entropy

MDPI

Article

Multi-Agent Hierarchical Graph Attention Actor–Critic Reinforcement Learning

Tongyue Li [1], Dianxi Shi [1,*], Songchang Jin [1], Zhen Wang [1,2], Huanhuan Yang [3] and Yang Chen [4]

[1] Academy of Military Sciences, Beijing 100097, China
[2] Tianjin Artificial Intelligence Innovation Center (TAIIC), Tianjin 300450, China
[3] College of Computer, National University of Defense Technology, Changsha 410073, China
[4] School of Computer Science, Peking University, Beijing 100871, China
* Correspondence: dxshi@nudt.edu.cn

Abstract: Multi-agent systems often face challenges such as elevated communication demands, intricate interactions, and difficulties in transferability. To address the issues of complex information interaction and model scalability, we propose an innovative hierarchical graph attention actor–critic reinforcement learning method. This method naturally models the interactions within a multi-agent system as a graph, employing hierarchical graph attention to capture the complex cooperative and competitive relationships among agents, thereby enhancing their adaptability to dynamic environments. Specifically, graph neural networks encode agent observations as single feature-embedding vectors, maintaining a constant dimensionality irrespective of the number of agents, which improves model scalability. Through the "inter-agent" and "inter-group" attention layers, the embedding vector of each agent is updated into an information-condensed and contextualized state representation, which extracts state-dependent relationships between agents and model interactions at both individual and group levels. We conducted experiments across several multi-agent tasks to assess our proposed method's effectiveness, stability, and scalability. Furthermore, to enhance the applicability of our method in large-scale tasks, we tested and validated its performance within a curriculum learning training framework, thereby enhancing its transferability.

Keywords: hierarchical graph attention; multi-agent reinforcement learning; curriculum learning

check for updates

Academic Editor: Adam Lipowski

Received: 14 November 2024
Revised: 12 December 2024
Accepted: 23 December 2024
Published: 25 December 2024

Citation: Li, T.; Shi, D.; Jin, S.; Wang, Z.; Yang, H.; Chen, Y. Multi-Agent Hierarchical Graph Attention Actor–Critic Reinforcement Learning. *Entropy* **2025**, *27*, 4. https://doi.org/10.3390/e27010004

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Multi-agent systems have attracted substantial research attention due to their widespread presence and crucial roles in various domains. In nature, these systems are evident in ecosystems and food chains [1,2]. In industrial applications, they are indispensable in automated manufacturing and robotics [3,4], intelligent transportation [5], coordinated patrol [6], formation control [7], and cooperative navigation [8]. Interactions within multi-agent systems are broadly categorized into cooperative and competitive relationships. Cooperative interactions involve agents working together to achieve shared goals, while competitive interactions imply that agents compete for resources to fulfil individual objectives [9]. Additionally, mixed cooperative–competitive relationships are common in many fields, exemplified by predation among animal groups and symbiotic relationships among fish [10,11]. However, as multi-agent systems are increasingly applied to more complex tasks with a growing number of agents, interaction information between agents is becoming more and more frequent. Agents need to interact with important targets

in a dynamic environment to acquire crucial information, which enables them to make informed decisions.

Multi-agent reinforcement Learning (MARL) is a crucial approach for addressing collaboration or competition problems in multi-agent systems [12,13]. Previous MARL research has predominantly focused on fully cooperative strategies, aiming to maximize the collective reward of agents, which is suitable for fully cooperative tasks. However, in both natural and industrial applications, mixed cooperative–competitive relationships are common and widespread, such as in multi-UAV pursuit and missile interception tasks [14–18]. In mixed cooperative–competitive tasks, cooperation and competition are often interwoven. The teammates collaborate to achieve common goals while simultaneously competing against opponents to pursue individual or subsystem interests [19]. Consequently, most fully cooperative MARL methods are inadequate for addressing these complex scenarios. Additionally, as multi-agent systems are increasingly applied to more complex tasks with a growing number of agents, large-scale modelling inevitably leads to the curse of dimensionality, greatly complicating training processes. A more fundamental challenge is the limited transferability of these models. Previous MARL methods have rarely considered strategy transferability, limiting their application to tasks with variable agent numbers and hindering their effectiveness in complex multi-agent tasks.

Multi-agent interactions can be naturally modeled as a graph, where nodes represent agents and edges represent their interactions [20]. Agents can share crucial information through connections to enable mutual learning and strategy optimization. However, as the frequency of agent interactions rises, information complexity escalates. Full graph connections and global information exchange are costly and can lead to information redundancy.

In practice, most real-world environments exhibit partial observability and limited communication (due to limited range or noisy sensors) [21]. Agents usually communicate only with neighbouring agents to reduce communication costs, meaning they must learn to collaborate based on local observations and limited communication. In multi-agent systems, where nearby agents have a more significant influence on each other's behaviour, agents should wisely choose whom to communicate with, bake key information into the graph architecture, and transmit valuable information to appropriate teammates to facilitate collaboration and learn more efficient strategies [22].

Motivated by the aforementioned discussions, we propose a multi-agent hierarchical graph attention actor–critic reinforcement learning method (MAHGAC). We model the multi-agent interactions as a graph and use a hierarchical graph attention mechanism (HGAT) to encode the local observations of each agent as a single node embedding vector. This vector encapsulates an information-condensed and contextualized state representation, aggregating the state dependencies among agents and capturing both individual and hierarchical relationships. Therefore, agents can learn to assign importance weights to neighbouring agents, dynamically select optimal teammates for cooperation or opponents for focus, learn more advanced strategies, and improve the performance of multi-agent systems in complex interactions. In summary, our work makes the following contributions:

- Multi-agent interactions are effectively modelled as graphs, where agents are represented as nodes, and their connections form edges through which information is exchanged. Graph attention networks encode each agent's local observations into a single node embedding vector. The dimensionality of this embedding vector remains constant regardless of the number of agents and generates a fixed-size environment representation, offering flexibility and scalability.
- We propose a hierarchical graph attention mechanism to optimize the efficient information extraction of agents in complex environments. The HGAT transforms the agents' observation information into an information-condensed and contextualized

state representation to capture relationships at both individual and hierarchical levels using the "inter-agent" and "inter-group" attention layers. By aggregating individual and hierarchical relationships, agents can better "understand" the dynamic environment changes, focus on interacting with the most relevant agents, and thus learn more "advanced" strategies.

- To validate the transferability of our method, we trained it within a curriculum learning framework. With curriculum learning, agents can gradually adapt to new tasks with varying numbers of agents, enabling the trained strategies to be effectively transferred to new tasks, thereby enhancing their transferability. Using curriculum learning, we successfully transferred a five-agent line formation strategy to a new task with ten agents.

The rest of this paper is structured as follows: Section 2 reviews existing works. Section 3 presents the problem formulation and relevant preliminary knowledge. Section 4 details the methodology, including mathematical formulations and the algorithm's training procedure. Section 5 outlines the experimental environment setup, baseline methods, and evaluation metrics and provides an analysis of the experimental results. Section 6 presents the transferability within the curriculum learning framework. Finally, Section 7 summarizes the conclusions.

2. Related Works

In multi-agent systems, agents must observe environmental information and exchange critical knowledge with one another to coordinate actions and achieve collective goals. Given the enormous volume of information, agents must determine which peers to prioritize in information sharing to foster cooperation and enhance overall system performance. Information acquisition and transfer among agents are crucial for effective strategy learning.

Recent research has increasingly employed attention mechanisms in multi-agent systems. Ma et al. [23] propose a novel interactive advantage actor–critic collaborative MARL method (IAC). The method utilizes a shared attention mechanism to evaluate the functions of each agent in value functions, considering the influence of teammates. Iqbal et al. [24] propose an actor–attention–critic (MAAC) MARL method, which uses an attention mechanism to select the relevant information for each agent at each step. The mechanism functions similarly to a differentiable key-value memory model, where each agent queries the currently relevant observational and action information from other agents and integrates this information into the value function estimation, computing the agent's gradient estimate.

Modelling multi-agent systems such as graphs with graph attention mechanisms has become a key method for solving information interactions between agents. Su et al. [25] presents an agent communication architecture that employs graph convolution to represent agent communication, where the flexibility of the graph structure enables the method to be applied to a variety of multi-agent systems. Liu et al. [26] propose a two-stage graph attention network (G2ANet). It establishes relationships between agents through a complete graph, utilizing soft and hard attention mechanisms to learn whether interactions exist between two agents and learn the importance of these interactions. Jiang et al. [27] propose a graph convolutional reinforcement learning (DGN) method, which adapts to the dynamics of the underlying graph of the multi-agent environment and learns cooperation using latent features through the convolutional layers with a gradually increased receptive field, and cooperation is further enhanced through temporal relationship regularization. Sun et al. [28] propose a multi-attention interaction modelling method (IMMA), utilizing multiple latent graphs to represent the interactions and attention of various independent types, thereby considering different strength relationships among agents. Niu et al. [22] propose a multi-agent graph attention communication (MAGIC), which uses the graph

attention communication protocol to allow agents to learn when to communicate and to whom they send information to.

In multi-agent systems, agents must prioritize peers for information sharing to foster cooperation and enhance overall system performance. Attention mechanisms are an effective method of optimizing information interactions between agents. However, as task difficulty and complexity increase, the volume and complexity of information exchanged between agents also grow. Agents must extract critical information from vast, dynamically changing environments and determine state dependencies between agents to facilitate more effective collaboration.

3. Preliminaries

3.1. Partially Observable Markov Game (POMG)

We abstracted multi-agent problem as a partially observable Markov game (POMG) based on the observational capabilities and dynamic attributes of N agents: $s \in S$ denotes the global state of the game. $a_i \in A_i$ is an action for agent i. State transition function defines the probability distribution of the agent transitioning to the next state based on the current actions of each agent: $S \times A_1 \times \cdots \times A_N \longrightarrow P(S)$. The reward for agent i is computed as a function of state s and joint action a as: $R_i : S \times A_1 \times \cdots \times A_N \longrightarrow R$. $o_i \in O_i$ denotes a local observation of agent i, which contains partial information from the global state $s \in S$. As shown in Figure 1, in the multi-agent pursuit environment, the observable space of the agent i includes the location information and velocity information of entities within the observation range $V(i)$:$o_i = \{s_j | j \in V(i)\}$. For pursuer p_5, the observation information is: the position $p_4 : (p^p_{x4}, p^p_{y4})$, the velocity $p_4 : (v^p_{x4}, v^p_{y4})$, the position of prey $e : (p^e_x, p^e_y)$, the velocity of prey $e : (v^e_x, v^e_y)$, and the position of obstacle $o_1 : (p^o_{x1}, p^o_{y1})$. The action space A is the speed of the agent at the next time (v_x, v_y). Each agent learns a strategy π_i:$O_i \rightarrow P(A_i)$. The agent i aims to maximize its discounted return: $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where $\gamma \in [0, 1]$ is a discount factor.

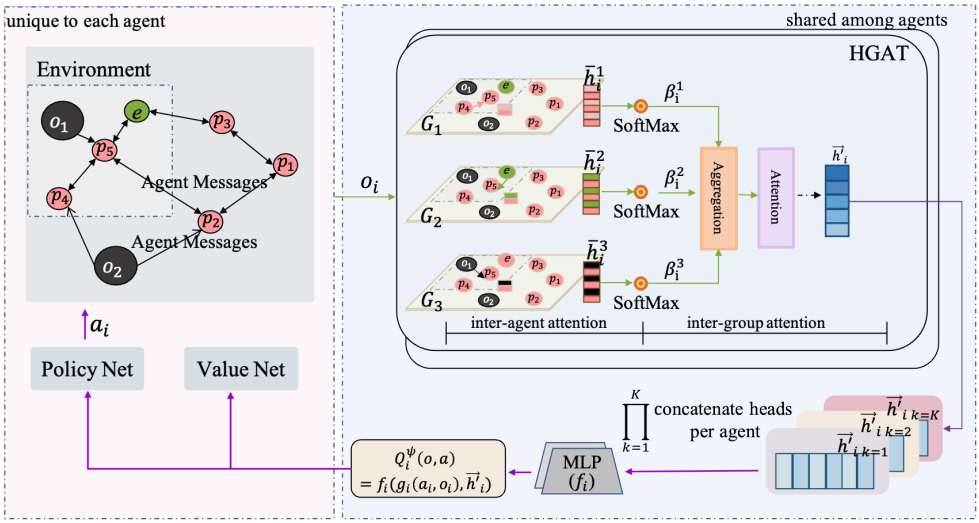


Figure 1. The overall structure of the MAHGAC. Left: an interactive multi-agent pursuit environment. Right: a shared HGAT module. The MAHGAC adopts the centralized training and decentralized execution (CTDE) training paradigm. During the training, adopting a centralized critic and sharing a hierarchical graph attention mechanism, agent i can obtain information from all agents and learn the importance weights of other agents in its vicinity. During the testing, each agent executes actions based on its own observations.

3.2. Graph Attention Network (GAT)

The graph attention network (GAT) is an effective model to process structured data that are represented as a graph. GAT proposes a way to calculate the target node embedding vector $\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_j} \alpha_{ij} W_{\vec{h}_j}\right)$ of the graph by aggregating the node embedding vectors $\vec{h}_j$ from the neighbour nodes $\{j \in \mathcal{N}_i\}$ that are connected with the target node i. The GAT uses self-attention to aggregate information from neighbouring nodes to adaptively match weights for different neighbours.

Figure 2 illustrates the information aggregation steps of GAT. For target node i, calculate the similarity coefficients e_{ij} between its neighbours $j \in \mathcal{N}_i$ and itself.

$$e_{ij} = a\left(\left[W\vec{h}_i \| W\vec{h}_j\right]\right), j \in N_i \tag{1}$$

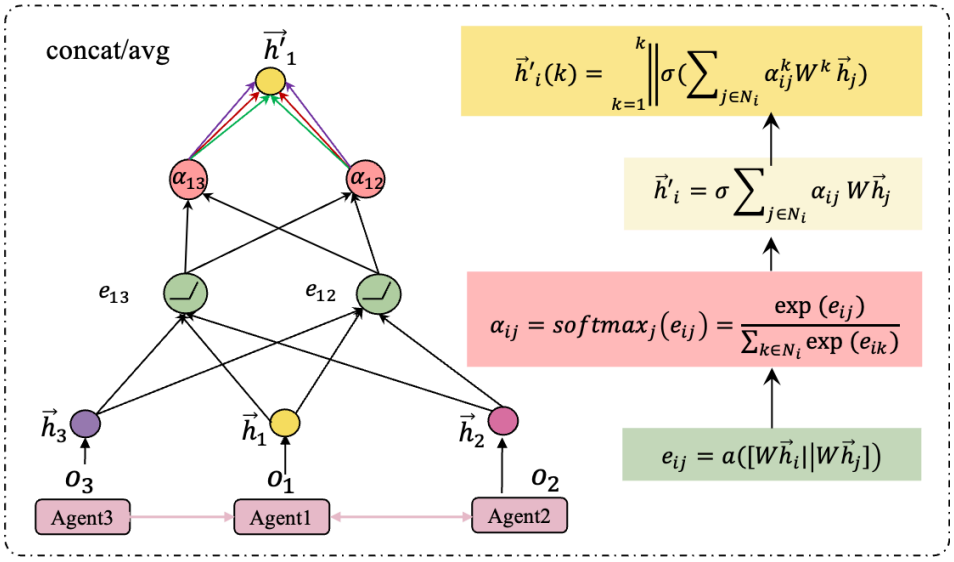


Figure 2. The information aggregation steps in GAT between connected agents. For agent 1, connected agent 2, and agent 3, calculate the attention weights of node 2: $\alpha_{12} = \frac{\exp(\text{LeakyReLU}(e_{12}))}{\exp(\text{LeakyReLU}(e_{12}))+\exp(\text{LeakyReLU}(e_{13}))}$ and node 3: $\alpha_{13} = \frac{\exp(\text{LeakyReLU}(e_{13}))}{\exp(\text{LeakyReLU}(e_{12}))+\exp(\text{LeakyReLU}(e_{13}))}$ towards node 1, respectively, to obtain the node embedding vector of agent 1 for a more robust state representation of the agent's feature information.

A linear mapping with a shared parameter matrix W increases the dimension of vertex features, $[.||.]$ concatenates the features after the transformation of node i, j, and $a(.)$ maps concatenated high-dimensional features into a single real number using a single-layer feed-forward neural network. Then, the calculated attention coefficient α_{ij} is:

$$\alpha_{ij} = \frac{\exp(LeakyReLU(e_{ij}))}{\sum_{z\in N_i}\exp(LeakyReLU(e_{iz}))} \tag{2}$$

where $z \in N_i$ represents the other nodes that are connected to the current node i, except for node j. According to the attention weights, the new feature $\vec{h}'_i$ of the target node i, after fusing the neighbourhood information, can be calculated by the feature weighted sum. In our study, we employ multi-head attention to measure the relevance interrelation among agents from different dimensions [29], enhancing the performance of GAT and the stability of the updated features, where k is the number of attention heads.

$$\vec{h}'_i(k) = \Big\|_{k=1}^{k} \sigma\left(\sum_{j\in N_i} \alpha_{ij}^k W^k \vec{h}_j\right) \tag{3}$$

4. Methods

We propose a novel multi-agent hierarchical graph attention actor–critic reinforcement learning method (MAHGAC), as illustrated in Figure 1. MAHGAC employs the multi-agent actor–critic reinforcement learning network, in which agents interact with the environment to learn strategies through trial and error. The observation o_i is encoded as a node embedding vector $\vec{h'}_i$ through a hierarchical graph attention network, serving as an information-aggregated and contextualized state representation that adaptively captures the state dependencies among agents. This feature-embedding information is shared among all agents, allowing each agent i to receive contributions from other agents at each time step. After processing through a two-layer MPL, the $\vec{h'}_i$ updates the action-value function $Q_i^{\psi}(o,a) = f_i((o_i,a_i),\vec{h'}_i)$, guiding better collaboration among agents and promoting the learning of "advanced" strategies for complex interactive multi-agent tasks.

4.1. Agents Communication

We model the multi-agent interaction as the graph $G = (V,E)$, as shown in Figure 3. The entities (agents and landmarks) in the environment are abstracted as nodes $n \in V$ on the graph, with edges $e \in E$ between nodes that allow for communication. Agents exchange information and learn collaborative interactions along the edges of the graph. The observation of the agents is encoded as a node embedded vector, and the GAT (Section 3.2) weights the nodes connected to each agent to aggregate information effectively. The GAT encodes each agent's local observations into a single node embedding vector, maintaining a constant dimensionality regardless of the number of agents. Thus, it generates a fixed-size environment representation. This stability allows for transferring learned policies to diverse scenarios, enhancing the model's scalability.

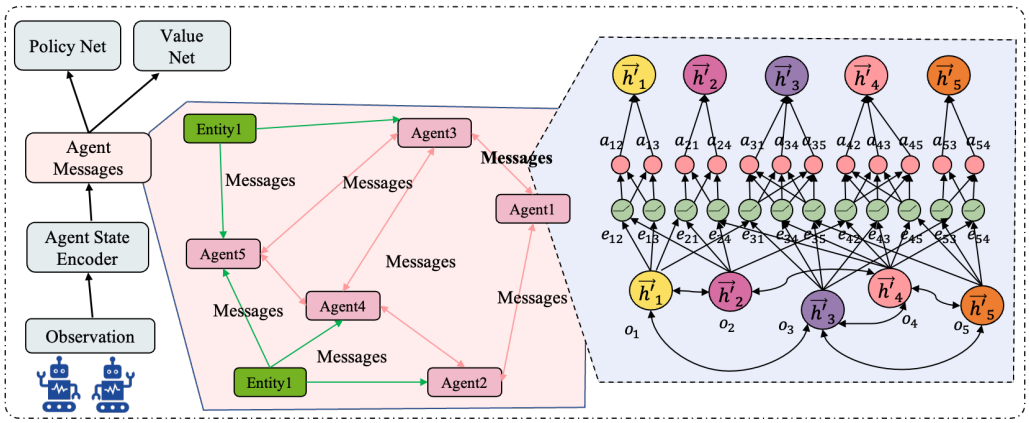


Figure 3. The information exchange process among agents. The connectivity diagram between agents, where nodes can exchange information through edges.

In practice, agents who are close to each other have a greater influence on each other's behaviour. We baked this critical information into the graph architecture, significantly enhancing the learning process. This approach provides a strong inductive bias for tasks with varying numbers of agents, enabling the model to perform exceptionally well in diverse and complex environments.

4.2. Hierarchical Graph Attention Network (HGAT)

As agents increase, information interactions in multi-agent systems grow increasingly complex. We propose the hierarchical graph attention network (HGAT), which enables agents to identify and prioritize critical interactive targets using "inter-agent" and "inter-group" attention layers. Thus, agents promote a deeper understanding of environmental information and make more effective decisions.

Step 1. Entities Clustering

We use prior knowledge or data to classify all of the entities in the environment (agents, landmarks, etc.) into different groups C^g. If it is a completely cooperative task, such as formation control, we classify all of the agents into one group. If it is cooperative navigation, we can cluster all of the agents into one group and the landmarks into another. If it is a mixed environment, such as a pursuit task, we can divide the pursuers into one group, the prey into another, and the obstacles into a separate group, as shown in Figure 4.

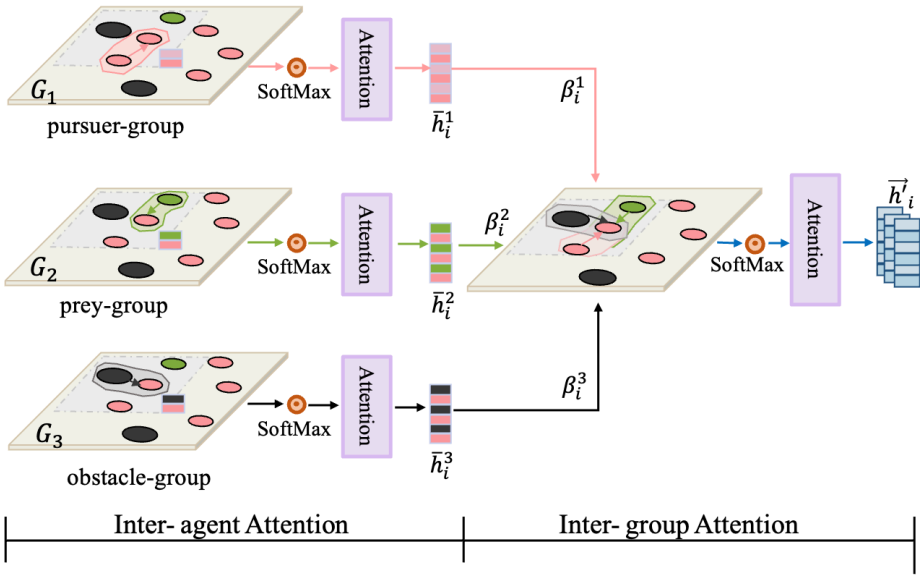


Figure 4. Hierarchical graph attention network. An example of the multi-agent pursuit task, where the entities in the environment are classified into 3 groups: pursuer-group, prey-group, and obstacle-group. In the "inter-agent" graph attention layer, attention weights are calculated between agents within each group, and then the aggregated feature vectors $\vec{h}_i^g$ are used as inputs for the "inter-group" graph attention layer to obtain information-aggregated and contextualized state representation $\vec{h'}_i$.

Step 2. "Inter-agent" Attention

The HGAT calculates the "inter-agent" attention and "inter-group" attention. Initially, through the node embedding vectors:

$$\vec{h}_{ij}^g = f_{ij}^g(s_i, s_j; W_{ij}^g) \tag{4}$$

where $i, j \in C^g \cap V(i)$ of neighbour nodes $j \in V(i)$ adjacent to agent i in each group.

The embedding vector of the target node aggregation embedding $\vec{h}_i^g$ is calculated:

$$\vec{h}_i^g = \sum_{j \in C^g \cap V(i)} \alpha_{ij}^g \vec{h}_{ij}^g \tag{5}$$

The "inter-agent" attention weight α_{ij}^g quantifies the importance of the embedding $\vec{h}_{ij}^g$ from agent j to agent i, computed based on the GAT:

$$\alpha_{i\cdot}^g \propto \exp\left(e_{i\cdot}^g\right) \tag{6}$$

where $e_{ij}^g = f_\alpha^g(s_i, s_j; W_\alpha^g)$. We employed multiple attention heads [29]; thus, the aggregated embedding of agent i is $\vec{h}_i^g$, where k is the number of attention heads.

$$\vec{h}_i^g = \prod_{k=1}^{K} \sum_{g=1}^{G} \sum_{j \in C^g \cap V(i)} \alpha_{ij}^g \vec{h}_{ij}^g \tag{7}$$

Step 3. "Inter-group" Attention

Then, HGAT computes the "inter-group" relationships, aggregates the group-level node embedding vector $\vec{h}_i^1, \ldots, \vec{h}_i^g$, and updates the feature-embedding vector $\vec{h'}_i$ of agent i that incorporates contextual information.

$$\vec{h'}_i = \sum_{g=1}^{G} \beta_i^g \vec{h}_i^g \tag{8}$$

where the "inter-group" attention weight β_i measures the contribution of agent i across different subgroups, which guides which group agent i should focus on more to achieve its goal.

$$\beta_i = (\beta_i^1, \ldots, \beta_i^G) \propto \exp(q_i) \tag{9}$$

$$q_i = \left[q_i^1, ..., q_i^G\right] = f_\beta([\vec{h}_i^1, ..., \vec{h}_i^G]; W_\beta) \tag{10}$$

Agent i obtains an information-condensed and contextualized state representation $\vec{h}_i \longrightarrow \vec{h'}_i$ using HGAT, which is particularly advantageous when addressing complex mixed cooperative–competitive multi-agent tasks. On the one hand, HGAT encodes the agent's observations o_i into node embedding vectors, where the dimensionality remains constant regardless of the number of agents, generating a fixed-size environment representation, thereby demonstrating scalability to larger tasks. On the other hand, by aggregating individual and group-level relationships, agents can discern the "role" they should play based on the relevance and importance of their contributions at each time step, enabling them to accomplish the final task more effectively.

4.3. Multi-Agent Actor-Critic

The node embedded feature vectors $\vec{h'}_i$ are passed through a two-layer MLP f_i and then input into the value and policy networks $Q_i^\psi(o, a) = f_i(g_i(a_i, o_i), \{\vec{h'}_j\}_{j \neq i})$. These networks predict the estimated state value and the probability distribution of all possible actions, respectively. Each agent selects an action from this distribution, executes the chosen action, and receives a reward from the environment based on these actions. To promote exploration and reduce the risk of converging to suboptimal deterministic policies, we adopt the modern and widely recognized maximum entropy reinforcement learning approach to learn a soft value function [30]. This approach incorporates an entropy term into the policy gradient:

$$\nabla_{\theta_i} J(\pi_\theta) = E_{s \sim D, a \sim \pi}[\nabla_{\theta_i} \log(\pi_{\theta_i}(a_i|o_i))(-\alpha \log\left(\pi_{\theta_i}(a_i|o_i)\right) + Q_i^\psi(o, a))] \tag{11}$$

Update all critics by minimizing a joint regression loss function through parameter sharing:

$$\mathcal{L}_Q(\psi) = \sum_{i=1}^{N} E_{(o,a,r,o') \sim D}[(Q_i^\psi(o, a) - y_i)^2] \tag{12}$$

$$y_i = r_i + \gamma E_{a' \sim \pi_{\bar{\theta}}(o')}[Q_i^{\bar{\psi}}(o', a') - \alpha \log(\pi_{\bar{\theta}_i}(a_i'|o_i'))] \tag{13}$$

where $\bar{\psi}$ and $\bar{\theta}$ are the parameters of the target critics and target policies, respectively. Q_i^ψ is the action-value estimate for agent i, which receives observations and actions for agents. α is the temperature parameter determining the balance between maximizing entropy and rewards.

4.4. MAHGAC Algorithm

The pseudocode for the MAHGAC method is depicted in Algorithm 1. We train using soft actor–critic, an off-policy actor–critic method for maximum entropy reinforcement learning [31]. During training, at each time point, generate a set of rollout, consisting of a tuple $(o_t, a_t, r_t, o_{t+1})_{1...N}$, which is added to the replay buffer (1×10^6). After an episode (25-time steps), reset the environment and perform 4 updates for the attention critic and all policies. For each update, we sample a mini-batch of 1024 time points from the replay buffer and then perform gradient descent on the $Q-$function loss and policy objective using an ADAM optimizer with a learning rate of 0.001 [32]. The agent encoder takes the 4-dim states as an input and outputs 128-dim embedding representations, and the encoders are a single ReLU fully connected (FC) layer. The communication module uses attention with 128-dim queries, keys, and values. Aggregated messages and the state of an agent are concatenated and updated by a single ReLU FC layer containing 128 neurons. We use $K = 3$ communication hops between the agents.

Algorithm 1 Training Procedure for MAHGAC

Initialize actor networks θ, critic network ψ
Initialize target networks $\bar{\psi}$ and $\bar{\theta}$, replay buffer D
for $episode = 1 \rightarrow M$ **do**
  Reset environments, and get initial o_i^e for each agent i
  for $t = 1 \rightarrow T$ **do**
    Select actions $a_i \sim \pi_i(\cdot|o_i)$ for each agent i
    Send actions to environment and get o_i', r_i for all agents
    Store transitions for all environments in D
    for $g = 1$ to G for agent group C^g **do**
      for agent $i = 1$ to $|C^g|$ in group C^g **do**
        Sample a minibatch $(o_{1...N}, a_{1...N}, r_{1...N}, o'_{1...N})$
        Calculate $Q_i^{\psi}(o_{1...N}, a_{1...N}), a_i' \sim \pi_i^{\bar{\theta}}(\bar{o}_i), Q_i^{\bar{\psi}}(o_{1...N}, a_{1...N})$
        Set $y_i = r_i + \gamma E_{a' \sim \pi_{\bar{\theta}}(o')}[Q_i^{\bar{\psi}}(o', a') - \alpha \log(\pi_{\bar{\theta}_i}(a_i'|o_i'))]$
        Update critic using $\mathcal{L}_Q(\psi) = \sum_{i=1}^{N} E_{(o,a,r,o') \sim D}[(Q_i^{\psi}(o,a) - y_i)^2]$
        Sample $(o_{1...N} \sim D)$
        Calculate $a_{1...N} \sim \pi_i^{\bar{\theta}}(o_i'), Q_i^{\psi}(o_{1...N}, a_{1...N})$
        Update policies using:
        $\nabla_{\theta_i} J(\pi_\theta) = E_{s \sim D, a \sim \pi}[\nabla_{\theta_i} \log(\pi_{\theta_i}(a_i|o_i))(-\alpha \log(\pi_{\theta_i}(a_i|o_i)) + Q_i^{\psi}(o,a))]$
      end for
    end for
    Update target network parameters for each agent group C^g
    $\bar{\psi}_g = \tau \bar{\psi}_g + (1-\tau)\psi_g$
    $\bar{\theta}_g = \tau \bar{\theta}_g + (1-\tau)\theta_g$
  end for
end for

5. Experiments

5.1. Experimental Settings

We evaluate the effectiveness of the MAHGAC method in the multi-agent cooperative navigation task, multi-agent formation control task, and multi-agent confronting pursuit task. We exploit a multi-agent particle environment (MPE https://github.com/openai/maddpg, accessed on 11 September 2024) where agents can move in 2×2 sq. units 2D space. The action space for each agent is discretized, allowing agents to control unit acceleration or deceleration in the X and Y directions.

- **Cooperative navigation**: Figure 5a shows that the environment consists of M agents and M landmarks. The objective for each agent is to reach a distinct landmark while

avoiding collisions with other agents. Each episode begins with M agents and M landmarks randomly initialised in the environment and ends after 25 time steps. During each episode, each agent receives a reward of $-d$ based on its distance to the nearest landmark and incurs a penalty of -1 if it collides with another agent. Landmarks are not preassigned to agents, and agents dynamically determine which landmarks to target based on environmental feedback. Ultimately, each agent occupies a unique landmark, completing the navigation task and learning collaborative strategies.

- **Linear formation**: As shown in Figure 5b, there are M agents and two landmarks. The agents aim to position themselves equally spaced along a line between the two landmarks. Each episode begins with the agents and landmarks randomly initialized and ends after 25 time steps. Each agent receives a reward of $-d$ based on the distance between its current position and the expected position along the line.
- **Regular polygonal formation**: As shown in Figure 5c, there are M agents and one landmark. The agents are required to position themselves into an M-sided regular polygonal formation with the landmark at its centre. Each episode starts with the agents, and the landmark randomly initializes and ends after 25 time steps. During the episode, each agent receives a reward of $-d$ based on the distance between its current position relative to the landmark and the expected position in the polygonal formation.
- **Confronting pursuit**: Figure 5d shows that the environment consists of M pursuers and N prey. The competitive game objective is that the M homogeneous pursuers pursue the N prey while the prey strives to escape. As pursuers have lower speed and acceleration compared with prey, they must cooperate effectively to succeed in their pursuit. Each pursuer obtains a positive reward of +10 when it catches the prey, while the prey incurs a negative reward of -10. To prevent prey from straying too far from a designated zone, they receive a negative reward if they leave this area. The environment also contains obstacles, and any agent colliding with an obstacle is penalized with a negative reward of -10.

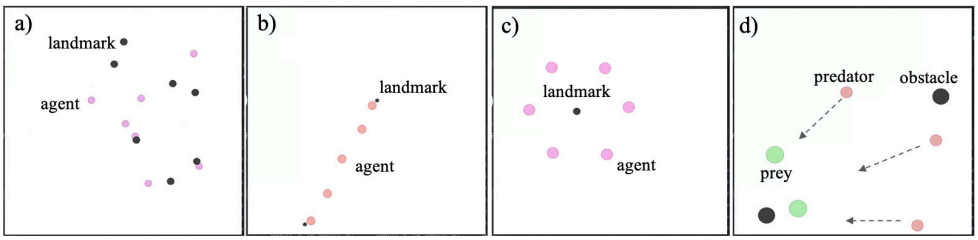


Figure 5. Experimental environments: (**a**) Cooperative navigation, where agents reach different landmarks while avoiding obstacles. (**b**) Linear formation, where agents form a line between two landmarks. (**c**) Regular polygon formation, where agents encircle landmarks to form a regular polygon. (**d**) Confronting pursuit, where pursuers collaborate to chase two prey, and when both prey are caught, the task is successful.

In our work, we conducted a series of comparative experiments to evaluate the performance of different MARL methods across various tasks. We selected four methods for comparison: MADDPG [33] without attention, G2ANet [26] with attention, DGN [27] with single-layer graph attention, and MAAC [24] with actor–attention–critic. All methods were tested under identical experimental and training settings. Each experiment was conducted five times, and the results were reported using average values and standard deviations to describe the range of outcomes. To comprehensively evaluate their performance, we employed two main evaluation metrics:

- Success rate (S%): percentage of tasks completed during evaluation episodes (higher is better).

- Mean episode length (MEL): average length of successful episodes during evaluation (lower is better).

5.2. Results

5.2.1. Effectiveness

Figure 6 illustrates the mean episode reward curves of each method across four tasks. Each curve represents the average results of multiple experiments, with the shaded area indicating the standard deviation. In both fully cooperative and mixed cooperative–competitive tasks, the reward curves of MAHGAC converge to higher levels, demonstrating superior performance compared with other methods. Additionally, MAHGAC outperforms other methods employing single-layer graph attention in mixed cooperative–competitive tasks. This advantage is attributed to the increased complexity of agent relationships, which demand more interactions and sophisticated information selection. MAHGAC adaptively extracts state-dependent relationships among multiple agents, enhancing information selection and strategy learning.

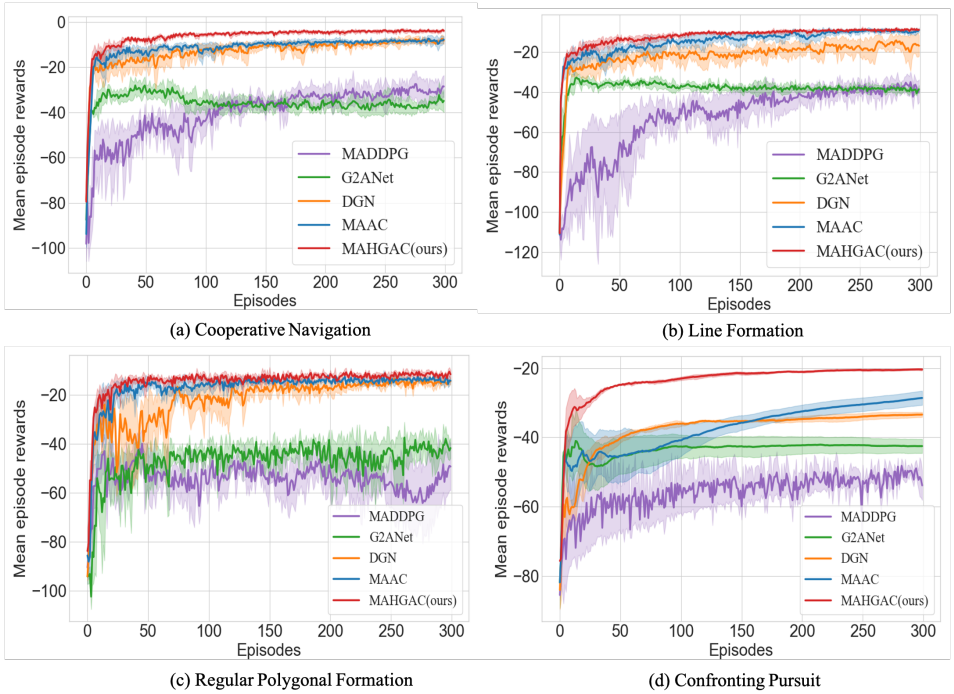


Figure 6. Subfigure (**a**) represents the mean episode rewards curves of 3 agents in the cooperative navigation task. (**b**) represents the mean episode rewards curves of 5 agents in the linear formation task. (**c**) represents the mean episode rewards curves of 4 agents in the regular polygonal formation task. (**d**) represents the mean episode rewards curves for the task of 3 pursuers cooperating to pursue 2 prey.

Table 1 presents the success rate (S%) and mean episode length (MEL) for each method in both fully cooperative and mixed cooperative–competitive tasks. Compared with MADDPG without attention, MAHGAC significantly outperforms in success rate across all tasks, while MEL remains consistent. Compared with G2ANet with attention, MAHGAC achieves higher success rates across all tasks and maintains lower MEL in cooperative navigation, linear formation, and pursuit tasks.

In fully cooperative tasks, MAHGAC demonstrates an average success rate improvement of 8.054% over DGN with single-layer graph attention, with the MEL being reduced by an average of 0.4. Compared with MAAC with actor–attention–critic, MAHGAC demon-

strates an average success rate improvement of 0.98% and a reduction in MEL by an average of 0.316.

In mixed cooperative–competitive tasks, MAHGAC improves average success rates by 19.942% and decreases MEL by 0.58 on average. Compared with MAAC, MAHGAC achieves an average success rate increase of 7.961% and an MEL reduction of 1.09 on average. MAHGAC demonstrates significant advantages in complex, fully cooperative tasks and mixed cooperative–competitive tasks, particularly those with abundant information.

Table 1. Success rate (S%) and mean episode length (MEL) of different methods in different tasks.

Method	Cooperative Navigation (N = 3)		Linear Formation (N = 5)		Regular Polygonal (N = 4)		Pursuit (N = 4)	
	S (%)	MEL	S (%)	MEL	S (%)	MEL	S (%)	MEL
MADDPG	47.389	3.86	55.938	7.12	65.994	5.54	53.935	5.84
G2ANet	54.983	5.04	61.214	8.22	70.186	5.37	58.420	6.77
DGN	79.083	4.10	79.617	7.70	87.710	5.53	66.708	6.30
MAAC	85.025	4.39	90.720	7.23	91.883	5.45	78.689	6.87
MAHGAC	86.378	3.72	91.330	7.11	92.865	5.29	86.650	5.78

Notably, MAHGAC surpasses other graph attention methods in handling complex agent interactions. This improvement is attributed to the HGAT mechanism, which models state dependencies at both individual agent and subgroup levels. The "inter-agent" graph attention layer effectively captures interactions within each subgroup, enabling agents to adjust their role positioning. The "inter-group" graph attention layer helps agents learn different subgroups' relationships, enhancing their ability to adapt to dynamic environments involving opponents, teammates, or obstacles. The HGAT mechanism provides agents with a clearer understanding of their roles ("Who am I?") and actions ("What should I do at each time point?").

5.2.2. Scalability

As shown in Figure 7, we compare the average episode rewards for the cooperative navigation task with different numbers of agents. As the number of agents and task complexity increase, interactions between agents become more complex, resulting in a sharp decline in the MADDPG rewards, indicating its limited scalability. Similarly, G2ANet, DGN, and MAAC show a decline in reward performance as agent numbers increase, albeit to different extents. In contrast, when the number of agents increases from 3 to 15, MAHGAC remains stable performance, demonstrating its scalability. Furthermore, the boxplot shapes reveal that MADDPG, G2ANet, and DGN exhibit relatively scattered convergence values across multiple tests. At the same time, MAHGAC and MAAC display more consistent convergence values, further indicating the generalization capability.

Table 2 shows the success rate for the cooperative navigation task with different numbers of agents. When the number of agents is three, all methods complete the task. However, as the number of agents increases to seven, both MADDPG and G2ANet struggle, whereas the success rate of MAHGAC improves by 16.07% compared with DGN with single-layer graph attention. As the number of agents increases to 11 and 15, MADDPG and G2ANet fail to complete the task, while DGN, MAAC, and MAHGAC can still be successful.

Moreover, compared with DGN with single-layer graph attention, the success rate of MAHGAC increases by 33.66% and 38.075% at N = 11 and N = 15, respectively. When compared with MAAC with actor–attention–critic, the success rate of MAHGAC increases by 4.343% and 6.074% at N = 11 and N = 15, respectively. As the number of agents increases, MAHGAC demonstrates exceptional stability, with a success rate standard deviation of 0.093, compared with 12.618 for DGN and 1.922 for MAAC. The trends in the boxplot in Figure 7 further affirm the superior stability of the MAHGAC. MAHGAC

maintains its performance with increasing agent numbers, showcasing robust scalability and remarkable stability.

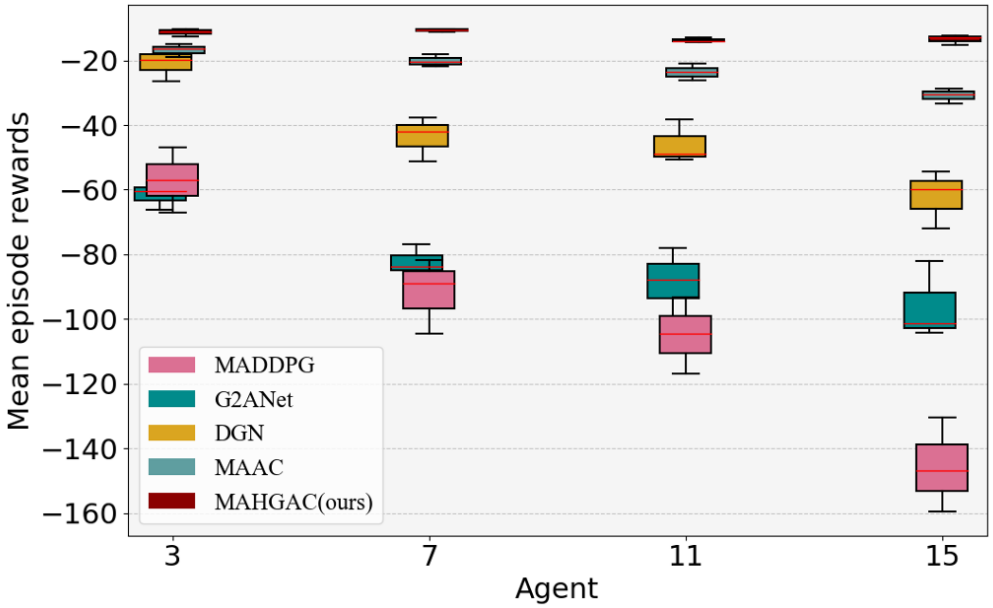


Figure 7. The average episode rewards of different methods in cooperative navigation task with different numbers of agents.

Table 2. Success rates (S%) for the cooperative navigation task.

Method	Cooperative Navigation (S%)			
	N = 3	N = 7	N = 11	N = 15
MADDPG	47.389	13.850	-	-
G2ANet	54.987	28.076	-	-
DGN	79.083	70.200	52.775	48.120
MAAC	85.025	83.883	82.092	80.121
MAHGAC	86.378	86.272	86.435	86.195

6. Curriculum Learning

As the number and dimensions of agents increase, training the model from scratch becomes increasingly time-consuming, leading to slower convergence rates. To address this challenge, we adopted the curriculum learning approach to facilitate model transferability. Curriculum learning is a training strategy that emulates the human learning process, advocating that models start learning from simple samples and progressively transition to more complex samples and tasks [34].

In this work, we conducted curriculum learning experiments by training our strategy within this framework and using a line formation task for verification. Notably, the purpose of this experiment was solely to verify that the MAHGAC method could be effectively transferred using curriculum learning.

In the MAHGAC, network parameters are shared between agents, facilitating the direct application of the policy π. Initially trained for a task Q with N agents, this policy can subsequently be fine-tuned to perform different tasks Q' with M agents. We trained on a line formation task with five agents. Upon reaching an 85% success rate, and we transferred the learned strategy to a 10-agent team.

Figure 8 illustrates transferring the 5-agent line formation strategy to a new 10-agent team using curriculum learning. As shown in Table 3, the 10-agent team successfully learned the cooperative strategy, achieving a 90.51% success rate and 21.82 MEL with

curriculum learning, demonstrating a 2.89% improvement in success rate and a 3.33 reduction in MEL. This comparison underscores the effectiveness of curriculum learning in accelerating and enhancing the completion of line formation tasks.

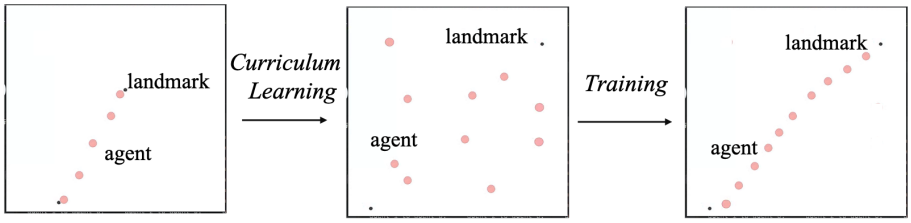


Figure 8. The 5-agent line formation strategy is transferred to a new 10-agent line formation task through curriculum learning.

Table 3. Curriculum learning for line formation.

Normal Training				Curriculum Learning Training			
N = 0		N = 10		N = 5		N = 10	
S (%)	**MEL**	**S (%)**	**MEL**	**S (%)**	**MEL**	**S (%)**	**MEL**
0	0	87.62	21.82	91.33	7.11	90.513	18.49

7. Conclusions

We propose an innovative multi-agent hierarchical graph attention actor–critic reinforcement learning method (MAHGAC) that leverages graph attention networks to encode agent's observations into a single node embedding vector, maintaining a constant dimensionality regardless of the number of agents, generating a fixed-size environment representation, offering flexibility and scalability. We introduce a hierarchical graph attention mechanism (HGAT) to further capture complex interactions. HGAT employs "inter-agent" and "inter-group" graph attention layers to update agents' observations into information-condensed and contextualized state representations. Compared with other attention communication, HGAT utilizes the topological structure of the graph, allowing for more flexible handling of high-dimensional agent information, effectively models state dependencies at individual and group levels, enabling agents to focus on interactions with the most relevant objects and learn more sophisticated strategies.

We conducted experiments to evaluate MAHGAC's effectiveness and scalability. Compared with baseline methods, MAHGAC demonstrates stability performance and superior scalability in both fully cooperative and mixed cooperative–competitive scenarios. Furthermore, we evaluated MAHGAC's transferability within a curriculum learning framework in the linear formation task. Experimental results show that MAHGAC not only sustains performance as the task complex but also exhibits superior stability, scalability, and transferability, offering new possibilities for addressing larger-scale tasks in practice.

Author Contributions: Methodology, T.L.; Software, Z.W.; Formal analysis, T.L.; Investigation, Y.C.; Resources, S.J.; Writing—original draft, T.L. and H.Y.; Funding acquisition, D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 91948303).

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The Author Zhen Wang was employed by the Tianjin Artificial Intelligence Innovation Center (TAIIC). The remaining authors declare that the research was conducted in

the absence of any commercial or financial relationships that could be seen as a potential conflict of interest.

References

1. Franks, N.R.; Worley, A.; Grant, K.A.; Gorman, A.R.; Vizard, V.; Plackett, H.; Doran, C.; Gamble, M.L.; Stumpe, M.C.; Sendova-Franks, A.B. Social behaviour and collective motion in plant-animal worms. *Proc. R. Soc. B Biol. Sci.* **2016**, *283*, 20152946. [CrossRef] [PubMed]
2. Perolat, J.; Leibo, J.Z.; Zambaldi, V.; Beattie, C.; Tuyls, K.; Graepel, T. A multi-agent reinforcement learning model of common-pool resource appropriation. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30.
3. Oliff, H.; Liu, Y.; Kumar, M.; Williams, M.; Ryan, M. Reinforcement learning for facilitating human-robot-interaction in manufacturing. *J. Manuf. Syst.* **2020**, *56*, 326–340. [CrossRef]
4. Zhang, K.; Yang, Z.; Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. In *Handbook of Reinforcement Learning and Control*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 321–384.
5. Mo, Z.; Li, W.; Fu, Y.; Ruan, K.; Di, X. CVLight: Decentralized learning for adaptive traffic signal control with connected vehicles. *Transp. Res. Part C Emerg. Technol.* **2022**, *141*, 103728. [CrossRef]
6. Farinelli, A.; Iocchi, L.; Nardi, D. Distributed on-line dynamic task assignment for multi-robot patrolling. *Auton. Robot.* **2017**, *41*, 1321–1345. [CrossRef]
7. Sui, Z.; Pu, Z.; Yi, J.; Wu, S. Formation control with collision avoidance through deep reinforcement learning using model-guided demonstration. *IEEE Trans. Neural Networks Learn. Syst.* **2020**, *32*, 2358–2372. [CrossRef] [PubMed]
8. Liu, L.; Luo, C.; Shen, F. Multi-agent formation control with target tracking and navigation. In Proceedings of the 2017 IEEE International Conference on Information and Automation (ICIA), Macau, Chin, 18–20 July 2017; IEEE: Piscataway Township, NJ, USA, 2017; pp. 98–103.
9. Ryu, H.; Shin, H.; Park, J. Cooperative and competitive biases for multi-agent reinforcement learning. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, Virtual, 3–7 May 2021; International Foundation for Autonomous Agents and Multiagent Systems: Liverpool, UK, 2021; AAMAS '21; pp. 1091–1099.
10. Hahn, C.; Ritz, F.; Wikidal, P.; Phan, T.; Gabor, T.; Linnhoff-Popien, C. Foraging swarms using multi-agent reinforcement learning. In Proceedings of the ALIFE 2020: The 2020 Conference on Artificial Life, Online, 13–18 July 2020; pp. 333–340.
11. Leitão, P.; Barbosa, J.; Trentesaux, D. Bio-inspired multi-agent systems for reconfigurable manufacturing systems. *Eng. Appl. Artif. Intell.* **2012**, *25*, 934–944. [CrossRef]
12. Stadler, M.; Banfi, J.; Roy, N. Approximating the value of collaborative team actions for efficient multiagent navigation in uncertain graphs. In Proceedings of the International Conference on Automated Planning and Scheduling, Prague, Czech Republic, 8–13 July 2023.
13. Tassel, P.; Kovács, B.; Gebser, M.; Schekotihin, K.; Kohlenbrein, W.; Schrott-Kostwein, P. Reinforcement learning of dispatching strategies for large-scale industrial scheduling. In Proceedings of the International Conference on Automated Planning and Scheduling, Virtual, 13–24 June 2022; Volume 32, pp. 638–646.
14. Xie, S.; Li, Y.; Wang, X.; Zhang, H.; Zhang, Z.; Luo, X.; Yu, H. Hierarchical relationship modeling in multi-agent reinforcement learning for mixed cooperative–competitive environments. *Inf. Fusion* **2024**, *108*, 102318. [CrossRef]
15. Tony, L.A.; Jana, S.; Varun, V.; Shorewala, S.; Vidyadhara, B.; Gadde, M.S.; Kashyap, A.; Ravichandran, R.; Krishnapuram, R.; Ghose, D. UAV collaboration for autonomous target capture. In *Proceedings of the Congress on Intelligent Systems: Proceedings of CIS 2021*; Springer: Berlin/Heidelberg, Germany, 2022; Volume 1, pp. 847–862.
16. Hausman, K.; Müller, J.; Hariharan, A.; Ayanian, N.; Sukhatme, G.S. Cooperative multi-robot control for target tracking with onboard sensing. *Int. J. Robot. Res.* **2015**, *34*, 1660–1677. [CrossRef]
17. Gong, X.; Chen, W.; Chen, Z. All-aspect attack guidance law for agile missiles based on deep reinforcement learning. *Aerosp. Sci. Technol.* **2022**, *127*, 107677. [CrossRef]
18. Shalumov, V. Cooperative online guide-launch-guide policy in a target-missile-defender engagement using deep reinforcement learning. *Aerosp. Sci. Technol.* **2020**, *104*, 105996. [CrossRef]
19. Wu, J.; Huang, Z. Promoting diversity in mixed complex cooperative and competitive multi-agent environment. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, Birmingham, UK, 21–25 October 2023; Association for Computing Machinery: New York, NY, USA; CIKM '23; pp. 4355–4359.
20. Munikoti, S.; Agarwal, D.; Das, L.; Halappanavar, M.; Natarajan, B. Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 15051–15071. [CrossRef] [PubMed]
21. Agarwal, A.; Kumar, S.; Sycara, K.P. Learning transferable cooperative behavior in multi-agent teams. *arXiv* **2019**, arXiv:1906.01202.

22. Niu, Y.; Paleja, R.R.; Gombolay, M.C. Multi-agent graph-attention communication and teaming. In Proceedings of the AAMAS, Virtual, 3–7 May 2021; pp. 964–973.
23. Ma, X.; Yang, Y.; Li, C.; Lu, Y.; Zhao, Q.; Jun, Y. Modeling the interaction between agents in cooperative multi-agent reinforcement learning. *arXiv* **2021**, arXiv:2102.06042.
24. Iqbal, S.; Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 2961–2970.
25. Su, J.; Adams, S.; Beling, P.A. Counterfactual multi-agent reinforcement learning with graph convolution communication. *arXiv* **2020**, arXiv:2004.00470.
26. Liu, Y.; Wang, W.; Hu, Y.; Hao, J.; Chen, X.; Gao, Y. Multi-agent game abstraction via graph attention neural network. *arXiv* **2019**, arXiv:1911.10715. [CrossRef]
27. Jiang, J.; Dun, C.; Huang, T.; Lu, Z. Graph convolutional reinforcement learning. *arXiv* **2018**, arXiv:1810.09202.
28. Sun, F.Y.; Kauvar, I.; Zhang, R.; Li, J.; Kochenderfer, M.J.; Wu, J.; Haber, N. Interaction modeling with multiplex attention. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 20038–20050.
29. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–11.
30. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
31. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
32. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
33. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Abbeel, O.P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30.
34. Wang, X.; Chen, Y.; Zhu, W. A survey on curriculum learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 4555–4576. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.