Entrophy Enhanced Multiagent Coordination Based on Hierarchical Graph Learning for Continuous Action Space

Yining Chen, Ke Wang, Guanghua Song, and Xiaohong Jiang

Abstract—In most existing studies on large-scale multiagent coordination, the control methods aim to learn discrete policies for agents with finite choices. They rarely consider selecting actions directly from continuous action spaces to provide more accurate control; therefore, they are normally unsuitable for more complex tasks. To solve the control issue of large-scale multiagent systems with continuous action spaces, we propose a novel multiagent reinforcement learning (MARL) approach named entropy-enhanced hierarchical graph continuous action multiagent coordination control method (EHCAMA) to derive stable continuous policies, by constructing a new network architecture in an actor-critic framework. By optimizing policies with maximum entropy learning, agents improve their exploration ability in training and acquire an excellent performance in execution. Further, we employ hierarchical graph attention networks (HGATs) and gated recurrent units (GRUs) to improve the scalability and transferability of our method. We simulate the performance of EHCAMA for cooperative tasks with both homogeneous and heterogeneous agents, and compare it with soft actor-critic-hierarchical graph recurrent network (SAC-HGRN), hierarchical graph attention-based multiagent actor-critic (HAMA), actor hierarchical attention critic (AHAC), and adaptive and gated graph attention network (AGGAT)-Comm. The experimental results show that our method consistently outperforms the baselines in large-scale multiagent scenarios.

Index Terms—Continuous action space, deep reinforcement learning (DRL), maximum entropy learning, multiagent.

I. INTRODUCTION

THE multiagent system has been applied in various domains during the last decade [1], such as compute games [2], [3], smart grids [4], [5], and the unmanned aerial vehicle (UAV) navigation [6], [7]. By achieving consensus among agents, the multiagent system can coordinate them to solve complex tasks efficiently in the real world. However, most previous works in large-scale multiagent systems simplify the control problem of agents by discretizing their actions as a small number of choices. Although such methods may help agents to learn a stable policy, it is still hard for discrete action spaces to model their actions accurately in some complex scenarios. To address the challenges in accurate control, we focus on large-scale multiagent systems with continuous action space and employ multiagent reinforcement learning (MARL) to coordinate agents for various scale tasks.

In standard MARL, agents learn their policies in two ways—off-policy and on-policy. The off-policy approaches [8], [9], [10] utilize the transitions stored in replay buffers to optimize deterministic policies, which brings good sample efficiency. However, they cannot prevent agents from converging to nonoptimal policies in continuous action spaces, because they use the maximization based on Q-learning [11] for training. As a result, the instability in these policies limits the applicability of off-policy learning in large-scale multiagent systems. The on-policy approaches [12], [13], [14], [15] derive more stable and robust policies by outputting the distribution of actions. Since they control the exploration by the stochasticity of their policies, the agents obtain better performance when executing real-world tasks. Unfortunately, these approaches require new samples for updating policies, thus they suffer from poor sample complexity. Moreover, this shortcoming becomes extremely severe as the scale of multiagent systems increases since agents need more samples to learn an effective policy.

We herein propose an MARL approach named entropy-enhanced hierarchical graph continuous action multiagent coordination control method (EHCAMA), which contains a graph learning network and an entropy-enhanced optimization for training. To derive efficient and stable policies in continuous action spaces, we implement EHCAMA on an actor-critic framework and train the stochastic actor with an off-policy training strategy. We introduce maximum entropy learning [16], [17] to augment the MARL training process with entropy maximization, which provides intelligent exploration for agents and helps them find the optimal policies. At the same time, we adopt hierarchical graph attention networks (HGATs) [18] and gated recurrent units (GRUs) [19] to summarize the information from agents’ observations, neighbors, and memories, improving its scalability and transferability in large-scale environments.

The rest of this article is organized as follows. We discuss related work in Section II and introduce the background about
MARL, HGAT, and maximum entropy learning in Section III. In Section IV, we present a detailed description of our method. The experimental results are shown in Section V. Finally, we conclude the article in Section VI.

II. RELATED WORK

Deep reinforcement learning (DRL) is an efficient paradigm to address the coordination issues in multiagent learning. It combines reinforcement learning (RL), which models the interaction between agents and environments, and deep neural networks (DNNs) that process high-dimensional observation data. The value-based DRL algorithms like deep Q-network (DQN) [20] compute Q-value for each action and select the one with the maximum Q-value. Therefore, they are only applicable to discrete action spaces. The actor-critic algorithms use a separate actor network to approximate the policies and directly adjust its parameters to maximize the expected rewards, which are feasible in continuous action space. Deep deterministic policy gradient (DDPG) [8], a popular actor-critic method, employs off-policy learning to derive deterministic policies for agents. However, it is hard for DDPG to resolve complex tasks since its policies are unstable and nonrobust to hyperparameter settings. On the contrary, the on-policy algorithms, such as trust region policy optimization (TRPO) [13] and proximal policy optimization (PPO) [14], [15], train stochastic policies with newly collected samples, which brings stability to agents but reduces their sample efficiency. The authors [21] studied off-policy stochastic control and presented a zero-step variant of stochastic value gradients (SVG(0)) that optimizes the policy with a experience database. Haarnoja et al. [22], [23] introduced maximum entropy learning into policy gradient method and proposed an off-policy algorithm named soft actor-critic (SAC). By incorporating an entropy term in the maximum expected return objective, it provides tractable stochastic policies for agents and encourages them to explore. In [24], the authors proposed a SAC-based algorithm for autonomous vehicles, which employs a bootstrapped architecture [25] to further improve the exploration.

Unlike the above RL methods that only consider single-agent scenarios, MARL focuses on learning decentralized policies and controls the multiagent system by achieving consensus among agents. A widely used solution in MARL is to learn each agent’s policy on a framework of centralized training and decentralized execution (CTDE), such as multiagent deep deterministic policy gradient (MADDPG) [26] and countfactual multiagent (COMA) [27]. These methods derive a decentralized policy for each agent with the gradient from their centralized critics. Dynamic decomposed MADDPG (DD-MADDPG) [28] uses two critic networks to learn the global Q-function and the local Q-function simultaneously, reducing the influence of other agents’ policies in training. It also applies a prioritized experience replay (PER) [29] based on TD-errors and time to accelerate convergence of policies. Following SAC, multiactor-attention-critic (MAAC) [30] adopts maximum entropy learning to control the exploration and avoid converging to nonoptimal policies. In addition, it constructs a centralized critic employing an attention mechanism [31] to selectively use information from the multiagent system, offering enhanced scalability to the number of agents. Actor hierarchical attention critic (AHAC) [32] labels agents according to their attributes and then employs a multthead hierarchical attention network (MHA) with an encoder based on the recurrent neural network (RNN) to extract the feature information from their action-observation history. However, these CTDE methods face two major challenges when applying them to a large-scale environment. First, it is difficult to optimize the individual actor network for each agent in large-scale multiagent systems. Second, due to lack of transferability, these methods must retrain networks before executing new tasks, rather than transferring trained policies.

To address the scalability and transferability issues, some MARL methods apply graph neural networks (GNNs) [33] with shared parameters to model the relationship between agents and derive decentralized policies for a set of homogeneous agents. Deep graph network (DGN) [34] represents the environment state as a graph and adopts graph attention networks (GATs) [35] to extract the relationships among agents. The authors [36], [37] use hard-attention to filter irrelevant agents and soft-attention to quantify the importance of the relationships between the remaining agents when processing received messages. The authors [38] proposed an adaptive and gated graph attention network (AGGAT) that adaptively selects neighbors through hard attention network and leverages a gated attention layer to refine the attention results. The AGGAT-based MARL method shows high performance and generalization ability in various scale environments.

In some real-world scenarios, the multiagent system may involve heterogeneous agents. However, the above GNN-based methods are designed to achieve consensus among homogeneous agents. They cannot adequately represent the various relationships by using a single graph. To address the coordination issue in heterogeneous environments, hierarchical graph attention-based multiagent actor-critic (HAMA) [18] categorizes all the agents into different groups using prior knowledge and employs a HGAT to extract the interagent and intergroup relationships. In [39], Du et al. presented a MARL approach named MA-HA for heterogeneous environments. They applied a heterogeneous GAT to selectively aggregate the information of neighbors and subsequently computed the individual Q-values of each agents. The authors [40] applied grouped attention mechanism to extract the features of entities in different groups and optimized each agent network by using the mutual information in a value decomposition framework [41]. Our earlier work [42] presented a scalable and transferable MARL method for large-scale multiagent systems in mixed cooperative-competitive environments. We used HGAT to extract features from observations while introducing interagent communication to allow adjacent agents to exchange high-dimensional vectors aggregated from observations. Meanwhile, it records long-term historical information in its recurrent unit. These improvements provide outstanding performance in large-scale cooperative and competitive tasks with discrete action spaces. In another previous work [43], we designed hierarchical
graph recurrent network (HGRN), a novel network structure based on HGAT and RNN, as well as SAC-HGRN, a MARL algorithm that integrated the proposed network with SAC. By optimizing maximum entropy objective, SAC-HGRN shows its superiority in both homogeneous and heterogeneous environments with discrete action spaces.

Our proposed EHCAMA differs from the previous works [42], [43] in which: 1) Jain et al. [42] employ $\epsilon$-greedy or $\epsilon$-categorical strategy to provide exploration in discrete action spaces for agents, whereas EHCAMA derives stochastic policies for continuous action spaces and optimizes them with maximum entropy learning. 2) Ren et al. constructed the output layer in SAC-HGRN’s actor network for discrete actions. We design a new recurrent unit that outputs the distribution of continuous actions and improve training process to learn better policies for continuous action spaces, including a graph learning network based on HGAT and SAC-HGRN shows its superiority in both homogeneous and heterogeneous environments.

B. HGAT

HGAT is a novel network structure in MARL, which aims to extract the relationships among different types of agents. The idea of HGAT is to represent the global state as a graph and process it through a network that stacks multiple GATs hierarchically. After receiving observations from the environment, agent $i$ uses GAT to aggregate embedding vectors $e_j$ from its neighbors in each group $l$ into $e_i^l = \sum_{j \in N^l_i} \alpha_{ij} W^l_{ij} e_j$, where $N^l_i$ denotes the set of agent $i$’s neighbors in group $l$. The attention weight $\alpha_{ij} \propto \exp(e_j^T W^l_{ij} W^l_{ij} e_j)$, $W^l_{ij}$, and $W^l_{ij}$ are the matrices that transform embedding vectors into “query,” “key,” and “value,” respectively. After that, another GAT aggregates $e_i^l$ from all groups into $e_i$, which summarizes the local states of its neighbors and their relationships.

C. Maximum Entropy Learning

In maximum entropy learning, agents aim to learn a soft value function that combines each agent’s total expected return and action entropy at each state. They train their policy $\pi$ with the object $J(\pi) = \sum_{t=0}^{T-1} E[r(t) + \alpha H(\pi(\cdot|s(t)))|s(t)],$ where $H(\pi(\cdot|s(t)))$ is an entropy term and $\alpha$ is the temperature hyperparameter that controls the stochasticity of $\pi$. Since they explore more widely, agents can avoid converging to nonoptimal policies and improve their performance.

IV. METHODS

In this section, we introduce our coordination control solution for large-scale multiagent systems with continuous action spaces, including a graph learning network based on HGAT and a maximum entropy MARL approach named EHCAMA.

A. Graph Representation and Network Architecture

Fig. 1 shows the graph representation of the global state of the environment. The first step is to cluster $M$ entities (including agents and landmarks) in the environment into $K$ groups according to their features. Because of the high differences between these heterogeneous entities, it is more efficient for agents to treat them separately. The second step is to construct an observation graph $G^O$ and a communication graph $G^C$, where the nodes denote entities while the edges mean two of them are in their observation range and communication range, respectively. In the third step, EHCAMA transforms the global

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 2. Network architecture of (a) the actor and (b) the critic in EHCAMA.

Fig. 3. Structure of the recurrent unit in the stochastic actor.

In the interagent communication stage, each agent $i$ shares the information from $o_i$, sending $e_i'$ to its neighbors, which helps it to cooperate better with other agents. Upon reception of message $m_i$, the second HGAT layer extracts features from $m_i$ and uses $A^c$ to calculate $e''_i$ like the first layer. To accelerate converging, we concatenate $e'_i$ and $e''_i$ into a vector and input it into the recurrent unit.

In some prior actor-critic approaches for continuous action spaces, agents learn deterministic policies to determine their actions, such as DDPG. Unfortunately, since the deterministic actors only use an additional random noise for exploration, it is hard for them to stabilize [46]. Thus, we introduce stochastic policies with Gaussian distributions to provide an intelligent exploration, thereby improving the performance of EHCAMA either in the training stage or in the executing stage. As shown in Fig. 3, we design a new recurrent unit that consists of a GRU layer and two fully connected layers. The GRU layer maintains the hidden states $h^\pi_i$ as the memory, which helps agent $i$ to recall the previous observations and record the new information from input vectors. After that, agent $i$ respectively calculates $\mu_i$ and $\sigma_i$ by two separate linear transforms and obtains the action distribution $\pi()$. In practice, we squash the Gaussian samples into $(-1, 1)$ as $a_i = \text{tanh}(u_i)$, where $u_i \sim N(\mu_i, \sigma_i^2)$ [23].

Similar to the actor network, the critic also uses HGAT layers to aggregate the information from observations and neighbors. Unlike our previous work [42], we cannot compute Q-values for each choice in continuous action spaces. Thus, the critic uses an individual linear encoder to process actions $a_i$, as shown in Fig. 2(b). The critic’s recurrent unit contains a GRU layer and a fully connected layer, which also maintains the hidden states. To reduce the overoptimism for Q-values, we use twin critic networks with different parameters and represent them as $Q^l_i$ and $Q^2_i$ for each group $l$ [9]. Their hidden states for each agent $i$ are denoted as $h^Q_i$ and $h^{Q_2}_i$, respectively.

B. Training Process With Entropy-Enhanced Optimization

In standard actor-critic methods, the training strategy aims to maximize the total expected return $R$ of the agent. However, it cannot be used to train stochastic policies in EHCAMA because it does not consider the optimization of exploration. This fatal defect brings extreme instability to policies and causes performance deterioration in complex tasks. To this end, we introduce maximum entropy learning to optimize exploration. By maximizing $R_E$ and entropy $H_i$ of each state, agent $i$ can adjust exploration to an appropriate degree. In addition, we apply target networks and a replay buffer to reinforce stability and sample-efficiency when training EHCAMA [20].

At each timeslot, EHCAMA stores an experience $(s, A^O, A^C, a, r, s', A'^O, A'^C, h, h')$ into a replay buffer $B$ shared by $N$ agents, where $a = (a_1, \ldots, a_N)$ and $r = (r_1, \ldots, r_N)$. $s'$, $A'^O$, and $A'^C$ represent the structured data of the next global state. $h$ contains the hidden states of actors and critics in all agents and $h'$ is the set of next hidden states. We set all hidden states to zero when we initialize a new episode.

In the training stage, we compute the objective for optimizing $\rho$ and $\theta_i^Q$ by reusing the previous experiences sampled from $B$. To evaluate each global state in terms of $R_E$ and $H_i$, we define a soft value function $V$ as:

$$V(o_i, m_i) = \min_{c=1,2} Q^c_i(o_i, m_i, a_i, h^\pi_i; \theta^Q_i) - \alpha_1 \log \pi_i(a_i | o_i, m_i, h^\pi_i; \theta^\pi_i)$$  (3)

where $\theta^Q_i$ and $\theta^\pi_i$ are the parameters of $Q^c_i$ and $\pi_i$, respectively. The temperature parameter $\alpha_1$ represents the importance of the entropy term for agent $i$ in group $l$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
For each critic network $Q^c_i$, we update parameters $\theta^Q_i$ by Bellman residual minimization as

$$
L(\theta^Q_i) = \frac{1}{N_t} \sum_{i=1}^{N_t} \mathbb{E} \left[ (y_i - Q^c_i(a_i, m_i, h^Q_i; \theta^Q_i))^2 \right]
$$

(4)

where $L(\theta^Q_i)$ is the loss function and $N_t$ means the number of agents in group $l$. The target value $y_i$ is calculated as

$$
y_i = r_i + \gamma V^\pi(o'_i, m'_i)
$$

(5)

where $o'_i$ and $m'_i$ respectively denote $i$’s next observation and received messages. The next soft value $V^\pi(o'_i, m'_i)$ is computed by the target networks $Q^c_i$ and $\pi^\pi_i$, whose parameters $\theta^Q_i$ and $\theta^\pi_i$ are transferred from $Q^c_i$ and $\pi^\pi_i$ via soft updates, respectively.

The stochastic actor $\pi^\pi_i$ is trained to maximize the soft value function $V(o_i, m_i)$ by the following gradient:

$$
\nabla_{\theta^\pi_i} J(\theta^\pi_i) = \frac{1}{N_t} \sum_{i=1}^{N_t} \mathbb{E} \left[ -\nabla_{\theta^\pi_i} \alpha_i \log \pi^\pi_i(a_i | o_i, m_i, h^\pi_i; \theta^\pi_i) 
+ \left( \nabla_{\alpha_i} \min_{c_{i,1,2}} Q^c_i(a_i, m_i, h^Q_i; \theta^Q_i) 
- \nabla_{\alpha_i} \log \pi^\pi_i(a_i | o_i, m_i, h^\pi_i; \theta^\pi_i) \right) \right] \times \nabla_{\theta^\pi_i} f(\epsilon_i; o_i, m_i, h^\pi_i; \theta^\pi_i).
$$

(6)

We apply the reparameterization trick to reduce the variance of $\nabla_{\theta^\pi_i} J(\theta^\pi_i)$. Thus, $\alpha_i$ in (6) is generated from the reparameterized policy function $f(\epsilon_i; o_i, m_i, h^\pi_i; \theta^\pi_i)$, where $\epsilon_i$ represents the noise vector sampled from the Gaussian distribution $\mathcal{N}(0, I)$.

V. SIMULATIONS

A. The Experimental Environment

In this section, we describe our experimental environment to test various capabilities of our proposed EHCAMA and baselines, including a homogeneous scenario, UAV recon, and a heterogeneous scenario, cooperative reconnaissance tasks, whose quality is evaluated based on coverage $C$, fairness $F$, and energy consumption $E$. We use an overall indicator $CFE$ that combines the score of $C$, $F$, and $E$ as the major metric in evaluation, which is defined as

$$
CFE = \frac{C \times F}{E}
$$

where

$$
C = \frac{1}{T} \sum_{t=1}^{T} \frac{n_C(t)}{n}
$$

$$
F = \frac{\left( \sum_{j=1}^{n} c_j \right)^2}{n \sum_{j=1}^{n} c_j^2}
$$

$$
E = \frac{1}{T \times N} \sum_{t=1}^{T} \sum_{j=1}^{N} E_i(t)
$$

(8)

where we calculate $F$ based on Jain’s fairness index [48]. $T$ means the total timeslots of a task, $n_C(t)$ denotes the number of PoIs covered by UAVs at timeslot $t$, and $c_j$ is the coverage time of PoI $j$ at the end of a task.
In simulations, we set the target area as $200 \times 200$ units and the number of PoIs as 120. $v_{\text{max}}$ is 10 units while $E_h$ and $E_m$ are both 0.5. The recon range, observation range, and communication range are set to 10, 15, and 30 units, respectively. In addition, we set $\alpha$ in Fig. 4(b) as 4 units and the length of each episode is 100 timeslots.

2) Cooperative Treasure Collection (CTC): As shown in Fig. 5, CTC is a heterogeneous environment that involves three types of agents, “hunters,” “red banks,” and “blue banks.” The hunters collect randomly distributed treasures of any color in their collection range and then deposit them into the banks of the corresponding color in their depositing range. The action spaces of hunters and banks are the same as that shown in Fig. 4(b). All agents can observe treasures’ positions in their observation range and exchange messages with each other within their communication range. Each hunter receives a reward $r_{\text{collect}} = 0.2$ when collecting a treasure and is rewarded as $r_{\text{deposit}} = 1$ for the depositing a treasure. Each bank obtain a storing reward $r_{\text{store}} = N$, where $N$ is the number of neighboring banks in the same group when depositing treasures of the corresponding color.

We deploy ten hunters and five banks of each color in an area of $40 \times 40$ units to collect treasures which respawn upon being collected. The number of red and blue treasures is 25 each. The collection range, depositing range, observation range, and communication range are 1, 2, 5, and 10 units, respectively. The maximum velocity of hunters is 2 units and banks’ is 0.8 unit. Each episode contains 500 timeslots.

B. Experimental Setup and Baseline Methods

We implement our proposed EHCAMA with PyTorch and simulate its performance on an Ubuntu 18.04 server with two NVIDIA RTX 3080 GPUs. Empirically, we set the hyperparameters of EHCAMA as follows. We apply Adam as the optimizer and set learning rate to 0.001. The discount factor $\gamma$ is 0.95 and the soft update rate $\tau$ is 0.01. The number of units in each fully connected layer and GRU layer is 256 while each HGAT layer contains four attention heads. We set the capacity of the replay buffer as 50k and the size of a minibatch as 128. The parameters of networks are updated four times every 100 timeslots when training.

As the experimental baselines for comparing the performances, we consider four algorithms, SAC-HGRN [43], HAMA [18], AHAC [32], and AGGAT-Comm [38]. Moreover, we emulate DDPG to implement a deterministic variant of our method named “DHCAMA” to verify the necessity of entropy-enhanced optimization. The comparison of our method and baselines is summarized in Table I. The usages of HGAT in SAC-HGRN and HAMA are different from our method. Specifically, SAC-HGRN uses HGAT for communication while HAMA processes local observation represented as a graph through HGAT and does not consider communication. AHAC adopts MHA to compress information of the multiagent system, which can be regarded as a HGAT layer with a complete graph. We modify the actor network of SAC-HGRN and AHAC for continuous action space and learn stochastic policies in a SAC framework. SAC-HGRN represents each agent’s observation as its local state and a set of pixel maps that indicate the position of entities in each group. In AHAC and AGGAT-Comm, the observation is a high-dimensional vector that contains all local states of each agent’s observed entities. We ensure the dimension of the observation fixed by padding zero to cover missing information. In UAV recon, each method is trained for 100k episodes and tested for 10k episodes. The temperature hyperparameter $\alpha$ of the maximum entropy methods is set to 2.5. In CTC, we train them for 10k episodes and set $\alpha$ to 0.001.

C. Experimental Results

We simulate the performance of our method in the experimental environments described in Section V-A and compare it with baselines. We first test the scalability of each method in UAV recon and indicate the impact of number of UAVs on four metrics (including CFE, coverage, fairness, and energy consumption) in Fig. 6. Note that all models except AHAC and AGGAT-Comm are trained with 20 UAVs and transferred to various scale scenarios. Then, we examine the transferability of three HGAT-based methods (EHCAMA, SAC-HGRN, AGGAT-Comm) and baselines are trained with 20 UA Vs and transferred to various scale scenarios.
As observed in Fig. 6(a), the performance of EHCAMA on CFE is consistently superior to baseline methods in various scale tasks. Compared with the second-best approach DHCAMA, our method increases CFE by 69.2%, 54.8%, 51.9%, and 50.0% for the number of UA Vs $N = \{10, 20, 30, 40\}$, respectively. We subsequently analyze three components in CFE from Fig. 6(b)–6(d) and make the following observations.

1) As indicated in Fig. 6(b), EHCAMA and DHCAMA obtain higher scores in terms of coverage than others, which means that our proposed network architecture can provide better policies for UA Vs to optimize their flight trajectories and achieve higher coverage. Besides, we notice that EHCAMA shows better performance over its deterministic variant because of the intelligent exploration provided by the maximum entropy learning. SAC-HGRN extracts features directly from raw pixel maps. As a result, it cannot learn as effectively as our method that uses HGAT to process the observations. Compared with EHCAMA, HAMA does not employ interagent communication and recurrent units, which drops the performance of UA Vs on cooperation and leads to low coverage. AHAC adopts RNN and MHA to approximate the Q-function but does not introduce them into...
actor networks. As a result, it can neither extract relational features from observations by GNN nor utilize spatiotemporal information to improve decision making. Although AGGAT-Comm has an AGGAT layer to achieve interagent communication, it applies a LSTM-based module instead of a GNN layer to process the local observation. In addition, AGGAT-Comm optimizes the standard maximum objective like REINFORCE algorithm [49], resulting in the estimated gradients with high variance. Therefore, AGGAT-Comm fails to learn cooperation policies for agents.

2) From Fig. 6(c), it can be observed that three maximum entropy methods, EHCAMA, SAC-HGRN, and AHAC outperform others from the aspect of fairness index. We suspect that the maximum entropy objective encourages UAVs to explore more widely and helps UAVs to cover PoIs more fairly than deterministic policies. Moreover, the score of EHCAMA is higher than SAC-HGRN’s, which denotes that our network architecture also plays a role in improving the performance on fairness. AHAC obtains the highest score with lower coverage than EHCAMA and SAC-HGRN. The main reason is that AHAC cannot optimize flight trajectories for UAVs by learning from raw observation vectors and they tend to cover each PoI for a nearly equal but brief time.

3) Fig. 6(d) shows that EHCAMA, SAC-HGRN, and AHAC consume more energy than other policies. This is because UAVs need to move longer distances for exploration to improve the performance on coverage and fairness. For example, when the number of UAVs is 20, EHCAMA gives 29.8% improvement on coverage and 27.4% on fairness compared to DHCAMA, while consuming only 3.4% more energy. HAMA and AGGAT-Comm show poor competence in terms of coverage and fairness, thus they prefer to reduce energy consumption for maximizing their rewards. EHCAMA performs better than SAC-HGRN and AHAC because our proposed network can help agents to control energy consumption.

Then, we show the experimental results of transfer learning in Fig. 7, where the nontransferred policies of EHCAMA, SAC-HGRN, and HAMA are trained and tested under same setting of scenarios. From Fig. 7, we see that the performance of EHCAMA does not appear to degradation when transferring to different scale tasks, which indicates superior transferability of our method under various scale tasks.

Fig. 8 indicates the learning curves of all methods in CTC, where each of them is trained with three different random seeds. It can be observed that EHCAMA and DHCAMA achieve high reward while HAMA and AGGAT-Comm are unable to train agents in CTC. With the help of the hierarchical communication based on HGAT, our methods can summarize the information from agents in different groups and derive efficient policies for hunters and banks to cooperate better in the heterogeneous environment. Furthermore, since maximum entropy learning provides an intelligent exploration for agents, the performance of EHCAMA in terms of reward and convergence is superior to those of DHCAMA, which suffer from the instability in DDPG-style training process. SAC-HGRN converges faster than DHCAMA, but due to its MLP-based feature extraction module, it still obtains lower reward compared with ours.

We consider the reasons why EHCAMA succeeds as follows. First, because HGAT can model the hierarchical relationship in observation, agents can extract features from structured data represented as graphs, which is better than raw vectors or pixel maps in nongraph-representation approaches. Second, the interagent communication and the hidden states in recurrent units provide spatiotemporal information from their neighbors and memories to agents, which aids them to cooperate better in a partially observable environment. Finally, maximum entropy learning makes the stochastic policies tractable for exploring. Intelligent exploration helps agents to learn the reward distribution on continuous action space and prevents them from optimizing their policies to local optimums, thereby bringing stability and robustness when executing and training.

D. Ablation Study

To verify the benefit of stochastic policies and maximum entropy learning, we compare the performance of EHCAMA with DHCAMA and two stochastic variants described as follows.
Similarly, since the second HGAT layer aggregates messages from all agents, we denote $G_i^O = (V_i^O, E_i^O)$ as the subgraphs of the observation graph $G^O$ related to agent $i$. The encoders spend $|V_i^O| \times d_a \times d_h$ time to transform local states. In the first HGAT layer, computing the embedding vectors of all groups takes $|V_i^O| \times d_h + |V_i^O| \times d_h$ [51] and the group-level aggregating takes additional $K \times d_h^2$ time. In the worst case, each agent observes all $M$ entities, which means that $|V_i^O|$ and $|E_i^O|$ both equal to $M$. Therefore, the temporal complexity for processing agent $i$’s observation by encoders and HGAT is $O(M)$. Similarly, since the second HGAT layer aggregates messages from all $N$ agents in the worst case, its temporal complexity is $O(N \times M)$. In the recurrent unit, the GRU layer takes $Bd_h^2$ [47] while outputting $\mu_i$ and $\sigma_i$ spends $2 \times d_h \times d_a$ time, where $d_a$ represents the dimension of agent $i$’s action. In summary, the overall temporal complexity of EHCAMA is $O(M)$. Since agents need to maintain the hidden states of their actor networks, the space complexity of each agent in the running stage is $O(1)$.

An advantage of EHCAMA is the lower space complexity of observations in the replay buffer. Our method stores the set of local states as observations into the replay buffer, whose space complexity is $O(M)$. Moreover, it stores the observation adjacency matrices $A^O$ with space complexity $O(N \times M)$ to represent the observation graph $G^O$. In contrast, SAC-HGRN requires multiple channels of pixel maps to represent each group of entities in the observations, which means the space complexity is $O(K \times N \times R_{obs})$, where $R_{obs}$ is the observation range. In practice, since SAC-HGRN prefers to increase the size of pixel maps for accurate representation, it needs more space to store observations than ours. HAMA has the same space complexity as EHCAMA since it represents the environment state as a graph. AHAC and AGGAT-Comm concatenate all local states as observation, whose space complexity is $O(N \times M)$. However, they have to store zero-padding for compensating missing information in the replay buffer, while EHCAMA uses 0 or 1 in adjacency matrices to represent the observation status of an entity. Therefore, our method obtains a lower space overhead in the training stage.

E. Computational Complexity Analysis

In a scenario involving $K$ groups of $M$ entities (including $N$ agents), we denote $G_i^O = (V_i^O, E_i^O)$ as the subgraphs of the observation graph $G^O$ related to agent $i$. The encoders spend $|V_i^O| \times d_a \times d_h$ time to transform local states. In the first HGAT layer, computing the embedding vectors of all groups takes $|V_i^O| \times d_h^2 + |E_i^O| \times d_h$ [51] and the group-level aggregating takes additional $K \times d_h^2$ time. In the worst case, each agent observes all $M$ entities, which means that $|V_i^O|$ and $|E_i^O|$ both equal to $M$. Therefore, the temporal complexity for processing agent $i$’s observation by encoders and HGAT is $O(M)$. Similarly, since the second HGAT layer aggregates messages from all $N$ agents in the worst case, its temporal complexity is $O(N \times M)$. In the recurrent unit, the GRU layer takes $Bd_h^2$ [47] while outputting $\mu_i$ and $\sigma_i$ spends $2 \times d_h \times d_a$ time, where $d_a$ represents the dimension of agent $i$’s action. In summary, the overall temporal complexity of EHCAMA is $O(M)$. Since agents need to maintain the hidden states of their actor networks, the space complexity of each agent in the running stage is $O(1)$.

VI. CONCLUSION

In this article, we propose an entropy-enhanced MARL method named EHCAMA to solve the control problem in large-scale multiagent systems with continuous action spaces. By employing entropy-enhanced optimization based on maximum entropy learning, agents in EHCAMA can learn stable stochastic policies to explore the environment intelligently. From experimental results, we observe that the performance of EHCAMA is superior to other baseline algorithms and our deterministic variant in the UAV recon and CTC scenarios. Meanwhile, our method also shows scalability and transferability in various scale multiagent systems.

However, EHCAMA still has limitations. Our method uses predefined rules to select neighbors of agents in the communication stage, which is difficult to apply such rules in some environments where the interactions between agents are complex. Besides, the temperature hyperparameter $\alpha$ of our method needs to be tuned finely for different tasks. As a future work, we will delve into developing learnable communication protocols for more adaptive neighbor selection, and try to improve the training strategy with automating entropy adjustment.

REFERENCES

[1] A. Dorri, S. S. Kanhere, and R. Jadak, “Multi-agent systems: A survey,” IEEE Access, vol. 6, pp. 28573–28593, 2018.
[2] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, “Monotonic value function factorisation for deep multiagent reinforcement learning,” J. Mach. Learn. Res., vol. 21, no. 1, pp. 7234–7284, 2020.
[3] K. Arulkumaran, A. Cully, and J. Togelius, “AlphaStar: An evolutionary computation perspective,” in Proc. Genetic Evol. Comput. Conf. Companion, 2019, pp. 314–315.
[4] C. P. Nguyen and A. J. Flueck, “Agent based restoration with distributed energy storage support in smart grids,” IEEE Trans. Smart Grid, vol. 3, no. 2, pp. 1029–1038, Jun. 2012.
Yining Chen received the B.Sc. degree in computer science and technology from Sichuan University, Chengdu, China, in 2015. He is currently pursuing the Ph.D. degree in computer science and technology with Zhejiang University, Hangzhou, China. His research interests include reinforcement learning and multiagent systems.

Ke Wang received the B.S. degree in automation from Zhejiang University of Technology, Hangzhou, China, in 2020. He is currently pursuing the Ph.D. degree in aerospace information technology with Zhejiang University, Hangzhou, China. His research interests include multiagent reinforcement learning and software-defined networking.

Guanghua Song received the B.S. degree in computer science from Nanjing University of Science and Technology, Nanjing, China, in 1989, and the M.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 1992 and 2003, respectively. He is currently a Full Professor with the School of Aeronautics and Astronautics, Zhejiang University, Hangzhou, China. His research interests include swarm intelligence, UAV intelligence, and aerospace information technology.

Xiaohong Jiang received the B.Sc. and M.Sc. degrees in computer science from Nanjing University, Nanjing, China, and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2003. She is an Associate Professor with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. Her research focuses on distributed systems, cloud computing, and data service.