Hierarchical Cooperative Swarm Policy Learning with Role Emergence

Tianle Zhang[1,2], Zhen Liu[1,2](✉), Zhiqiang Pu[1,2], Tenghai Qiu[1] and Jianqiang Yi[1,2]

Abstract—**Swarm systems can cooperatively and efficiently accomplish specified complex tasks. Recent works have shown the potential of multi-agent reinforcement learning methods to study behavior policies of swarm systems. However, it is difficult for them to complete complex swarm tasks efficiently. In human society, role assignment can effectively help humans understand complex tasks and decompose them into simple certain subtasks. Inspired by this, we propose a two-level hierarchical cooperative swarm policy learning framework with role emergence based on hierarchical deep reinforcement learning for distributed swarm systems. In this framework, roles are dynamic and emergent. Agents with the same role tend to collectively complete a certain subtask. Specifically, each agent uses a higher-level swarm policy to dynamically select a role for itself in a role space and at a higher temporal scale, while it uses a lower-level swarm policy to perform the responsibilities of the selected role in a primitive action space. Meanwhile, hierarchical swarm policies with partial observation are centrally trained and decentrally executed, where agents' local interaction modules and extrinsic team rewards are designed to promote cooperation among agents. In addition, an intrinsic reward is defined to enable different roles to be identified by agents' longer-term behaviors, which implicitly associates the roles with responsibilities. Simulation results show that our method can learn and generate emergent, dynamic and identifiable roles, which helps swarm systems to reliably and efficiently accomplish complex tasks in a shorter time.**

Index Terms—**distributed swarm system, role emergence, hierarchical multi-agent reinforcement learning.**

I. INTRODUCTION

Swarm systems have attracted the attention of many researchers in recent years because of their unique benefits and great potential applications. Their applications can be founded in warehousing logistics, search and rescue scenarios, formation control [1]–[3], etc. Most of the swarm systems used in these applications are to use many identical agents to collectively accomplish a common target. Meanwhile, each agent in the swarm systems has simple and limited capabilities, e.g. partial observability, local interactivity and restricted manipulation. This provides high requirements for efficient cooperation among agents. Besides, the number of agents in the swarm systems may be large-scale and uncertain. Hence, finding swarm policies that enable swarm systems to efficiently accomplish complex tasks remains great challenging.

Recently, multi-agent reinforcement learning (MARL) has shown great potential in multi-agent systems, and many deep MARL methods have been proposed [4]–[9]. MARL provides a promising approach to develop the behavior policies of swarm systems. However, most of MARL methods cannot be directly applied to swarm systems due to some challenges, including the scalability of swarm policies and the complexity of swarm tasks. To achieve scalability, some MARL methods adopt a mechanism that all agents in the same swarm system share and learn a policy network [10]. But, their centralized training manner will encounter dimensional curse with the increase of the number of agents. Meanwhile, the simple sharing mechanism is not always effective on many complex swarm tasks that require agents to learn many skills. An effective way to efficiently accomplish complex tasks is to assign a corresponding subtask to each agent, which can be seen as a division of labor. This brings forward a question, that is, how to realize the dynamic division of labor and parameter sharing for swarm systems, so as to improve the efficiency of completing complex tasks.

Inspired by the division of labor in nature and human society, a concept that comes to mind is role assignment. Each role has individual responsibility for the whole task. Agents with the same role take on the same responsibilities, and can form a small team to collectively accomplish specified subtasks. Complex tasks can be decomposed by roles associated with responsibilities equivalent to subtasks. The role theory has been widely researched in sociology, economics, and organization theory. Some researchers have introduced the concept of role into multi-agent systems [11]–[13]. However, these works need to use prior domain knowledge for task decomposition and predefine the responsibilities of each role, which is not suitable for complex swarm systems in dynamic and uncertain environments and prevents the swarm system from adapting to the environments [14].

Motivated by the above discussions, we propose a novel hierarchical cooperative swarm policy learning framework with role emergence (HCSP-RE) to learn role-oriented cooperative swarm policies, so as to enable distributed swarm systems to reliably and efficiently accomplish complex tasks. In this paper, the key technical and simulation contributions are as follows. 1) We construct a two-level role-oriented hierarchical policy scheme for each agent in distributed swarm systems by defining a high-level action space as a set of all roles. The scheme consists of a higher-level policy that chooses and maintains a role for many time steps, and a lower-level policy that takes primitive actions according to the assigned role. 2) We design a centralized training and decentralized execution mechanism for both high-level and low-level policies, where

[1]Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, e-mail: tianle-zhang@outlook.com, liuzhen@ia.ac.cn, zhiqiang.pu@ia.ac.cn, tenghai.qiu@ia.ac.cn, jianqiang.yi@ia.ac.cn

[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China.

agents' interaction module and extrinsic team rewards are designed to promote cooperation among agents in the same swarm system. This mechanism avoids the limitations of centralized training in large-scale agents. 3) We define an intrinsic reward for lower-level policies to enable different roles to be identified by agents' long-term behaviors. This makes the roles contain enough information about agents' long-term behaviors, which implicitly associates the roles with responsibilities. 4) simulation results show the effectiveness and superiority of HCSP-RE compared with the existing methods.

II. BACKGROUND

A. Partially Observable Markov Games

In this paper, we consider a complex swarm task that can be modeled by a partially observable Markov game (POMG) [15] which is an multi-agent extension of Markov decision processes. A POMG for n agents is defined by a set of states $\mathcal{S}$ describing the possible state of an environment, a set of actions $\mathcal{A}_1$, ... ,$\mathcal{A}_n$ and a set of observations $O_1, ..., O_n$ for the agents. Based on the local observation $o_i \in O_i$, agent i takes action $a_i \in \mathcal{A}_i$ in the environment. Then, agent i obtains a reward by a reward function $r_i : \mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_n \mapsto \mathbb{R}$. The state of the environment evolves to next state according to the state transition function $P : \mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_n \mapsto \mathcal{S}$. Meanwhile, the initial state of the environment is determined by a distribution $\rho : \mathcal{S} \mapsto [0, 1]$. Agent i aims to maximize its own expected cumulative return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where $\gamma \in [0, 1]$ is a discount factor and T is the time horizon.

B. Proximal Policy Optimization

In this paper, proximal policy optimization (PPO) [16] is adopted as the basic training algorithm. PPO is a novel policy gradient algorithm [17] for deep reinforcement learning, and it is an actor-critic framework. PPO uses two deep neural networks approximate a value function (critic) V_ϕ and a policy function (actor) π_θ respectively, where ϕ and θ are critic and actor network parameters respectively. The objective of PPO is to update an actor that maximizes expected advantage function of the current actor $\pi_{\theta_{old}}$. In addition, in order to ensure that the performance of the new actor is gradually improved, the optimization objective of PPO is as follow:

$$\begin{aligned} J_{\pi_\theta}^{PPO} = \mathbb{E}[min(&\frac{\pi_\theta}{\pi_{\theta_{old}}} A^{\pi_{\theta_{old}}}(s, a), \\ &clip(\frac{\pi_\theta}{\pi_{\theta_{old}}}, 1 - \epsilon, 1 + \epsilon) A^{\pi_{\theta_{old}}}(s, a))], \end{aligned} \tag{1}$$

where $clip(\frac{\pi_\theta}{\pi_{\theta_{old}}}, 1 - \epsilon, 1 + \epsilon)$ limits $\frac{\pi_\theta}{\pi_{\theta_{old}}}$ in the interval $[1 - \epsilon, 1 + \epsilon]$, and $A^{\pi_{\theta_{old}}}(s, a) = Q^{\pi_{\theta_{old}}}(s, a) - V_\phi(s)$ is the advantage estimate function. The action value function $Q^{\pi_{\theta_{old}}}(s, a)$ is approximately evaluated by temporal difference operation, i.e., $Q^{\pi_{\theta_{old}}}(s, a) = r + \gamma V_\phi(s') - V_\phi(s)$.

C. Hierarchical Multiagent Reinforcement Learning

In single-agent hierarchical reinforcement learning, multiple layers of policies are trained to perform decisions and controls at higher levels of temporal and behavioral abstraction [18]. In hierarchy of policies, the lowest policy applies actions to the environment, while the higher level policies are able to plan over a longer time scale. Similar to single-agent hierarchical learning, multiagent hierarchical policies can also be learned from high-level and low-level perspectives through temporal abstraction [19]. In the learning process, high-level and low-level policies are trained independently without gradient transfer between levels. Recently, an efficient learning method is that centralized MARL algorithm and independent reinforcement learning are used to train high-level and low-level policies respectively [20]. But, there will be some limitations when the method is applied to distributed swarm systems.

D. Distributed Swarm Systems

In this paper, we study a swarm system where a group of self-organizing homogeneous agents tries to collectively accomplish specified tasks. The swarm system with n homogenous agents is distributed and self-organized. The illustration of the swarm system is shown in Fig. 1. For simplicity, we assume that geometry of the agents in the swarm system is modeled as a disc. The basic state of agent i contains its own position $p_i = [p_i^x, p_i^y]^T$ and velocity $v_i = [v_i^x, v_i^y]^T$, i.e., $s_i = [p_i, v_i]^T$. The kinematic model of agent i is a double integrator model, and the action of the agent is denoted as $a_i = [F_i^x, F_i^y]^T$, where F_i^x, F_i^y are forces on the agent in x and y directions. Meanwhile, the number of the agents in the swarm system is uncertain and may be large-scale. Obviously, the environment of the agents in the swarm system is uncertain and dynamic. In addition, the main challenges in this paper are as follows.

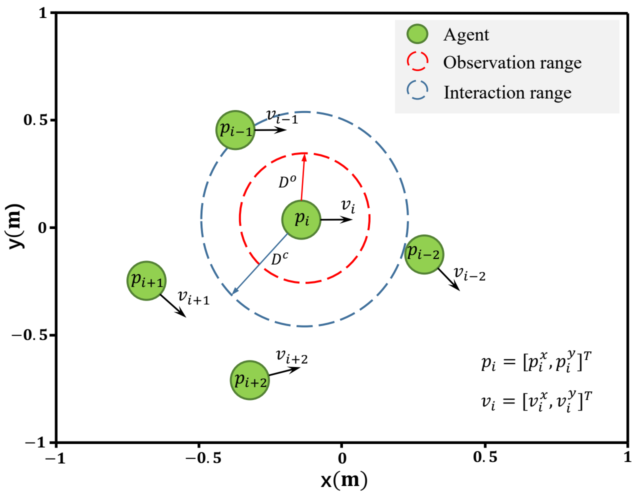


Fig. 1. Illustration of a distributed swarm system. Green solid circles represent agents in the swarm system. Red and blue dotted circles represent the observation range and interaction range of each agent respectively.

1) *Partial Observability:* In the distributed swarm system, each agent has a limitation on observability. Each agent can only perceive its own state and the states of neighbors within the observation range D^o of the agent.

2) *Local Interactivity:* In swarm systems, agents often need to interact with collaborators to facilitate cooperation among

agents. However, in the distributed swarm systems, each agent only interacts with some of the homogeneous agents within the range of its local interaction D^c, instead of all agents in the same swarm system.

3) Role Assignment: to efficiently complete the specified complex tasks, the swarm system need a clear role assignment. Meanwhile, due to high uncertainty of the swarm system, the role assignment should be dynamic. Each agent has an assigned role, and performs the responsibility of the role by a long-term behavior. Through dynamic role assignment, the swarm system can achieve labor division for all agents to decompose complex tasks, which is also higher-level intelligent cooperation among the agents. In this paper, roles in the swarm system are emergent without being defined in advance, and the role of each agent changes dynamically at a higher time scale. The responsibilities of roles are also determined automatically by learning approaches. The process of responsibility determination is equivalent to the process of task decomposition, which can be understood that responsibilities are equivalent to subtasks. Meanwhile, responsibilities can also be reflected through the agents' longer-term behaviors.

Hence, this paper mainly need to find a swarm policy for the distributed swarm system to realize dynamic role assignment, which implicitly implements task decomposition via emergent roles associated with responsibilities that are equivalent to subtasks.

III. APPROACH

In this section, we propose a novel hierarchical cooperative swarm policy learning framework with role emergence, called HCSP-RE, which introduces the dynamic and identifiable role concept into distributed swarm systems and promotes them to reliably and efficiently accomplish complex tasks. Firstly, the overall design of HCSP-RE is given. Then, role-oriented distributed hierarchical swarm policies with centralized training and decentralized execution are presented in detail. Next, to associate each role with different responsibilities, we define an identifier-based intrinsic reward to enable the role to be identifiable by agents' longer-term behaviors. Finally, the training algorithm of HCSP-RE is presented.

A. Overall Design of HCSP-RE

The overall structure of HCSP-RE is given in Fig. 2. At the higher-level policy, the temporally-extended extrinsic team reward is used to train a high-level decentralized actor-critic network for decentralized dynamic assignment of role ρ_i to each agent i, $i = 1, ..., n$. At the lower-level policy, conditioned on the assigned role, a low-level decentralized actor-critic network takes primitive actions to swarm environments, that is, it performs the responsibilities of roles. The trajectory of agent i, τ_i, generated by the lower-level policy under the assigned role ρ_i is collected into a dataset $\mathcal{D} = \{\{(\rho_i, \tau_i)\}_{i=1}^{n}\}$, which is used to train a role identifier q to approximately estimate the true posterior probability as $q(\rho_i|\tau_i)$ that predicts roles from trajectories. Meanwhile, The likelihood of posterior probability, $logq(\rho_i|\tau_i)$, is acted as the intrinsic reward for the lower-level policy to associate roles with responsibilities and learn identifiable roles.

In HCSP-RE, the experiences of all agents are used to train one higher-level policy network, one lower-level policy network and one identifier network, that is, agents in the same swarm system share all the learning parameters including higher-level policy, lower-level policy and identifier networks. This sharing method avoids dimensional curse of large-scale agents in swarm systems, and achieves the scalability of swarm policies. Besides, all perception modules have the same network structure and different parameters, so do interaction modules.

B. Role-oriented Distributed Hierarchical Swarm Policies with Role Assignment and Performing Responsibility

In the role-oriented distributed hierarchical swarm policies as shown in the middle of Fig . 2, the distributed higher-level policy is designed to implement role assignment and the lower-level policy performs the responsibility of the role assigned by the higher-level policy. In the following, take agent i as an example to present the higher-level and lower-level policies in detail.

Higher-level Policy: Role Assignment

Agent i uses the higher-level policy to dynamically assign a role $\rho_i \in \mathcal{Z}$ for itself according to the partial observation $o_i \in O$ ($o_i = \{s_j | j \in N^o(i)\}$) and local interactive message $m_i^h \in M^h$ ($m_i^h = \{h_j^h | j \in N^c(i)\}$) of the agent in a role space $\mathcal{Z}$ every k time steps, where $\mathcal{Z}$ denotes a set of role variables z_j ($j = 1, ..., q$) with one-hot encoding, i.e., $\mathcal{Z} = \{z_1, ..., z_q\}$ (q represents the number of roles), $M^h = \{h_1^h, ..., h_n^h\}$ denotes a set of high-level interactive messages, $N^o(i)$ is some neighborhood (include itself) within the observation range D^o of agent i, and $N^c(i)$ is some neighborhood (include itself) within the interaction range D^c of agent i. In particular, the network structure of the higher-level policy is a coupled actor-critic network. The high-level critic network $c_i^h : O \times M^h \mapsto \mathbb{R}$ consists of a perception module P_i^h, an interaction module I_i^h and a high-level value module V_i^h that are two fully-connected (FC) layers, i.e., $c_i^h = V_i^h(mean(\tilde{h}_i^h, \tilde{h}_{-i}^h))$ ($-i$ denotes all agents except for agent i). The high-level actor network $\mu_i : O \times M^h \mapsto \mathcal{Z}$ makes up of a perception module P_i^h, an interaction module I_i^h and a role module A_i^h that are two FC layers. Herein, The perception module and interaction module are both graph attention networks [21] based on Transformer [22], and the two modules can process information of uncertain number of agents in swarm systems. The perception module is designed to extract effective high-level environment feature h_i^h from o_i, and the feature h_i^h encodes the spatial relations between agent i and the surrounding environment. The interaction module is designed to cooperatively negotiate role division with surrounding homogeneous agents by utilizing interactive message m_i^h, yielding high-level interaction feature $\tilde{h}_i^h$ which contains cooperative information with neighbor agents. In addition, in the structure of the higher-level policy, the high-level critic network is used to evaluate the output of the high-level actor network in the training phase. The actor network is used as

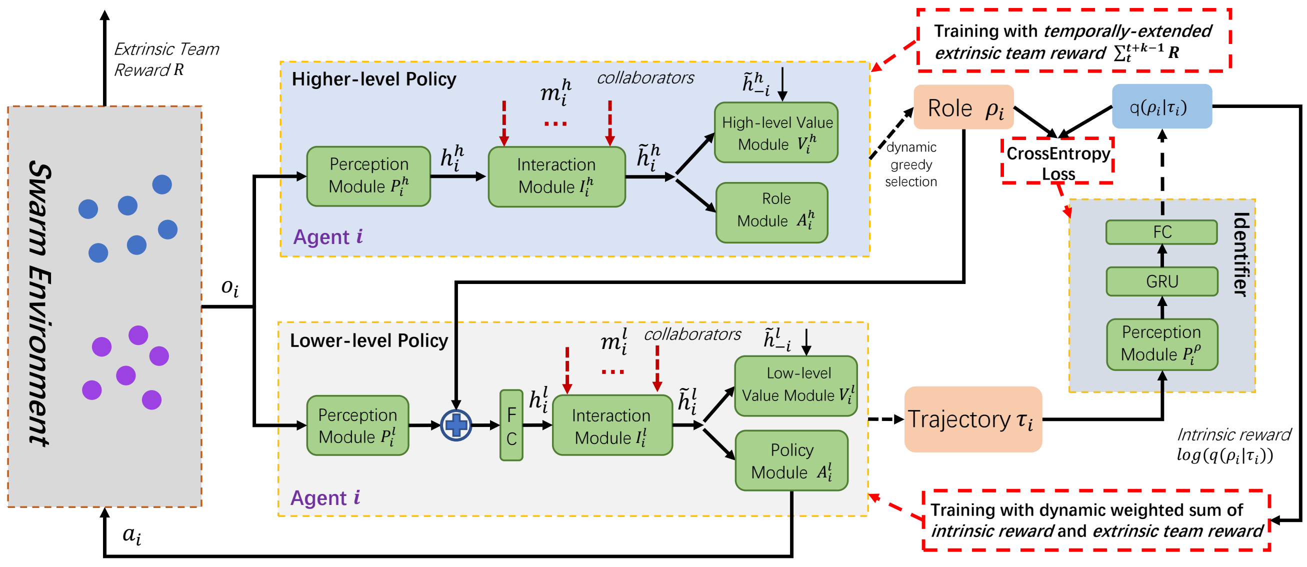

Extrinsic Team Reward R
Swarm Environment
o_i
a_i
Higher-level Policy
Perception Module P_i^h
h_i^h
m_i^h
collaborators
Interaction Module I_i^h
$\tilde{h}_i^h$
$\tilde{h}_{-i}^h$
High-level Value Module V_i^h
Role Module A_i^h
Agent i
Training with temporally-extended extrinsic team reward $\sum_t^{t+k-1} R$
Role ρ_i
dynamic greedy selection
CrossEntropy Loss
$q(\rho_i|\tau_i)$
Identifier
FC
GRU
Perception Module P_i^p
Lower-level Policy
Perception Module P_i^l
FC
h_i^l
m_i^l
Interaction Module I_i^l
$\tilde{h}_i^l$
$\tilde{h}_{-i}^l$
Low-level Value Module V_i^l
Policy Module A_i^l
Agent i
Trajectory τ_i
Intrinsic reward $log(q(\rho_i|\tau_i))$
Training with dynamic weighted sum of intrinsic reward and extrinsic team reward

Fig. 2. Hierarchical cooperative swarm policy learning framework with role emergence.

the higher-level policy, where it outputs the probabilities of roles, and the role with the highest probability is chosen by greedy method.

In addition, the higher-level policy with actor-critic framework adopts a centralized training and decentralized execution manner [4]. In this training phase, to cooperatively learn effective role assignment for distributed higher-level policy, besides the interaction module with collaborators, we design an extrinsic team reward $R : S \times \{\mathcal{A}\}_{i=1}^n \mapsto \mathbb{R}$ that maps global state and joint actions to a scalar reward. Meanwhile, the higher-level policy is implemented each k time steps, that is, each choice of role ρ_i for agent i is sustained for k time steps, which enables the lower-level policy to fully perform the responsibilities of the role with enough time. Furthermore, the higher-level policy learns to assign roles for the lower-level policy to optimize temporally-extended extrinsic team reward R_H that is the sum of extrinsic team rewards R obtained during the execution interval k of the higher-level policy, i.e., $R_H := \sum_t^{t+k-1} R^t$. Hence, the higher-level policy can be obtained through optimizing the following objective,

$$\underset{\mu_i}{argmax}\, \mathbb{E}_{o_i^t, P, \rho_i^t \sim \mu_i, a_i^t \sim \pi_i}[\sum_{\tilde{t}=1}^{\tilde{T}} \gamma^{\tilde{t}} \sum_{t=\tilde{t}}^{\tilde{t}+k-1} R^t)], \quad (2)$$

where P denotes the swarm environment transition probability,$\tilde{t}$ represents the number of the higher-level policy execution times in each episode, γ is a high-level discount factor, $\tilde{t} = k * t$ and π_i is the actor of the lower-level policy given in the following.

Lower-level Policy: Performing Role Responsibility

Each agent uses the lower-level policy to perform the responsibility of the role assigned by the higher-level policy in a primitive action space. Conditioned on an assigned role $\rho_i \in \mathcal{Z}$, an agent's observation $o_i \in O$ and interaction message $m_i^l \in M^l$ ($m_i^l = \{h_j^l | j \in N^c(i)\}$) from collaborators where $M^l = \{h_1^l, ..., h_n^l\}$ denotes a set of low-level interactive messages, the lower-level policy outputs a primitive action a_i in the action space $\mathcal{A}$ interacting with the environment. Similar to the network structure of the higher-level policy, the network structure of the lower-level policy is also an actor-critic network. The low-level critic network $c_i^l : O \times M^l \times \mathcal{Z} \mapsto \mathbb{R}$ consists of a perception module P_i^l, an interaction module I_i^l and two-FC low-level value module V_i^l, i.e., $c_i^l = V_i^l(mean(\tilde{h}_i^l, \tilde{h}_{-i}^l))$. The low-level actor network $\pi_i : O \times M^l \times \mathcal{Z} \mapsto \mathcal{A}$ makes up of a perception module P_i^l, an interaction module I_i^l and two-FC policy module A_i^l. Herein, the perception module is also used to extract effective low-level environment feature h_i^l from o_i and the interaction module is also adopted to promote low-level cooperation among roles to better perform responsibilities from the roles, yielding low-level interaction feature $\tilde{h}_i^l$. Besides, the low-level critic network is responsible for evaluating the output of the low-level actor network in the training. The low-level actor network is used as the lower-level policy, which directly outputs a primitive action to the swarm environment.

Similar to the higher-level policy, the lower-level policy also adopts the centralized training and decentralized execution manner with an actor-critic framework. For its training, the lower-level policy needs to learn to choose primitive actions to generate useful and identifiable behaviors for its acting role, that is, it produces corresponding behaviors to perform the responsibility of the role. Hence, a low-level reward function R_L is designed to drive the lower-level policy to achieve its responsibilities. The low-level reward is dynamic weighted sum of extrinsic team reward R and intrinsic reward R_I, i.e., $R_L = R + \alpha R_I$, where the team reward is used to guarantee that the generated behaviors are useful for team performance, and the intrinsic reward given in the next subsection is designed to promote the association of roles with responsibilities, which enables roles to be identifiable by agents' long-term behaviors.

Hence, the lower-level policy can be obtained by maximizing the low-level reward, i.e.,

$$\underset{\pi_i}{argmax}\, \mathbb{E}_{o_i^t, P, \rho_i^t \sim \mu_i, a_i^t \sim \pi_i}[\sum_{t=1}^{T} \gamma^t R_L^t]. \quad (3)$$

C. Role Identifiability via Identifier-based Intrinsic Rewards

In role-oriented swarm systems, agents with same roles have same responsibilities for complex tasks. Each role has individual responsibility that can be reflected through corresponding behaviors. Intuitively, a role is a comprehensive pattern of behavior, and each role is identifiable by agents' long term behaviors. Therefore, we would like associate a role with its responsibility by enabling the role ρ_i to be identified by agents' longer-term behavior, that is, trajectory τ_i. This can be achieved by maximizing the mutual information between the individual trajectory and the role, i.e., $I(\tau_i;\rho_i)$, which measures the amount of information that the roles contain about agent' longer-term behavior. But, maximizing the mutual information is often intractable. Hence, we introduce a variational posterior estimator $q_\xi(\rho_i|\tau_i)$ parameterized with ξ to approximately estimate the true posterior $p(\rho_i|\tau_i)$, which derives a tractable lower bound on $I(\tau_i;\rho_i)$ [14]. In addition, since the role ρ_i assigned by the higher-level policy is a fixed value and not a variable for the lower-level policy, the lower bound of the mutual information can be expressed as follows [23],

$$I(\tau_i;\rho_i) \geq \mathbb{E}_{\tau_i, \rho_i \sim p(\rho_i|\tau_i)}[log(q_\xi(\rho_i|\tau_i))]. \quad (4)$$

The posterior estimator q_ξ is designed as an identifier as shown in the right of Fig. 2, which is used to predict the role ρ_i^t from trajectory $\tau_i^{t,k} = (o_i^t, a_i^t, \cdots, o_i^{t+k-1}, a_i^{t+k-1})$ generated by the lower-level policy under the role ρ_i. For q_ξ, we use a perception module P_i^o to process observation states in the trajectory and gated recurrent unit (GRU) [24] to encode trajectory information. The network parameter of the identifier is the parameter ξ. The identifier parameter is updated using a dataset $\mathcal{D} = \{\{(\rho_i, \tau_i)\}_{i=1}^n\}$ of role-trajectory pairs, where each makes up of the role ρ_i assigned by the higher-level policy and the corresponding trajectory τ_i generated by the lower-level policy under the role ρ_i.

To maximizing the mutual information, we only need to maximize its lower bound. Hence, we take the lower bound as the intrinsic reward function for the lower-level policy, i.e., $R_I = log(q_\xi(\rho_i|\tau_i))$. Through the operation, the lower-level policy optimizes the intrinsic reward, which means to maximize $I(\tau_i;\rho_i)$. This can enable the assigned role to be identified through agents' long-term behaviors generated by the lower-level policy, which promotes the association of roles with responsibilities.

D. Training Algorithm

Algorithm 1 is designed to optimize the objectives in (2) and (3) to achieve role emergence for HCSP-RE. In the training algorithm, we adopt an improved proximal policy optimization (PPO) [16] algorithm based on a coupling actor-critic network framework. Specifically, the higher-level policy networks (the high-level critic and actor) and lower-level policy networks (the low-level critic and actor) are updated by minimizing high-level total loss $L_h = \beta_1 L_{c^h} + \beta_2 L_\mu - \beta_3 H_h$ and low-level total loss $L_l = \beta_1 L_{c^l} + \beta_2 L_\pi - \beta_3 H_l$ respectively, where $\beta_1, \beta_2, \beta_3$ are hyper-parameters. The high-level total loss L_h is conducted by weighted summation of high-level value loss L_{c^h}, action loss L_μ and action entropy H_h. The low-level total loss L_l is conducted by weighted summation of low-level value loss L_{c^l}, action loss L_π and action entropy H_l,

$$\begin{aligned} L_{c^h} =& \mathbb{E}[(R_H + \gamma c^h(o_i', m_i'; w_h^-) - c^h(o_i, m_i; w_h))^2], \\ L_{c^l} =& \mathbb{E}[(R_L + \gamma c^l(o_i', m_i', \rho_i'; w_l^-) - c^l(o_i, m_i, \rho_i; w_l))^2], \end{aligned} \quad (5)$$

$$\begin{aligned} L_\mu =& -\mathbb{E}[min(\frac{\mu(o_i, m_i; w_h)}{\mu(o_i, m_i; w_h^-)}, clip(\frac{\mu(o_i, m_i; w_h)}{\mu(o_i, m_i; w_h^-)}, \\ & 1-\epsilon, 1+\epsilon)) * (R_H + \gamma c^h(o_i', m_i; w_h^-) - c^h(o_i, m_i; w_h)], \\ L_\pi =& -\mathbb{E}[min(\frac{\pi(o_i, m_i, \rho_i; w_l)}{\pi(o_i, m_i, \rho_i; w_l^-)}, clip(\frac{\pi(o_i, m_i, \rho_i; w_l)}{\pi(o_i, m_i, \rho_i; w_l^-)}, \\ & 1-\epsilon, 1+\epsilon)) * (R_L + \gamma c^l(o_i', m_i, \rho_i; w_l^-) - c^l(o_i, m_i, \rho_i; w_l)], \end{aligned} \quad (6)$$

$$\begin{aligned} H_h =& -\sum \mu(o_i, m_i; w) log(\mu(o_i, m_i; w)), \\ H_l =& -\sum \pi(o_i, m_i, \rho_i; w) log(\pi(o_i, m_i, \rho_i; w)). \end{aligned} \quad (7)$$

Herein, H_h or H_l is specially designed to encourage exploration for agents through penalizing the entropy of actor μ or π. Meanwhile, the identifier q_ξ is updated with a cross entropy loss $L_q = CrossEntropy(\rho_i, \hat{\rho}_i)$ where $\hat{\rho}_i$ is the output of q_ξ, which can be viewed as a supervised learning process with the sample dataset $\mathcal{D}$. In addition, we periodically evaluate the performance of the learned swarm policies, e.g., success rate. If performance exceeds $S_{threshold}$, we gradually increase the weight α to better promote the association of responsibilities performed by the lower-level policy with the given roles.

IV. SIMULATIONS

In this section, simulations are conducted to verify the effectiveness of the proposed HCSP-RE including a fully cooperative *coverage* task and a predatory-prey-style *pursuit* task. This aims to demonstrate the effectiveness and efficiency of the proposed method. The key point of the demonstration is whether the proposed method can improve the efficiency of task completion.

A. Task Description

Coverage: This task requires a swarm system with n homogeneous agents to cooperatively reach n landmarks. In other words, the agents need to cover all of the landmarks. Meanwhile, the corresponding relationship between agents and landmarks is not determined in advance. This task is similar to the cooperative navigation task in [4].

Pursuit: This task is that a swarm system with n slower predators need to cooperatively hunt m ($m < n$) faster preys in a randomly generated environment. Specifically, the preys are

Algorithm 1 Training Algorithm for HCSP-RE

```
1: Initialize high-level critic $c^h$ and actor $\mu$, low-level critic $c^l$, and actor $\pi$, identifier $q$, high-level replay buffer $\mathcal{B}_H$, low-level replay buffer $\mathcal{B}_L$, and trajectory-role dataset $\mathcal{D}$
2: for each episode do
3:   $o^t$ = env.reset()
4:   for each step $t = 1, ..., T$ in episode do
5:     if $t$ mod $k = 0$ then
6:       if $t > 1$ then
7:         Compute $R_H^t = \sum_{j=1}^{k} R^{t-j}$, Store $(o^{t-k}, \rho^{t-k}, R_H^t)$ into $\mathcal{B}_H$, Store $(\rho^{t-k}, \tau^{t-k,k})$ into $\mathcal{D}$
8:         Compute intrinsic reward $R_I$
9:         Recompute $R_L^t = R_L^t + \alpha R_I$ from $t-1$ to $t-k$
10:      end if
11:      Assign new $\rho_i$ by high-level actor $\mu_i$, $i = 1, ..., n$
12:    end if
13:    Get $a^t$ from low-level actor $\pi$
14:    $o^{t+1}, R^t$ = env.step($a_t$)
15:    Compute $R_L^t = R^t$, Store $(o^t, a^t, R_L^t)$ in $\mathcal{B}_L$
16:  end for
17:  Compute high-level and low-level advantage estimates with generalized advantage estimator (GAE) [25] using $c^h$ and $c^l$, and store into $\mathcal{B}_H$ and $\mathcal{B}_L$ respectively for each episode.
18:  if size of $\mathcal{B}_L \geq N_{threshold}$ then
19:    Update $c^h$ and $\mu$ by minimizing high-level total loss $L_h = \beta_1 L_{c^h} + \beta_2 L_\mu - \beta_3 H_h$ with sampling from $\mathcal{B}_H$
20:    Update $c^l$ and $\pi$ by minimizing low-level total loss $L_l = \beta_1 L_{c^l} + \beta_2 L_\pi - \beta_3 H_l$ with sampling from $\mathcal{B}_L$
21:    Update $q$ by minimizing $L_q$ calculated with sampling from $\mathcal{D}$
22:    Empty $\mathcal{B}_H$, $\mathcal{B}_L$, $\mathcal{D}$
23:  end if
24:  if evaluation success rate $\geq S_{threshold}$ then
25:    $\alpha = \alpha + \alpha_s$
26:  end if
27: end for
```

controlled by an improved fixed policy based on the Voronoi escape strategy [26] and are confined to a closed world.

B. Simulation Settings

In simulations, all task environments are built on the multi-agent particle environment (MAPE) [4], where agents with double integrator dynamics models take discrete actions in a two-dimensional world. In each task, the agents can only partially observe the surrounding environment and interact with local collaborators. This is also a characteristic of distributed swarm systems. In addition, to speed up training process, we adopt curriculum learning with model reload [27] by progressively increasing the number of training agents in a swarm system for each task.

In *coverage* task, we design a shaped team reward by adding the negative minimum distance of all agents from each landmark. In *pursuit* task, a swarm system obtains +10 team reward when one live prey is hunted by predators. Meanwhile, a shaped team reward is designed to minimize the distance between the swarm system and all preys, which aims to speed up the learning process. Besides, in the training or test phase, when one of the following two conditions is satisfied, the episode will be terminated. The first is that the task is successfully accomplished by the swarm system. Another one is that the episode reaches 50 time steps. In addition, in the *Pursuit* task, the state of agent i contains the life state s_i^l, i.e., $s_i = [p_i, v_i, s_i^l]^T$, where $s_i^l = 1$ represents the state of being alive and $s_i^l = 0$ represents the state of being dead. For implementation details, we set $\gamma = 0.99$, $N_{threshold} = 5000$, $q = 4$, $\beta_1 = 0.5$, $\beta_2 = 1.0$, $\beta_3 = 0.01$, $S_{threshold} = 0.9$, $\epsilon = 0.2$, $\alpha_s = 0.1$, $D^o = 2.5$ $D^c = 3$, execution interval of higher-level policy $k = 5$ and the dimension of discrete action space is 4. In addition, to fully validate the superiority of the proposed method, we adopt the method proposed in [21] (TRANSFER) as a baseline method. This is because this method can be applied in swarm systems with a variable number of agents.

C. Simulation Results

Quantitative Evaluation. In the *coverage* task, we set a swarm system with $n = 12$ agents to occupy $n = 12$ landmarks. In the *coverage* task, a swarm system with $n = 15$ predators needs to hunt $m = 5$ preys. To speed up training process, we adopt curriculum learning manner with model reload. In particular, the policy is firstly learned in a team with small number of agents. When the success rate of evaluation for the learned policy exceeds a threshold (90%), this policy is transferred to a team with more agents to be trained until the number of agents reaches the set requirement, e.g., $n \in \{3, 6, 12\}$ for the *coverage* task and $(n, m) \in \{(3, 1), (6, 2), (9, 3), (12, 4), (15, 5)\}$ for the *pursuit* task. Through this manner, we obtain the learned policy and test its performance. Simulation test results are shown in Table I. The test results (mean and standard deviation) are obtained by running 1000 episodes in five independently seeded environments, where each seeded environment runs 200 episodes. As expected, our method shows superior performance than the baseline method in terms of success rate, episode reward and episode length, especially in the episode length. The results of the episode length show that our method can complete the task in a shorter time. This superior performance demonstrates that emergent roles by our method can improve the efficiency of task completion. We infer that this is because the complex tasks can be decomposed into common subtasks by assigning different roles associated with responsibilities to agents in swarm systems, which accelerates the completion time of tasks with a clear role division. On the contrary, the baseline method need to take a long time to finish the tasks due to the lack of efficient labor division of roles with cooperation for complex tasks. In short, our method has high efficiency of task completion for complex swarm systems. Though not by a large margin, this superiority indicates the benefits of the proposed method with emergent roles.

In order to further verify the generalization of our method, the learned swarm policy is performed in scenarios where

TABLE I

Simulation test performance of our method and baseline method in *coverage* and *pursuit* tasks. *Success Rate: percentage of episodes successfully completed by the agents. Mean Episode Reward: mean of rewards for each episode of the agents. Episode Length: successful episode length.*

Task	Method	Success Rate	Mean Episode Reward	Episode Length (Average / 75th / 90th Percentile)
coverage ($n = 12$)	TRANSFER	0.94±0.03	-7.20±0.21	20.65±0.35 / 24.00±0.63 / 28.80±1.17
	HCSP-RE (ours)	**0.96**±0.01	-6.52±0.16	**16.19**±0.37 / **18.20**±0.40 / **21.60**±0.80
pursuit ($n = 15, m = 5$)	TRANSFER	0.94±0.02	0.89±0.15	26.52±0.38 / 30.60±0.80 / 37.00±0.89
	HCSP-RE (ours)	**0.98**±0.01	1.51±0.09	**24.14**±0.46 / **26.20**±0.40 / **33.00**±1.79

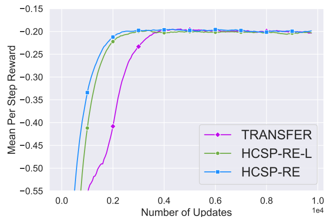
TABLE II

Generalization performance of our method in the *coverage* and *pursuit* tasks.

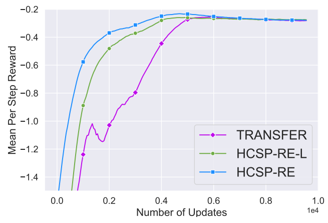
Task	Number of agents	Success Rate	Mean Episode Reward	Episode Length (Average / 75th / 90th Percentile)
coverage	$n = 9$	**0.98**±0.01	-5.69±0.07	**16.44**±0.28 / 18.40±0.49 / 21.80±0.75
	$n = 14$	**0.92**±0.01	-7.48±0.10	**16.67**±0.33 / 18.80±0.40 / 22.20±0.75
pursuit	$n = 12, m = 4$	**0.95**±0.01	0.82±0.10	**25.35**±0.54 / 28.00±1.10 / 37.20±2.40
	$n = 18, m = 6$	**0.97**±0.01	1.80±0.03	**24.48**±0.16 / 27.00±0.63 / 34.20±0.40

the number of agents is different from that of the training. The results are shown in Table II. As expected, our method still has good generalization performance. This is because the dynamic role assignment of our method can adapt well to various uncertain environments. Hence, the generalization performance further demonstrates the effectiveness and advantage of introducing role assignment into swarm systems.

Ablation Studies. To understand the superior performance of HCSP-RE, we carry out ablation studies to test the contribution of its main components and analysis its main hyper parameters. Firstly, ablation studies regarding main components of HCSP-RE is implemented, and the results are shown in Fig. 3. The HCSP-RE-L method is a ablation method without the higher policy, that is, there is no role assignment. The results show that HCSP-RE has better performance than HCSP-RE-L and TRANSFER in term of the convergence rate. This fully validates the effectiveness of the higher policy of the HCSP-RE method with role assignment. It can be seen from the Fig. 3 that with the increase of the number of agents or the task becomes more difficult, the advantage of HCSP-RE is greater.



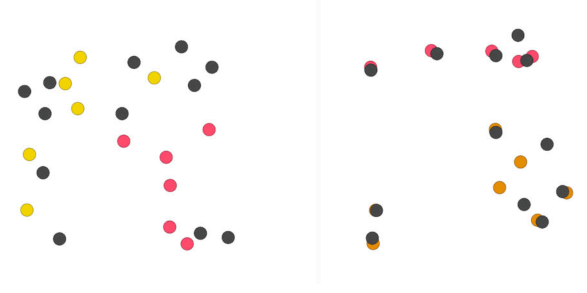
(a) Mean per step reward vs. number of updates in 3-agent coverage task



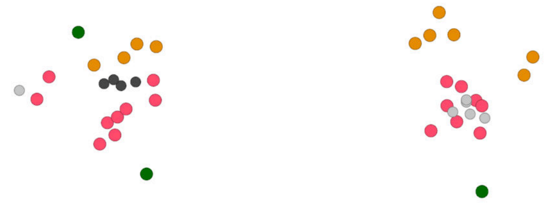
(b) Mean per step reward vs. number of updates in 7-agent coverage task

Fig. 3. Ablation studies regarding main components of HCSP-RE.

Role Emergence. To fully analyze and verify the effectiveness of the proposed method, we further investigate the emergence of roles in some scenarios as shown in Fig. 4. Black solid circles represent landmarks or live preys, grey solid circles represent dead preys and other color solid circles represent agents in swarm systems. Different color solid circles



(a) Coverage scenario 1 (b) Coverage scenario 2



(c) Pursuit scenario 1 (d) Pursuit scenario 2

Fig. 4. Illustrations of keyframes in *coverage* and *pursuit* tasks.

denotes agents acting as different roles. It can be seen from the figure shown in Fig. 4 that a swarm system can assign different roles to agents to complete a complex task. The complex task can be decomposed by defining roles associated with simple subtasks using our learning framework. Agents with the same role collectively accomplish a subtask. This enables the complex task to be performed more efficiently. In the following, let us give an intuitive explanation of Fig. 4. In Coverage scenario 1 (Fig. 4(a)), the swarm system have a division of two roles, i.e., red agents and yellow agents. Agents with the same role form a small team to cooperatively accomplish a subtask, e.g., occupying some automatically assigned landmarks. Meanwhile, to complete the task more efficiently, the emergent roles for agents are dynamic by the role assignment of the higher-level policy. In Coverage scenario 2 (Fig. 4(b)), the division of labor for different roles is clearly reflected in the task by the proposed method. In Pursuit scenario 1 (Fig. 4(c)), the predator swarm system uses the division of roles to cooperatively hunt preys. This is easy to come up with different pursuit strategies to intelligently and farsightedly capture preys. In Pursuit scenario 2 (Fig. 4(d)), the predator swarm system have captured all preys with emergent roles associated with responsibilities. In particular, red agents are responsible for hunting all preys, and orange and green agents are responsible for preventing preys from escaping. The comprehensibility and interpretability of roles are our future research point.

V. Conclusions

In this paper, HCSP-RE is proposed to enable distributed swarm systems to emerge roles to efficiently accomplish complex tasks, which implicitly draws connection to the division of labor and task decomposition by defining roles. Specifically, each agent use the higher-level policy to assign a role for it, and the lower-level policy to perform the responsibilities of the assigned roles. Meanwhile, these hierarchical swarm policies are centrally trained and decentrally executed. Furthermore, an intrinsic reward for the lower-level policy is designed to associate roles with responsibilities. Simulation results verify the superiority of the proposed method, and provides role concept into swarm systems to explain and promote cooperation within agent teams for complex tasks. In the future, the comprehensibility and interpretability of roles are directions of our research.

Acknowledgment

This work was supported by the National Key Research and Development Program of China under Grant 2020AAA0103404, 2018AAA0101005, the Strategic Priority Research Program of Chinese Academy of Sciences under Grant No. XDA27030403, and Science and Technology Development Fund of Macau (No.0025/2019/AKP).

References

[1] Y. Liu, L. Wang, H. Huang, M. Liu, and C.-z. Xu, "A novel swarm robot simulation platform for warehousing logistics," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2017, pp. 2669–2674.

[2] R. D. Arnold, H. Yamaguchi, and T. Tanaka, "Search and rescue with autonomous flying robots through behavior-based cooperative intelligence," *Journal of International Humanitarian Action*, vol. 3, no. 1, p. 18, 2018.

[3] Z. Sui, Z. Pu, J. Yi, and S. Wu, "Formation control with collision avoidance through deep reinforcement learning using model-guided demonstration," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[4] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6382–6393.

[5] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[6] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 7265–7275.

[7] H. Mao, W. Liu, J. Hao, J. Luo, D. Li, Z. Zhang, J. Wang, and Z. Xiao, "Neighborhood cognition consistent multi-agent reinforcement learning," *arXiv preprint arXiv:1912.01160*, 2019.

[8] T. Wang, J. Wang, C. Zheng, and C. Zhang, "Learning nearly decomposable value functions via communication minimization," in *International Conference on Learning Representations*, 2019.

[9] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," in *International Conference on Learning Representations*, 2019.

[10] M. Hüttenrauch, S. Adrian, G. Neumann *et al.*, "Deep reinforcement learning for swarm systems," *Journal of Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019.

[11] A. Campbell and A. S. Wu, "Multi-agent role allocation: issues, approaches, and multiple perspectives," *Autonomous agents and multi-agent systems*, vol. 22, no. 2, pp. 317–355, 2011.

[12] M. Cossentino, V. Hilaire, A. Molesini, and V. Seidita, *Handbook on agent-oriented design processes*. Springer, 2014.

[13] K. M. Lhaksmana, Y. Murakami, and T. Ishida, "Role-based modeling for designing agent behavior in self-organizing multi-agent systems," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 01, pp. 79–96, 2018.

[14] T. Wang, H. Dong, V. Lesser, and C. Zhang, "Roma: Multi-agent reinforcement learning with emergent roles," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[15] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*.

[17] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation." in *NIPs*, vol. 99. Citeseer, 1999, pp. 1057–1063.

[18] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*. Curran Associates, Inc., 2019, pp. 3303–3313.

[19] H. Tang, J. Hao, T. Lv, Y. Chen, Z. Zhang, H. Jia, C. Ren, Y. Zheng, Z. Meng, C. Fan *et al.*, "Hierarchical deep multiagent reinforcement learning with temporal abstraction," *arXiv preprint arXiv:1809.09332*, 2018.

[20] J. Yang, I. Borovikov, and H. Zha, "Hierarchical cooperative multi-agent reinforcement learning with skill discovery," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1566–1574.

[21] A. Agarwal, S. Kumar, K. Sycara, and M. Lewis, "Learning transferable cooperative behavior in multi-agent team," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS'2020)*. IFMAS, 2020.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[23] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "Maven: Multi-agent variational exploration," 2019.

[24] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014.

[25] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[26] Z. Zhou, W. Zhang, J. Ding, H. Huang, D. M. Stipanović, and C. J. Tomlin, "Cooperative pursuit with voronoi partitions," *Automatica*, vol. 72, pp. 64–72, 2016.

[27] W. Wang, T. Yang, Y. Liu, J. Hao, X. Hao, Y. Hu, Y. Chen, C. Fan, and Y. Gao, "From few to more: Large-scale dynamic multiagent curriculum learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7293–7300.