799	5537	4891	4692	7550	6570	4693	4441	7535	10453	2746
9	4899	3627	6896	4255	707	6973	9717	**337**	4881	2650	4174	4279	5274	3430	4162	3972	4096	4793	1739
10	6316	3261	6989	5772	**587**	6163	8672	931	5640	3153	5233	4931	5546	3010	5319	4720	4930	5366	1800

Note: The size $|N| \times |V| \times |Q| \times |B|$ of the cases 1–10 are $50 \times 5 \times 2 \times 4$, $50 \times 8 \times 2 \times 4$, $80 \times 8 \times 2 \times 4$, $80 \times 8 \times 4 \times 4$, $80 \times 10 \times 4 \times 4$, $80 \times 10 \times 4 \times 6$, $100 \times 10 \times 4 \times 4$, $100 \times 10 \times 4 \times 6$, $100 \times 12 \times 4 \times 4$, $100 \times 12 \times 4 \times 6$.

Table A3. AGVs travel time of CDA algorithm and assignment rules under different cases.

Cases	AGVs Travel Time (s)																		
	LTT	**STT**	**LPT**	**SPT**	**GUT**	**LUT**	**LQ-LUT**	**LQ-GUT**	**SQ-LUT**	**SQ-GUT**	**SQ-LPT**	**SQ-SPT**	**LQ-LPT**	**LQ-SPT**	**SQ-LTT**	**SQ-STT**	**LQ-LTT**	**LQ-STT**	**CDA**
1	4530	4379	4882	4380	4595	4294	4599	4637	4734	4746	4634	4862	4724	4565	4606	4874	4602	4730	**3901**
2	4483	4229	4853	4460	4609	4450	4431	4331	4616	4666	4604	4752	4336	4676	4604	4752	4387	4405	**3779**
3	7149	7212	8030	7291	7173	6991	7125	7013	7673	7605	7493	7833	7278	7199	7445	7913	7312	7493	**5937**
4	7025	6994	7921	7043	7077	6811	7108	7101	7384	7362	7330	7514	7210	7381	7300	7520	6983	7274	**6048**
5	6919	7061	7707	7033	7095	6982	6984	7043	7193	7097	7019	7219	7091	7093	7045	7223	6833	7033	**6412**
6	7041	7263	8064	7264	7311	7148	7452	7545	7350	7709	7295	7358	7396	7277	7239	7352	7277	7259	**6093**
7	8735	8685	9682	8691	8807	8620	8641	8953	8975	9309	8929	9135	8605	8933	8835	9141	8695	8733	**7569**
8	8999	9053	10,220	9101	9353	8988	9021	9103	9325	9509	9217	9417	9093	9234	9241	9323	9251	9304	**7565**
9	8628	8724	9815	8677	8809	8741	8883	8722	8735	8997	8645	8927	8719	8761	8631	8825	8756	8642	**7325**
10	8888	8785	9941	8823	8948	8824	8811	8960	9264	9835	9304	9334	8899	9168	9350	9308	9075	8795	**7482**

Note: The size $|N| \times |V| \times |Q| \times |B|$ of the cases 1–10 are $50 \times 5 \times 2 \times 4$, $50 \times 8 \times 2 \times 4$, $80 \times 8 \times 2 \times 4$, $80 \times 8 \times 4 \times 4$, $80 \times 10 \times 4 \times 4$, $80 \times 10 \times 4 \times 6$, $100 \times 10 \times 4 \times 4$, $100 \times 10 \times 4 \times 6$, $100 \times 12 \times 4 \times 4$, $100 \times 12 \times 4 \times 6$.

Table A4. Tasks delay rate of CDA algorithm and assignment rules under different cases.

Cases	Tasks Delay Rate																		
	LTT	**STT**	**LPT**	**SPT**	**GUT**	**LUT**	**LQ-LUT**	**LQ-GUT**	**SQ-LUT**	**SQ-GUT**	**SQ-LPT**	**SQ-SPT**	**LQ-LPT**	**LQ-SPT**	**SQ-LTT**	**SQ-STT**	**LQ-LTT**	**LQ-STT**	**CDA**
1	0.24	0.18	0.24	0.2	0.14	0.12	0.26	**0.10**	0.22	0.18	0.18	0.24	0.22	0.14	0.20	0.22	0.16	0.22	**0.10**
2	0.12	0.14	0.16	0.10	0.10	0.10	0.20	**0.06**	0.16	0.16	0.14	0.14	0.14	0.14	0.14	0.14	0.10	0.10	0.08
3	0.123	0.14	0.15	0.14	0.09	0.09	0.19	**0.04**	0.10	0.15	0.09	0.09	0.14	0.14	0.08	0.09	0.09	0.10	0.08
4	0.29	0.21	0.34	0.28	0.28	0.21	0.34	**0.13**	0.26	0.25	0.24	0.26	0.30	0.24	0.24	0.25	0.25	0.24	0.20
5	0.19	0.21	0.24	0.21	0.19	0.15	0.25	**0.09**	0.21	0.22	0.20	0.19	0.16	0.18	0.20	0.19	0.18	0.21	0.18
6	0.23	0.19	0.25	0.21	0.19	0.15	0.28	**0.13**	0.20	0.21	0.20	0.20	0.24	0.21	0.20	0.20	0.19	0.23	0.13
7	0.22	0.18	0.22	0.22	0.19	0.15	0.29	**0.07**	0.21	0.20	0.22	0.20	0.25	0.17	0.20	0.20	0.19	0.21	0.15
8	0.20	0.18	0.27	0.22	0.20	0.17	0.28	**0.12**	0.22	0.18	0.20	0.20	0.20	0.19	0.20	0.18	0.20	0.22	0.14
9	0.18	0.16	0.17	0.15	0.15	0.14	0.25	**0.08**	0.18	0.14	0.15	0.17	0.19	0.17	0.15	0.17	0.17	0.16	0.12
10	0.22	0.14	0.20	0.20	0.15	0.13	0.24	**0.09**	0.20	0.14	0.20	0.18	0.20	0.13	0.20	0.18	0.20	0.20	0.12

Note: The size $|N| \times |V| \times |Q| \times |B|$ of the cases 1–10 are $50 \times 5 \times 2 \times 4$, $50 \times 8 \times 2 \times 4$, $80 \times 8 \times 2 \times 4$, $80 \times 8 \times 4 \times 4$, $80 \times 10 \times 4 \times 4$, $80 \times 10 \times 4 \times 6$, $100 \times 10 \times 4 \times 4$, $100 \times 10 \times 4 \times 6$, $100 \times 12 \times 4 \times 4$, $100 \times 12 \times 4 \times 6$.

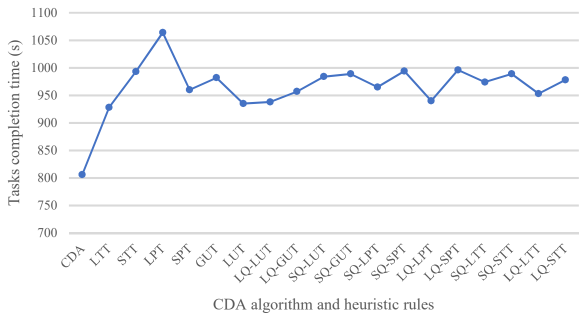

Figure A1. Curve of tasks completion time of CDA algorithm and assignment rules under different cases.

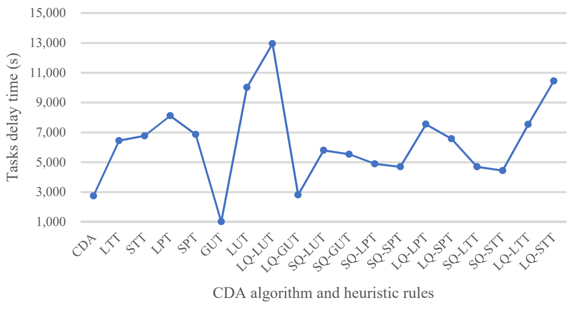

Figure A2. Curve of tasks delay time of CDA algorithm and assignment rules under different cases.

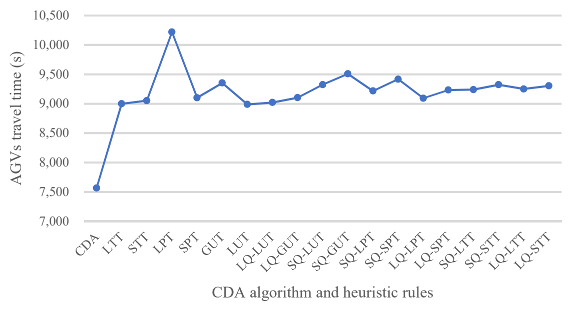

Figure A3. Curve of AGVs travel time of CDA algorithm and assignment rules under different cases.

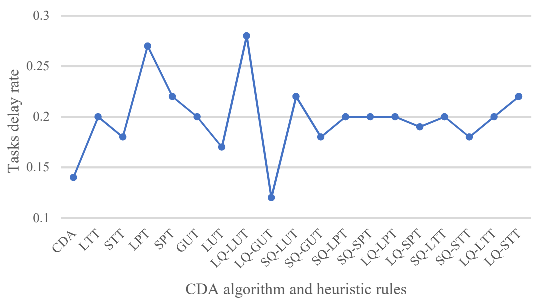

Figure A4. Curve of tasks delay rate of CDA algorithm and assignment rules under different cases.

Table A5. Scheduling results of different algorithms with different cases.

Cases	$\lvert N\rvert \times \lvert V\rvert \times \lvert Q\rvert \times \lvert B\rvert$	Tasks Completion Time (s)			Tasks Delay Time (s)			AGVs Travel Time (s)			Delay Rate of Tasks			$GPA_{CDA-AGA}$(%)	$GPA_{CDA-RHPA}$(%)
		AGA	RHPA	CDA	AGA	RHPA	CDA	AGA	RHPA	CDA	AGA	RHPA	CDA		
11	200-10-4-6	1591	1618	1584	7694	12,876	7604	15,871	15,433	15,513	0.1085	0.14	0.11	0.44	2.15
12	200-10-4-8	1618	1645	1616	10,282	12,112	8343	15,834	15,518	15,636	0.1072	0.14	0.12	0.12	1.79
13	200-12-4-6	1310	1345	1330	9288	11,509	8533	15,588	15,106	15,467	0.1186	0.15	0.12	−1.50	1.13
14	200-12-4-8	1315	1327	1305	5952	12,973	8439	15,406	15,327	15,163	0.1117	0.13	0.12	0.77	1.69
15	300-12-4-8	1994	2011	1960	13,921	23,110	17,319	22,567	22,873	23,070	0.1057	0.12	0.12	1.73	2.60
16	300-12-6-10	2065	1991	2030	28,736	29,521	32,325	24,067	22,915	23,694	0.1886	0.16	0.18	1.72	−1.92
17	300-12-6-10	2062	1983	2010	23,991	36,527	31,237	24,026	23,262	23,446	0.1699	0.18	0.20	2.59	−1.34
18	300-13-6-8	1838	1821	1803	23,171	25,389	20,073	23,424	22,654	22,548	0.1557	0.16	0.14	1.94	1.00
19	300-13-6-10	1846	1890	1779	29,140	32,153	24,933	23,676	22,499	22,682	0.1683	0.19	0.17	3.77	6.24
20	300-13-8-10	1828	1829	1807	26,471	36,714	26,131	22,993	22,922	22,872	0.2014	0.21	0.21	1.16	1.22
21	400-13-6-8	2398	2435	2320	43,908	47,248	35,032	30,865	29,290	29,538	0.1,455	0.17	0.14	3.36	4.96
22	400-13-6-10	2471	2459	2384	37,124	52,999	47,076	30,935	30,198	31,038	0.1501	0.18	0.17	3.65	3.15
23	400-13-8-10	2374	2379	2337	55,548	64,424	57,329	30,011	30,033	29,902	0.243	0.24	0.19	1.58	1.80
24	400-14-6-8	2270	2224	2188	27,675	42,383	27,089	31,515	30,096	29,943	0.126	0.14	0.12	3.75	1.65
25	400-14-6-10	2282	2187	2083	32,490	43,594	34,507	30,617	29,822	29,820	0.1397	0.15	0.14	9.55	4.99
26	400-14-8-10	2265	2228	2175	58,538	48,992	44,950	31,033	30,217	29,870	0.2191	0.18	0.21	4.13	2.44
27	500-14-6-8	2817	2775	2649	46,780	75,977	66,679	37,914	37,388	38,196	0.1315	0.17	0.15	6.34	4.76
28	500-14-6-10	2832	2794	2708	55,802	64,888	41,159	38,436	38,102	38,247	0.1461	0.15	0.13	4.58	3.18
29	500-14-8-10	2764	2705	2611	68,922	94,360	73,322	37,949	36,953	37,204	0.1852	0.2	0.2	5.86	3.60
30	500-15-8-10	2722	2632	2558	73,674	78,205	76,810	38,937	38,548	37,876	0.1691	0.2	0.19	6.41	2.98

References

1. Luo, J.; Wu, Y. Scheduling of container-handling equipment during the loading process at an automated container terminal. *Comput. Ind. Eng.* **2020**, *149*, 106848. [CrossRef]
2. Saidi-Mehrabad, M.; Dehnavi-Arani, S.; Evazabadian, F.; Mahmoodian, V. An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. *Comput. Ind. Eng.* **2015**, *86*, 2–13. [CrossRef]
3. Xu, B.; Jie, D.; Li, J.; Yang, Y.; Wen, F.; Song, H. Integrated scheduling optimization of U-shaped automated container terminal under loading and unloading mode. *Comput. Ind. Eng.* **2021**, *162*, 107695. [CrossRef]
4. Li, J.; Yang, J.; Xu, B.; Yang, Y.; Wen, F.; Song, H. Hybrid Scheduling for Multi-Equipment at U-Shape Trafficked Automated Terminal Based on Chaos Particle Swarm Optimization. *J. Mar. Sci. Eng.* **2021**, *9*, 1080. [CrossRef]
5. Kim, K.H.; Bae, J.W. A look-ahead dispatching method for automated guided vehicles in automated port container terminals. *Transp. Sci.* **2004**, *38*, 224–234. [CrossRef]
6. Iris, Ç.; Christensen, J.; Pacino, D.; Ropke, S. Flexible ship loading problem with transfer vehicle assignment and scheduling. *Transp. Res. Part B Methodol.* **2018**, *111*, 113–134. [CrossRef]
7. Iris, Ç.; Lam, J.S.L. Recoverable robustness in weekly berth and quay crane planning. *Transp. Res. Part B Methodol.* **2019**, *122*, 365–389. [CrossRef]
8. Degris, T.; White, M.; Sutton, R.S. Off-policy actor-critic. *arXiv* **2012**, arXiv:1205.4839.
9. Rashidi, H.; Tsang, E.P.K. A complete and an incomplete algorithm for automated guided vehicle scheduling in container terminals. *Comput. Math. Appl.* **2011**, *61*, 630–641. [CrossRef]
10. Grunow, M.; Günther, H.O.; Lehmann, M. Strategies for dispatching AGVs at automated seaport container terminals. In *Container Terminals and Cargo Systems*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 155–178.
11. Pjevčević, D.; Vladisavljević, I.; Vukadinović, K.; Teodorović, D. Application of DEA to the analysis of AGV fleet operations in a port container terminal. *Procedia-Soc. Behav. Sci.* **2011**, *20*, 816–825. [CrossRef]
12. Skinner, B.; Yuan, S.; Huang, S.; Liu, D.; Cai, B.; Dissanayake, G.; Pagac, D. Optimisation for job scheduling at automated container terminals using genetic algorithm. *Comput. Ind. Eng.* **2013**, *64*, 511–523. [CrossRef]
13. Luo, J.; Wu, Y. Modelling of dual-cycle strategy for container storage and vehicle scheduling problems at automated container terminals. *Transp. Res. Part E Logist. Transp. Rev.* **2015**, *79*, 49–64. [CrossRef]
14. Angeloudis, P.; Bell, M.G.H. An uncertainty-aware AGV assignment algorithm for automated container terminals. *Transp. Res. Part E Logist. Transp. Rev.* **2010**, *46*, 354–366. [CrossRef]
15. Klerides, E.; Hadjiconstantinou, E. Modelling and solution approaches to the multi-load AGV dispatching problem in container terminals. *Marit. Econ. Logist.* **2011**, *13*, 371–386. [CrossRef]
16. Cai, B.; Huang, S.; Liu, D.; Dissanayake, G. Rescheduling policies for large-scale task allocation of autonomous straddle carriers under uncertainty at automated container terminals. *Robot. Auton. Syst.* **2014**, *62*, 506–514. [CrossRef]
17. Xin, J.; Negenborn, R.R.; Lodewijks, G. Rescheduling of interacting machines in automated container terminals. *IFAC Proc. Vol.* **2014**, *47*, 1698–1704. [CrossRef]
18. Kim, J.; Choe, R.; Ryu, K.R. Multi-objective optimization of dispatching strategies for situation-adaptive AGV operation in an automated container terminal. In Proceedings of the 2013 Research in Adaptive and Convergent Systems, Montreal, QC, Canada, 1–4 October 2013; pp. 1–6. [CrossRef]
19. Han, B.A.; Yang, J.J. Research on adaptive job shop scheduling problems based on dueling double DQN. *IEEE Access* **2020**, *8*, 186474–186495. [CrossRef]
20. Briskorn, D.; Drexl, A.; Hartmann, S. Inventory-based dispatching of automated guided vehicles on container terminals. In *Container Terminals and Cargo Systems*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 195–214.
21. Choe, R.; Kim, J.; Ryu, K.R. Online preference learning for adaptive dispatching of AGVs in an automated container terminal. *Appl. Soft Comput.* **2016**, *38*, 647–660. [CrossRef]
22. Fotuhi, F.; Huynh, N.; Vidal, J.M.; Xie, Y. Modeling yard crane operators as reinforcement learning agents. *Res. Transp. Econ.* **2013**, *42*, 3–12. [CrossRef]
23. Hu, H.; Jia, X.; He, Q.; Fu, S.; Liu, K. Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. *Comput. Ind. Eng.* **2020**, *149*, 106749. [CrossRef]
24. Tang, X.; Qin, Z.; Zhang, F.; Wang, Z.; Xu, Z.; Ma, Y.; Ye, J. A deep value-network based approach for multi-driver order dispatching. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1780–1790. [CrossRef]
25. Yao, Z.; Wang, Y.; Meng, L.; Qiu, X.; Yu, P. DDPG-Based Energy-Efficient Flow Scheduling Algorithm in Software-Defined Data Centers. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 6629852. [CrossRef]
26. Luo, S.; Lin, X.; Zheng, Z. A novel CNN-DDPG based AI-trader: Performance and roles in business operations. *Transp. Res. Part E Logist. Transp. Rev.* **2019**, *131*, 68–79. [CrossRef]
27. Ying, C.; Chow, A.H.F.; Chin, K.S. An actor-critic deep reinforcement learning approach for metro train scheduling with rolling stock circulation under stochastic demand. *Transp. Res. Part B Methodol.* **2020**, *140*, 210–235. [CrossRef]
28. Yang, Y.; Zhong, M.; Dessouky, Y.; Postolache, O. An integrated scheduling method for AGV routing in automated container terminals. *Comput. Ind. Eng.* **2018**, *126*, 482–493. [CrossRef]

29. Iqbal, S.; Sha, F. Actor-attention-critic for multi-agent reinforcement learning. International Conference on Machine Learning. *arXiv* **2019**, arXiv:1810.02912.
30. Zhong, M.; Yang, Y.; Dessouky, Y.; Postolache, O. Multi-AGV scheduling for conflict-free path planning in automated container terminals. *Comput. Ind. Eng.* **2020**, *142*, 106371. [CrossRef]
31. Liu, D.; Wang, W.; Wang, L.; Jia, H.; Shi, M. Dynamic Pricing Strategy of Electric Vehicle Aggregators Based on DDPG Reinforcement Learning Algorithm. *IEEE Access* **2021**, *9*, 21556–21566. [CrossRef]
32. Díaz-Madroñero, M.; Mula, J.; Jiménez, M.; Peidro, D. A rolling horizon approach for material requirement planning under fuzzy lead times. *Int. J. Prod. Res.* **2017**, *55*, 2197–2211. [CrossRef]
33. Sun, Z.W. The world's First! Zhenhua Heavy Industry Releases New Technology for Container Terminal Loading and Unloading, China Water Transport Network 2019. Available online: http://app.zgsyb.com/news.html?aid=530549 (accessed on 25 November 2021).