Received February 9, 2021, accepted February 23, 2021, date of publication February 26, 2021, date of current version March 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3062457

A Novel Hierarchical Soft Actor-Critic Algorithm for Multi-Logistics Robots Task Allocation

HENGLIANG TANG, ANQI WANG, FEI XUE, JIAXIN YANG, AND YANG CAO
School of Information, Beijing Wuzi University, Beijing 101149, China

Corresponding author: Hengliang Tang (tanghengliangbwu@163.com)

This work was supported in part by the National Natural Science Foundation of China, Foundation of Ministry of Education of China, under Grant 20YJCZH200, in part by the Beijing Youth Top-Notch Talent Plan of High-Creation Plan under Grant 2017000026833ZK25, in part by the Canal Plan-Leading Talent Project of Beijing Tongzhou District under Grant YHLB2017038; in part by the Project of Beijing Municipal Education Commission of China under Grant KM202110037002 and Grant KM202010037002, in part by the Grass-roots Academic Team Building Project of Beijing Wuzi University under Grant 2019XJJCTD04, in part by the Beijing Key Laboratory of Intelligent Logistics System under Grant BZ0211, and in part by the Beijing Intelligent Logistics System Collaborative Innovation Center.

ABSTRACT In intelligent unmanned warehouse goods-to-man systems, the allocation of tasks has an important influence on the efficiency because of the dynamic performance of AGV robots and orders. The paper presents a hierarchical Soft Actor-Critic algorithm to solve the dynamic scheduling problem of orders picking. The method proposed is based on the classic Soft Actor-Critic and hierarchical reinforcement learning algorithm. In this paper, the model is trained at different time scales by introducing sub-goals, with the top-level learning a policy and the bottom level learning a policy to achieve the sub-goals. The actor of the controller aims to maximize expected intrinsic reward while also maximizing entropy. That is, to succeed at the sub-goals while moving as randomly as possible. Finally, experimental results for simulation experiments in different scenes show that the method can make multi-logistics AGV robots work together and improves the reward in sparse environments about 2.61 times compared to the SAC algorithm.

INDEX TERMS Multi-logistics robot, task allocation, deep reinforcement learning, actor-critic, hierarchical reinforcement learning.

I. INTRODUCTION

The logistics industry has entered the era of intelligent logistics [1] which is highly informative, automated, intelligent and networked. The efficient operation of each part of the intelligent logistics system requires the support of the intelligent storage system. The intelligent storage system [2], [3] uses the Internet of things technology to sense the storage status in real-time, and applies artificial intelligence technology for data processing and analysis. Compared with the traditional storage system, the intelligent storage system has higher efficiency, higher fault tolerance rate, lower labor cost, and strong robustness. A large amount of information will be generated during the operation of the intelligent storage system, which is characterized by the dynamic nature of order information, goods information and storage information. Therefore, a large number of warehousing logistics robots and artificial intelligence technologies are needed to optimize decision-making [4]–[6]. The dynamic task allocation [7] problem of orders belongs to a part of picking work, which includes the process of orders batch, orders task allocation, path planning, picking, packing and shipping. In the storage system, orders information usually has dynamic problems such as multiple and miscellaneous categories of goods, high frequency and large batch, etc. Therefore, the most important thing to study the intelligent storage system is to orders dynamic task allocation.

Currently, the intelligent storage system has a large scale, which requires the cooperation of multiple storage logistics robots in the actual work process. The storage logistics robots mainly use AGVs, represented by amazon's KIVA robot [8]–[10], etc. How to make multiple robots complete multiple tasks in collaboration are still the key point and difficulty in realizing the intelligent storage system of multiple mobile robots. Many logistics robot dynamic task allocation problem is to point to in a state known storage environment, using real-time environmental condition, according to the task allocation results of optimum distribution of tasks to

The associate editor coordinating the review of this manuscript and approving it for publication was Meng-Lin Ku.

logistics robot, and combined with a path planning algorithm, produced from starting point to the shelves and shelves to choose no static obstacles and dynamic collision conflict of the optimal path, in the process, as far as possible little system of time cost.

How to perform a collaborative dynamic task assignment for multi-logistics robots is the primary content of this paper. The traditional approach treats task assignment as a path planning problem for single or multiple robots. It is assumed that the AGV robot plans the trajectory from the current location to the target location in the warehouse environment. Although useful in many situations, traditional task assignment algorithms are limited by their flexibility in practice. The complexity and dynamic obstacles of the environment can increase the instability of the system and reduce the computational efficiency. Besides, most traditional algorithms do not consider dynamic task assignment and can only be solved based on the static order information. Thus, this paper proposed a hierarchical Actor-Critic algorithm to address the core problem of multi-logistics robot task allocation.

In recent years, with the great success of AlphaGo [11]–[13] in the field of go, deep reinforcement learning algorithms have been greatly applied in various fields of study. Deep reinforcement learning algorithm combines the feature that reinforcement learning can realize the interaction between agent and environment through reward mechanism and the advantage that deep learning can extract features of high-dimensional data. It directly extracts the features from the massive environmental data of complex dynamics by using deep neural network, and finally learns the optimal strategy of the agent. Soft Actor-Critic(SAC) [14] is a state-of-the-art deep reinforcement learning algorithm that solves both discrete and continuous control problems. Compared with the traditional Deep Deterministic Policy Gradient(DDPG) [15] algorithm, SAC uses a stochastic policy. Compared with deterministic policy, stochastic policy have more advantages in practical robot control. Therefore, our research focuses on the SAC.

When we extend the task assignment to multi-robot systems, feature quantification and conflict resolution among robots in multi-robot systems become more complex, and traditional deep reinforcement learning methods face the dimensional disaster problem. This is because the state space of an agent increases in a complex dynamic environment, and therefore the parameters and the memory required for the training process increase dramatically. To solve the dimensional catastrophe of traditional deep reinforcement learning, this paper combines hierarchical reinforcement learning(HRL) with the SAC algorithm. Hierarchical reinforcement learning algorithms layer the learning process to reduce the complexity of the algorithm. the HRL approach has multiple layers of learning strategies, each layer being able to be controlled at different time levels. In this paper, task assignment is decided centrally by the assigned robots, and multiple logistics robots interact sparsely in completing the task and planning the actual path. The learning policies of the logistics robots do not communicate in the neural network and the logistics robots make autonomous decisions based on their observed information. In this paper, a deep reinforcement learning-based task assignment method for multiple logistics robots is proposed. By sharing parameters and models, the robots can learn the policies of other robots.

This paper has some contributions on multi-logistics robots task allocation algorithms as follows:

1. The Hierarchical Soft Actor-Critic is proposed based on soft actor-critic, an off-policy DRL algorithm based on maximum entropy reinforcement learning framework, the actor of controller aims to maximize expected intrinsic reward while also maximizing entropy.

2. By combining maximum entropy reinforcement learning with hierarchical structure by introducing the intrinsic reward mechanism, the model is trained at different time scales. Thus, the method can solve the multi-logistics robot task allocation problem.

This paper is organized as follows: Section 2 describes the Actor-Critic algorithms and their applications in multi-robot task allocation problems. In Section 3, the proposed algorithm is summarized, and a novel Hierarchical soft Actor-Critic algorithm is introduced. Section 4 presents the design of the multi-logistics robots task assignment simulation experiment with the proposed algorithm and analyzes the experiment results. Section 5 summarizes this work.

II. RELATED WORK

Benefiting from the development of the deep reinforcement learning, deep reinforcement learning has shown great potential to solve the problem of task allocation in the large-scale unmanned warehouse. Agents learn to optimize policy for complex problems by deep reinforcement learning, which can extract features of high-dimensional state spaces through deep neural networks.

A. ACTOR-CRITIC ALGORITHMS

Deep reinforcement learning can be divided into value-based RL(Reinforcement Learning) and policy-based RL [16]–[18], and combining the algorithm the two results in a series of Actor-Critic (AC) algorithms. The basic idea of Actor-Critic is as follows: the actor selects the policy based on the state and the policy selection function. The critic evaluates the actor's policy based on the value function and guides the actor to improve the policy. Mnih *et al.* [19] first proposed the Asynchronous Advantage Actor-Critic(A3C) and obtained some good results. Also, Advantage Actor-Critic(A2C), another optimization algorithm proposed by Konda and Tsitsiklis [20] also achieved good results.

DeepMind also proposed the Deep Deterministic Policy Gradient (DDPG) [15], which successfully solved the problem of learning in continuous action space. To solve the overestimation problem, Fujimoto *et al.* [21] also proposed the Twin Delayed Deep Deterministic policy gradient algorithm(TD3). Also, Haarnoja *et al.* [14] proposed an off-policy Soft Actor-Critic algorithm (SAC) for maximum entropy reinforcement learning. Compared with DDPG, SAC uses a stochastic policy and has achieved very good results on the open benchmark, which could be directly applied to real robots.

B. MULTI-AGENT ACTOR-CRITIC ALGORITHMS

Deep reinforcement learning and multi-agent systems [22], [23] have complementary advantages. Peng *et al.* [24] proposed an AC-based multi-agent bi-directional collaboration network. Both actor and critic networks use a bi-directional Long Short-Term Memory(LSTM) [25] architecture to connect agents in series. Mao *et al.* [26] proposed a general cooperative actor-critic network and the AC-Cnet architecture. In a partially observed environment, the communication protocol between agents was learned from zero. Yang *et al.* [27] proposed mean-field reinforcement learning to approximate inter-agent interactions used the mean interaction between the global or neighboring agents, designed mean-field Q learning and mean-field Actor-Critic algorithms for multiple agents, and analyzed the convergence. Lowe *et al.* [28] extended the DDPG method to multi-agent learning, modeled the opponent by observing the opponent's past behavior, constructed a global critic function to evaluate the global state-autonomous action, and trained a set of agent policies to improve the robustness of the algorithm. Foerster *et al.* [29] proposed an actor-critic counterfactual multi-agent (COMA) policy gradient method, which uses a centralized critic function to evaluate joint actions. Each agent uses its actor policy network to make decisions.

C. ALGORITHMS FOR TASK ALLOCATION

The achievements of the research on multi-agent dynamic task allocation [30]–[32] are mainly based on heuristic intelligent algorithms. Intelligent algorithms mainly use environmental learning or heuristic search, such as A* algorithms [33], evolutionary algorithms [34]–[36], and neural network-based methods, etc. Evolutionary algorithms based on simulated organisms mainly include ant colony algorithm(ACO), genetic algorithm or algorithms combining the two. Existing ACO has a high calculational time complexity and is prone to fall into the local minimum when solving high-dimensional space problems. The genetic algorithm [37] can only approximate the global optimal solution, and is not fast enough to be combined with other intelligent algorithms. Besides, there are related researches based on the search algorithms, which compute large amounts when performing global searches, while local search methods require heuristic rules. Recently, the Deep Reinforcement Learning algorithms are applied to the scheduling problems. Wang *et al.* [38] apply a DQN [39] to guide the scheduling problems. Zeng [40] and others used reinforcement learning Q-Learning algorithm to consider the overall system state for scheduling. Shahrabi *et al.* [41] used reinforcement learning with a Q-factor algorithm to the scheduling method which considers random job arrivals and machine breakdowns.

III. PRELIMINARIES

In this section, we first describe the problem of multi-logistics robot task allocation. Then, we describe hierarchical soft actor-critic(HSAC) method for task scheduling.

A. PROBLEM FORMULATION

Logistics Robots Dynamic Task Assignment (LRDTA) is a dynamic scheduling problem for cargo-to-person mode order picking using AGV carts in an unmanned warehouse. After the order arrives at the system, first, the order is assigned to the picking table using an intelligent algorithm. Second, the shelves needed in the order are assigned to the logistics robots, which assign the location information of the shelves, perform path planning, and move the shelves from the current position to the target position. After that, the shelves are transported to the designated picking table, and the shelves are transported back to the original position after the picking is completed. Finally, the logistics robots move from the current position to the target shelf position, and so on. Finally, the logistics robot moves from the current position to the target shelf position, and so on until it reaches the termination state, i.e., all orders have been picked.

Input parameters include warehouse layout and order lists. The order lists are collected from real warehouse procedure. The objective function of LRDTA is the shortest total outbound time:

$$\min\max\ ITC(R_i, S_{R_i}) \tag{1}$$

$$ITC = \sum_{P=1}^{k}\sum_{j=1}^{S_{R_i}}\left[\left(c_{R_iS_j} \times x_{P_s\eta_{R_is_j}}\right) + IC\right] \tag{2}$$

$$IC = \left(\sum_{j=1}^{m} x_{P_s\eta_{R_is_j}}\right) \times t_{\text{pick}} - c_{R_is_j} \times x_{P_s\eta_{R_i}s_j} \tag{3}$$

where R denotes the robot, P denotes the picking table, S_{R_i} is the set of tasks for the robot, $c_{R_iS_j}$ is the cost of the robot to complete the task, and t_{pick} denotes the time for order picking and packing as a constant. $x_{P_s\eta_{R_is_j}}$ indicates whether the shelf j carried by AGV robot i is picked at picking table s.

The constrains are as followings. Equations 4 and 5 constrain a task to be assigned to only one picking table and robot, respectively.

$$\sum_{P_1}^{P_l} x_{P_s\eta_{R_is_j}} = 1 \cdots \mathrm{R}_i = \mathrm{R}_1, \mathrm{R}_2, \ldots, \mathrm{R}_m \tag{4}$$

$$\sum_{R_1}^{R_m} \eta_{R_is_j} = 1 \tag{5}$$

Deep reinforcement learning(DRL) is an important area of machine learning and artificial intelligence. DRL focus on the problem of how agents can learn policies by interacting directly with their environment to maximize long-term returns. Traditional DRL is based on the Markov Decision Process (MDP), which can be expressed as a five-tuple $<S, A, P, R, \gamma>$. S is a finite set of states, a state s in the set belongs to S. A is a finite set of actions, an action a in the set belongs to A, and A is a set of actions that can be performed in the state s. P is a state transition equation, which means that the agent will jump to state s' with probability p after performing an action a in the state s, R is the reward function. γ is the discount factor, and γ belongs to [0,1]. MDP indicates that the next state of the environment is only related to the current state and not to the previous state. However, in a complex environment or difficult task, too large state space of the agent can lead to a rapid increase in learning parameters and storage space. Facing of dimensional disaster, DRL has a hard time achieving the desired results, so hierarchical reinforcement learning(HRL) is proposed.

HRL breaks down a complex problem into several sub-problems, which can be solved separately with better results than solving the whole problem directly. HRL needs to be performed on a semi-Markov decision process(SMDP). In the SMDP, the number of steps from the current state to the next state is a random variable τ, i.e., after selecting an action a in a state s, it will transition to the next state s' with the probability after the τ step. The probability of state transition $P(s', \tau|s, a)$ is the joint probability of s and τ.

This paper aims to provide a task allocation method with the deep reinforcement learning algorithm for the multi-logistic robot system. We try to find such a learnable policy module $\pi: S \times A \rightarrow [0, 1]$, which can guide the agent to choose the action. Given a policy π and a states$s$, the action-value function represents the expected cumulative reward that can be obtained by performing the action a in the state s of a given policy π. Standard reinforcement learning requires maximizing the expected value of the sum of rewards.

$$Q(s_t, a_t) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)] \tag{6}$$

B. SOFT ACTOR-CRITIC ALGORITHM

In the multi-logistics robot task allocation problem, the relationship between the environment state inputs and the output control law can be very complex. In our work, we consider some modification of the Soft Actor-Critic(SAC) as the basic framework.

SAC is a remarkable deep reinforcement learning algorithm to address the discrete and continuous control problems. Compared with the classic DDPG, SAC uses a stochastic policy, which has certain advantages over deterministic policy in actual robot control. Stochastic policy is achieved through maximum entropy. The idea of maximum entropy keeps any useful action or trajectory from being overlooked. To obtain good policies to achieve dynamic task assignment, logistic robots need to learn stable rules between perception and decision-making, and avoid collisions and collaborate in some cases. This method uses maximum entropy goal to learn policy for more complex tasks:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t, a_t)\sim\rho_\pi}[r(s_t, a_t) + \alpha H(\pi(\cdot|s_t))] \tag{7}$$

α is a hyper-parameter of the temperature coefficient, used to adjust the focus of entropy value. The state value function $V(s_t)$ and action-state value function $Q(s_t, a_t)$ of the maximum entropy reinforcement learning can be expressed as Equation(3) and Equation(4):

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p(s_{t+1}, \tau|s_t, a_t)}[V(s_{t+1})] \tag{8}$$

$$V(s_t) = \mathbb{E}_{a_t\sim\pi}[Q(s_t, a_t) - \alpha \log \pi(a_t|s_t)] \tag{9}$$

There are a number of conceptual and practical advantages to using maximum entropy. First, the policy learned can be used as an initialization for more complex tasks, while integrating with hierarchical architecture. Second, the policy can capture near-optimal action and allow for greater exploration. Lastly, the maximum entropy provides a stronger robustness and generalization.

This algorithm constructs neural networks $Q_\theta(s_t, a_t)$ and $\pi_\phi(a_t|s_t)$to represent action-state value function $Q(s_t, a_t)$ and policy π. At each time step, the actor-network and the critic-network are updated by uniformly sampling minibatch from the memory buffer. The algorithm also creates a target network of $Q(s_t, a_t)$ and policy π for soft updates through the learned network, which can greatly improve the stability of learning.

The loss function for the critic-networks can be formulated as Equation(5):

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t)\sim D)} \left[\frac{1}{2}\left(Q_\theta\,(s_t, a_t) - \widehat{Q}(s_t, a_t)\right)^2\right] \tag{10}$$

with

$$\widehat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t-1}\sim p}[V_{\overline{\psi}}(s_{t+1})] \tag{11}$$

The loss function for the actor-networks when training policy π_ϕ can be formulated as Equation(7):

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}, a_t\sim\pi_\phi}\left[\log \pi_\phi\,(a_t|s_t) - \frac{1}{\alpha} Q_\theta\,(s_t, a_t) + \log Z\,(s_t)\right] \tag{12}$$

IV. HIERARCHICAL SOFT ACTOR-CRITIC

Prior methods have proposed the off-policy actor-critic method in the maximum entropy RL framework. The method proposed incorporates three key ingredients: an actor-critic architecture with policy and value function networks, an entropy maximization to enable stability and exploration and a hierarchical framework combined with an intrinsic motivation learning mechanism.

In this section, we will add the hierarchical framework [42], [43] to the SAC as the hierarchical soft actor-critic(HSAC). In a multi-logistics robot system, the observation of each robot contains information about the other robots. If other robots are regarded only as dynamic obstacles without considering how to collaborate with others, the efficiency of task allocation for unmanned warehouse selection will be greatly reduced.

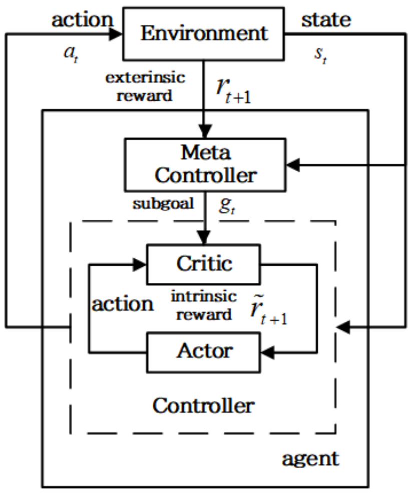


FIGURE 1. The structure of the HSAC.

A. META-CONTROLLER/CONTROLLER FRAMEWORK

To address the aforementioned issues, we proposed a hierarchical soft actor-critic(HSAC). Compared with the classic soft actor critic for multi-logistic task allocation, the HSAC algorithm has a two-layer structure, namely Meta-controller and Controller. The Meta-controller takes the state s_t as input and selects a new subgoal g_t. The Controller uses state and selected subgoal to select the action until the subgoal is reached or the episode is terminated. Two models are used in the controller to approximate the output action and action-value function respectively. In this paper, we only construct a two-level hierarchical structure, but the proposed approach can be extended to a greater depth of hierarchy.

As is shown in Figure 1, the Meta-controller provides subgoal g_t for the Controller. At each time step t, the controller's actor model outputs action a_t and policy based on state s_t and subgoal g_t. The action a_t, along with the state s_t and subgoal g_t serves as the input to the controller's critic model, which ultimately outputs value $Q(s_t, a_t; g_t)$. When the controller implements a subgoal g_t, the meta-controller receives a state s_t from the environment and select a subgoal from the set of subgoals.

The environment provides the state s_t and extrinsic reward r_{t+1}. Meta-controller and controller use separate networks inside, and the controller has both actor and critic models inside. The meta controller gets the subgoal g_t, and generates policies by estimating the value function $Q(s_t; g_t)$. The controller uses the state s_t and g_t as inputs, and generates the policies by estimating $Q(s_t, a_t; g_t)$ and the action $a_t$by estimating $\pi(a_t|s_t, g_t)$. The environment receives the action a_t and generates a new state s_{t+1} and extrinsic reward r_{t+1}. The controller's critic judges whether the internal goal g_t has been completed according to the new state s_{t+1}. If the episode ends or the subgoal is complete, the meta-controller will select a new subgoal. If the subgoal is not complete, the controller's critic will provide the intrinsic reward $\tilde{r}_{t+1}$. if it is completed, a new subgoal is generated, and if it is not completed, it gives internal feedback r.

The objective function of the controller is to maximize the future accumulated intrinsic reward

$$\tilde{G}_t = \sum_{t'=t}^{t+T} \gamma^{t'-t}\tilde{r}_t(g_t) \tag{13}$$

where T is the time to complete the subgoal g_t. The objective function of the meta-controller is to maximize the future cumulative extrinsic reward

$$G_t = \sum_{t'=t}^{\tau} \gamma^{t'-t} r_t \tag{14}$$

where τ is the time of the final step in a episode. We can use two different value functions to learn policy for meta-controller and controller's critic models as Equation(10) and Equation(11):

$$Q(s_t) = \mathbb{E}_{(s_t,a_t)\sim\pi_g}[\sum_{t'=t}^{\tau} \gamma^{t'-t}(r(s_t) - \alpha \log \pi_g(\cdot|s_t))] \tag{15}$$

where π_g is the policy over the meta-controller.

$$\begin{aligned} Q(s_t, a_t; g_t) = \mathbb{E}_{(s_t,a_t)\sim\pi_{ag}}[&\sum_{t'=t}^{T} \gamma^{t'-t}\tilde{r}(s_t, a_t; g_t) \\ &- \alpha \log \pi_{ag}(a_t|s_t, g_t))] \end{aligned} \tag{16}$$

where π_{ag} is the policy over the controller.

In other words, we update the policy in terms of the Kullback-Leibler divergence according to Equation(12):

$$\pi_{new} = \arg \min_{\pi' \in \Pi_{ag}} D_{KL}(\pi'(g_t|s_t) || \frac{\exp(Q^{\pi_{old}}(s_t; g_t))}{Z^{\pi_{old}}(s_t)}) \tag{17}$$

The partition function $Z^{\pi_{old}}(s_t)$ normalizes the distribution, which is constant for the policy $\pi\,(g_t|s_t)$ and can be directly ignored in the actual calculation. The controller's experience replay buffer $\mathcal{D}_1$ consists of a set of $(s_t, a_t, g_t, \tilde{r}_t, s_{t+1})$. The meta-controller's experience replay buffer $\mathcal{D}_2$ consists of a set of $(s_t, g_t, G_{t:t+T}, g_{t+T})$.

We use the V network, policy network and Q network to approximate the value of meta-controller, the policy of the controller's actor model and the value of the controller's critic model. We will consider a parameterized value function $Q_\psi(s_t; g_t)$, $Q_\theta(s_t, a_t; g_t)$ and policy $\pi_\phi(a_t|s_t, g_t)$. To train the meta-controller value function, we minimize the loss function based on the extrinsic reward from the environment:

$$\begin{aligned} J_Q(\psi) = \mathbb{E}_{(s_t,g_t,G_{t:t+T},g_{t+T})\sim\mathcal{D}_2}[&\frac{1}{2}(Q_\psi\,(s_t; g_t) \\ &- \mathbb{E}_{a_t\sim\pi_\phi}[Q_\theta\,(s_t, a_t; g_t) - \log \pi_\phi\,(a_t|s_t, g_t)])^2] \end{aligned} \tag{18}$$

The minimized loss function for updating parameters of the controller's critic is Equation(14).

$$J_Q(\theta) = \mathop{\mathbb{E}}_{(s_t,a_t,g_t,)\sim\mathcal{D}_1}\left[\frac{1}{2}\Big(Q_\theta\,(s_t, a_t; g_t) - \hat{Q}\,(s_t, a_t; g_t)\Big)^2\right] \tag{19}$$

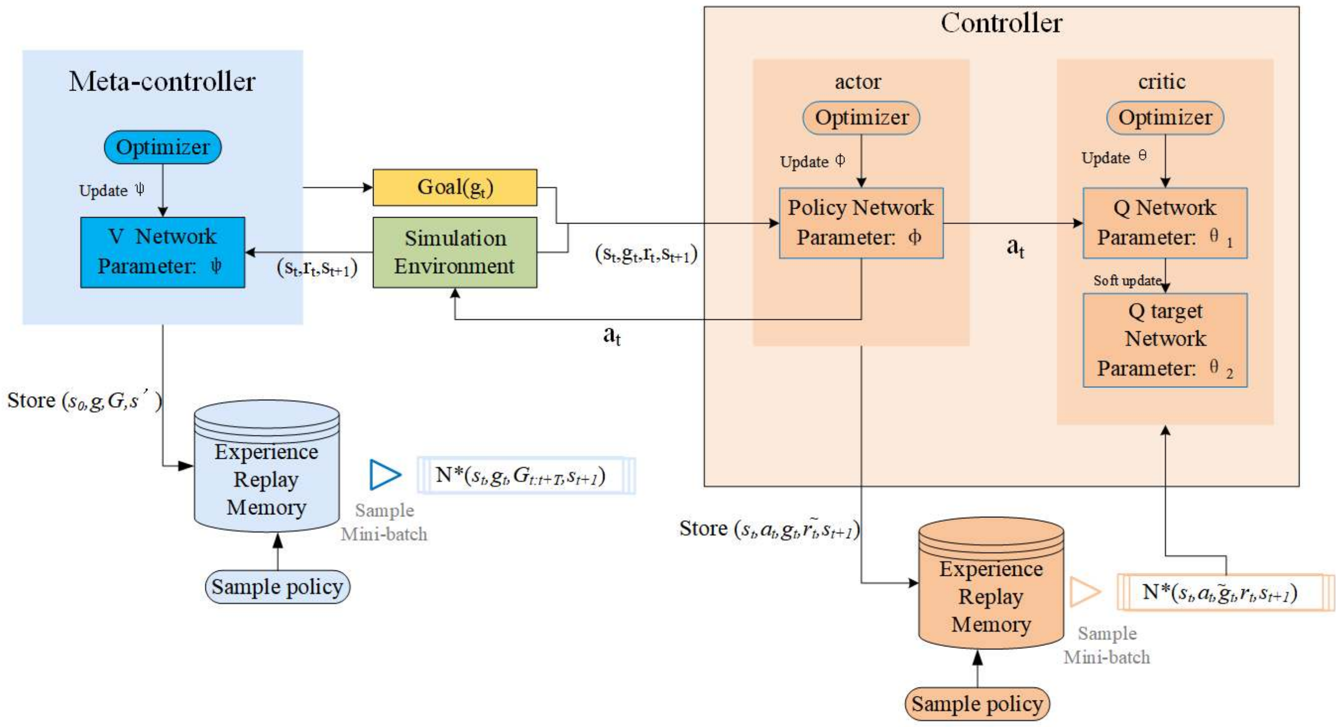


FIGURE 2. **The network structure of the Soft Actor-Critic for task allocation.**

with

$$\hat{Q}\left(s_t, a_t; g_t\right) = \tilde{r}\left(s_t, a_t; g_t\right) + \gamma \mathbb{E}_{s_{t+1}\sim p}[V_{\overline{\psi}}(s_{t+1})] \quad (20)$$

Policy parameters can be learned by minimizing the Kullback-Leibler divergence in Equation(16).

$$\begin{aligned} J_{\pi_{ag}}(\phi) = \mathbb{E}_{(s_t,a_t,g_t)\sim\mathcal{D}_1}[&\log \pi_\phi\left(a_t|s_t, g_t\right) \\ &- \frac{1}{\alpha}Q_\theta\left(s_t, a_t; g_t\right) + \log Z\left(s_t, g_t\right)] \end{aligned} \quad (21)$$

The algorithm achieves the goal of unmanned warehouse order picking by sharing experience replay memory and policies. The robots use the same policy to plan the path from the start to the destination, and the robot's trajectory is stored in a shared experience buffer. The HSAC algorithm is given in Algorithm 1.

B. NETWORK STRUCTURE

In this section, we overview the network structure, and the perception information of the policy module includes the information about the unmanned warehouse environment and the information about the robot. Some parameters related to task allocation are added to the framework, such as the set of shelves allocated to the robot and the set of orders allocated to the picking station. It enables the policy module to consider both the picking station state and the robot state, which is essential for task assignment.

There are three types of neural networks in our module, namely V network, Policy network and Q network. As shown in Figure 2, they all use perceptual information as input. V network passes data through a fully connected layer and outputs state value. Policy network passes the data through the fully connected layer and then through a Gaussian distribution layer, and outputs the sampled actions and policies. The sampled actions output from the Policy network and the perception information are used as input to the Q network, and finally Q network output the state-action value.

C. REWARD SHAPING

Intrinsic reward is the core idea behind the learning of value functions in the meta-controller and the controller. There is sparse delayed feedback in task scheduling. Thus, logistics robots cannot effectively explore the environment and achieve the task. By specifying and quantifying the task goals, it is determined whether the robot can learn the desired policy. When logistics robots complete tasks in unmanned warehouses, rewards are scarce, and it is hard to learn appropriate policies to achieve goals. To solve this problem, we use subgoals to help logistics robots complete a series of scheduling tasks. In the method proposed in this paper, the unmanned warehouse scheduling is divided into three stages. First, a group of robots are assigned tasks and arrive at the location of the shelf that needs to be moved from their starting positions. Then, the logistics robot carries the shelves to the picking table. Finally, when the task of order picking is complete, the robot returns the shelves. There are different parts of the reward function: a reward for completing tasks, a collision penalty, and a reward for approaching. Except for the positive reward for completing the task, the other settings

Algorithm 1 Hierarchical Soft Actor-Critic

```
Input: $\theta_1, \theta_2, \psi, \phi$
1:  Specify Subgoals space $\mathbf{G}$
2:  Initialize parameter vectors $\theta, \psi, \phi$
3:  Initialize experience memories $\mathcal{D}_1$ and $\mathcal{D}_2$
4:  for each episode do
5:      Initialize state $s_0 \in S$, $s \leftarrow s_0$
6:      $G \leftarrow 0$
7:      for each step do
8:          $a_t \sim \pi_\phi(a_t|s_t)$
9:          $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
10:         $\mathcal{D}_1 \leftarrow \mathcal{D}_1 \bigcup (s_t, a_t, g_t, \tilde{r}_t, s_{t+1})$
11:         Sample from $\mathcal{D}_1$ and compute $\nabla J_Q(\theta)$,$\nabla J_\pi(\phi)$
12:         Update controller's parameters, $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$
13:         Update controller's parameters, $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$
14:         Sample from $\mathcal{D}_2$ and compute $\nabla J_Q(\psi)$
15:         Update meta-controller's parameters, $\psi \leftarrow \psi - \lambda_Q \nabla_\psi J_Q(\psi)$
16:         $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
17:         $s \leftarrow s'$,$G \leftarrow G + r$
18:     end for
19:     until s is terminal or subgoal is attained
20:     $\mathcal{D}_2 \leftarrow \mathcal{D}_2 \bigcup (s_0, g, G, s')$
21: end for
Output: $\theta_1, \theta_2, \psi, \phi$
```

are negative rewards.

$$r\left(s_t, a_t\right) = \begin{cases} \sum_1^n -k\left(dist_{t+1} - dist_t\right) & \text{at each time step } t \\ r_{\text{collision}} & \text{if the robot collide} \\ r_{\text{task}} & \text{if achieves the task} \end{cases} \tag{22}$$

where $(dist_{t+1} - dist_t)$ represents the degree of change between the robot's position and the target's position, from time step t to $t+1$. The distance is calculated using Manhattan distance. k is the weight of the reward robot's approach to the target position during the path planning process. The reward for completing the subgoals is the intrinsic reward:

$$\tilde{r}\left(s_t, a_t; g_t\right) = \begin{cases} \min\left(r\left(s_t, a_t\right), -1\right) & \text{if } s_{t+1} \text{ achieves } g_t \\ r_{\text{subgoal}} & \text{if } s_{t+1} \text{ achieves } g_t \end{cases} \tag{23}$$

The rewards for each logistics robot are aggregated into a set of rewards. When any of the robots collide, the environment will be reset and the episode will end.

V. EXPERIMENTS

To evaluate the performance of the HSAC algorithm, sufficient experiments were conducted in an unmanned warehouse simulation environment which is implemented by Python. To build an unmanned warehouse environment for logistics robots, an SMDP was used to model the problem of the unmanned warehouse scheduling. The proposed method is implemented on a PC with 16G RAM, i7-8750H processor, and Geforce GTX1060Ti under Windows 10 operating system.

TABLE 1. The parameters of the warehouse environment.

	Size	Tasks	Robots
A	tiny	2	2
B	tiny	4	2
C	small	4	4
D	small	8	4
E	medium	6	6
F	medium	12	6

A. MULTI-AGV UNMANNED WAREHOUSE

Our experimental environment and experimental environment parameter settings are derived from the research results of Christianos F. et al[44]. The experiment simulates the unmanned warehouse scenes. The size of the warehouse is set to either tiny(10×11), small(10×20),medium(16×20), and large(20×29). At each epoch, orders were randomly generated and distributed over N shelves. And there are H AGV robots in the environment. The experimental unmanned warehouse parameter settings are shown in Table 1. The environment requires AGV robots to move requested shelves to the picking stations and back to the shelve's original location. For a logistics robot, other robots can be regarded as obstacles. In the experiment, the robots were randomly set in different initial positions.

The action space for each AGV robots is A={Forward, Turn Left, Turn Right, Load/Unload Shelf}. The AGV robots can move beneath shelves when they do not carry a shelf, but when carrying a shelf, robots must use the corridors visible.

The environment is partially-observable with a very sparse reward since robots have a limited view area and are rewarded only when shelves are delivered successfully. The observation of an AGV robot consists of a 3×3 square centered on the robot. It contains information about the surrounding robots and shelves.

The environment used in our experiment is shown in Figure 3, the necessary modules of an unmanned warehouse such as shelves, picking stations, and robots are built in the simulation. The position of the blue square represents the distribution of the shelves in the unmanned warehouse. The yellow pentagon represents the AGV robots with a random initial position. The black square represents the picking station.

B. MULTI-LOGISTIC ROBOT TASK ALLOCATION EXPERIMENT

The flow chart of HSAC algorithm to solve the dynamic task assignment of multi-logistics robots is shown in Figure 4. First, at each iteration, the task list of the environment is obtained. Then, the tasks are assigned to the AGV robots. For each AGV robot, first, the state and tasks of the environment are obtained. Then, the Meta-Controller generates the robot's

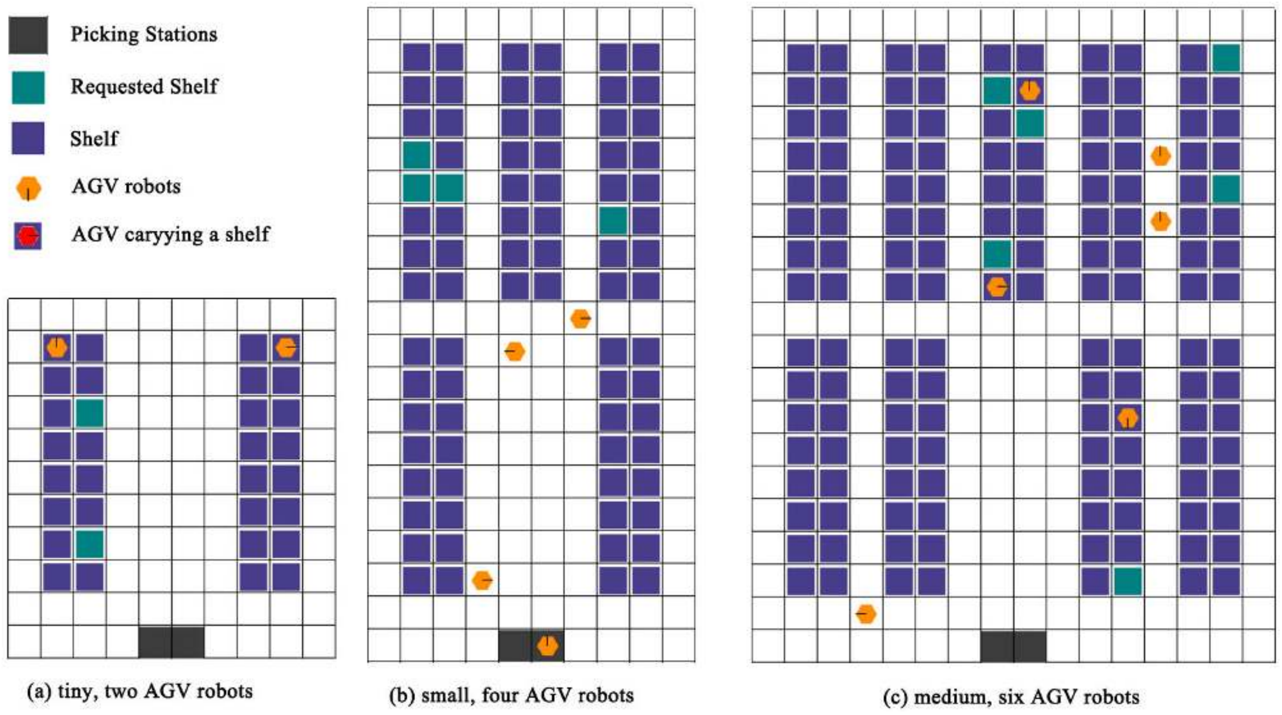
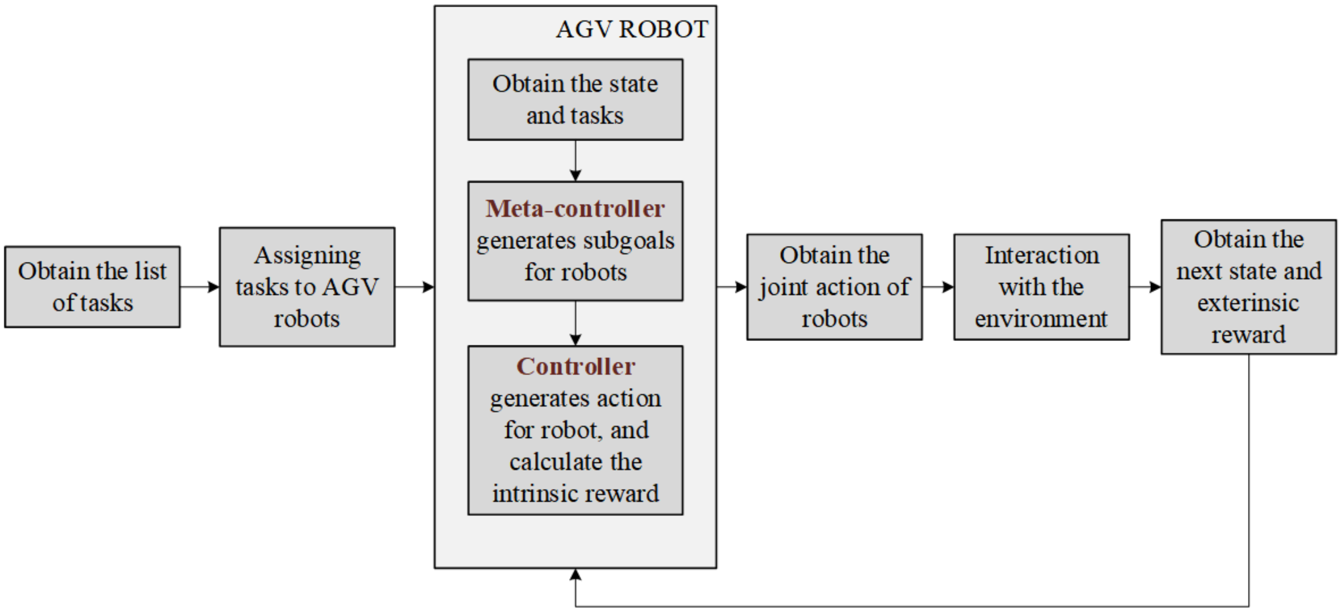


FIGURE 3. The road layout of unmanned warehouse.



AGV ROBOT
Obtain the state and tasks
Meta-controller generates subgoals for robots
Controller generates action for robot, and calculate the intrinsic reward
Obtain the list of tasks
Assigning tasks to AGV robots
Obtain the joint action of robots
Interaction with the environment
Obtain the next state and exterinsic reward

FIGURE 4. The flow chart of experiment.

subgoals. Finally, the Controller generates the robot's actions and calculates the intrinsic rewards. Next, the robot's actions are united to interact with the environment and obtain the next state and extrinsic reward. The previous steps are cycled until all tasks of the environment are completed.

We present the Hierarchical Soft Actor-Critic algorithm(HSAC), which builds on the Soft Actor-Critic algorithm by applying the modifications describes in the last section to solve dimensional disaster problem with consideration of hierarchical architecture. The framework of Controller maintains a pair of critics along with a single actor. Our hyperparameter adjustments were referenced from the SAC [14] and fine-tuned during the experiment.Below, Table 2 shows the hyper-parameters of the HSAC.

The goal of our experiment is to understand how the sample complexity and stability of our method compares with the following baseline deep reinforcement learning algorithms.

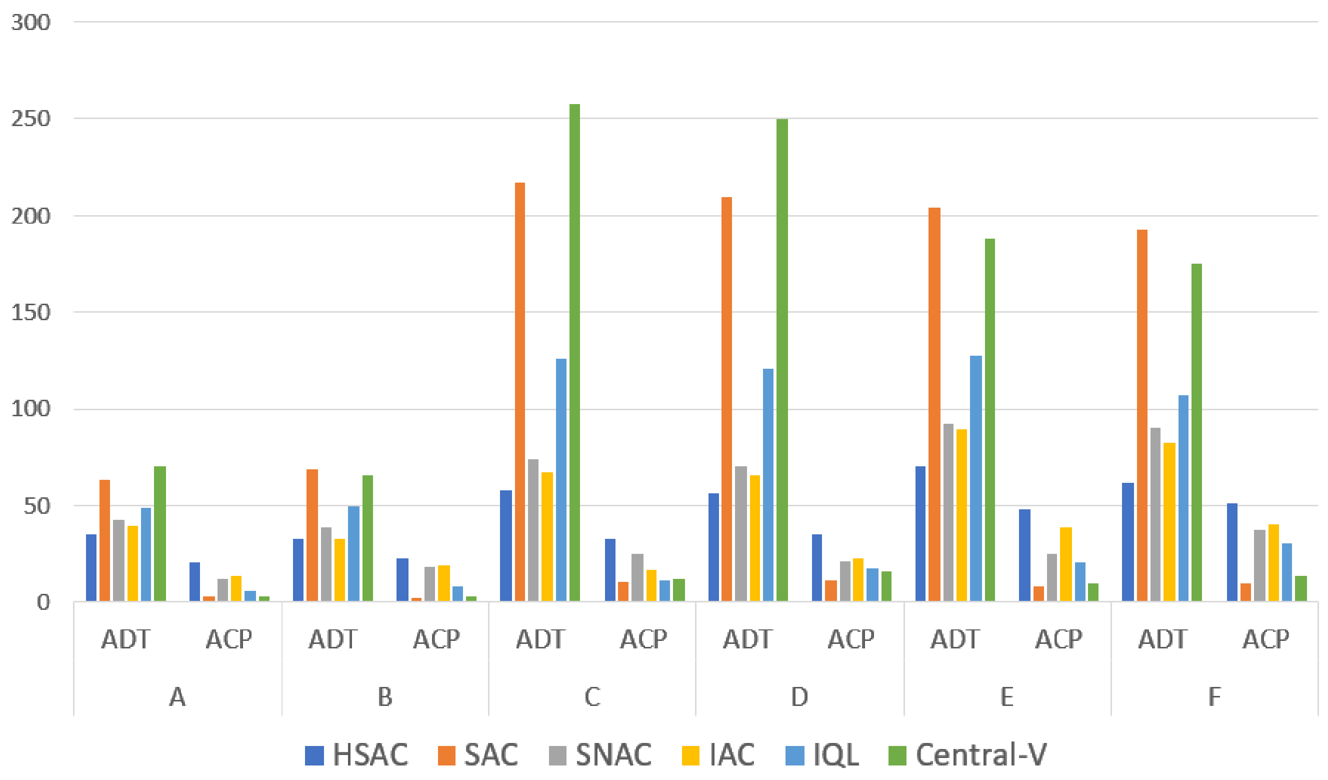


FIGURE 5. The results of metrics.

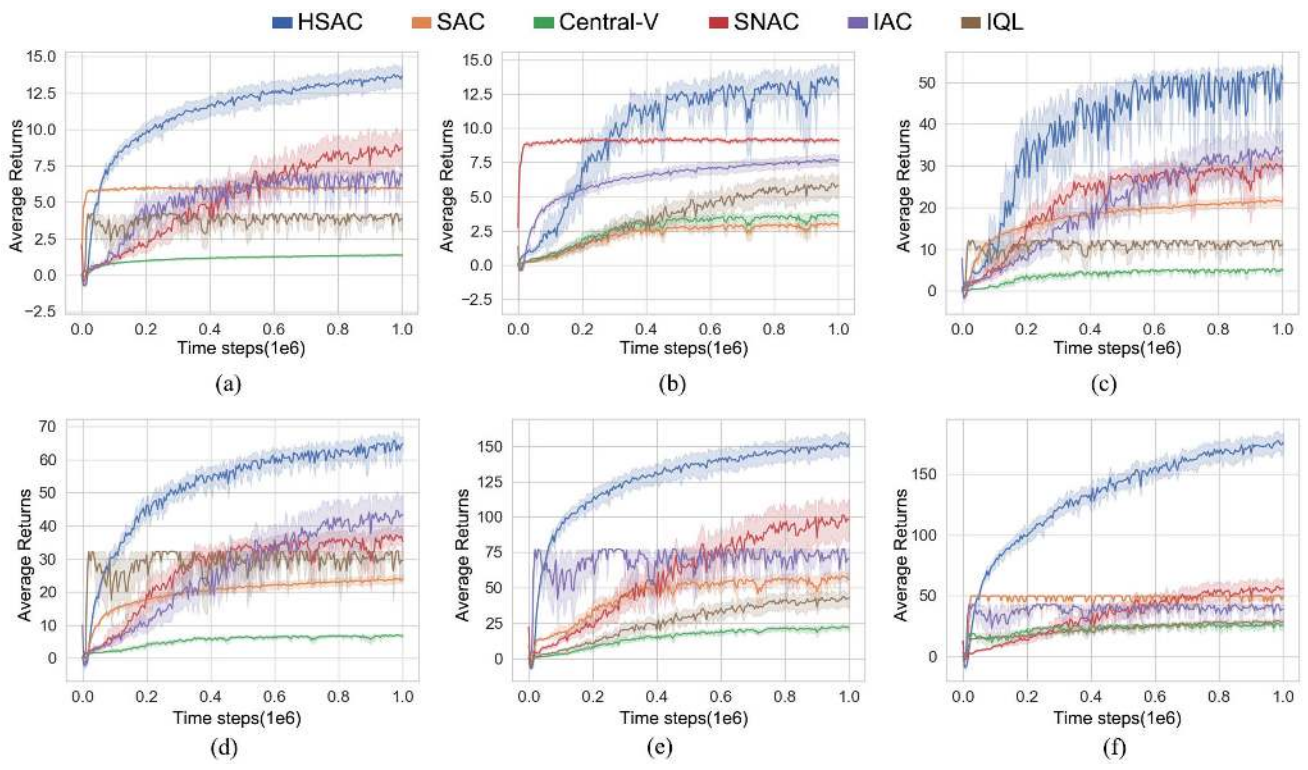


FIGURE 6. The average reward of different algorithms.Description of the average returns of algorithms in episodes of different 5 random seeds.

- **Soft Actor-Critic(SAC):** SAC is an off-policy actor-critic deep reinforcement learning algorithm based on the maximum entropy reinforcement learning framework. By combining off-policy updates with a stable stochastic actor-critic formulation, SAC achieves state-of-the-art performance.

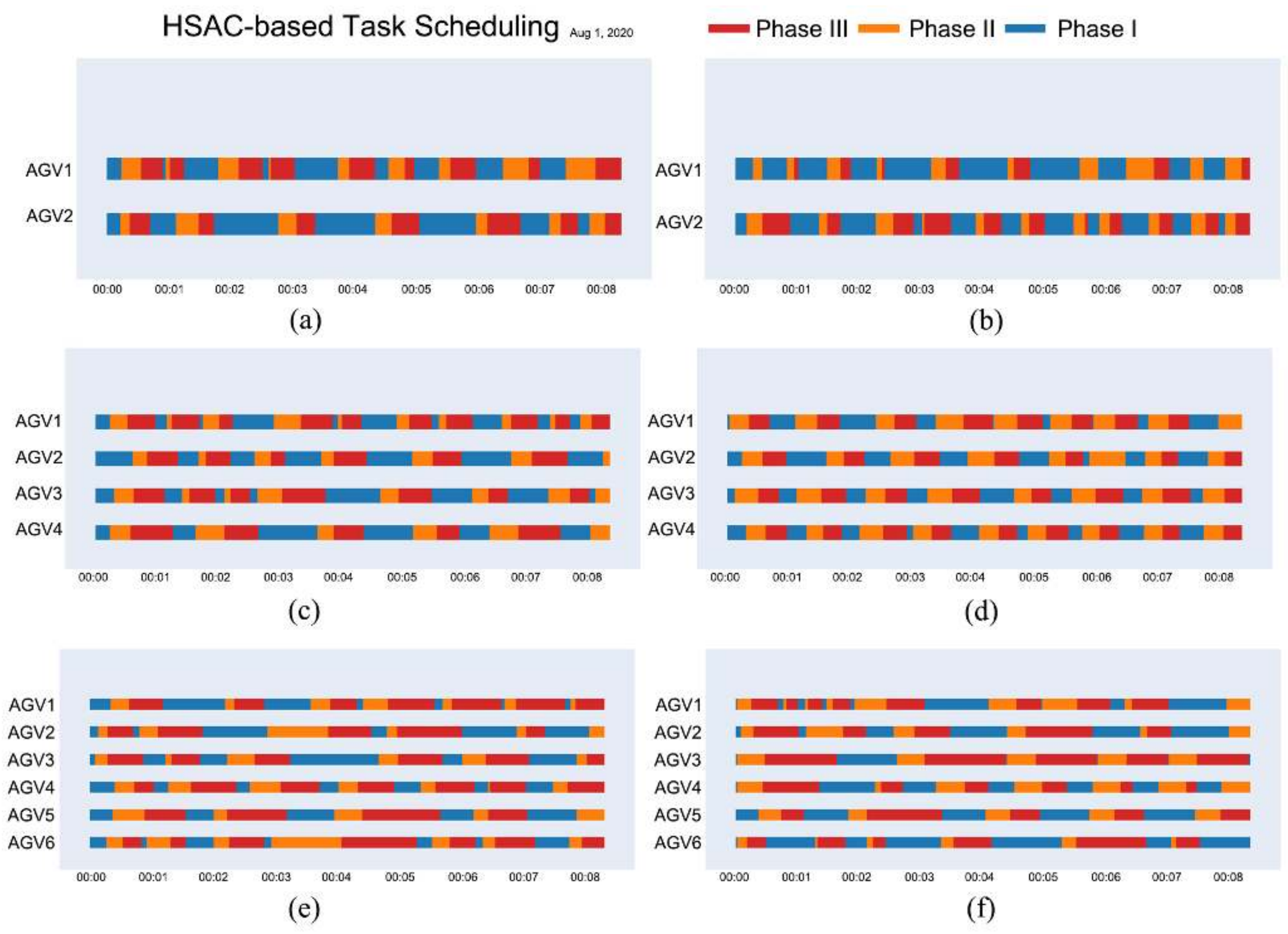


FIGURE 7. The Gantt charts of the HSAC with different warehouse environments.

TABLE 2. Hyperparameters used for HSAC results.

Hyperparameter	Value
Layers	2 fully connected layers
Fully connected layer hidden units	128
Batch size	64
Replay buffer size	131072
Discount rate	0.99
Steps per learning update	4
Learning iterations per round	1
Learning rate	0.0003
Optimizer	Adam
Weight initializer	He
Fixed network update frequency	1024
Loss	Mean squared error
Initial random steps	1024
Entropy target	0.98*(-log(1/|A|))

- **Shared Network Actor-Critic(SNAC):** SNAC trains a single shared policy among all agents. During training, the policy and value loss gradients are summed and used to optimize the shared parameters. Importance sampling is not required since all trajectories are on-policy.
- **Independent Actor-Critic(IAC):** Each agent has its policy network and it's trained separately only using its own experience. IAC uses an actor-critic algorithm for each agent and treating other agents as part of the environment. Arguably, independent learning is one of the most straightforward approaches to MARL and serves as a reasonable baseline due to its simplicity.
- **Independent Q-Learning(IQL):** Each agent has a decentralized state-action value function that is conditioned only on the local history of observations and actions of each agent. Each agent receives its local history of observations and updates the parameters of the Q-value network using the standard Q-learning optimization.
- **Central-V:** Central-V is an actor-critic algorithm in which the actor approximates the individual policy and the critic learns a joint state value function. It extends existing on-policy actor-critic algorithms, such as A2C or PPO, by applying centralized critics conditioned on the state of the environment rather than the individual history of observations.

In order to better evaluate the performance of the algorithm in scheduling tasks, two metrics are proposed: average distance traveled by the robot per task(ADT) and average task accomplished per epoch(ACP). The results of metrics in three different experiments as shown in Figure 5.

TABLE 3. Max average return over 5 trials of 1 million time steps.

	HSAC	SAC	SNAC	IAC	IQL	Central-V
A	**12.78**	5.86	8.62	6.33	4.24	1.85
B	**16.63**	2.94	9.88	8.62	5.46	2.93
C	**52.04**	24.23	36.79	40.17	12.26	5.38
D	**58.31**	26.76	38.23	45.07	15.81	6.90
E	**156.57**	56.79	60.24	42.88	32.17	25.90
F	**189.53**	52.55	69.30	43.65	36.34	28.92

A comparative analysis of the above results leads to the following conclusions.

Since the SAC algorithm does not introduce subgoals and internal rewards, it is difficult to learn the best strategy for completing the task in a sparsely rewarded experimental environment, resulting in the robot wasting a lot of time exploring the environment, whereas with HSAC, which introduces subgoals and internal rewards, the robot is better able to find paths that are closer to the subgoals through internal reward feedback, thus achieving the better results. For six different experimental contexts, the larger the map size, the more sparse the feedback from the environment, and the more difficult it was for the method to learn strategies for completing the task.

In terms of the average path the robot takes to complete each task, we expect the robot to complete the task in a shorter distance, and HSAC achieves the best results with a larger improvement. When the number of robots is constant and the number of assignable tasks increases, the average path required to complete each task decreases. When the map is larger, the average path required to complete each task increases.

In terms of the idle rate of the robot, the smaller the idle rate of the robot, the better. From the experimental results, it can be seen that the best results were achieved for the idle rate of HSAC in each context. When the number of robots remained constant and the number of assignable tasks increased, the idle rate of the robots decreased.

In terms of the number of tasks completed per episodes, we expected the bot to complete more tasks in an episodes. However, the robot does not complete more number of tasks when the map is maximal. Our proposed algorithm achieves better results.

In practical applications, unmanned warehouses generally take the number of order picks completed per hour as an evaluation index, and with each time step as 1 second, it is possible to obtain an average of 61.73 time steps per order in the F scenario, which would result in 58.318 completed orders per hour.

The goal of our experiment is to understand how the sample complexity and stability of our method compares with prior deep reinforcement learning algorithms. For scheduling tasks with high decision space, the HSAC technique with a hierarchical structure can improve scheduling efficiency. As shown in Table 3 and Figure 6 are the learning curves and maximum average feedback of the robot under 6 experimental scenarios. The experiments were run under 5 random seeds for more than 1 million time steps and the parameters were updated every 5000 steps for a total of 1024 episodes performed. The algorithms do not use the intrinsic reward function proposed in this paper except for the HSAC algorithm.

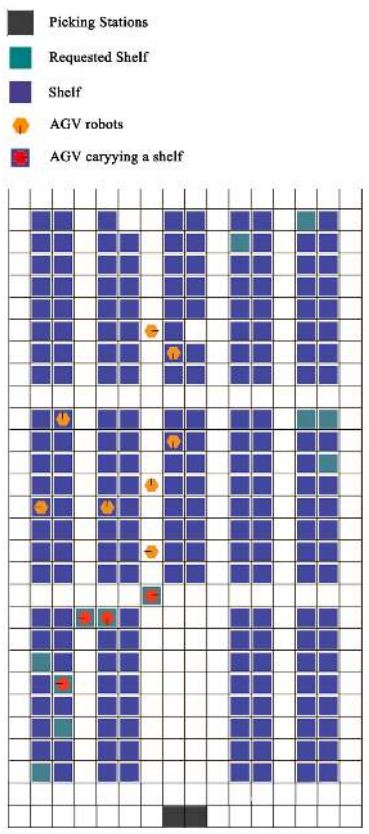


FIGURE 8. The road layout of unmanned warehouse in supplementary experiments.

From the learning curve in Figure 6, it can be seen that the HSAC algorithm reward value changes less when the Agent's task size and map size change, and can better adapt to changes in the Agent's task size, i.e., it can better adapt to the dynamic external environment. The HSAC algorithm works best and converges efficiently.

During the order picking process of task assignment and path planning, the robot's state can be divided into three types: Phase 1, Phase 2, and Phase 3. Phase 1 is the process of the robot finding the location of the shelf from its own position when it receives the task. Phase 2 is the process of the robot picking up the shelves to locate the picking table after it has found the location of the shelves. Phase 3 is the process of the robot returning the shelf to its original location after the order picking is complete. Figure 7 shows a Gantt chart of the robot working at different stages in different test scenarios with a maximum number of 500 steps. For the robot, the length of Phase 2 and Phase 3 should be similar, while the length of Phase 1 is different due to the different starting position of the robot at each time it receives the task. We want the algorithm to be able to spend close time in Phase 2 and Phase 3. Therefore, it can be seen that the robots

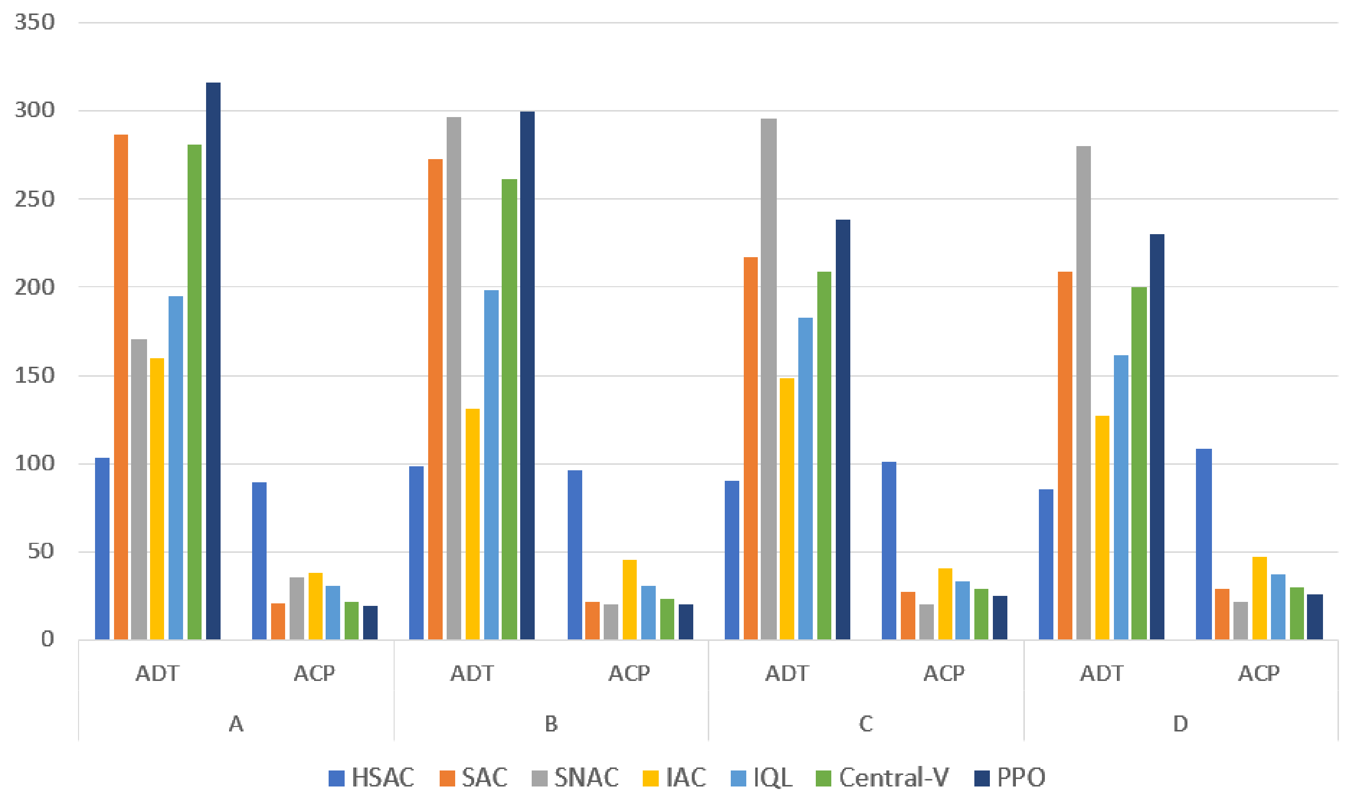


FIGURE 9. The results of supplementary experiment.

TABLE 4. The parameters of the supplementary environment.

	Tasks	Robots
A	12	12
B	24	12
C	15	15
D	30	15
E	18	18
F	36	18
G	20	20
H	40	20

with the HSAC algorithm have a lower idle rate and stable performance.

Through quantitative and qualitative analysis of various evaluation metrics, this paper designs a hierarchical structure based on the SAC algorithm and performs simulation experiments. The experimental results show that the internal reward mechanism based on subgoals can effectively learn the behavioral strategies of complex task assignments at different stages of order assignment and path planning, and the maximum entropy algorithm can effectively enhance robustness and exploration of the environment. Therefore, comparing the simulation results, the proposed HSAC algorithm improves significantly, which further verifies the superiority of the algorithm and proves that the HSAC algorithm is more robust and stable in performance.

C. SUPPLEMENTARY EXPERIMENTS

To verify the performance of the method proposed in this paper, we performed additional experiments in the simulation environment. In addition, we added experimental results for PPO [45], an off-policy RL algorithm for state-of-the-art performance. The road network layout of the experiment is shown in Figure 9 and Figure 10, and it can be seen that the size of the map is close to the size of the actual scene. The supplementary experiments was carried out under the warehouse layout as shown in Figure 8. As shown in Table 4, we have conducted experiments for the cases of 12, 15, 18, and 20 number of robots.

From the experimental results, it can be seen that HSAC achieves optimal results for both. For both ADT and ACP, HSAC reduced the average robot walking distance per task and the time required to complete the task compared to SAC. Because the HSAC algorithm uses a hierarchical mechanism to divide the path planning process of the task into three phases, where the intrinsic reward-based directed exploration is performed in each phase. The average robot distance traveled per task and the time required to complete the task were less effective in the SAC algorithm before improvement because the SAC algorithm trained independent policies for each robot in reward-sparse dynamic environment exploration, and the optimal action selection policy could not be learned in a limited amount of time.

The time required for each task completion was slightly higher than the average robot walking distance due to the fact that the robot's actions in completing the task included turning left, turning right, and load/unload shelves in addition to moving forward. For ACP, the algorithm is most effective

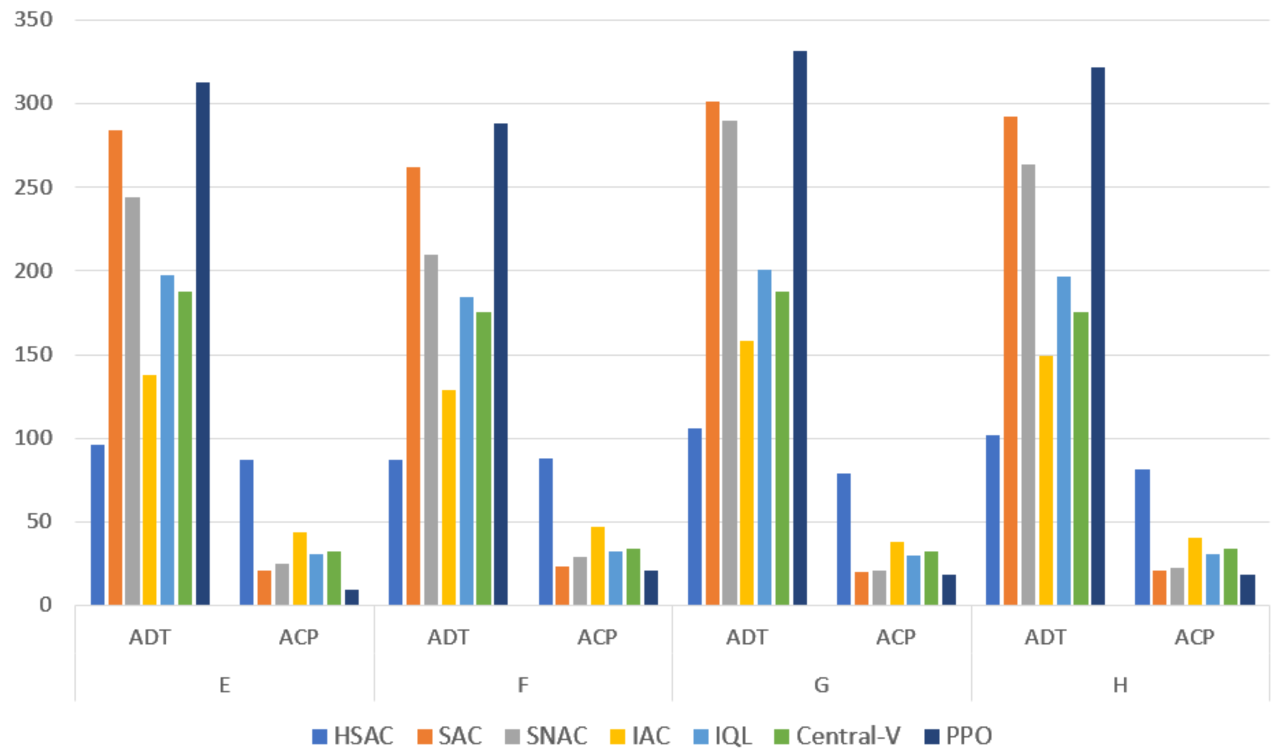


FIGURE 10. The results of supplementary experiment.

TABLE 5. The results of HSAC in the supplementary environment.

	A	B	C	D
ADT	103.77	98.62	90.12	85.52
ACP	89.83	96.27	101.08	108.32
	E	**F**	**G**	**H**
ADT	96.32	87.17	105.89	101.44
ACP	87.16	87.9	79.02	81.65

when the number of robots is 15. This is because when the number of robots increases, the probability of congestion of robots at the target location also increases, so it is not the case that the algorithm is more effective when the number of robots is higher.

Table 5 shows the performance of the HSAC algorithm for ADT and ACP with different number of robots and number of tasks. It can be seen that the effectiveness of the algorithm improves as the number of robots increases, but not infinitely. And the effectiveness of the algorithm starts to decrease after the number of robots 15, because too many robots will make the warehouse crowded, which is in line with the expected result. Although adding robots can improve efficiency to some extent, the performance of the algorithm remains stable within a certain range.

Based on the experimental results, we can conclude that, first, the HSAC algorithm is optimal even in a large map environment with sparse rewards. Second, the average number of tasks completed by the HSAC algorithm per robot per iteration, 7.49, 8.02, 8.42, 9.02, 7.26, 7.33, 6.59 and 6.804, respectively, basically maintained relatively stable results and verified the stability of the algorithm.

VI. CONCLUSION

This work proposed a hierarchical soft actor-critic, an off-policy maximum entropy hierarchical deep reinforcement learning algorithm and applied the method to multi-logistic robot task allocation. The method retains the benefits of soft actor-critic and hierarchical reinforcement learning. The proposed HSAC algorithm takes robot environment observations as input of neural network to directly control AGV robots to shuttle between shelves and picking stations in a dynamic environment. The performance of the method with other algorithms in a multi-robot environment is evaluated in a simulation environment. The experimental results show that the method can teach multi-logistics robots to work together to complete order picking tasks with high efficiency.

However, this paper only considers AGV systems for multi-robot picking work in unmanned warehouses. If we want to apply it to other AGV robot systems, it remains to be further investigated whether the task assignment system needs to be modified and whether it will produce a different assignment effect. In addition, the data set for the training model in this paper is from the simulation environment, we did not collect data in the real AGV robot system in order to save the research cost, and the subsequent research can consider to train and modify the neural network of the model by collecting data from the real AGV robot system.

REFERENCES

[1] Y. Fu and J. Zhu, "Operation mechanisms for intelligent logistics system: A blockchain perspective," *IEEE Access*, vol. 7, pp. 144202–144213, 2019.

[2] X. Zhang, "Intelligent storage system based on 'Internet+' logistics," in *Proc. Int. Conf. Appl. Techn. Cyber Secur. Intell.*, Anhui, China, Jun. 2019, pp. 811–816.

[3] F. Xue, H. Tang, Q. Su, and T. Li, "Task allocation of intelligent warehouse picking system based on multi-robot coalition," *KSII Trans. Internet Inf. Syst.*, vol. 7, pp. 3566–3582, Jul. 2019.

[4] J. Zeng, L. Qin, Y. Hu, Q. Yin, and C. Hu, "Integrating a path planner and an adaptive motion controller for navigation in dynamic environments," *Appl. Sci.*, vol. 9, no. 7, p. 1384, Apr. 2019.

[5] M. B. Radac and R. E. Precup, "Data-driven model-free tracking reinforcement learning control with VRFT-based adaptive actor-critic," *Appl. Sci.*, vol. 9, no. 9, p. 1807, Apr. 2019.

[6] H. Bae, G. Kim, J. Kim, D. Qian, and S. Lee, "Multi-robot path planning method using reinforcement learning," *Appl. Sci.*, vol. 9, no. 15, p. 3057, Jul. 2019.

[7] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," 2020, *arXiv:2001.11231*. [Online]. Available: http://arxiv.org/abs/2001.11231

[8] T.-M. Wang, Y. Tao, and H. Liu, "Current researches and future development trend of intelligent robot: A review," *Int. J. Autom. Comput.*, vol. 15, no. 5, pp. 525–546, Apr. 2018.

[9] A. J. Moshayedi, L. Jinsong, and L. Liao, "AGV (automated guided vehicle) robot: Mission and obstacles in design and performance," *J. Simul. Anal. Novel Technol. Mech. Eng.*, vol. 12, pp. 5–18, Nov. 2019.

[10] Y. Wu and D. Ge, "Key technologies of warehousing robot for intelligent logistics," in *Proc. 1st Int. Symp. Manage. Social Sci. (ISMSS)*, Wuhan, China, Apr.2019, pp. 13–14.

[11] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, p. 484, Jan. 2016.

[12] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[13] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," Jan. 2018, *arXiv:1801.01290*. [Online]. Available: http://arxiv.org/abs/1801.01290

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," Sep. 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971

[16] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A survey of deep reinforcement learning in video games," Dec. 2019, *arXiv:1912.10944*. [Online]. Available: http://arxiv.org/abs/1912.10944

[17] D. Zhang, X. Han, and C. Deng, "Review on the research and practice of deep learning and reinforcement learning in smart grids," *CSEE J. Power Energy Syst.*, vol. 4, no. 3, pp. 362–370, Sep. 2018.

[18] S. Bhagat, H. Banerjee, Z. Ho Tse, and H. Ren, "Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges," *Robotics*, vol. 8, no. 1, p. 4, Jan. 2019.

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, New York, NY, USA, Jun. 2016, pp. 1928–1937.

[20] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, Cambridge, MA, USA, Nov. 1999, pp. 1008–1014.

[21] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," Feb. 2018, *arXiv:1802.09477*. [Online]. Available: http://arxiv.org/abs/1802.09477

[22] A. OroojlooyJadid and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," Aug. 2019, *arXiv:1908.03963*. [Online]. Available: http://arxiv.org/abs/1908.03963

[23] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[24] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang, "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play StarCraft combat games," Mar. 2017, *arXiv:1703.10069*. [Online]. Available: http://arxiv.org/abs/1703.10069

[25] J. Zhou and W. Xu, "End-to-end learning of semantic role labeling using recurrent neural networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics, 7th Int. Joint Conf. Natural Lang. Process.*, Beijing, China, Jul. 2015, pp. 1127–1137.

[26] H. Mao, Z. Gong, Y. Ni, and Z. Xiao, "ACCNet: Actor-coordinator-critic net for 'Learning-to-communicate' with deep multi-agent reinforcement learning," Jun. 2017, *arXiv:1706.03235*. [Online]. Available: http://arxiv.org/abs/1706.03235

[27] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," Feb. 2018, *arXiv:1802.05438*. [Online]. Available: http://arxiv.org/abs/1802.05438

[28] R. Lowe, Y. Wu, A. Tamar, A. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 6382–6393.

[29] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," May 2017, *arXiv:1705.08926*. [Online]. Available: http://arxiv.org/abs/1705.08926

[30] M. Kim, D.-K. Han, J.-H. Park, and J.-S. Kim, "Motion planning of robot manipulators for a smoother path using a twin delayed deep deterministic policy gradient with hindsight experience replay," *Appl. Sci.*, vol. 10, no. 2, p. 575, Jan. 2020.

[31] N. Liu, Y. Cai, T. Lu, R. Wang, and S. Wang, "Real–sim–real transfer for real-world robot control policy learning with deep reinforcement learning," *Appl. Sci.*, vol. 10, no. 5, p. 1555, Feb. 2020.

[32] J. Kim and H. Lee, "Adaptive human–machine evaluation framework using stochastic gradient descent-based reinforcement learning for dynamic competing network," *Appl. Sci.*, vol. 10, no. 7, p. 2558, Apr. 2020.

[33] C. Liu, Q. Mao, X. Chu, and S. Xie, "An improved A-Star algorithm considering water current, traffic separation and berthing for vessel path planning," *Appl. Sci.*, vol. 9, no. 6, p. 1057, Mar. 2019.

[34] D. Laha and J. N. D. Gupta, "An improved cuckoo search algorithm for scheduling jobs on identical parallel machines," *Comput. Ind. Eng.*, vol. 126, pp. 348–360, Dec. 2018.

[35] W. Deng, J. Xu, and H. Zhao, "An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem," *IEEE Access*, vol. 7, pp. 20281–20292, 2019.

[36] Z. Jia, J. Yan, J. Y. T. Leung, K. Li, and H. Chen, "Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities," *Appl. Soft Comput.*, vol. 75, pp. 548–561, Feb. 2019.

[37] A. A. R. Hosseinabadi, J. Vahidi, B. Saemi, A. K. Sangaiah, and M. Elhoseny, "Extended genetic algorithm for solving open-shop scheduling problem," *Soft Comput.*, vol. 23, no. 13, pp. 5099–5116, Jul. 2019.

[38] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, "Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 39974–39982, 2019.

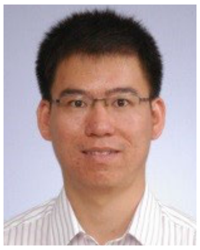
[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[40] Q. Zeng, Z. Yang, and X. Hu, "A method integrating simulation and reinforcement learning for operation scheduling in container terminals," *Transport*, vol. 26, no. 4, pp. 383–393, Jan. 2012.

[41] J. Shahrabi, M. A. Adibi, and M. Mahootchi, "A reinforcement learning approach to parameter estimation in dynamic job shop scheduling," *Comput. Ind. Eng.*, vol. 110, pp. 75–82, Aug. 2017.

[42] J. Rafati and D. C. Noelle, "Learning representations in model-free hierarchical reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.* Honolulu, HI, USA: Hilton Hawaiian Village, 2019, pp. 10009–10010.

[43] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst.* Barcelona, Spain: Centre Convencions Internacional Barcelona, Dec. 2016, pp. 3675–3683.
[44] F. Christianos, L. Schäfer, and S. V. Albrecht, "Shared experience actor-critic for multi-agent reinforcement learning," Jun. 2020, *arXiv:2006.07169*. [Online]. Available: http://arxiv.org/abs/2006.07169
[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," Jun. 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347



HENGLIANG TANG received the B.Sc. and Ph.D. degrees from the Beijing University of Technology, in 2005 and 2011, respectively. He is currently a Professor with the Beijing Wuzi University. His main research interests include computer vision and the IoT information technology.

ANQI WANG received the B.Sc. degree from Beijing Wuzi University, in 2016, where she is currently pursuing the M.Sc. degree. Her main research interests include deep reinforcement learning and the IoT information technology.

FEI XUE received the B.Sc. degree from the University of Jinan, in 2006, the M.Sc. degree from the Taiyuan University of Science and Technology, in 2011, and the Ph.D. degree from the Beijing University of Technology, in 2016, respectively. He is currently an Associate Professor with the Beijing Wuzi University. His main research interests include computational complexity theory and optimization.

JIAXIN YANG received the B.Sc. degree from Beihang University, in 2017. He is currently pursuing the M.Sc. degree with the Beijing Wuzi University. His main research interests include computer vision and the IoT information technology.

YANG CAO received the B.Sc. and M.Sc. degrees from the Taiyuan University of Science and Technology, in 2011 and 2015, respectively, and the Ph.D. degree from the Beijing University of Technology, in 2019. He is currently a Lecturer with Beijing Wuzi University. His main research interests include machine learning and big data analysis.