entropy

Article

Scalable and Transferable Reinforcement Learning for Multi-Agent Mixed Cooperative–Competitive Environments Based on Hierarchical Graph Attention

Yining Chen [1], Guanghua Song [1,*], Zhenhui Ye [2] and Xiaohong Jiang [2]

[1] School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China; ch19930611@zju.edu.cn
[2] College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China; zhenhuiye@zju.edu.cn (Z.Y.); jiangxh@zju.edu.cn (X.J.)
* Correspondence: ghsong@zju.edu.cn

Abstract: Most previous studies on multi-agent systems aim to coordinate agents to achieve a common goal, but the lack of scalability and transferability prevents them from being applied to large-scale multi-agent tasks. To deal with these limitations, we propose a deep reinforcement learning (DRL) based multi-agent coordination control method for mixed cooperative–competitive environments. To improve scalability and transferability when applying in large-scale multi-agent systems, we construct inter-agent communication and use hierarchical graph attention networks (HGAT) to process the local observations of agents and received messages from neighbors. We also adopt the gated recurrent units (GRU) to address the partial observability issue by recording historical information. The simulation results based on a cooperative task and a competitive task not only show the superiority of our method, but also indicate the scalability and transferability of our method in various scale tasks.

Keywords: multi-agent; deep reinforcement learning; partial observability

check for updates

Citation: Chen, Y.; Song, G.; Ye, Z.; Jiang, X. Scalable and Transferable Reinforcement Learning for Multi-Agent Mixed Cooperative–Competitive Environments Based on Hierarchical Graph Attention. *Entropy* **2022**, *24*, 563. https://doi.org/10.3390/e24040563

Academic Editors: Yaniv Altshuler, Francisco Camara Pereira and Eli David

Received: 28 March 2022
Accepted: 15 April 2022
Published: 18 April 2022

1. Introduction

The last few years witnessed the rapid development of the multi-agent system. Due to its ability to solve complex computing or coordinating problems [1], it has been widely used in different fields, such as computer networks [2,3], robotics [4,5], etc. In the multi-agent system, agents try to learn their policies and execute tasks collaboratively, in either cooperative or competitive environments, by making autonomous decisions. However, in large-scale multi-agent systems, partial observability, scalability, and transferability are three important issues to be addressed for developing efficient and effective multi-agent coordination methods. Firstly, it is impossible for agents to learn their policies from the global state of the environment, as it contains massive information about a large number of agents. Therefore, they need to communicate with other agents in some ways, to reach consensus on decision making. Secondly, the previous methods either learn a policy to control all agents [6] or train their policies individually [7], which is difficult to be extended for large-scale agents. Thirdly, in existing deep-learning-based methods, the policies are trained and tested under the same number of agents, making them untransferable to different scales.

In this paper, we propose a scalable and transferable multi-agent coordination control method, based on deep reinforcement learning (DRL) and hierarchical graph attention networks (HGAT) [8], for mixed cooperative-competitive environments. By regarding the whole system as a graph, HGAT helps agents extract the relationships among different groups of entities in their observations and learn to selectively pay attention to them, which brings high scalability when applying in large-scale multi-agent systems. We enforce inter-agent communication to share agents' local observations with their neighbors and process the received messages through HGAT; therefore, agents can reach consensus by learning from their local observations and information aggregated from the neighbors. Moreover, we introduce the gated recurrent unit (GRU) [9] into our method to record the historical information of agents and utilize it when determining actions, which optimizes the policies under partial observability. We also apply parameter sharing to make our method transferable. Compared with previous works, our method achieves a better performance in mixed cooperative–competitive environments while acquiring high scalability and transferability.

The rest of this paper is organized as follows. In Section 2, we review the related works. We describe some background knowledge of multi-agent reinforcement learning and hierarchical graph attention networks in Section 3. In Section 4, we describe a cooperative scenario, UAV recon and a competitive scenario, predator-prey. We present the mechanism of our method in Section 5. The simulation results are shown in Section 6. We discuss advantages of our method in Section 7 and draw the conclusion in Section 8. The list of abbreviations is shown in Abbreviations.

2. Related Work

Multi-agent coordination has been studied extensively in recent years and implemented in various frameworks, including heuristic algorithms and reinforcement learning (RL) algorithms. In [10], the authors presented a solution to the mission planning problems in multi-agent systems. They encoded the assignments of tasks as alleles and applied the genetic algorithm (GA) for optimization. The authors of [11] designed a control method for the multi-UAV cooperative search-attack mission. UAVs employ ant colony optimization (ACO) to perceive surrounding pheromones and plan flyable paths to search and attack fixed threats. The authors of [12] focused on the dynamic cooperative cleaners problem [13], and presented a decentralized algorithm named "sweep" to coordinate several agents to cover an expanding region of grids. It was also used to navigate myopic robots who cannot communicate with each other [14]. In [15], the authors designed a randomized search heuristic (RSH) algorithm to solve the coverage path planning problem in multi-UAV search and rescue tasks, where the search area is transformed into a graph. The authors of [16] proposed a centralized method to navigate UAVs for crowd surveillance. They regarded the multi-agent system as a single agent and improved its Quality of Service (QoS) by using an on-policy RL algorithm state-action-reward-state-action (SARSA) [17]. Ref. [18] proposed a distributed task allocation method based on Q-learning [19] for cooperative spectrum sharing in robot networks, where each robot maximizes the total utility of the system by updating its local Q-table.

However, as the scale of multi-agent systems increases, the environment becomes more complex while the action space of the whole system expands exponentially. It is difficult for heuristic algorithms and the original RL methods to coordinate agents since they need more time and storage space to optimize their policies. Combining deep neural networks (DNNs) and RL algorithms, deep reinforcement learning (DRL) is widely used for multi-agent coordination in cooperative or competitive environments. It extracts features from the environment state with DNN and uses them to determine actions for agents, which brings better performance. Moreover, since the environment is affected by the action of all agents in multi-agent systems, it is hard for adversarial deep RL [20] to train another policy to generate possible disturbances from all agents. Semi-supervised RL [21] also fails to apply in multi-agent systems, as it cannot learn to evaluate the contribution of each agent from the global state and their actions. DRL can either control the whole multi-agent system by a centralized policy (such as [6]) or control agents individually in a distributed framework called multi-agent reinforcement learning (MARL). In a large-scale environment, MARL is more robust and reliable than the centralized methods because each agent can train its policies to focus on its local observation instead of learning from the global state.

The goal of MARL is to derive decentralized policies for agents and impose a consensus to conduct a collaborative task. To achieve this, the multi-agent deep deterministic policy gradient (MADDPG) [22] and counterfactual multi-agent (COMA) [23] construct a centralized critic to train decentralized actors by augmenting it with extra information about other agents, such as observations and actions. Compared with independent learning [24], which only uses local information, MADDPG and COMA can derive better policies in a non-stationary environment. However, it is difficult for these approaches to be applied in a large-scale multi-agent system, as they directly use the global state or all observations when training. Multi-actor-attention-critic (MAAC) [25] applies the attention mechanism to improve scalability by quantifying the importance of each agent through the attention weights. Deep graph network (DGN) [26] regards the multi-agent system as a graph and employs a graph convolutional network (GCN) with shared weight to process information from neighboring nodes, which also brings high scalability. Ref. [8] proposed a scalable and transferable model, named the hierarchical graph attention-based multi-agent actor-critic (HAMA). It clusters all agents into different groups according to prior knowledge and constructs HGAT to extract the inter-agent relationships in each group of agents and inter-group relationships among groups, aggregating them into high-dimensional vectors. By using MADDPG with shared parameters to process those vectors and determine actions, HAMA can coordinate agents better than the original MADDPG and MAAC when executing cooperative and competitive tasks.

Various MARL-based methods have recently been proposed for multi-agent coordination. Ref. [27] designed a distributed method to provide long-term communication coverage by navigating several UAV mobile base stations (UAV-MBSs) through MADDPG. Ref. [7] presented a MADDPG-based approach that jointly optimizes the trajectory of UAVs to achieve secure communications, which also enhanced the critic with the attention mechanism, such as [25]. The authors of [28] adopted GCN to solve the problem for large-scale multi-robot control. Ref. [29] separated the search problem in indoor environments into high-level planning and low-level action. It applied trust region policy optimization (TRPO) [30] as the global and local planners to handle the control at different levels. In our previous work, we proposed the deep recurrent graph network (DRGN) [31], a novel method that is designed for navigation in a large-scale multi-agent system. It constructs inter-agent communication based on a graph attention network (GAT) [32] and applies GRU to recall the long-term historical information of agents. By utilizing extra information from neighbors and memories, DRGN performs better than DQN and MAAC when navigating a large-scale UAV-MBS swarm to provide communication services for targets that are randomly distributed on the ground.

The difference between our method and the previous works are summarized as follows. DRGN represents the observation as a pixel map of the observable area and processes it by DNN. Our method regards the global state as a graph where the nodes represent the entities in the environment and employs HGAT to process the observation. It is more effective for our method to learn relationships between agents and entities through HGAT. Moreover, our method spends less space to store the observation than DRGN, as the scale of the observation in our method is independent of the observation range. In HAMA, each agent observes up to K nearest neighboring entities per type, where K is a constant. Our method considers that agents can observe all entities inside the observation range and uses an adjacency matrix to denote the relationships of the observation, which can describe the actual observation of agents more accurately than HAMA. In addition, our method introduces GRU and HGAT-based inter-agent communication to provide extra information for agents, so they can optimize policies for coordination by learning from historical information and neighbors.

3. Background

3.1. Multi-Agent Reinforcement Learning (MARL)

The process of MARL is regarded as a decentralized partially observable Markov decision process (Dec-POMDP) [33]. In MARL, each agent i observes the environment state s and obtains a local observation o_i. Then, it selects an action according to its policy π_i. The environment executes the joint actions $\boldsymbol{a} = (a_1, \cdots, a_N)$ and transforms s to the next state s'. After execution, each agent acquires a reward $r_i = \mathcal{R}_i(s, \boldsymbol{a})$ and a next observation o'_i from the environment. Each agent aims to optimize its policy to maximize its total expected return $R_i = \sum_{t=0}^{T} \gamma^t r_i(t)$, where T is a final timeslot, and $\gamma \in [0, 1]$ is the discount factor.

Q-learning [19] and policy gradient [34] are two popular RL methods. The idea of Q-learning is to estimate an state-action value function $Q(s, a) = \mathbb{E}[R]$ and select the optimal action to maximize $Q(\cdot)$. Deep Q-network (DQN) [35], a Q-learning-based algorithm, uses a DNN as a function approximator and trains it by minimizing the loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}[(y - Q(s, a|\theta))^2] \tag{1}$$

where θ is the parameter of the DNN. The target value y is defined as $y = r + \gamma \max_{a'} Q'(s', a')$ [35], where Q' is the target network, whose parameters are periodically updated from θ. DQN also applies a replay buffer to stabilize learning.

Policy gradient directly optimizes the policy π to maximize $\mathcal{J}(\theta^\pi) = \mathbb{E}[R]$ and updates parameters based on the gradient [34]:

$$\nabla_{\theta^\pi} \mathcal{J}(\theta^\pi) = \mathbb{E}_{s \sim p^\pi, a \sim \pi}[\nabla_{\theta^\pi} \log \pi(a|s, \theta^\pi) Q(s, a)] \tag{2}$$

where p^π is the state distribution. $Q(s, a)$ can be estimated by samples [36] or a function approximator, such as DQN, which leads to the actor–critic algorithm [37].

3.2. Hierarchical Graph Attention Network (HGAT)

HGAT is an effective method for processing hierarchically structured data represented as a graph and introduced into MARL to extract the relationships among agents. By stacking multiple GATs hierarchically, HGAT firstly aggregates embedding vectors e^l_{ij} from neighboring agents in each group l into e'^l_i and subsequently aggregates e'^l_i from all groups into e'_i. The aggregated embedding vector e'_i represents the hierarchical relationships among different groups of neighbors.

4. System Model and Problem Statement

In this section, we describe the settings of a multi-agent cooperative scenario, UAV recon and a competitive scenario, predator-prey.

4.1. UAV Recon

As shown in Figure 1a, we deploy N UAVs into a hot-spot area to scout n point-of-interests (PoIs) for T timeslots, where PoIs are randomly distributed. As we consider our UAVs to move at the same altitude, the area of our mission is two-dimensional. Each UAV has a circled recon area whose radius is considered as a recon range. If the Euclidean distance between a UAV and a PoI is less than the recon range, we consider the PoI to be covered.

In the beginning, each UAV is deployed in a random position. At each timeslot t, each UAV i determines its acceleration $acc_i \in \{(acc, 0), (-acc, 0), (0, acc), (0, -acc), (0, 0)\}$ as its action. The action space of i is discrete. The energy consumption of i is defined as:

$$E_i = E_h + \frac{v_i}{v_{max}} E_m \tag{3}$$

where v_i is the velocity of i and v_{max} is the maximum velocity of UAVs. E_h and E_m are the energy consumption for hovering and movement, respectively.

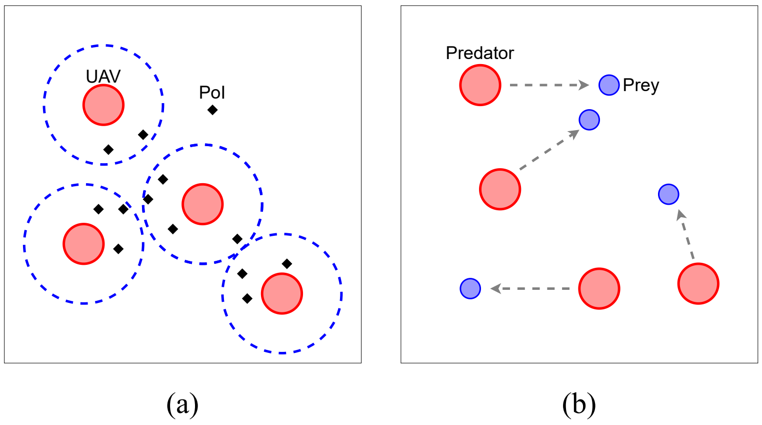

Figure 1. Illustrations of (**a**) *UAV Recon* and (**b**) *Predator-Prey*.

In our scenario, our goals of UAVs are to cover more PoIs more fairly with less energy consumption. To evaluate the quality of tasks, we consider three metrics: coverage C, fairness F, and energy consumption E. The score of C denotes the proportion of covered PoIs, which is defined as:

$$C = \frac{n_C(t)}{n} \tag{4}$$

where $n_C(t)$ is the number of covered PoIs at timeslot t.

The score of fairness denotes how fair all PoIs are covered. Here, we use Jain's fairness index [38] to define the score of F as:

$$F = \frac{(\sum_{j=1}^{n} c_j)^2}{n \sum_{j=1}^{n} c_j^2} \tag{5}$$

while c_j is the coverage time of PoI j.

Finally, UAVs need to control energy consumption in tasks. We define the score of E as:

$$E = \frac{1}{N} \sum_{i=1}^{N} E_i \tag{6}$$

When executing recon missions, each UAV needs to observe local states of other UAVs and PoIs to determine its action. The local state of UAV i is defined as $s_i = (P_i, v_i)$, where P_i and v_i are the position and the velocity of i, respectively. Each PoI j's local state $s_j = (P_j)$. If a PoI is in UAV i's observation range, we consider the PoI is observed by i. If another UAV j is in i's communication range, we consider i can communicate with j. To train UAV's policy, we define a heuristic reward r_i as:

$$r_i = \frac{\eta_1 \times r_{indv} + \eta_2 \times r_{shared}}{E_i} - p_i \tag{7}$$

where p_i is a penalty factor. When UAV i flies across the border, it is penalized by p_i. $r_{indv} = -1$ if no PoIs is covered by i individually, otherwise $r_{indv} = n_{indv}$, where n_{indv} means the number of PoIs that are only covered by i. $r_{shared} = 0$ if i does not share PoIs with others, otherwise $r_{shared} = \frac{n_{shared}}{N_{share}}$, where n_{shared} denotes the number of PoIs which are covered by N_{share} neighboring UAVs. η_1 and η_2 are the importance factor of $r_{individual}$ and r_{shared}, respectively. We empirically set $\eta_1 \gg \eta_2$ to encourage UAVs to cover more PoIs and avoid overlapping with others.

4.2. Predator-Prey

As shown in Figure 1b, we deploy $N_{predator}$ predators to eliminate N_{prey} prey.

Both of them are controlled by a DRL-based method. If the distance between a predator and a prey is less than predators' attack range, we consider the prey to be eliminated. The goal of the predators is to eliminate all prey, while the goal of the prey is to escape from the predators. The speed of the predators is slower than the prey speed, so they need to cooperate with each other when chasing prey.

The action space of the predators and the prey is the same as the UAVs in the UAV recon scenario. The local state of each predator or prey is defined as $s_i = (P_i, v_i)$, where P_i and v_i are the position and the velocity of a predator or prey, respectively. We consider that each predator and prey can observe the local state of adversaries inside its observation range, while it can communicate with companions inside its communication range. The eliminated preys can neither be observed nor communicate with others. To evaluate the performance of predators and prey, we define the score as:

$$S = \frac{T - T_{eliminate}}{T} \tag{8}$$

where T is the total timeslots of an episode, while $T_{eliminate}$ is the timeslot when all prey are eliminated.

When predator i eliminates prey j, i will obtain a positive reward, while j will obtain a negative reward. When all prey are eliminated, the predators will get an additional reward.

5. HGAT-Based Multi-Agent Coordination Control Method

To achieve the goals of two scenarios described in Section IV, we present a multi-agent coordination control method based on HGAT for mixed cooperative–competitive environments. In our method, the global state of the environment is regarded as a graph, containing the local state of agents and the relationship among them. Each agent summarizes the information from the environment by HGAT and subsequently computes the Q-value and action in a value-based or actor–critic framework.

5.1. HGAT-Based Observation Aggregation and Inter-Agent Communication

In the multi-agent system, the environment involves multiple kinds of entities, including agents, PoIs, etc. As they are heterogeneous, agents need to treat their local states and model their relationships separately. Thus, we categorize all entities into different groups at the first step of execution in cooperative or competitive scenarios. As shown in Figure 2, M entities (containing N agents) are clustered into K groups and represent the environment's state as graphs. The agents construct an observation graph $G^{\mathcal{O}}$ and a communication graph $G^{\mathcal{C}}$ respectively based on their observation ranges $\mathcal{O}_1, \cdots, \mathcal{O}_N$ and communication ranges $\mathcal{C}_1, \cdots, \mathcal{C}_N$. The edges of $G^{\mathcal{O}}$ represent that the entities can be observed by agents, while the edges of $G^{\mathcal{C}}$ represent that two of the agents can communicate with each other. The adjacency matrix of $G^{\mathcal{O}}$ and $G^{\mathcal{C}}$ are $\mathbf{A}^{\mathcal{O}}$ and $\mathbf{A}^{\mathcal{C}}$, respectively. i's observation is defined as $o_i = \{s_j | j \in \mathcal{O}_i\}$. Its received messages from the others is $m_i = \{m_{ji} | j \in \mathcal{C}_i\}$, where s_j is agent j's local state and m_{ji} is the message that j sends to i.

At each timeslot, the agents use the network shown in Figure 3 to determine their actions according to $\mathbf{s}$, $\mathbf{A}^{\mathcal{O}}$, and $\mathbf{A}^{\mathcal{C}}$ received from the environment, where $\mathbf{s} = (s_1, , \cdots, s_M)$. The parameters of the network are shared among the agents in the same group. The network contains three components, a set of encoders, two stacked HGAT layers, and a recurrent unit, which consists of a gated recurrent unit (GRU) layer and a fully connected layer. GRU is a variant of the recurrent neural network (RNN). To summarize the information in each agent i's observation o_i, the first HGAT layer processes o_i into a high-dimensional aggregated embedding vector e'_i as shown in Figure 4. Firstly, the encoder which consists of a fully connected layer transforms the local states from each group l into embedding vectors as $e_j = f_e^l(s_j)$, where f_e^l means the encoder for group l. e_j is the embedding vector for entity j in group l. Then, it aggregates e_j as ${e'}_i^l = \sum_j \alpha_{ij} \mathbf{W}_v^l e_j$ [32], where $\mathbf{W}_v^l$ is a matrix that transforms

e_j into a "value". The attention weight α_{ij} represents the importance of the embedding vector e_j from j to i, which is calculated by softmax as $\alpha_{ij} \propto \exp({e_j}^{\mathrm{T}} {\mathbf{W}_k^l}^{\mathrm{T}} \mathbf{W}_q e_i)$ [32] if $a_{i,j}^{\mathcal{O}}$ in $\mathbf{A}^{\mathcal{O}}$ is 1, otherwise $\alpha_{ij} = 0$. $\mathbf{W}_k$ and $\mathbf{W}_q$ transform a embedding vector into a "key" and a "query", respectively. $\mathbf{A}^{\mathcal{O}}$ is used for selection so that only the local states from $\mathcal{O}_i$ are summarized. To improve the performance, we use the multiple attention heads here. Finally, ${e'}_i^l$ from all groups are aggregated into e_i' by a fully connected layer f_G, as:

$$e_i' = f_G(\|_{l=1}^K {e'}_i^l) \tag{9}$$

where $\|$ represents the concatenation operation. We do not apply another GAT for aggregating, such as HAMA, as our approach has less computing overhead.

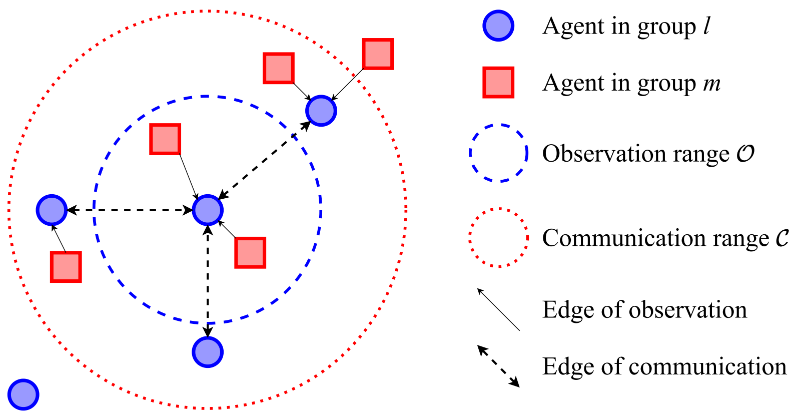

Figure 2. The clustering of agents and their topology.

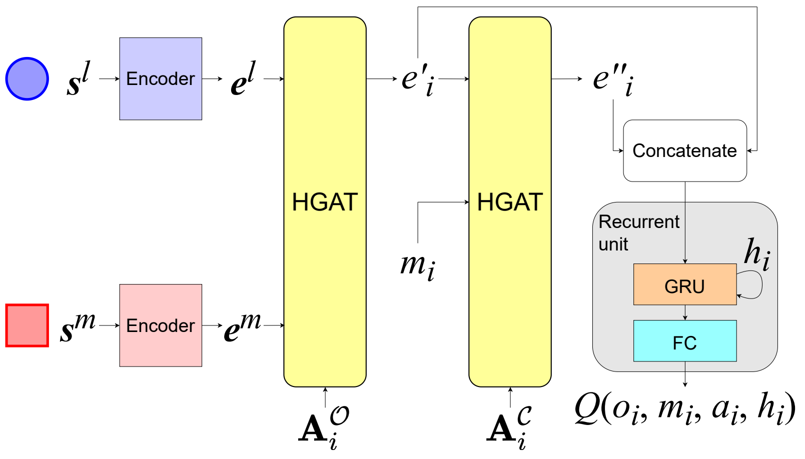

Figure 3. The overall structure of the network. s^l and e^l represent the local states and embedding vectors of agents in group l. $\mathbf{A}_i$ denotes the ith row of $\mathbf{A}$.

After calculating e_i', agent i sends it as a message m_{ij} to each neighboring agent j in $\mathcal{C}(i)$. Inter-agent communication helps agents to share their observations with neighbors, which brings a better performance in coordination. To summarize each agent i's received messages m_i, the second HGAT layer processes m_i and aggregates it into another embedding vector e_i'' by the same means as shown in Figure 4. The adjacency matrix used here is $\mathbf{A}^{\mathcal{C}}$ instead of $\mathbf{A}^{\mathcal{O}}$. Our method is capable of inner-group and inter-group communication and can easily extend to a multi-hop by stacking new HGAT layers.

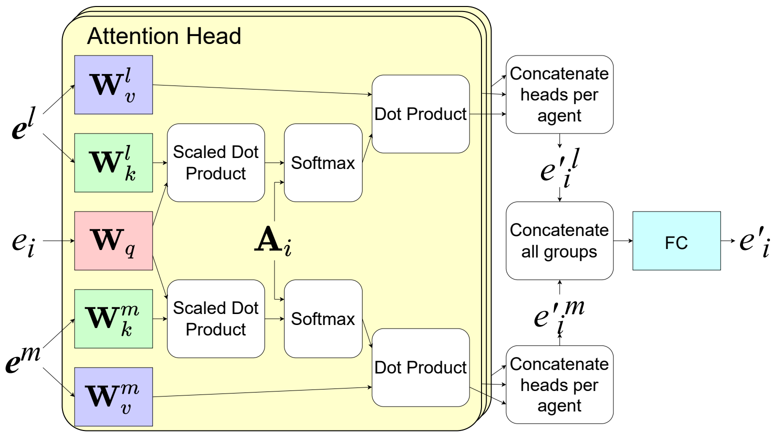

Figure 4. The architecture of an HGAT layer.

5.2. Implementation in a Value-Based Framework

This implement is based on DQN. Each agent i maintains hidden states h_i for the recurrent unit and calculates its Q-values by a Q-network, as shown in Figure 3. Similar to DQN, our method also employs a target network with the same structure.

We introduce the skip-connection strategy by concatenating e'_i and e''_i as an input of the recurrent unit when computing the Q-value, so agents can use the information both from their observation and others'. The Q-value is calculated as:

$$Q^l(o_i, m_i, a_i, h_i) \approx f_R^l(e'_i, e''_i, h_i) \tag{10}$$

where Q^l represents the Q-network of group l where i belongs, f_R^l means the recurrent unit in Q^l, and a_i is the action determined by i according to Q-values. We apply ϵ-greedy policy [35] to balance the exploitation and exploration as:

$$a_i = \begin{cases} \arg\max_{a \in \mathcal{A}_i} Q^l(o_i, m_i, a, h_i), & \textit{with probability } 1-\epsilon \\ \textit{random}(\mathcal{A}_i), & \textit{with probability } \epsilon \end{cases} \tag{11}$$

where $\mathcal{A}_i$ is the action space of i.

After executing the joint actions $\boldsymbol{a} = (a_1, \cdots, a_N)$, the environment transforms the current state to the next and sends the next local states $\boldsymbol{s}'$, the next adjacency matrix $\mathbf{A}'^{\mathcal{O}}$ and $\mathbf{A}'^{\mathcal{C}}$, and the reward r_i to each agent i. The experience $(\boldsymbol{s}, \mathbf{A}^{\mathcal{O}}, \mathbf{A}^{\mathcal{C}}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}', \mathbf{A}'^{\mathcal{O}}, \mathbf{A}'^{\mathcal{C}}, \boldsymbol{h}, \boldsymbol{h}')$ is stored in a shared replay buffer B, where $\boldsymbol{r} = (r_1, \cdots, r_N)$, $\boldsymbol{h} = (h_1, \cdots, h_N)$, and $\boldsymbol{h}' = (h'_1, \cdots, h'_N)$. h'_i is the next hidden state that the Q-network outputs when agent i calculates Q-values. h_i is initialized to zero at the beginning of an episode.

To training the Q-network of each group, we sample H experiences from B as a minibatch and minimize the loss:

$$\mathcal{L}(\theta_l^Q) = \frac{1}{N_l} \sum_{i=1}^{N_l} \mathbb{E}[(y_i - Q^l(o_i, m_i, a_i, h_i | \theta_l^Q))^2] \tag{12}$$

where N_l means the number of agents in group l and θ_l^Q denote the parameters of Q^l. y_i is the target value that calculated by the target network Q'^l, as:

$$y_i = r_i + \gamma \max_{a' \in \mathcal{A}_i} Q'^l(o'_i, m'_i, a', h'_i | \theta_l^{Q'}) \tag{13}$$

where o'_i and m'_i are i's next observation and next received messages, respectively. $\theta_l^{Q'}$ denote the parameters of Q'^l, which are periodically updated from θ_l^Q.

5.3. Implementation in an Actor–Critic Framework

Our method can also be implemented on the actor–critic framework. In this implementation, each agent i has an actor network and a critic network, maintaining hidden states h_i^π and h_i^Q. After obtaining $\mathbf{s}$, $\mathbf{A}^\mathcal{O}$ and $\mathbf{A}^\mathcal{C}$, agent i in group l computes the probability of actions as:

$$\pi^l(o_i, m_i, h_i^\pi) \approx f_R^{\pi^l}(e'^\pi_i, e''^\pi_i, h_i^\pi) \tag{14}$$

where π^l represents the actor network of group l and $f_R^{\pi^l}$ represents the recurrent unit in π^l. We employ the ϵ-categorical policy here. Agent i determines an action based on $\pi^l(o_i, m_i, h_i^\pi)$ with probability $1-\epsilon$ and makes a random choice with probability ϵ. The critic network Q^l subsequently calculates Q-values, such as the value-based framework. The hidden states h_i^π and h_i^Q and the next hidden states h'^π_i and h'^Q_i are stored in the replay buffer, where h'^π_i and h'^Q_i are the outputs of π^l and Q^l, respectively.

The critic network of each group is trained by minimizing the loss $\mathcal{L}(\theta_l^Q)$, which is computed as (13). As the actor–critic framework selects actions according to $\pi^l(o_i, m_i, h_i^\pi)$ instead of the maximum Q-value, we use the expectation of the next state's Q-value to calculate the target value y_i as:

$$y_i = r_i + \gamma \sum_{a' \in \mathcal{A}_i} \pi'^l(a'|o'_i, m'_i, h'^\pi_i, \theta_l^{\pi'}) Q'^l(o'_i, m'_i, a', h'^Q_i|\theta_l^{Q'}) \tag{15}$$

where $\theta_l^{\pi'}$ and $\theta_l^{Q'}$ are the parameters of target network π'^l and Q'^l, respectively.

The actor network of each group is trained according to the gradient:

$$\nabla_{\theta_l^\pi}(\mathcal{J}(\theta_l^\pi)) = \frac{1}{N_l} \sum_{i=1}^{N_l} \mathbb{E}[\log \pi^l(a_i|o_i, m_i, h_i^\pi, \theta_l^\pi)(Q^l(o_i, m_i, a_i, h_i^Q|\theta_l^Q) - b_i)] \tag{16}$$

where the baseline b_i is designed to reduce variance and stabilize training [39], which is defined as:

$$b_i = \sum_{a \in \mathcal{A}_i} \pi^l(a|o_i, m_i, h_i^\pi, \theta_l^\pi) Q^l(o_i, m_i, a, h_i^Q|\theta_l^Q) \tag{17}$$

After training, $\theta_l^{\pi'}$ and $\theta_l^{Q'}$ are updated as $\theta_l^{\pi'} \leftarrow \tau\theta_l^\pi + (1-\tau)\theta_l^{\pi'}$, and $\theta_l^{Q'} \leftarrow \tau\theta_l^Q + (1-\tau)\theta_l^{Q'}$, respectively [40].

Our method can be extended to continuous action space by estimating the expectation of b_i with Monte Carlo samples or a learnable state value function $V(o_i, m_i)$ [23].

6. Simulation

6.1. Set Up

To evaluate the performance of our method, we conduct a series of simulations on an Ubuntu 18.04 server with 2 NVIDIA RTX 3080 GPUs. We implement a value-based (VB) version and an actor–critic (AC) version of our method in PyTorch. Each fully connected layer and GRU layer contains 256 units. The activation functions in encoders and HGAT layers are ReLU [41]. The number of attention heads is 4. Empirically, we set the learning rate of the optimizer to 0.001, and the discount factor γ to 0.95. The replay buffer size is 50 K and the size of a minibatch is 128. ϵ is set to 0.3. For the value-based version, The target networks are updated every five training steps. For the actor–critic version, we set τ to 0.01. The networks are trained every 100 timeslots and update their parameters four times in a training step.

We compare our method with four MARL baselines, including DGN, DQN, HAMA, and MADDPG. For non-HGAT-based approaches, each agent concatenates all local states in its observation into a vector, while padding 0 for unobserved entities. The parameters of networks are shared among agents in all baselines except MADDPG. We use the Gumbel-Softmax reparameterization trick [42] in HAMA and MADDPG to make them trainable in discrete action spaces. DGN is based on our proposed algorithm [31], which applies a GAT layer for inter-agent communication. We train our method and each baseline for 100 K episodes and test them for 10 K episodes.

6.2. UAV Recon

As summarized in Table 1, we deploy several UAVs in a 200 × 200 area where 120 PoIs are distributed. The penalty factor p in (7) is set to 1. We evaluate the performance of our method in the test stage under different number of UAVs and compare it with baselines.

Table 1. Experiment parameters of UAV recon.

Parameters	Settings
Target Area	200 × 200
Number of PoIs	120
Recon Range	10
Observation Range	15
Communication Range	30
Maximum Speed	10/s
Energy Consumption for Hovering	0.5
Energy Consumption for Movement	0.5
Penalty Factor p	1
Importance Factor η_1	1
Importance Factor η_2	0.1
Timeslots of Each Episode	100

Figure 5 shows the performance of each method in terms of coverage, fairness, and energy consumption under different numbers of UAVs. Note that both two versions of our method are trained with 20 UAVs and transferred to a different scale of UAV swarms. From Figure 5a,b, we observe that our method outperforms all baselines in terms of coverage and fairness. Compared with DGN and DQN, our method employs HGAT to extract features from observation, which is more effective than processing raw observation vectors directly. Therefore, our method helps UAVs to search PoIs and better optimize their flight trajectories. Although HAMA also applies HGAT, UAVs cannot cooperate as effectively as our method, owing to the lack of communication. In our method, the UAVs communicate with others and process received messages by another HGAT layer. Furthermore, the recurrent unit helps UAVs to learn from the hidden states, which induces a better performance. In MADDPG, each UAV trains an individual network and concatenates observations and actions of all agents into a high-dimensional vector as an input of the critic. As the networks in MADDPG expands exponentially to the scale of the agents, it is hard to be trained effectively and efficiently in large-scale multi-agent systems. As a consequence, the MADDPG consumes more time to train but obtains the lowest score.

Figure 5c indicates that our method consumes less energy than DGN and DQN. As their flight trajectories are better, UAVs can cover more PoIs fairly while consuming less energy. The energy consumption of HAMA is considerable with our method in low-scale environments and increases when the number of UAVs is up to 40. MADDPG fails to improve coverage and fairness, so it tends to save on energy to maximize its reward.

To test the capability of transferred learning, we compare the transferred policies with those trained under the same settings of testing. As shown in Figure 6, the performance does not deteriorate when the policy is transferred to execute with 10, 30, or 40 UAVs, which indicates that our method is highly transferable under various numbers of UAVs.

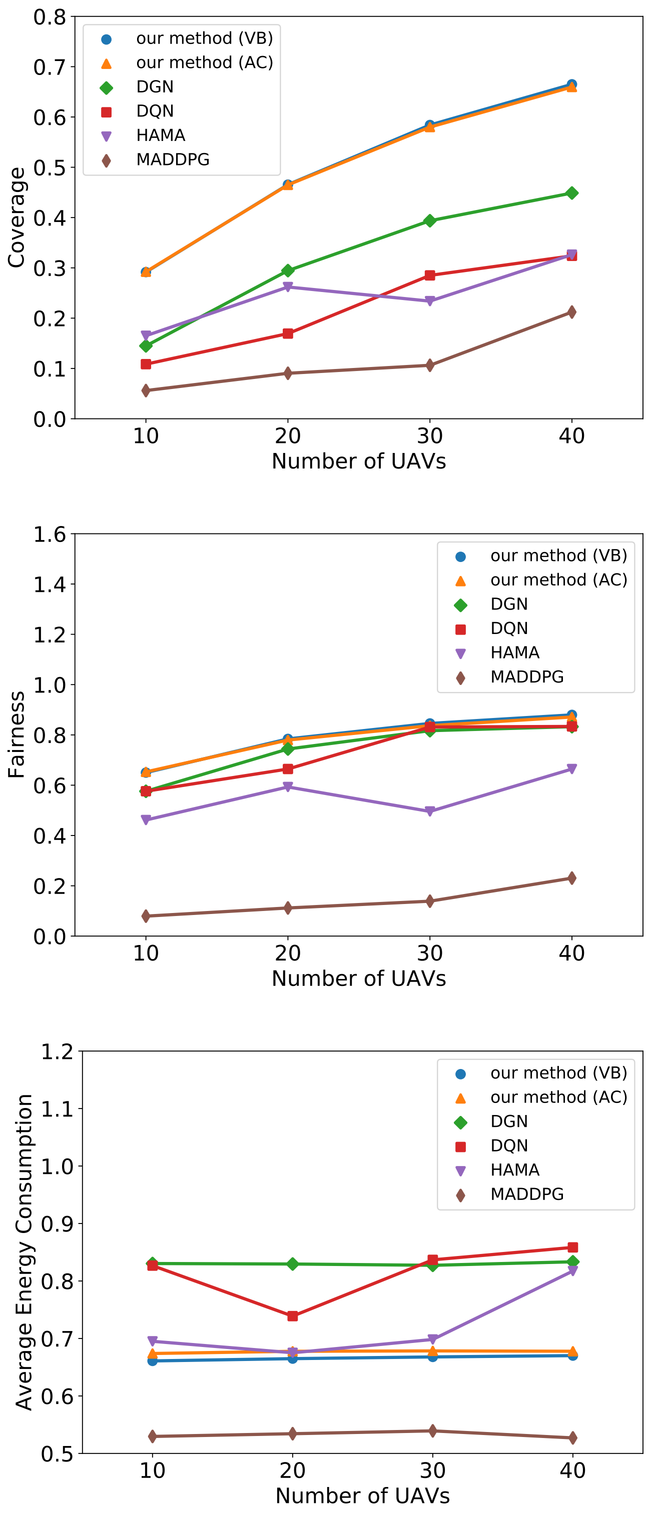

Figure 5. Simulation results of all methods on coverage, fairness, and energy consumption under different number of UAVs.

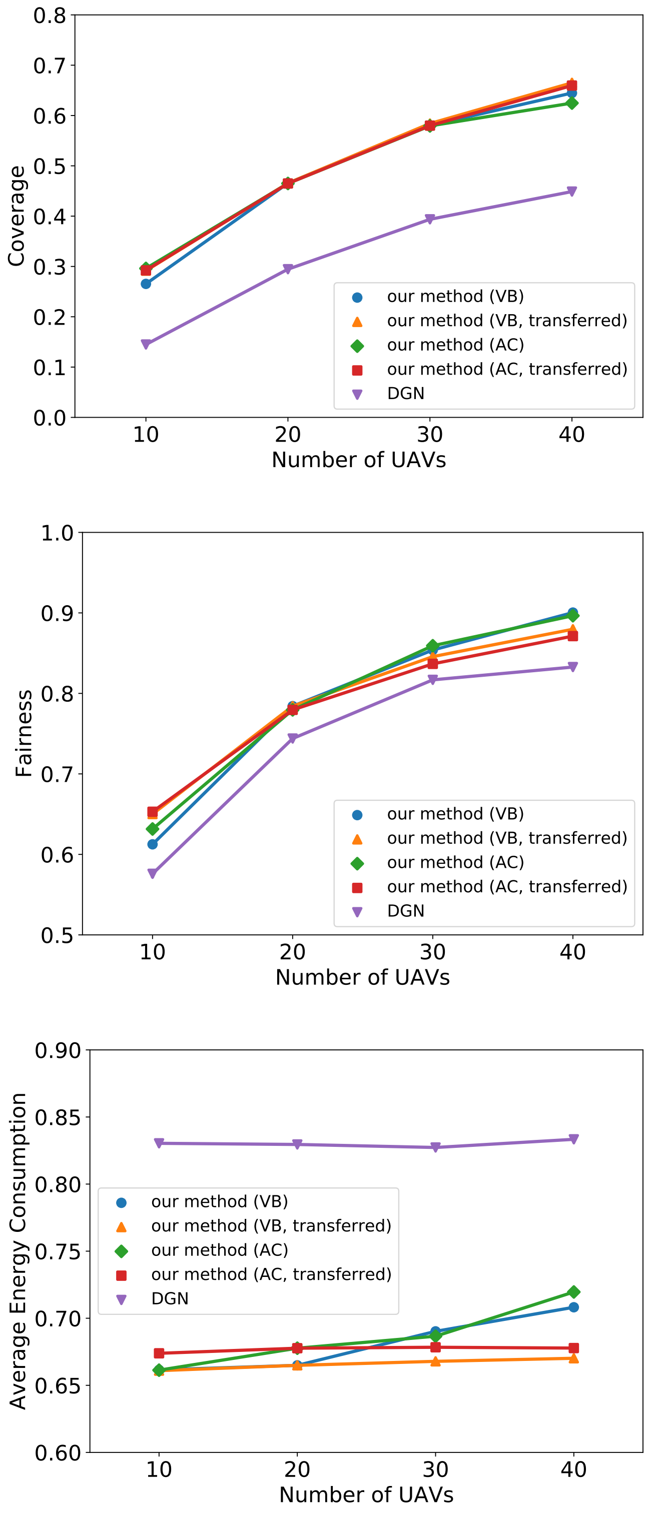

Figure 6. Simulation results of transfer learning on coverage, fairness, and energy consumption under different number of UAVs.

6.3. Predator-Prey

As summarized in Table 2, we deploy five predators in a 100 × 100 area to eliminate five prey. We set the attack reward of predators and prey to 10 and −10, respectively. The additional reward is set as $r_{additional} = 10 \times S$. We train the policy by the value-based version of our method and test it by competing with other policies trained by different methods.

Table 2. Experiment parameters of predator-prey.

Parameters	Settings
Target Area	100×100
Number of Predators	5
Number of Preys	5
Attack Range	8
Observation Range	30
Communication Range	100
Maximum Speed of Predators	10/s
Maximum Speed of Preys	12/s
Timeslots of Each Episode	100

Table 3 indicates that our method shows its superiority over all baselines in both roles of predator and prey. By introducing GRU and inter-agent communication, the predators obtain more information from hidden states and neighbors to decide which prey to capture. It is more flexible for predators to determine whether to chase prey individually or cooperatively. Similarly, GRU and inter-agent communication also bring more information to prey, so they can choose from various strategies to survive. For example, prey can escape from predators by their faster speed or sacrifice one of them to distract predators.

Table 3. The mean and standard deviation of scores in predator-prey.

Predator	**Prey**		
	Our Method	**DGN**	**DQN**
Our method	**0.331** ± 0.088	**0.535** ± 0.086	**0.591** ± 0.101
DGN	0.051 ± 0.060	0.271 ± 0.095	0.386 ± 0.095
DQN	0.014 ± 0.034	0.173 ± 0.086	0.120 ± 0.078
Predator	**Prey**		
	Our Method	**HAMA**	**MADDPG**
Our method	**0.331** ± 0.088	**0.787** ± 0.027	**0.472** ± 0.098
HAMA	0.051 ± 0.050	0.351 ± 0.050	0.403 ± 0.091
MADDPG	0.038 ± 0.048	0.239 ± 0.090	0.051 ± 0.057

7. Discussion

The experimental results indicate that the performance of our method is superior to those of others in both cooperative and competitive scenarios. We assume that three components, including HGAT, GRU, and inter-agent communication, are the key factors which induce the success of our method. To validate our hypothesis, we conduct an ablation study in Appendix A to clarify the necessity of each component (HGAT, GRU, and inter-agent communication).

From Table A1, we observe a significant deterioration in performance when removing HGAT or GRU, while disabling inter-agent communication also induces a decrease in terms of coverage and fairness. To explain the necessity of each component, we assume the following reasons. Firstly, HGAT plays an important role in summarizing observations. Not only does it process the local states from all groups, but it also quantifies their importance with attention weights. In addition, HGAT models the hierarchical relationships among agents as a graph, which is effective for them to optimize their policies in dynamic environments. Secondly, GRU makes a significant contribution to overcoming the limitation of partial observability. When determining actions, GRU helps agents to remember the historical information recorded in the hidden states, such as the position of the observed PoIs in the UAV recon. It is beneficial for agents to improve their performance by getting what they cannot observe from the hidden states. Finally, inter-agent communication expands agents' horizons. By sending high-dimensional embedding vectors, they share their observations with others. With the help of HGAT, agents can cooperate better by using extensive information from those vectors in decision making.

Compared with non-HGAT-based approaches, our method has another advantage in the replay buffer. As they concatenate all local states into a vector and pad 0 for unobserved entities, the space complexity of observations in their replay buffer is $O(N \times M)$, where N and M means the number of agents and entities, respectively. However, our method only stores local states, whose space complexity is $O(M)$. Although it has to store the adjacency matrices $\mathbf{A}^{\mathcal{O}}$ to represent the relationship among agents, this is more economical than storing observations in terms of storage, as an adjacency matrix represents the relationship between agents by a bit.

8. Conclusions

In this paper, we propose a scalable and transferable DRL-based multi-agent coordination control method for cooperative and competitive tasks. This method introduces HGAT, GRU, and inter-agent communication into DRL to improve performance in mixed cooperative–competitive environments. By intensive simulations, our method shows its superiority over DGN, DQN, HAMA, and MADDPG both in UAV recon and predator-prey.

In the future, we will improve our method by introducing an adaptive policy base on the action entropy of the agent to provide a more intelligent exploration. We will evaluate the performance of the entropy-based policy and compare it with the ϵ-greedy and ϵ-categorical policies. Specifically, we will test the capabilities of automating entropy adjustment under different entropy targets in large-scale multi-agent systems. Furthermore, we will try to extend our method into continuous policies and evaluate its performance in cooperative and competitive scenarios with continuous action space.

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

Funding: This research was funded by the National key R & D program of China sub project "Emergent behavior recognition, training and interpretation techniques" grant number 2018AAA0102302.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

AC	Actor–Critic
ACO	Ant Colony Optimization
COMA	COunterfactual Multi-Agent
Dec-POMDP	Decentralized Partially Observable Markov Decision Process
DGN	Deep Graph Network
DNN	Deep Neural Network
DQN	Deep Q-Network
DRGN	Deep Recurrent Graph Network
DRL	Deep Reinforcement Learning

GA	Genetic Algorithm
GAT	Graph Attention neTwork
GCN	Graph Convolutional Network
GRU	Gated Recurrent Unit
HAMA	Hierarchical Graph Attention-Based Multi-Agent Actor–Critic
HGAT	Hierarchical Graph Attention neTwork
MAAC	Multi-Actor-Attention-Critic
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MARL	Multi-Agent Reinforcement Learning
PoI	Point-of-Interest
QoS	Quality of Service
RL	Reinforcement Learning
RSH	Randomized Search Heuristic
SARSA	State-Action-Reward-State-Action
TRPO	Trust Region Policy Optimization
UAV	Unmanned Aerial Vehicle
UAV-MBS	Unmanned Aerial Vehicle Mobile Base Station
VB	Value-Based

Appendix A. Ablation Study

In the ablation study, we test our method and three variants in UAV recon with 20 UAVs and summarized the performances in Table A1. The variants are described as follows:

- Without H: removing the first HGAT layer;
- Without G: removing GRU in the recurrent unit;
- Without C: disabling inter-agent communication.

Table A1. The mean and standard deviation of three metrics in the ablation study ($N = 20$).

Model	Metric		
	Coverage	Fairness	Energy Consumption
Our method	**0.466** ± 0.046	**0.784** ± 0.061	**0.665** ± 0.015
Without H	0.103 ± 0.022	0.696 ± 0.087	0.710 ± 0.011
Without G	0.349 ± 0.065	0.726 ± 0.116	0.730 ± 0.017
Without C	0.412 ± 0.058	0.749 ± 0.081	0.668 ± 0.023

References

1. Dorri, A.; Kanhere, S.S.; Jurdak, R. Multi-agent systems: A survey. *IEEE Access* **2018**, *6*, 28573–28593. [CrossRef]
2. Gutierrez-Garcia, J.O.; Sim, K.M. Agent-based cloud bag-of-tasks execution. *J. Syst. Softw.* **2015**, *104*, 17–31. [CrossRef]
3. Claes, R.; Holvoet, T.; Weyns, D. A decentralized approach for anticipatory vehicle routing using delegate multiagent systems. *IEEE Trans. Intell. Transp. Syst.* **2011**, *12*, 364–373. [CrossRef]
4. Ota, J. Multi-agent robot systems as distributed autonomous systems. *Adv. Eng. Inform.* **2006**, *20*, 59–70. [CrossRef]
5. Iñigo-Blasco, P.; Diaz-del Rio, F.; Romero-Ternero, M.C.; Cagigas-Muñiz, D.; Vicente-Diaz, S. Robotics software frameworks for multi-agent robotic systems development. *Robot. Auton. Syst.* **2012**, *60*, 803–821. [CrossRef]
6. Liu, C.H.; Chen, Z.; Tang, J.; Xu, J.; Piao, C. Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2059–2070. [CrossRef]
7. Zhang, Y.; Mou, Z.; Gao, F.; Jiang, J.; Ding, R.; Han, Z. UAV-enabled secure communications by multi-agent deep reinforcement learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 11599–11611. [CrossRef]
8. Ryu, H.; Shin, H.; Park, J. Multi-agent actor-critic with hierarchical graph attention network. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 7236–7243. [CrossRef]
9. Cho, K.; Van Merriënboer, B.; Bahdanau, D.; Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv* **2014**, arXiv:1409.1259.
10. Ramirez-Atencia, C.; Bello-Orgaz, G.; R-Moreno, M.D.; Camacho, D. Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms. *Soft Comput.* **2017**, *21*, 4883–4900. [CrossRef]
11. Zhen, Z.; Xing, D.; Gao, C. Cooperative search-attack mission planning for multi-UAV based on intelligent self-organized algorithm. *Aerosp. Sci. Technol.* **2018**, *76*, 402–411. [CrossRef]

12. Altshuler, Y.; Wagner, I.; Yanovski, V.; Bruckstein, A. Multi-agent Cooperative Cleaning of Expanding Domains. *Int. J. Robot. Res.* **2010**, *30*, 1037–1071. [CrossRef]
13. Altshuler, Y.; Pentland, A.; Bruckstein, A.M. *Swarms and Network Intelligence in Search*; Springer: Berlin/Heidelberg, Germany, 2018.
14. Altshuler, Y.; Bruckstein, A.M. Static and expanding grid coverage with ant robots: Complexity results. *Theor. Comput. Sci.* **2011**, *412*, 4661–4674. [CrossRef]
15. Cho, S.W.; Park, H.J.; Lee, H.; Shim, D.H.; Kim, S.Y. Coverage path planning for multiple unmanned aerial vehicles in maritime search and rescue operations. *Comput. Ind. Eng.* **2021**, *161*, 107612. [CrossRef]
16. Apostolopoulos, P.A.; Torres, M.; Tsiropoulou, E.E. Satisfaction-aware data offloading in surveillance systems. In Proceedings of the 14th Workshop on Challenged Networks, Los Cabos, Mexico, 25 October 2019; pp. 21–26. [CrossRef]
17. Rummery, G.A.; Niranjan, M. *On-Line Q-Learning Using Connectionist Systems*; Citeseer: University Park, PA, USA, 1994; Volume 37.
18. Shamsoshoara, A.; Khaledi, M.; Afghah, F.; Razi, A.; Ashdown, J. Distributed cooperative spectrum sharing in uav networks using multi-agent reinforcement learning. In Proceedings of the 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 11–14 January 2019; pp. 1–6. [CrossRef]
19. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
20. Pinto, L.; Davidson, J.; Sukthankar, R.; Gupta, A. Robust adversarial reinforcement learning. In Proceedings of the International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; pp. 2817–2826.
21. Finn, C.; Yu, T.; Fu, J.; Abbeel, P.; Levine, S. Generalizing skills with semi-supervised reinforcement learning. *arXiv* **2016**, arXiv:1612.00429.
22. Lowe, R.; WU, Y.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems 30*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.;Curran Associates, Inc.: Red Hook, NY, USA, 2017; pp. 6379–6390.
23. Foerster, J.N.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
24. Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the Tenth International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 1993; pp. 330–337.
25. Iqbal, S.; Sha, F. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Chaudhuri, K., Salakhutdinov, R., Eds.; PMLR: Long Beach, CA, USA, 2019; Volume 97, pp. 2961–2970.
26. Jiang, J.; Dun, C.; Huang, T.; Lu, Z. Graph convolutional reinforcement learning. *arXiv* **2018**, arXiv:1810.09202.
27. Liu, C.H.; Ma, X.; Gao, X.; Tang, J. Distributed energy-efficient multi-UAV navigation for long-term communication coverage by deep reinforcement learning. *IEEE Trans. Mob. Comput.* **2019**, *19*, 1274–1285. [CrossRef]
28. Khan, A.; Tolstaya, E.; Ribeiro, A.; Kumar, V. Graph policy gradients for large scale robot control. In Proceedings of the Conference on Robot Learning, Cambridge, MA, USA, 16–18 November 2020; pp. 823–834.
29. Walker, O.; Vanegas, F.; Gonzalez, F.; Koenig, S. A deep reinforcement learning framework for UAV navigation in indoor environments. In Proceedings of the 2019 IEEE Aerospace Conference, Big Sky, MT, USA, 2–9 March 2019; pp. 1–14.
30. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the International Conference on Machine Learning, Lille, France, 7–9 July 2015; pp. 1889–1897.
31. Ye, Z.; Wang, K.; Chen, Y.; Jiang, X.; Song, G. Multi-UAV Navigation for Partially Observable Communication Coverage by Graph Reinforcement Learning. *IEEE Trans. Mob. Comput.* **2022**. [CrossRef]
32. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
33. Bernstein, D.S.; Givan, R.; Immerman, N.; Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.* **2002**, *27*, 819–840. [CrossRef]
34. Sutton, R.S.; McAllester, D.A.; Singh, S.P.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2000; pp. 1057–1063.
35. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [CrossRef] [PubMed]
36. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
37. Konda, V.R.; Tsitsiklis, J.N. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2000; pp. 1008–1014.
38. Jain, R.K.; Chiu, D.M.W.; Hawe, W.R. *A Quantitative Measure of Fairness and Discrimination*; Eastern Research Laboratory, Digital Equipment Corporation: Hudson, MA, USA, 1984.
39. Weaver, L.; Tao, N. The optimal reward baseline for gradient-based reinforcement learning. *arXiv* **2013**, arXiv:1301.2315.
40. Heess, N.; Hunt, J.J.; Lillicrap, T.P.; Silver, D. Memory-based control with recurrent neural networks. *arXiv* **2015**, arXiv:1512.04455.
41. Nair, V.; Hinton, G.E. Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on Machine Learning (ICML 2010), Haifa, Israel, 21–24 June 2010; pp. 807–814.
42. Jang, E.; Gu, S.; Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv* **2016**, arXiv:1611.01144.